EasyChair Preprint

№ 767

Software Test-Run Reliability Modeling with Non-homogeneous Binomial Processes

Yunlu Zhao, Tadashi Dohi and Hiroyuki Okamura

February 2, 2019

Software Test-Run Reliability Modeling with Non-homogeneous Binomial Processes

Yunlu ZHAO
Hiroshima University
1-4-1, Kagamiyama
Higashi-Hiroshima, 739-8527 Japan
Email: mirrorZY1@rel.hiroshima-u.ac.jp

Tadashi DOHI
Hiroshima University
1-4-1, Kagamiyama
Higashi-Hiroshima, 739-8527 Japan
Email: dohi@rel.hiroshima-u.ac.jp

Hiroyuki OKAMURA
Hiroshima University
1-4-1, Kagamiyama
Higashi-Hiroshima, 739-8527 Japan
Email: okamu@rel.hiroshima-u.ac.jp

***Abstract*—While the number of test runs (test cases) is often used to define the time scale to measure quantitative software reliability, the common calendar-time modeling with non-homogeneous Poisson processes (NHPPs) is approximately applied to describe the time scale and the software fault-count phenomena as well. In this paper we give a conjecture that such an approximate treatment is not theoretically justified, and propose a simple test-run reliability modeling framework based on non-homogeneous binomial processes (NHBPs). We show that the Poisson-binomial distribution plays a central role in the software test-run reliability modeling, and apply it to the software release decision. In numerical experiments with seven software fault count data we compare the NHBP based software reliability models (SRMs) with their corresponding NHPP based SRMs and refer to an applicability of NHBP based software test-run reliability modeling.**

Keywords: software reliability, test-run reliability, non-homogeneous binomial process, non-homogeneous Poisson process, Poisson-binomial distribution, goodness-of-fit, prediction, software release decision.

I. Introduction

Software reliability is an attribute of software qualities to ensure that software system can continuously provide its service without any failure. In particular, since quantitative software reliability evaluation is useful to control the software testing, there have been a variety of approaches on quantification. One of the most attractive approaches is a model-based approach that describes software reliability growth phenomena in the system testing phase. Software reliability model (SRM) is defined as a stochastic model to represent the software reliability growth phenomena through the software fault counts. From the view point of mathematical tractability, non-homogeneous Poisson processes (NHPPs) are commonly used to describe software fault detection processes. In fact, during the last four decades, a great number of NHPP based SRMs have been proposed by many authors [25],[27],[39].

Quantitative software reliability is defined as the probability that software system does not fail during a specified time period under specified conditions, and can be regarded as a function of time. The commonly used time scale for software reliability assessment is the calendar time such as testing day, week and month. Since the fault count data are easily observed on the testing period measured with the calendar time and our concern is to predict the software reliability for the operational period measured with the calendar time, it may be appropriate to define the time scale for software reliability assessment as the calendar time. On one hand, it can be pointed out that the software testing is not always made uniformly on the calendar time in actual software development projects. For this issue, Musa *et al.* [27] recommend to use the test execution time measured with the CPU time as an alternative time scale. In fact, this is more informative than the calendar time data, because the execution time data and the calendar time data are considered as the software fault-detection time data (complete data) and the software fault count (grouped) data at discrete points (incomplete data), respectively. Though several software fault-detection time data sets have been open in the literature [25],[27], unfortunately, it is known that such data are seldom available in real industry.

Since the main reason why the software test execution time is introduced is to represent the net time scale to measure the testing progress, another idea would be to use the number of test runs (test cases) in the system testing. Yamada *et al.* [42],[44] and Yamada and Osaki [43] propose the so-called discrete NHPP (D-NHPP) based SRMs by replacing the real-valued mean value functions depending on time in an NHPP by the integer-valued ones depending on the number of test runs. Since then, several authors [18],[20],[21] concentrate to develop the similar but somewhat different discrete analogs to the continuous NHPP based SRMs. Okamura *et al.* [29],[30] develop a unified approach for D-NHPP based SRMs and their effective parameter estimation algorithms based on the EM (Expectation-Maximization) principle. Ishii *et al.* [19] consider the D-NHPP based SRMs with two discrete time scales which consist of the discretized calendar time and the number of test runs. Shibata *et al.* [38] provide a software metrics-based modeling framework on the discrete-time scale in the D-NHPP based SRM. Apart from the D-NHPP based SRM, Dewanji *et al.* [10] also apply a logistic regression model to utilize several software metrics data in a flight control software system. Worwa [41] proposes an interesting

discrete SRM under more complex program testing process, and derives the mean value function of the number of software faults experienced. However, he does not refer to the detailed statistical estimation of the underlying SRM and never validate his model with real software fault data.

At the first look, the D-NHPP based SRMs in [18],[20],[21],[42],[43][44] are different from the common continuous NHPP based SRMs. However, it can be easily seen by checking the likelihood functions that these two SRMs with different time scales are mathematically equivalent when the underlying continuous cumulative distribution functions are discretized appropriately by their associated discrete cumulative distribution functions, such like the relation between an exponential distribution and a geometric distribution (see [29],[30]), if the software fault count (grouped) data are observed. Hence, we recognize that the difference between the calendar time modeling and the execution time modeling is important because the corresponding likelihood functions are different from each other, but want to emphasize that no remarkable difference between the continuous NHPP based SRM modeling and the D-NHPP based SRMs exists. When the number of test runs is used for the time scale, we suppose for simplicity that each software test run can detect at most one software fault and that the number of software fault detected at each test run is regarded as a binary random variable taking 0 or 1. These assumptions may not always hold in the actual testing because one test run may be able to detect multiple faults, but can be validated to describe globally the debugging phenomena in the discrete-time scale such as the number of test runs. Because detecting multiple software faults by only one test run is quite rare. Cai [4] considers such a software test-run reliability modeling. In the subsequent paper Cai *et al.* [5] apply an NHPP based SRM to the test-run data sets for two real software systems (space program and SESD program). They generate multiple test data sets by changing the order of test cases, and try to estimate the sample mean and sample variance of the cumulative number of software faults. Looking at the underlying fault count data in [5], it is easily found that our binary assumption holds in except very a few test cases. Their claim is that the use of NHPP for such data is questionable because the sample mean cannot be regarded to equal the sample variance.

However, the above claim is not convinced from the following reasons: (i) The sample paths generated by changing the order of test cases are not the paths sampled from the underlying NHPP, because they are considered as conditioned paths with a given number of residual faults, which is called the Poisson bridge (see *e.g.* [33]). (ii) The model parameters are estimated by means of the maximum likelihood estimation with the grouped data regardless of the discrete complete data. More specifically when the discrete time scale is the number of test runs, it must be greater than or equal to the actual cumulative number of software faults with probability one. However, since the Poisson random variable denoting the cumulative number of software faults is not bounded, *i.e.*, there exists a positive probability that the actual cumulative number of faults is greater than the total number of test runs for D-NHPP based SRMs under the assumption on the binary random variables, the Poissonian assumption contradicts the possible upper bound of the cumulative number of software faults. This fact implies that the D-NHPP based SRMs are not justified theoretically to describe the software fault detection phenomena with respect to the number of test runs, and the conjecture that the underlying binary sequence based on the test runs does not follow an NHPP based SRM can be derived.

In this paper we propose a simple test-run reliability modeling framework based on non-homogeneous binomial processes (NHBPs). Finkelstein [15] considers a similar failure model to our software test-run reliability model, and develop continuous analog estimators (CAEs) with the well-known power-law type NHPP by Duane [12]. Bhattacharyya *et al.* [1] and Bhattacharyya and Ghosh [2] prove the large-sample properties of the resulting CAEs including consistency and asymptotic normality. It is worth noting that the CAEs in [1],[2],[15] are essentially equivalent to the maximum likelihood estimators of NHPP based SRMs with software-fault detection time data, by removing zero-fault count data. So, regarding the binary data based on the software test-runs as a discrete time data, it is possible to estimate the model parameters with NHPP based SRMs in the sense of approximation, because the information on zero-fault count is dropped for the analysis. At the same time, the binary software test-run data can be also viewed as a grouped data. In general, the existence of two likelihood functions for an NHPP based SRM on the discrete time scale such as the binary data seems to be in contradiction to the likelihood principle, because a probability model must have a one-to-one correspondence to the likelihood function. For this controversial argument, we show that the maximum likelihood estimates of NHPP based SRMs are exactly same even though the underlying data are regarded as either software fault-detection time data or software fault count (grouped) data. This fact has not been known in the software reliability engineering literature [25],[27],[39]. Hence, from the similar standpoint to Finkelstein [15], Bhattacharyya *et al.* [1] and Bhattacharyya and Ghosh [2], we take the position that the underlying test-run data are described by NHBPs with given fault-detection probability, and that the common NHPP based SRMs are regarded as approximations of NHBPs under the binary assumption.

To our best knowledge, this paper is the first challenge to treat the software reliability assessment exactly in the test-run reliability modeling. As Bhattacharyya and Ghosh [2] point out, it is easy to conduct the maximum likelihood estimation of NHBP based SRMs with the discrete binary data, but calculating their associated reliability metrics, such as quantitative software reliability, fault-free probability, etc. is not so trivial. Noting that the cumulative distribution function of NHBP is given by a Poisson-binomial distribution, we apply an efficient computation scheme [8] to obtain the exact reliability estimates for several kinds of NHBP based SRMs. Also, we can apply the well-known Le Cam's Poisson approximation [24],[40] to the NHBP based SRMs and show that the common

NHPP based SRMs can be derived approximately from their associated NHBP based SRMs. Further we apply our NHBP modeling framework to the software release decision, which is a practically important decision making in the software project management. In numerical experiments with software fault count data for seven real software programs, we investigate the goodness-of-fit and the predictive performances of our NHBP based SRMs, and compare them with the NHPP analogs. Also, we predict the best timing to release software products to the users or market in order to achieve a satisfactory software reliability level.

II. NHPP BASED SOFTWARE RELIABILITY MODELING

Let $\{N_i;\ i = 0, 1, 2, \cdots\}$ be the cumulative number of software faults detected by the first i-th test run ($i = 0, 1, 2, 3, \cdots$). Suppose that

(A-1) $N_0 = 0$,

(A-2) $\{N_i;\ i = 0, 1, 2, \cdots\}$ has independent increments, *i.e.* for any collection of the numbers of test runs $i_1, i_2, \cdots, i_k$ $(0 < i_1 < i_2 < \cdots < i_k)$, where k random variables, $N_{i_1}, N_{i_2} - N_{i_1}, \cdots, N_{i_k} - N_{i_{k-1}}$, are statistically independent from each other.

(A-3) For any of the numbers of test runs i_s and i_k $(0 \leq i_s < i_k; s < k)$,

$$\Pr\{N_{i_k} - N_{i_s} = n\} = \frac{\{\Lambda_{i_k} - \Lambda_{i_s}\}^n}{n!} e^{-\{\Lambda_{i_k} - \Lambda_{i_s}\}}, \quad (1)$$

where $\Lambda_i = \Lambda_i(\boldsymbol{\theta}) = \mathrm{E}[N_i]$ is called the mean value function with model parameter $\boldsymbol{\theta}$ and is non-decreasing in i $(= 0, 1, 2, \cdots)$.

Under the assumptions (A-1)–(A-3), the counting process in discrete time, $\{N_i;\ i = 0, 1, 2, \cdots\}$, is called the discrete non-homogeneous Poisson process (D-NHPP) having the probability mass function (pmf):

$$\Pr\{N_i = n \mid N_0 = 0\} = \frac{\Lambda_i^n}{n!} e^{-\Lambda_i}, n = 0, 1, 2, \cdots. \quad (2)$$

Suppose that each software test run can detect at most one software fault. If the discrete time scale i is the number of test runs, it holds that $N_i \leq i$ with probability one. However, since the Poisson random variable N_i is not bounded for an arbitrary i, *i.e.*, there exists a positive probability of $N_i > i$ for D-NHPP based SRMs, and the Poissonian assumption contradicts the possible upper bound of the cumulative number of software faults. This fact implies that the D-NHPP based SRMs are questionable to describe the software fault detection phenomena with respect to the number of test runs.

Let X_i $(i = 1, 2, \ldots, m)$ denote the test result which is an independent binary random variable taking the value of 0 (failure) or 1 (success), where m is the total number of test cases. The success means the detection of one software fault. Let $R = \sum_{i=1}^{m} X_i$ and $f(j)$ be the number of successes by m test runs and the number of test runs by the first j-th success $(j = 0, 1, \ldots, R)$, respectively. Finkelstein [15] focuses on only the power-law type model Duane [12], $\Lambda_m = pm^{\beta}$, with the model parameter $\boldsymbol{\theta} \in (p, \beta)$, and derives the continuous analog estimators (CAEs):

$$\hat{p} = Rm^{-\hat{\beta}}, \quad \hat{\beta} = R\Big\{\sum_{j=1}^{R} \log(m/f(j))\Big\}^{-1}. \quad (3)$$

However, it can be easily seen that the CAEs above are the standard maximum likelihood estimators of $\boldsymbol{\theta}$ when the underlying data are regarded as the discrete fault-detection time data. More precisely, let $0 < X_{(1)} < X_{(2)} < ... < X_{(R)}$ be the R success times of test run. Then the log likelihood function is given by

$$\log L_R(\boldsymbol{\theta}) = \sum_{j=1}^{R} \log\{\Lambda_{f(j)} - \Lambda_{f(j-1)}\} - \Lambda_m. \quad (4)$$

Hence the CAEs are the solutions of $\operatorname{argmax} \log L_R(\boldsymbol{\theta})$ satisfying the necessary condition (likelihood equations) $\partial \log L_R(\boldsymbol{\theta}) / \partial \boldsymbol{\theta} = 0$.

On the other hand, if the underlying binary data can be regarded as a grouped data, then the log likelihood function is given by

$$\log L_R(\boldsymbol{\theta}) = \sum_{j=1}^{R} (y_j - y_{j-1}) \log\{\Lambda_{f(j)} - \Lambda_{f(j-1)}\} - \Lambda_m - \sum_{j=1}^{R} \log[(y_j - y_{j-1})!]. \quad (5)$$

Note that there is no difference between Eq .(4) and Eq. (5) because the values of $y_j - y_{j-1}$ are always 1s for all succsesses $j = 1, 2, \ldots, R$, so that Eq. (5) is reduced to Eq. (4). It can be seen that the likelihood function with the binary data in NHPP based SRMs is uniquely given.

In the following section, we introduce the non-homogeneous binomial processes (NHBPs) and clarify a relationship between NHBP based SRMs and NHPP based SRMs.

III. NHBP BASED SOFTWARE RELIABILITY MODELING

Next, we consider a simple SRM based on NHBPs to describe the software fault count phenomena with respect to the number of test runs tried in the system testing. If one software fault is detected at the i-th test case, then $X_i = 1$ with fault-detection (success) probability p_i $(0 < p_i < 1)$, otherwise $X_i = 0$ with probability $1 - p_i$. Then the stochastic process $N_i = \sum_{k=1}^{i} X_k$ is said an NHBP [2] and represents the cumulative number of software faults detected by the first i-th test run. Figure 1 illustrates the configuration of software test-run reliability modeling, where the cumulative number of software faults increases as the number of success test runs increases. Suppose that m test-cases are tried in the system testing. Then, the expected cumulative number of software faults detected by m test runs is easily obtained by

$$\mathrm{E}[N_m] = \Lambda_m = \sum_{i=1}^{m} p_i. \quad (6)$$

It is immediate to see that $m \geq \Lambda_m$ which is an essential requirement in our test-run reliability modeling. Consider a

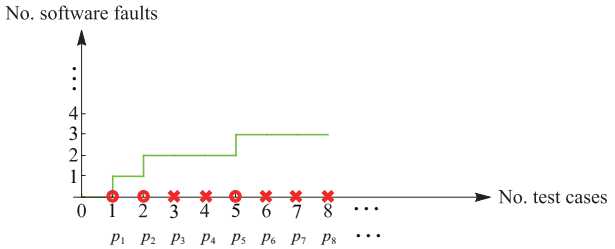


Fig. 1: Configuration of software test-run reliability modeling

special case where the stochastic process is homogeneous, *i.e.*, the fault-detection probability is constant, $p_i = p$, for all i. Then it is straightforward to see

$$\Pr\{N_m = n\} = \binom{m}{n} p^n (1-p)^{m-n}, \tag{7}$$

which is the elementary binomial pmf with

$$\mathrm{E}\left[N_m\right] = mp, \quad \mathrm{Var}\left[N_m\right] = mp(1-p). \tag{8}$$

As the most plausible example in NHBPs, Finkelstein [15] considers a linear relation of $\mathrm{E}\left[N_m\right]$ on a log-log scale as a reasonable growth model for single-mission systems, and proposes the power-law type model from the analogy to Duane model in continuous time [12]; $p_i = p\left[i^\beta - (i-1)^\beta\right]$ $(i = 1, 2, \ldots, m)$ and $\Lambda_m = pm^\beta$, where p $(0 < p < 1)$ denotes the probability that no fault is detected at time $i = 1$ and β $(0 < \beta < 1)$ is viewed as a reliability growth parameter. Note that this model is somewhat different from the well-known discrete Weibull pmf $p_i = p^{(i-1)^\beta} - p^{i^\beta}$ [28]. Let $\boldsymbol{\theta} = (p, \beta)$ be the model parameter of p_i, say, $p_i = p_i(\boldsymbol{\theta})$. Suppose that the binary random sequence $X_1, X_2, \cdots, X_m$ is observed. Then, the log likelihood function $\log L_m(\boldsymbol{\theta})$ with m test runs is given by

$$\log L_m(\boldsymbol{\theta}) = \sum_{i=1}^{m} \left\{ X_i \log p_i(\boldsymbol{\theta}) + (1 - X_i) \log(1 - p_i(\boldsymbol{\theta})) \right\}. \tag{9}$$

By maximizing $\log L_m(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, we get the maximum likelihood estimators of $\boldsymbol{\theta}$.

Once the model parameter $\boldsymbol{\theta}$ is estimated, we need to evaluate the pmf, $\Pr\{N_m = n\}$. From an intuitive probabilistic argument, it holds that

$$\Pr\{N_m = n\} = \sum_{A \in F_n} \prod_{i \in A} p_i \prod_{j \in A^c} (1 - p_j), \tag{10}$$

where F_n is the set of all subsets of n positive integers that can be selected from 1 to m, A^c is the complement of A. The explicit form of pmf for the NHBP is not known but can be calculated recursively [8] by

$$\Pr\{N_m = 0\} = \prod_{i=1}^{m} (1 - p_i), \tag{11}$$

$$\Pr\{N_m = n\} = (1/n) \sum_{j=1}^{n} (-1)^{j-1} \Pr(N_m = n - j) \times \sum_{i=1}^{m} \{p_i/(1-p_i)\}^j. \tag{12}$$

The probability of having n successful trials out of m in Eqs. (11) and (12) is called the Poisson-binomial distribution. The calculation of the recursive equation in Eq. (12) has not been trivial for a long time. For instance, Fernandez and Williams [14] apply the discrete Fourier transform and develop a specific computation algorithm of the Poisson-binomial pmf. However, for the middle size of m, the recursive equation in Eq.(12) can be solved numerically within real time on the recently high power CPU.

Next we approximate NHBP based SRMs by NHPP based SRMs and clarify the inter-relationship. From the Le Cam's Poisson approximation [24],[40], it holds that

$$\sum_{n=0}^{\infty} \left| \Pr\{N_m = n\} - \frac{\Lambda_m^n e^{-\Lambda_m}}{n!} \right| < 2 \sum_{i=1}^{m} p_i^2, \tag{13}$$

where $\Lambda_m = \sum_{i=1}^{m} p_i$. If $p_i = p = \Lambda_m/m$, then the right-hand side of Eq.(12) is $2\Lambda_m^2/m$, which shows that the binomial distribution asymptotically approaches to the Poisson distribution as m is sufficiently large. Hence we have

$$\Pr\{N_m = n\} \approx \frac{\Lambda_m^n e^{-\Lambda_m}}{n!}. \tag{14}$$

From Eq.(14), an arbitrary NHBP based SRM can be approximated by an NHPP based SRM with the mean value function $\Lambda_m = \sum_{i=1}^{m} p_i$ which is different from the existing D-NHPP based SRMs [29],[30].

IV. Discrete Fault-Detection Models

Next we develop the discrete probability models, p_i $(i = 1, 2, \ldots, m)$, to describe the software reliability growth phenomena. The power-law type model by Finkelstein [15] is a discrete version of Duane model [12] and the mean value function $\Lambda_m = pm^\beta$ is monotonically increasing in m. Also, since the pmfs of many discrete probability distributions with positive support are unimodal functions, the mean value function $\Lambda_m = \sum_{i=1}^{m} p_i$ does not saturate to a certain level like an exponential function. One idea to represent the reliability growth phenomena is to introduce the discrete hazard rate function [37] for the fault-detection probability p_i. Dissimilar to continuous probability distributions, the hazard rate functions of discrete pmfs are defined as probability. Hence, it is assumed that the fault-detection probability is given by a discrete hazard rate function with monotone properties. Especially, if the discrete hazard rate function is decreasing [13],[23], it can be expected that the mean value function $\Lambda_m = \sum_{i=1}^{m} p_i$ is a concave and increasing function of m. In this paper we propose the following fourteen fault-detection models:

Model 0 [15]:
$p_i = p[i^\beta - (i-1)^\beta]$ $(0 < p < 1,\ 0 < \beta < 1)$

Model 1 (constant hazard rate):
$p_i = p$ $(0 < p < 1)$

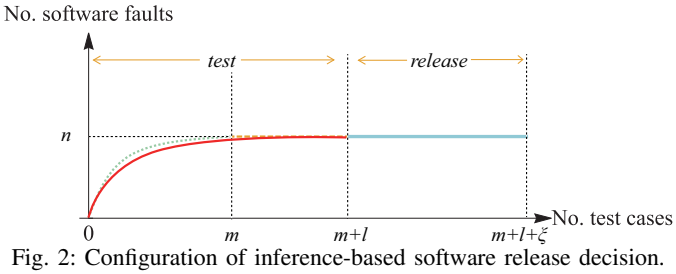


Fig. 2: Configuration of inference-based software release decision.

Model 2 (negative binomial hazard rate):
$p_i = \frac{\binom{p+i-2}{p-1}\beta^p(1-\beta)^{i-1}}{1-\sum_{k=1}^{i-1}\binom{p+k-2}{p-1}\beta^p(1-\beta)^{i-1}}$
$(p = 1, 2, 3, \cdots,\ 0 < \beta < 1)$

Model 3 (discrete IFR) [34]:
$p_i = 1 - p/(\beta i + 1)\ \ (0 < p \leq 1,\ \beta \geq 0)$

Model 4 (discrete DFR) [34]:
$p_i = p/(\beta i + 1)\ \ (0 < p \leq 1,\ \beta \geq 0)$

Model 5 (discrete parametric) [34]:
$p_i = 1 - e^{-p(i+1)^\beta}\ \ (\mu > 0,\ -\infty < \beta < \infty)$

Model 6 (discrete log-logistic hazard rate) [7]:
$p_i = \frac{1-p}{p^{\log i^{-\beta}+1}+1}\ \ (0 < p < 1,\ -\infty < \beta < \infty)$

Model 7 (discrete truncated logistic hazard rate) [7]:
$p_i = \frac{(1-p)p^\beta}{p^{-\beta+i^{i+1}}+1}\ \ (0 < p < 1,\ -\infty < \beta < \infty)$

Model 8 (discrete Burr hazard rate) [22]:
$p_i = 1 - p^{\log\left[\frac{(i+1)^\beta+1}{i^\beta+1}\right]}\ \ (0 < p < 1,\ \beta > 0)$

Model 9 (discrete pareto hazard rate) [22]:
$p_i = 1 - p^{\log\left(\frac{i+2}{i+1}\right)}\ \ (0 < p < 1)$

Model 10 (discrete Weibull hazard rate) [28]:
$p_i = 1 - p^{i^\beta-(i-1)^\beta}\ \ (0 < p < 1,\ 0 < \beta < 1)$

Model 11 (discrete Lindley hazard rate) [17]:
$p_i = \frac{(p-1)\left(\log p^{i+1}-1\right)+p\log p}{1-(i+1)\log p}\ \ (0 < p < 1)$

Model 12 (discrete gamma hazard rate) [6]:
$p_i = 1 - \frac{\Gamma\left(p, \frac{i+1}{\beta}\right)}{\Gamma\left(p, \frac{i}{\beta}\right)}\ \ (p > 0,\ \beta > 0)$

Model 13 (discrete skew logistic hazard rate) [3]:
$p_i = \frac{1-p}{p^{i+1}+1}\ \ (0 < p < 1).$

V. Software Release Decision

We apply the NHBP based SRMs to the software release decision. Okumoto and Goel [32] formulate two software release problems to determine the optimal software release timing. Dalal and McIntosh [9], Dohi *et al.* [11], Fujii *et al.* [16], Momotaz and Dohi [26], Okamura *et al.* [31], Pham and Zhang [35], Saito *et al.* [36] consider different software release problems under several optimization criteria. Among them, it is realistic to consider the reliability criterion to make the software release decision. The quantitative software reliability is defined as the probability that a software system does not fail during a pre-specified operational period of time. Suppose that m test-runs were tried in the system testing and that n software faults were detected and fixed. In this situation, one wishes to keep $100(1-\alpha)\%$ software reliability during the operational phase. Since the time scale employed here is the number of test-cases, it is assumed that the execution length of software in the operational phase is described by ξ test runs. This assumption does seem to be validated when the test cases are designed in accordance with the model-based testing. However, it is noted that $100(1-\alpha)\%$ software reliability with small α cannot be attained just after n software faults were detected, because the longer the zero-fault period, the higher software reliability.

Hence our concern is not to derive the stopping time to test the software. Similar to Fujii *et al.* [16], we consider an inference-based software release decision. That is, if the zero-fault period measured by the number of test runs is l, *i.e.*, we infer that no fault is detected in the period $(m, m+l]$, then the software is released after consuming $m+l$ test cases, otherwise, the fault-free period, l, is measured again just after a fault is detected and fixed. Figure 2 depicts the configuration of our inference-based software release decision. Then the software reliability function for an arbitrary number of software executions in the operational phase ξ is given by

$$\begin{aligned} R &= \Pr\{N_{m+l+\xi} = n | N_{m+l} = n\} \\ &= \prod_{i=m+l+1}^{m+l+\xi} (1-p_i) \geqslant 1-\alpha. \end{aligned} \tag{15}$$

Note that the model parameters $\boldsymbol{\theta} = (p, \beta)$ in NHBP based SRMs at $m+l$ test cases are estimated under the hypothesis that no software fault is detected in the period $(m, m+l]$. Substituting $(X_1,\ X_2, \cdots, X_m, X_{m+1}, \cdots, X_{m+l}) = (x_1, x_2, \cdots, x_m, 0, \cdots, 0)$ into the log likelihood function $\log L_{m+l}(\boldsymbol{\theta})$ in Eq.(9) and maximizing it lead to the inference-based maximum likelihood estimates of $\boldsymbol{\theta}$. It is intuitive to understand that the software reliability function increases as the zero-fault length l increases. Hence the inference-based software release time at the current number of test cases m is interpreted as $m + l^*$ with the minimum zero-fault period $l^* = \min\{l = 1, 2, \ldots; R \geqslant 1-\alpha\}$ so as to satisfy the reliability requirement level R.

VI. Numerical Examples

A. Data Sets

We analyze seven actual software fault datasets, which consist of a pair of the cumulative number of software faults and the cumulative number of test cases. DS1 used in [5] is measured from the space program which is subjected to random testing with a software tool kit, SRATE (Software Reliability Analysis, Testing, and Evaluation). It consists of 9,564 lines of C language code (6,218 executable), and involves several functions as an interpreter for an array definition language (ADL). The program reads a file that contains several ADL statements, and checks the contents of the file for adherence to the ADL grammar and to specific consistency rules. If the

TABLE I: Data sets.

	No. software faults	No. test cases
DS1	33	1200
DS2	25	469
DS3	7	520
DS4	8	518
DS5	22	537
DS6	19	390
DS7	14	3864

ADL file is correct, the program outputs an array data file containing a list of array elements, positions, and excitations; otherwise, outputs error message. In our experiment, we inject 38 software faults in advance. Upon a failure on the test run, only one failure-causing fault is removed. The total number of test cases prepared is 1200. However, since no fault is detected after 200th test run, we use only 300 test cases including additional 100 zero-fault counts in the experiment.

DS2 is also used in [5] and measured from SESD (Software Environment for Software Data Collection) program, which is a grammar analyzer used in a software environment for software data collection. It comprises 3559 lines of C++ code with 3,179 lines being executable. The SESD program generates five outputs; the number of lines of code, the number of total usages of operators, the number of total usages of operands, the number of distinct usages of operators, and the number of distinct usages of operands, for one input. Although 28 software faults are reported in the original test, three of them cannot be detected by only the test runs prepared. Hence, we remove these three faults and inject the remaining 25 faults.

DS3, DS4, DS5, DS6 and DS7 are the representative fault data sets known as Siemens reference programs which are reported by Siemens Corporate Research for a study of the fault detection capabilities of control-flow and data-flow coverage criteria. These programs perform a variety of tasks, such as "schedule", "schedule2", "tcas", "totinfo", and "replace". For instance, D3 is "schedule" which is a priority scheduler with 412 lines of C language code. It outputs the binary data till the last fault is detected by comparing the results between test-runs with and without fault. In this program, it is possible to generate an arbitrary number of test cases, so we execute 520 test runs in our experiment. Table I presents the detailed information on the fault data sets in the 7 development projects.

B. Goodness-of-fit and Predictive Performances

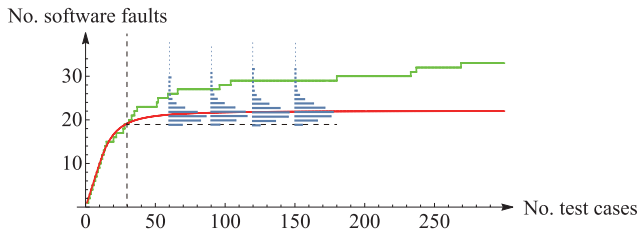
The model parameters are estimated by means of the maximum likelihood method, where the likelihood functions for NHBP and NHPP based SRMs are given in Eqs. (9) and (4), respectively. We observe 10%, 30%, 50%, 70% and 90% of the whole data sets in DS1 $\sim$ DS7 as the training data, and estimate the model parameters at each observation point. Once the model parameter $\boldsymbol{\theta}$ is obtained by maximizing the log likelihood functions, we calculate Akaike Information Criterion (AIC) and the mean squares error (MSE) with the maximum likelihood estimates $\hat{\boldsymbol{\theta}}$ [25],[27]. In each model category of NHBP or NHPP, the goodness-of-fit performance for the past observation is measured by the above criteria, where the smaller AIC (MSE) denotes the better goodness-of-fit model in terms of the smaller distance between our SRM and the real probability model (realization or data). In Tables II and III, we select the best SRM with the minimum AIC at each observation point and calculate MSE as well. For instance, Model 5 [34] is the best discrete fault-detection model in almost all cases with DS1 except at 10% observation point for NHPP based SRM. In the comparison of AIC in NHBP and NHPP based SRMs, it can be seen that our NHBP based SRMs provide the smaller AIC in all cases. On the other hand, NHPP based SRMs show the smaller MSE only in 10 cases out of 35 cases. Especially, NHPP based SRMs give the better goodness-of-fit results for DS2.

Next we predict the future behavior of the cumulative number of software faults at each observation point, where the prediction length is the remaining number of test cases. In Tables II and III, we calculate the predictive log likelihood (PLL) and the predictive mean squares error (PMSE) [25],[27] as prediction metrics, where the larger PLL and smaller PMSE show the better predictive performances. In a fashion similar to the goodness-of-fit performance, it can be observed that our NHBP based SRMs give the larger PLL than the corresponding NHPP based SRMs in all cases. Since the number of free parameters is exactly same for two SRMs, we can recognize that our NHBP based SRM tends to give the better predictive performance. Looking at the PMSE, it is seen that NHPP based SRMs outperform their associated NHBP based SRMs in 9 cases out of 35 cases. Even in the predictive performance, we can show the superority of our NHBP based SRMs in the discrete time scale.

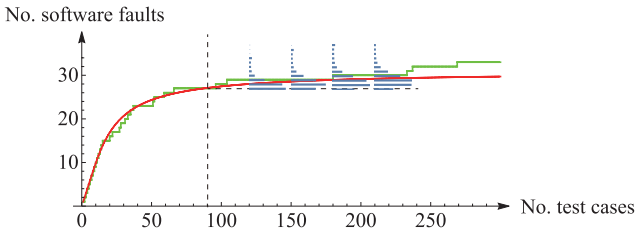
In Figs. 3 and 4, we plot the prediction of the cumulative number of software faults with the best NHBP and NHPP based SRMs with the minimum AIC, respectively, where DS1 is assumed. In both figures, the red line denotes the mean value function for prediction and the green line shows the realization. The predictive Poisson-binomial pmf on the cumulative number of software faults is calculated by using Eqs. (11) and (12) for NHBP based SRMs. In the earlier testing phase, the prediction distribution of the cumulative number of software faults has the long-tail. But as the progress goes on, the shape of pmf is skewed gradually and the variance tends to be much smaller. We zoom up the predictive distributions at 10% observation point with DS1 in Figs. 5 and 6, where the prediction length from the observation point changes from 30 to 120 test runs in the future. From these figures we find that our NHBP based SRMs provide different variances as well as the mean value functions from the corresponding NHPP based SRMs. Since the mean value functions are exactly same as their variance in NHPP based SRMs, our NHBP based SRMs with the Poisson-binomial distribution possess rather different features statistically.

C. Software Release Decision

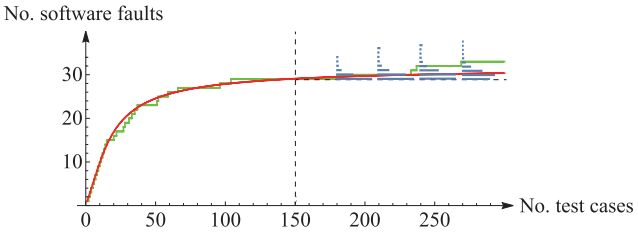
After trying all the test cases in DS1 $\sim$ DS7, consider the case where l additional test runs are tried, where l denote



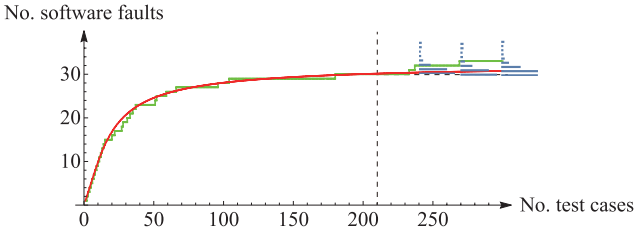
(i)Prediction at 10% observation point with M5.



(ii)Prediction at 30% observation point with M5.

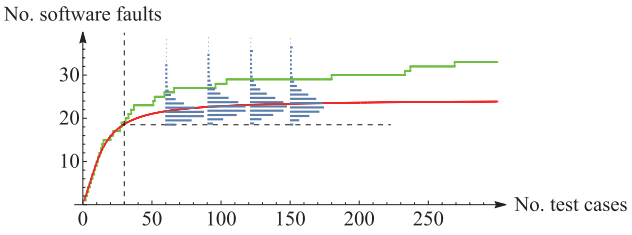


(iii)Prediction at 50% observation point with M5.

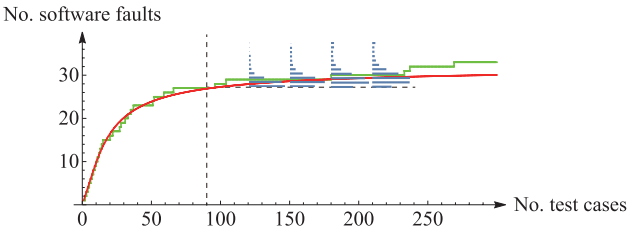


(iv)Prediction at 70% observation point with M5.

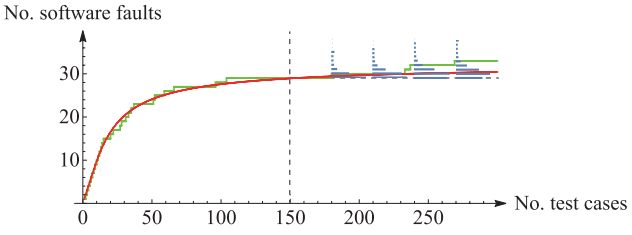
Fig. 3: Prediction of the cumulative number of software faults by the best NHBP based SRM with the minimum AIC (DS1).



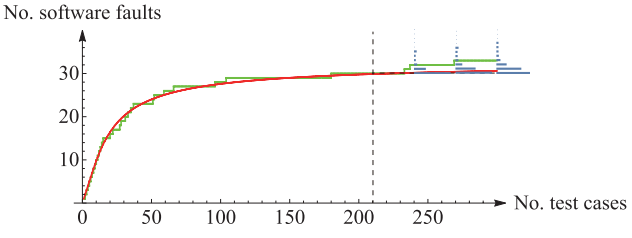
(i)Prediction at 10% observation point with M9.



(ii)Prediction at 30% observation point with M5.



(iii)Prediction at 50% observation point with M5.



(iv)Prediction at 70% observation point with M5.

Fig. 4: Prediction of the cumulative number of software faults by the best NHPP based SRM with the minimum AIC (DS1).

the virtual testing length. We suppose that the software test terminates after $m + l$ test runs and that the software system is released at that time, if no software fault is observed with the additional l test runs. Define the software test-run reliability as the probability that no software fault is detected during the discrete time interval $(m + l, m + l + \xi]$, where ξ is the number of executions in the operational phase in Eq. (15). In the experiments we set as $\xi = 1,000$ and $\alpha = 0.1$, so that the software reliability requirement is assumed to be 90%.

In Table IV, Table V, Table VI and Table VII we calculate the predictive software reliability with varying virtual testing lengh l, where 100% denote the same virtual length as the whole test runs m. In these tables, the underlined value denotes the software reliability which is greater than 90% in each data set. We also indicate the virtual testing length to achieve 90% software reliability in the last column. For instance, in DS4, 2,500% length is needed to guarantee 90% reliability requirement level with NHBP based SRM. From these results we find that rather long virtual testing length is required to achieve over 90% reliability requirement. In DS1, since $m = 1,200$, if no software fault is detected with additional test runs $l = 7,200$, the software reliability is 91.8% with ξ operational executions after releasing the software. This implies that it is actually difficult to keep the high level software reliability requirement such as 90%.

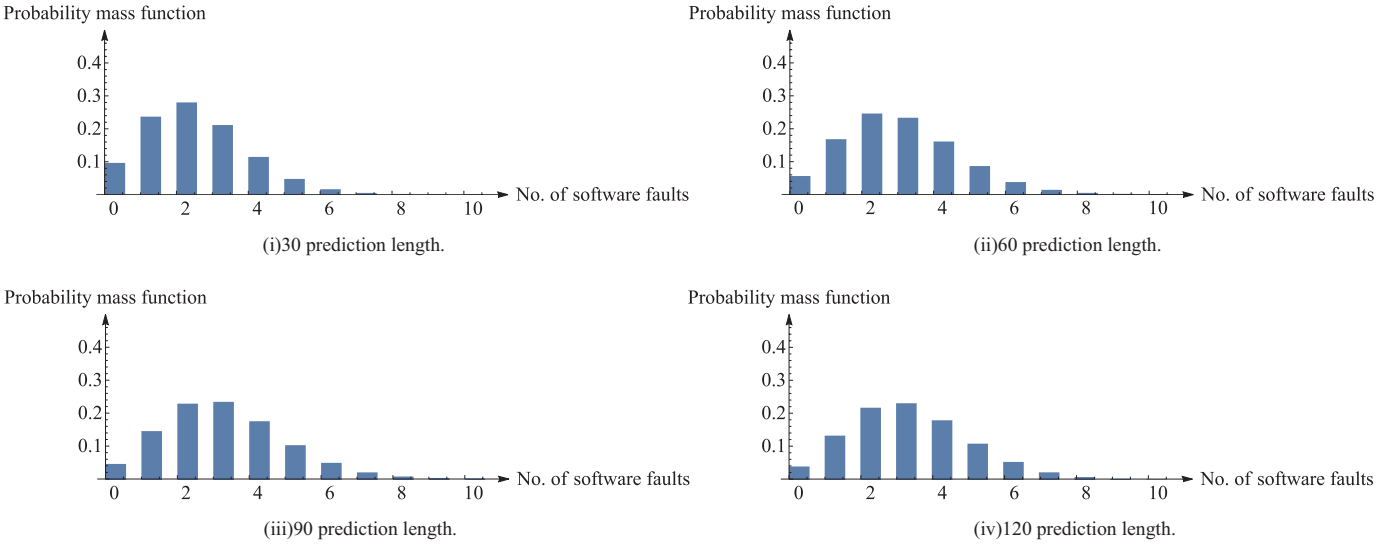


Fig. 5: Predictive pmf of the cumulative number of software faults by the best NHBP based SRM with the minimum AIC (M5) at 10% observation point (DS1).

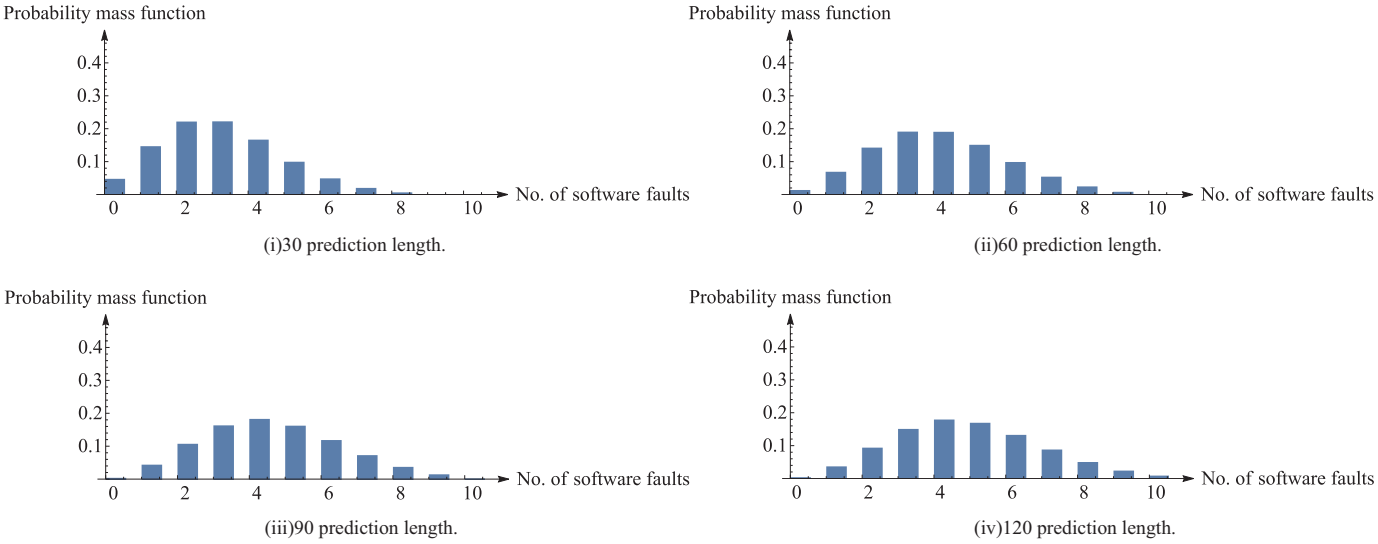


Fig. 6: Predictive pmf of the cumulative number of software faults by the best NHPP based SRM with the minimum AIC (M5) at 10% observation point (DS1).

VII. Conclusion

In this paper we have proposed a simple test-run reliability modeling framework based on non-homogeneous binomial processes (NHBPs) and compared them with the corresponding NHPP based SRMs in numerical experiments. In the software test run reliability modeling, since the NHPP based SRM could not be justified from the physical constraint, we have investigated an applicability of NHBP based SRM and its approximate solution. From the viewpoints of Akaike information criterion and predictive log likelihood, our NHBP based SRMs outperformed their associated NHPP based SRMs. The NHBP based SRMs were applied to the software release decision to achieve the minimum software reliability requirement. It was shown that relatively longer zero-fault counts in additional testing are needed to guarantee 90% software reliability. In future, we will consider the Bayesian estimation in the software test-run reliability modeling. Also it will be a challenging issue to develop NHBP based SRMs with software metrics such as the size metrics, complexity, etc.

References

[1] G. K. Bhattacharyya, A. Fries and R. A. Johnson, "Properties of continuous analog estimators for a discrete reliability growth model", *IEEE Transactions on Reliability*, vol. 38, pp. 373–378, 1989.

[2] G. K. Bhattacharyya and J. K. Ghosh, "Asymptotic properties of estimators in a binomial reliability growth model and its continuous-time analog", *Journal of Statistical Planning and Inference*, vol. 29, pp. 43–53, 1991.

TABLE II: Goodness-of-fit and predictive performances for NHBP based SRMs.

	Best SRM	AIC	MSE	PLL	PMSE
DS1					
10%	M5	27.017	0.357	-64.323	60.401
30%	M5	69.388	0.604	-29.381	3.119
50%	M5	85.934	0.385	-20.277	1.904
70%	M5	96.115	0.343	-15.405	2.765
90%	M5	124.717	2.958	-0.343	0.524
DS2					
10%	M4	55.706	5.407	-44.544	336.531
30%	M9	81.946	11.207	-19.761	15.537
50%	M9	95.756	12.531	-12.460	3.365
70%	M5	107.534	5.603	-6.217	0.741
90%	M5	109.672	6.974	-5.196	0.064
DS3					
10%	M9	27.730	0.122	-18.099	0.836
30%	M9	38.805	0.146	-12.571	0.133
50%	M9	50.049	0.093	-6.890	0.257
70%	M9	50.885	0.162	-6.492	0.069
90%	M9	51.475	0.281	-6.233	0.020
DS4					
10%	M0	13.547	0.069	-36.004	12.666
30%	M0	42.879	0.324	-17.224	0.308
50%	M0	53.877	0.294	-11.695	0.276
70%	M0	65.212	0.252	-6.157	0.470
90%	M10	66.425	0.413	-5.706	0.028
DS5					
10%	M9	50.211	0.572	-45.004	0.901
30%	M9	96.425	0.348	-22.008	2.451
50%	M9	115.949	0.411	-12.182	1.689
70%	M9	127.395	0.770	-6.416	1.028
90%	M9	129.699	2.225	-5.257	0.079
DS6					
10%	M9	41.600	0.348	-32.095	6.019
30%	M9	84.870	1.586	-11.059	19.747
50%	M9	90.180	4.962	-7.703	4.206
70%	M9	93.300	7.292	-6.037	0.846
90%	M5	95.134	4.814	-5.234	0.041
DS7					
10%	M9	47.882	0.231	-26.271	2.232
30%	M9	61.627	0.434	-19.375	0.212
50%	M9	73.723	0.362	-13.326	0.187
70%	M9	86.712	0.261	-6.810	0.112
90%	M9	99.870	0.492	-0.223	0.017

TABLE III: Goodness-of-fit and predictive performances for NHPP based SRMs

	Best SRM	AIC	MSE	PLL	PMSE
DS1					
10%	M9	53.586	1.583	-75.013	135.514
30%	M5	98.132	0.594	-55.596	2.553
50%	M5	114.645	0.453	-49.343	2.335
70%	M5	124.893	0.346	-44.976	3.337
90%	M5	152.761	1.119	-32.689	0.269
DS2					
10%	M5	66.914	2.645	-56.006	1.117
30%	M5	87.113	1.682	-44.372	0.581
50%	M5	99.578	1.418	-37.218	0.208
70%	M5	111.890	1.420	-31.152	0.385
90%	M5	112.334	1.075	-31.129	0.299
DS3					
10%	M9	28.461	0.660	-22.059	1.209
30%	M9	39.458	0.951	-17.536	1.241
50%	M9	50.908	1.324	-12.889	0.984
70%	M9	51.575	1.342	-12.494	0.714
90%	M9	52.075	1.220	-12.235	0.556
DS4					
10%	M9	15.128	0.162	-36.419	10.663
30%	M9	44.176	0.733	-22.825	0.733
50%	M9	55.257	0.682	-18.192	0.309
70%	M9	66.681	0.824	-13.405	0.095
90%	M9	67.386	0.529	-13.091	0.021
DS5					
10%	M9	55.368	0.572	-57.204	0.901
30%	M9	101.884	0.348	-40.034	2.451
50%	M9	121.478	0.411	-32.186	1.689
70%	M9	132.924	0.759	-27.429	1.046
90%	M9	135.245	2.225	-26.263	0.079
DS6					
10%	M9	47.558	0.348	-43.490	6.019
30%	M9	91.753	1.586	-28.978	19.747
50%	M9	96.936	4.962	-25.686	4.206
70%	M5	99.025	3.065	-26.214	5.058
90%	M5	99.662	4.304	-26.481	5.880
DS7					
10%	M9	50.031	0.231	-34.272	2.232
30%	M9	63.683	0.434	-28.382	0.212
50%	M9	75.788	0.362	-23.329	0.187
70%	M9	88.801	0.261	-17.811	0.112
90%	M9	101.987	0.492	-12.223	0.017

[3] D. Bhati, S. Chakraborty, S. G. Lateef, "An alternative discrete skew logistic distribution", *Working Paper*, *arXiv preprint arXiv:* 1604.01541, 2016.

[4] K.-Y. Cai, "Towards a conceptual framework of software run reliability modeling", *Information Science*, vol. 126 pp. 137–163, 2000.

[5] K.-Y. Cai, D.-B. Hu, C.-G. Bai, H. Hu and T. Jing, "Does software reliability growth behavior follow a non-homogeneous Poisson process", *Information and Software Technology*, vol. 50, pp. 1232–1247, 2008.

[6] S. Chakraborty and D. Chakravarty, "Discrete gamma distributions: properties and parameter estimations", *Communications in Statistics - Theory and Methods*, vol. 41, no. 18, pp. 3301-3324, 2012.

[7] S. Chakraborty and D. Chakravarty, "A new discrete probability distribution with integer support on $(-\infty, \infty)$", *Communications in Statistics - Theory and Methods*, vol. 45, no. 2, pp. 492–505, 2016.

[8] S. X. Chen and J. S. Liu, "Statistical applications of the Poisson-binomial and conditional Bernoulli distributions", *Statistica Sinica*, vol. 7, pp. 875-892, 1997.

[9] S. R. Dalal and A. A. McIntosh, "When to stop testing for large software systems with changing code", *IEEE Transactions on Software Engineering*, vol. 20, no. 4, pp. 318–323, 1994.

[10] A. Dewanji, D. Sengupta and A. K. Chakraborty, "A discrete time model for software reliability with application to a flight control software", *Applied Stochastic Models in Business and Industry*, vol. 27, pp. 723–731, 2011.

[11] T. Dohi, Y. Nishio and S. Osaki, "Optimal software release scheduling based on artificial neural networks", *Annals of Software Engineering*, vol. 8, pp. 167–185, 1999.

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

[13] N. Ebrahimi "Classes of discrete decreasing and increasing mean-residual-life distribution", *IEEE Transactions on Reliability*, vol. R-35, no. 4, pp. 403–405, 1986.

[14] M. Fernandez and S. Williams, "Closed-form expression for the Poisson-binomial probability density function", *IEEE Transactions on Aerospace Electronic Systems*, vol. 46, pp. 803–817, 2010.

[15] J. M. Finkelstein, "A logarithmic reliability growth model for single-mission systems", *IEEE Transactions on Reliability*, vol. R-32, pp. 508–511, 1983.

[16] T. Fujii, T. Dohi, H. Okamura and T. Fujiwara, "A software accelerated life testing model", *Proceedings of 16th IEEE Pacific Rim International*

TABLE IV: Inference-based software release decision with NHBP based SRM.

Data Set	DS1	DS2	DS3	DS4	DS5	DS6	DS7
Best SRM	M5	M5	M9	M10	M9	M5	M9
100%	0.437	0.286	0.457	0.161	0.071	0.357	0.468
200%	0.656	0.487	0.583	0.311	0.165	0.577	0.602
300%	0.762	0.617	0.664	0.422	0.257	0.689	0.686
400%	0.837	0.707	0.720	0.519	0.337	0.765	0.742
500%	0.879	0.762	0.761	0.573	0.406	0.815	0.782
600%	0.904	0.805	0.792	0.636	0.464	0.848	0.811
700%	0.924	0.836	0.816	0.683	0.512	0.875	0.834
800%	0.937	0.861	0.835	0.710	0.554	0.896	0.852
900%	0.947	0.881	0.851	0.742	0.589	0.910	0.867
1000%	0.956	0.893	0.864	0.766	0.620	0.923	0.879
1100%	0.962	0.906	0.875	0.776	0.647	0.932	0.890
1200%	0.966	0.914	0.885	0.795	0.671	0.939	0.898
1300%	0.971	0.925	0.893	0.810	0.692	0.946	0.906
1400%	0.974	0.931	0.900	0.820	0.710	0.950	0.912
1500%	0.978	0.938	0.907	0.835	0.727	0.954	0.918
1600%	0.978	0.944	0.912	0.844	0.741	0.957	0.923
1700%	0.980	0.948	0.917	0.855	0.755	0.962	0.927
1800%	0.983	0.952	0.922	0.863	0.767	0.965	0.931
1900%	0.984	0.955	0.926	0.871	0.778	0.969	0.935
2000%	0.986	0.958	0.929	0.876	0.788	0.970	0.938
$R \geqslant 0.9$	600%	1100%	1400%	2500%	4400%	900%	1300%

TABLE V: Inference-based software release decision with NHPP based SRM.

Data Set	DS1	DS2	DS3	DS4	DS5	DS6	DS7
Best SRM	M5	M5	M9	M9	M9	M5	M9
100%	0.447	0.752	0.455	0.402	0.076	0.619	0.483
200%	0.677	0.867	0.583	0.536	0.176	0.789	0.616
300%	0.791	0.917	0.665	0.624	0.271	0.866	0.698
400%	0.855	0.943	0.721	0.686	0.353	0.907	0.752
500%	0.893	0.958	0.762	0.731	0.423	0.931	0.791
600%	0.918	0.968	0.793	0.765	0.480	0.947	0.819
700%	0.935	0.975	0.817	0.792	0.529	0.958	0.841
800%	0.948	0.980	0.836	0.814	0.570	0.966	0.859
900%	0.957	0.983	0.852	0.832	0.605	0.972	0.873
1000%	0.964	0.986	0.865	0.846	0.635	0.976	0.885
1100%	0.969	0.988	0.876	0.859	0.662	0.979	0.894
1200%	0.973	0.990	0.886	0.870	0.685	0.982	0.903
1300%	0.977	0.991	0.894	0.879	0.705	0.984	0.910
1400%	0.980	0.992	0.901	0.887	0.723	0.986	0.916
1500%	0.982	0.993	0.907	0.894	0.739	0.988	0.922
1600%	0.984	0.994	0.913	0.901	0.753	0.989	0.926
1700%	0.986	0.994	0.918	0.906	0.766	0.990	0.931
1800%	0.987	0.995	0.922	0.911	0.778	0.991	0.935
1900%	0.988	0.995	0.926	0.916	0.789	0.992	0.938
2000%	0.989	0.996	0.930	0.920	0.798	0.992	0.941
$R \geqslant 0.9$	600%	300%	1400%	1600%	4100%	400%	1200%

Symposium on Dependable Computing (PRDC-2010), pp. 8592, IEEE CPS, 2010.

[17] E. Gmez-Dniz and E. Caldern-Ojeda, "The discrete Lindley distribution: properties and applications", *Journal of Statistical Computation and Simulation*, vol. 81, no. 11, pp. 1405–1416, 2011.

[18] S. Inoue and S. Yamada, "Discrete software reliability assessment with discretized NHPP models", *Computer & Mathematics with Applications*, vol. 51, no. 2, pp. 161–170, 2006.

[19] T. Ishii, T. Fujiwara and T. Dohi, "Bivariate extension of discrete software reliability modeling with number of test cases", *International Journal of Reliability, Quality and Safety Engineering*, vol. 15, no. 1, pp. 1–17, 2008.

[20] P. K. Kapur, S. Younes and S. Agarwala, "A general discrete software reliability growth model", *International Journal of Modelling and Simulation*, vol. 18, no. 1, pp. 60–65, 1998.

[21] P. K. Kapur, O. M. Pal Singh, O. Shatnawi and A. Gupta, "A discrete NHPP model for software reliability growth with imperfect fault debugging and fault generation", *International Journal of Performability Engineering*, vol. 2, no. 4, pp. 351–368, 2006.

[22] H. Krishna and P. S. Pundir, "Discrete Burr and discrete Pareto distributions", *Statistical Methodology*, vol. 6, no. 2, pp. 177-188, 2009.

[23] N. A. Langberg, R. V. Leon, J. Lynch and F. Proschan, "Extreme points of the class of discrete decreasing failure rate life distributions", *Mathematics of Operations Research*, vol. 5, no. 1, pp. 35–42, 1980.

[24] L. Le Cam, "An approximation theorem for the Poisson binomial distribution", *Pacific Journal of Mathematics*, vol. 10, no. 4, pp. 1181–1197, 1960.

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

[26] B. Momotaz and T. Dohi, "Optimal stopping time of software system test via artificial neural network with fault count data", *Journal of Quality in Maintenance Engineering*, vol. 24, no. 1, pp. 22–36, 2018.

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

[28] T. Nakagawa and S. Osaki, "The discrete Weibull distribution", *IEEE Transactions on Reliability*, vol. R-24, pp. 300–301, 1975.

[29] H. Okamura, A. Murayama and T. Dohi, "EM algorithm for discrete software reliability models: a unified parameter estimation method", *Proceedings of The 8th IEEE International Symposium on High Assurance Systems Engineering (HASE-2004)*, pp. 219–228, IEEE CPS, 2004.

[30] H. Okamura, A. Murayama and T. Dohi, "A unified parameter estimation algorithm for discrete software reliability models", *Opsearch*, vol. 42, no. 4, pp. 355–377, 2006.

[31] H. Okamura, T. Dohi and S. Osaki, "Bayesian inference for credible intervals of optimal software release time", *Advances in Software Engineering & Its Applications (ASEA 2011)* (T.-H. Kim, H. Adeli, H.-K. Kim, H.-J. Kang, K. J. Kim, A. Kiumi and B.-H. Kang, eds.), CCIS, vol. 257, pp. 377–384, Springer-Verlag, 2011.

[32] K. Okumoto and A. L. Goel, "Optimal release time for software systems based on reliability and cost criteria", *Journal of Systems and Software*, vol. 1, pp. 315–318, 1980.

[33] K. Okumura, H. Okamura and T. Dohi, "On the effect of the order of test cases in the modified exponential software reliability growth model", *Proceedings of 2011 Second International Conference on Networking and Computing (ICNC-2011)*, pp. 294–296, IEEE CPS, 2011.

[34] W. J. Padgett and J. D. Spurrier, "On discrete failure models", *IEEE Transactions on Reliability*, vol. R-34, pp. 253–256, 1985.

[35] H. Pham and X. Zhang, "A software cost model with warranty and risk costs", *IEEE Transactions on Computers*, vol. 48, no. 1, pp. 71-75, 1999.

[36] Y. Saito, T. Moroga and T. Dohi, "Optimal software release decision based on nonparametric inference approach", *Journal of the Japan Industrial Management Association*, vol. 66, pp. 396–405, 2016.

[37] M. Shaked, J. G. Shanthikumar and J. B. Valdez-Torres, "Discrete hazard rate functions", *Computers and Operations Research*, vol. 22, no. 4, pp. 391–402, 1995.

[38] K. Shibata, K. Rinsaka and T. Dohi, "Dynamic software reliability modeling with discrete-test metrics; how good is it?", *International Journal of Industrial Engineering*, vol.14, no. 4, 332–339, 2007.

[39] N. D. Singpurwalla and S. P. Wilson, *Statistical Methods in Software Engineering*, Springer-Verlag, New York, 1999.

[40] J. M. Steele, "Le Cam's inequality and Poisson approximation", *The American Mathematical Monthly*, vol. 101, pp. 48–54, 1994.

[41] K. Worwa, "A discrete-time software reliability-growth model and its application for predicting the number of errors encountered during program", *Control and Cybernetics*, vol. 34, no. 2, pp. 589–606, 2005.

[42] S. Yamada, S. Osaki and H. Narihisa, "Software reliability growth modeling with number of test runs", *Transactions of the IECE of Japan*, vol. E67, pp. 79–83, 1984.

[43] S. Yamada and S. Osaki, "Discrete software reliability growth models", *Applied Stochastic Models and Data Analysis*, vol. 1, pp. 65–77, 1985.

[44] S. Yamada, S. Osaki and H. Narihisa, "Discrete models for software reliability evaluation", *Reliability and Quality Control* (A. P. Basu, ed.), pp. 401–412, North-Holland, New York, 1986.