Using the Methods of Statistical Data Analysis to Improve the Trustworthiness of Software Reliability Modeling

YI-FAN HOU\textsuperscript{1}, CHIN-YU HUANG\textsuperscript{2}, (Member, IEEE), AND CHIH-CHIANG FANG\textsuperscript{2}

\textsuperscript{1}Institute of Information Systems and Applications, National Tsing Hua University, Hsinchu 30013, Taiwan
\textsuperscript{2}Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan

Corresponding author: Chin-Yu Huang (cyhuang@cs.nthu.edu.tw)

This work was supported by the Ministry of Science and Technology, Taiwan, under Grant MOST 108-2221-E-007-033-MY3 and Grant MOST 110-2221-E-007-035-MY3.

ABSTRACT In the past, several software reliability growth models (SRGMs) have been proposed to predict various aspects of software reliability, and non-homogeneous Poisson process (NHPP)-based SRGMs have drawn much attention. However, it should be noted that, in practice, blindly applying SRGMs might not lead to meaningful results when the characteristics of the data (i.e., data type) do not fit the characteristics of selected SRGMs. Moreover, there currently exist no guidelines for project managers or developers to follow in selecting the most appropriate SRGMs for a particular failure dataset. On the other hand, waiting to collect a substantial amount of failure data before fitting SRGMs might be unfeasible in many cases. In this paper, we propose using statistical data analysis methods to investigate the characteristics of software failure data and to improve the accuracy of software reliability growth modeling. Here, SRGMs will be broadly partitioned into two categories: “NHPP models” and “non-NHPP models.” Similarly, software failure data can be classified into two categories: “NHPP-type data” and “non-NHPP-type data.” We propose a two-phase method to verify the hypothesis that failure data follow an NHPP. Through the proposed two-phase and Laplace trend tests, we will be able to infer whether a dataset is the NHPP type and whether it is properly used by NHPP-based SRGMs. In this case, the final choice for the most suitable NHPP model for the software failure process would be made, and the software reliability prediction result can be deemed trustworthy and helpful after correctly judging whether the dataset can be classified or not as being NHPP based. Experiments based on 25 real software failure data were presented and analyzed. Using our proposed method, project managers and/or developers will be able to analyze and use collected failure data to help improve both planning accuracy and software quality.

INDEX TERMS Software reliability, failure data, NHPP, SRGMs, statistical analysis.

I. INTRODUCTION

In modern society, software plays a crucial role in several institutions and organizations. People rely heavily on software in many aspects of their lives. Presently, source code for developing software systems can be roughly divided into two categories: closed source software (CSS, e.g., Microsoft Office, Apple’s iOS, etc.) and free and open source software (FOSS, e.g., Mozilla Firefox, Ubuntu Linux, GCC, Open Office, etc.). To measure the reliability of CSS and FOSS systems, software practitioners must track the failure-occurrence process based on collected failure data. When more and more failure data or customer-reported bugs are obtained, it is possible to more precisely predict software reliability, failure intensity, the number of remaining faults, and so on. Software reliability is typically defined as the probability of the failure-free operation of a software program for a specified time in a specified environment [1].

Many software reliability growth models (SRGMs) have been published in the past, and several efforts have been made to estimate software reliability from collected datasets [2]–[9]. A wide variety of SRGMs that can be classified based on several different classification criteria have been proposed. For example, Musa and Okumoto [1], [7]
developed a classification scheme for software reliability models in terms of five attributes: time domain (calendar time or execution time), category (the total number of failures that can be experienced in infinite time as finite or infinite), type (the failure quantity distribution), class (finite failures category only), and family (infinite failures category only).

Pham [8] once reported that there were two main types of SRGMs: deterministic and probabilistic. In addition, Gokhale and Trivedi [9] argued that SRGMs can be categorized as data-domain and time-domain models.

In practice, most existing SRGMs typically assume that the number of faults detected within a specified time interval follows the non-homogeneous Poisson process (NHPP) with a mean value function (MVF) [3]. Theoretically, the homogeneous Poisson process (HPP) model typically assumes that the interarrival times between failures are independent and identically distributed and that the expected value of the number of failures in a time interval depends only on the length of the interval and not its start point. In actuality, the NHPP models form a prominent class, and there are some studies that discuss the assumptions of the NHPP model [5], [6]. NHPP differs from HPP in that the failure rate varies with time instead of being constant.

Over the past three decades, NHPP-type models have provided a novel conceptual and analytical framework for describing failure occurrence during software testing. Xie [10] reported that NHPP models were simple and easy to use and was the type of software reliability model most widely applied. Lyu [11] also indicated that the NHPP model formed the basis for the models using the observed number of faults per unit of time group. Musa et al. [1] studied and described existing Poisson-type models as special cases of the underlying general NHPP. The applications of NHPP models include important and safety-critical systems, such as the shuttle’s on-board system software, weapon systems, and so on [11]–[19].

It is worth noting that there exist no policies or guidelines that can be followed to select or use the most appropriate SRGMs for a particular failure dataset. There is a practical need to discuss and develop a set of recommendations for filtering and smoothing collected failure datasets. Consequently, in this paper, SRGMs are broadly partitioned into two categories: “NHPP-based models” and “non-NHPP models.” We also suppose that software failure data can be broadly classified into two groups: “NHPP-type data” and “non-NHPP-type data.” Here, a simple table can be created, as shown in Table 1, in which the row and column variables have the same number of categories.

Referring to Table 1, the row variable is the “Data type” and the column variable is the “Model type,” and the table dimension is 2 × 2. The figure’s first row indicates cases for NHPP-based models occurring with NHPP-type data and for the NHPP-based models occurring with the non-NHPP-type data. The first column shows a case for the NHPP-based models occurring with each of the two data categories. Altogether, we have the following four categories:

1. NHPP-based models with NHPP-type data
2. NHPP-based models with non-NHPP-type data
3. Non-NHPP models with NHPP-type data
4. Non-NHPP models with non-NHPP-type data

As depicted in Table 1, of the four combinations, we will pay more attention to the NHPP-based models with non-NHPP-type data, rather than non-NHPP models with either NHPP-type or non-NHPP-type data. That is, we would like to thoroughly investigate the possibility that before we estimate the reliability by using SRGM(s), we are willing to accept the false conclusion that the collected failure dataset is the NHPP type when it is not. This part should be vital for engineers and project managers before they plan to use SRGMs to estimate software reliability from real datasets. Thus, cases for non-NHPP models with both NHPP-type and non-NHPP-type data are not discussed in our study.

In this paper, we first propose a two-phase method to verify the hypothesis that failure data follow an NHPP with a decreasing intensity function. The first phase of the proposed method is to examine the whole dataset as an interval. If a dataset does not pass the first phase of examination, it will be adjusted by region partitioning and/or by removing outliers. Then, it will be examined again in the second phase of the proposed method. Moreover, the Laplace trend test is also performed, and we calculate the Laplace trend factor [11]. The Laplace test is widely used to identify trends in grouped data or time series [1]. In actuality, the Laplace test is one of the most commonly used methods in software reliability engineering because it is often found to be the most appropriate method when failures and fault removals follow the NHPP [20].

Through the proposed two-phase procedure and the Laplace trend test, we will be able to infer whether a dataset is the NHPP type and whether it is properly used by NHPP-based SRGMs. That is, the final decision of choosing the most suitable NHPP model(s) to model the software failure process would be made after correctly judging whether the selected dataset is definitely NHPP type or not. In our experiments, 25 real failure datasets will be examined. Approximately 28% (7/25) of the selected failure datasets passed the first phase of our proposed method without any data adjustment. A case study was conducted for the second phase of our proposed method with a dataset that failed the examination for both NHPP characteristics, and it passed the examination after some data adjustment.

The remainder of this paper is designed as follows. Section II provides a survey of related works, and some discussions about selected software failure datasets that are said to be definitely NHPP type or not are also presented. Our proposed two-phase test procedure is presented in Section III.
In Section IV, some experiments based on 25 real software failure data are presented and analyzed. Finally, some concluding findings are described in Section V.

II. BACKGROUND AND RELATED WORK

The human error process introduces defects into the code, and the run selection process determines which code is being executed at any given time. Software failure is the departure of the external results of the program operation from the software product requirements. The number of failures experienced in a time interval is a common way to describe the failure random process. It consists of a sequence of “point events,” where each point has no memory and is statistically independent of the past. This behavior is called nonstationary and can be modeled using an NHPP [21]. The behavior of NHPP is, in a sense, completely random and determined entirely by the rate at which failures occur. One characteristic of Poisson processes is that the mean is equal to the variance at any given time [1]. That is, if \( N(t) \) is the NHPP with MVF \( \mu(t) \), the variance of \( N(t) \) also equals to \( \mu(t) \). In fact, we can view the failure process from a different perspective. If we let \( H(t) \) denote failures remaining at time \( t \), both the mean and variance of \( H(t) \) are given as \( \omega(t) = \mu(\infty) - \mu(t) \). That is, the variance of \( H(t) \) decreases as software testing proceeds. Knowing the number of remaining faults is vital and useful in software development.

As mentioned in the preceding section, numerous SRGMs have been published in the literature and are applicable to the late stages of testing in software development, which can provide useful information in predicting and improving the reliability of software products. Some classical SRGMs based on NHPP, such as the Goel-Okumoto model, the Yamada delayed S-shaped model, Goel’s generalized GO model, and the Inflection S-shaped model, are defined in the following section. However, it has to be noted that most SRGMs are based on calendar time (e.g., the Jelinski-Moranda Model), staff time (e.g., the Shooman Model), or computer time (e.g., the Musa Model) [1], [3], [11]. We will briefly review these SRGMs in the following sections.

A. THE GOEL-OKUMOTO MODEL

The GO model was first proposed by Goel and Okumoto and was the most famous NHPP model in the field of software reliability modeling [2], [3], [11]. The GO model is also called the exponential model. Considering failure detection as NHPP with an exponentially decaying rate function, the MVF of the GO model is given by [2], [3], [11]:

\[
m(t) = a(1 - e^{-bt}),
\]

where \( a \) is the expected number of initial faults and \( b \) is the fault detection rate per fault in the steady state. The GO model was validated using Navy software failure data and was also implemented in CASRE and SMERS tools [22].

B. THE YAMADA DELAYED S-SHAPED MODEL

The Yamada Delayed S-shaped model is a modification of the NHPP to obtain an S-shaped curve for the cumulative number of failures detected where the failure rate initially increases and later (exponentially) decays [3], [11]. Its MVF is defined by

\[
m(t) = a(1 - (1 + bt)e^{-bt}),
\]

C. THE INFLECTION S-SHAPED MODEL

The Inflection S-shaped model was proposed by Ohba and can be regarded as a mutant of the logistic curve. The Inflection S-shaped model solves a technical condition with the G-O model and is widely used in Japan. The MVF of the Inflection S-shaped model is defined as [3], [11]:

\[
m(t) = \frac{a(1 - e^{-bt})}{1 + ce^{-bt}},
\]

where \( c \) is the inflection parameter and \( c = (1/\psi) - 1 \). Here \( \psi \) is the inflection rate that indicates the ratio of the number of detectable faults to the total number of faults in the software.

D. THE GENERALIZED GOEL-OKUMOTO MODEL

Goel once proposed a simple generalization of the Goel-Okumoto model, and the MVF of the generalized Goel-Okumoto model is given by the following [3], [11]:

\[
m(t) = a(1 - e^{-bt^m}),
\]

where \( m \) is the testing quality. Note that the fault detection rate of Goel’s generalized GO model is time dependent and determined by the quality of the testing parameter. It was shown that this model could provide a better goodness-of-fit than the GO model.

E. THE MUSA-OKUMOTO LOGARITHMIC MODEL

The Musa–Okumoto logarithmic model assumes that all faults are equally likely to occur and are independent of each other [1]. As one of the best predictive models, the Musa–Okumoto model belongs to the selected models in the ANSI/AIAA. The failure intensity of the Musa-Okumoto logarithmic model is given by [1], [3], [11], [16]:

\[
\lambda(t) = \lambda_0 \exp(-\theta m(t)),
\]

where \( \lambda_0 > 0 \) is the initial failure intensity and \( \theta \) is the failure rate decay parameter. The expected number of failures is a logarithmic function of the execution time:

\[
m(t) = \frac{1}{\theta} \ln(\lambda_0 \theta t + 1).
\]

In the past, Tamura and Yamada [24], [25] proposed a method of system-wide reliability assessment, considering big data on cloud computing. They used software failure-occurrence time data and the cumulative number of detected faults data by applying the hazard rate model and stochastic differential equation and performed a cluster analysis for the software fault data by using k-means clustering. There is also some research on applying data mining in reliability and risk assessment. For example, Maimon et al. [26] proposed a fuzzy approach to evaluating unreliable data in a relational
database. Saitta et al. [27] also presented a system identification methodology that makes use of data mining techniques to improve identification reliability. Additionally, Rekha and Parvathi [28] reviewed big data analytics and software project risk. Herdt et al. [29] reported that combined analysis of failure data, such as control charts, SRGMS, and Laplace trend analysis, would add new insights into the testing evaluation.

Additionally, Rocco [30] mentioned that the objective of data mining was either prediction or description. Predictive data mining involves using some variables or fields in the dataset to predict unknown values of the variables of interest. Descriptive data mining involves finding patterns and relationships described by the data that can be interpreted. Meeker and Hong [31] reviewed some applications where field reliability data are used and explored some of the opportunities to use reliability data to provide stronger statistical methods and to predict (and operate) the system performance. However, they also argued that big data would not solve all reliability problems, and detailed knowledge of the physics of failure is exceedingly important to provide some degree of assurance for the resulting predictions.

Actually, analyzing the characteristics of failure data is a vital task for software developers and project managers. In general, all records of fault detection and corrections are kept in the software change management system and predefined test reports. Project managers and developers will be able to analyze and use collected failure data to help improve both planning accuracy and software quality. However, some research has also indicated that the early phase of data may not be proper for curve fitting due to the low stability of the data [32]. Stability problems occur in the early phases of system testing, where the data may exhibit insufficient reliability growth for accurate estimates. Additionally, the effects of data discrepancies could be very pronounced, especially in the early stage of the software development life cycle (SDLC), when data is sparse [11].

On the other hand, waiting to collect a substantial amount of failure data before being able to fit SRGMs might be uneconomic in many cases. Parameter estimates would also differ greatly from long-run values in such a case. Reliability growth could be examined [11], [33]; however, this discussion does not touch the fundamental characteristic of the data itself. Although most software reliability models are based on some assumptions, the focus is usually on the failure rate. Only a few works are concerned with whether the data follows those assumptions. Lewis and Shedler [34] discussed the assumption test for NHPP-based models. They proposed methods for the statistical analysis of nonstationary univariate stochastic point processes and sequences of positive random variables, which are frequently encountered in computer systems.

However, most of the research has usually assumed the observed/collected data, followed by an NHPP with an increasing, decreasing, or remaining constant failure intensity, so that they would be able to select appropriate SRGMs to monitor the failure trends. Cai et al. [35], [36] presented two controlled software experiments to examine whether software reliability behavior follows an NHPP in general, which was distinctly different from the existing popular belief of software reliability modeling. Based on Cai et al.’s experiments, they made some conclusions and further invalidated the NHPP assumptions accordingly. Notably, their experimental results were demonstrated in terms of three statistical tests (i.e., the Cramér-von Mises test, the CPIT test, and the Chi-square test).

Moreover, the CPIT test has other deficiencies. For example, Miller [37] and Gaudoin [38] reported that the GO model and the Jelinski-Moranda (JM) model were indistinguishable based on the CPIT test, but the prediction results made by the two models on the same dataset may be significantly different. Gaudoin also argued that the CPIT tests were probably not very powerful. In Cai et al.’s experiments, the mean and variance were calculated based on the cumulative number of observed faults. Lin et al. [39] also justified the application of the NHPP models in software development. One of the most important applications regarding NHPP models is to determine the optimal software release time in late software testing; therefore, they believed that the NHPP models are still valuable and applicable.

From the above-mentioned discussions, it can be said that most research has assumed that failure data follow an NHPP, but we definitely have to correctly judge whether the failure dataset is said to be absolutely NHPP-type or not. Thus, we can later pick the most appropriate SRGM(s) to depict the software failure process and create plans for software testing and debugging. So far, there is no single SRGM that is universal to all situations [3], [11]. Consequently, software project managers and developers need a quick and cost-effective method to make sure the data follow an NHPP before adopting the failure data. In actuality, as project managers and developers learn to make software fault prediction and failure data analysis, they will also learn to make better software development plans and to better control the number of faults they detect and remove.

Finally, data analysis and curve fitting are not easy in statistics. They require several steps, from dealing with the data to choosing a model [40], [41]. If the dataset has extraordinary points or missing values, it should be handled. Parameters such as mean, variance, correlation, and distribution of the data should be discussed. After such analysis, model selection and fitting are performed. If the goodness of fit for the data is poor, other models will be considered. However, in software reliability modeling, we would assume that the dataset is a certain type, and we can immediately conduct curve fitting. The contrast is remarkable. As data science gains increasing attention, the importance of the data itself should not be ignored in software reliability modeling.

There are many statistical tests that can be used to establish a proper testing procedure. Rather than propose a new type of test in the following sections, we propose a combined method (i.e., a two-phase test and Laplace trend test) to verify
the hypothesis that failure data follow an NHPP with the decreasing intensity function.

**Example 1**: Here, let us illustrate our ideas with an example. The selected data are listed in Table 2, and they were released by Tohma et al. [42]. The test/debugging data consisted of both a modular test and a combinational test. The software system was a control application system, and the test or debugging data were recorded on a daily basis. Over the course of 46 weeks, 266 software faults were removed [11]. Here, we need to count the frequencies of values from the raw data to restructure them into frequency count data. The results are given in Table 3, and we need the information to calculate the probability of occurrences. Note that detecting zero faults per week occurs 15 times, detecting one fault per week occurs one time, and so on. The probability mass function of the frequency count form and the cumulative distribution function of the frequency count form are plotted in Figures 1 and 2, respectively. Referring to Figures 1 and 2, we can intuitively find that the observed probability of the frequency count data differs markedly from the expected probability. It is worth noting that one important property of the Poisson distribution is that the variance of the distribution is equal to the mean. In this example, the mean is 3.07 of the frequency count data, while the variance is 13.92. These data may also be far from following the Poisson distribution from this aspect. Let us further see the results if we select some SRGMS for software reliability modeling and prediction based on Tohma’s dataset. Three famous NHPP-based models, the Goel-Okumoto model (GO) (see Eq. (1)), the inflection S-shaped model (ISS) (see Eq. (3)), and the generalized Goel-Okumoto model (GGO) (see Eq. (4)), are selected, and the MLE method [11] is used to estimate the parameters of selected models (estimates for GO: \(a = 349.81, b = 0.0311;\) ISS: \(a = 302.87, b = 0.0596, c = 1.0109;\) GGO: \(a = 305.93, b = 0.0223, m = 1.1785). Figure 3 presents the curve-fitting results. It can be seen from Figure 3 that the fitting results may not be good for all selected models. Thus, it is acceptable or reasonable for software engineers (and/or developers) to check whether a software failure dataset is Poisson distributed before we perform software reliability modeling or plan to model the failure behavior of developed software by using NHPP-type models. From this example, we can clearly see that blindly applying SRGM(s) might not lead to meaningful results when the characteristics of the data do not fit the selected SRGM(s).

### III. TWO-PHASE TEST FOR NHPP

#### A. INVESTIGATING THE ASSUMPTION OF NHPP

As addressed in Section I, there are four possible combinations of data types and model types categorized in terms of NHPP and non-NHPP types, and we will mainly focus on discussing the collected failure dataset, whether it is NHPP type, before we estimate the reliability by using SRGM(s). Note that previous works usually simply accept the categorization of the failure dataset as NHPP type, whether or not it is. This may be due to the complexity of conducting the assumption test [11]. The traditional way to determine a model is by examining the graphical similarity and goodness of fit of the...
data and SRGMs. However, the characteristics of the failure data are not confirmed, which may cause low credibility in terms of the fitting results.

In statistics, a full process to determine a model contains many steps [40], [41], which include observing the collected data, adjusting the data (dealing with missing values and outliers), testing the underlying distribution, fitting several models, and finally choosing a proper model. The process may have too many steps in application domains, such as computer science, yet without such a detailed procedure, it may reduce the accuracy of the results. In this study, we propose a method for balancing both efficiency and effectiveness to test the data characteristics and is not too complex to perform. Before introducing the proposed method to test the characteristics of failure data, we briefly review and investigate the assumption of NHPP [43].

A counting process \( \{N(t) \}, t \geq 0 \) is said to be an NHPP with an intensity function \( \lambda(t), t \geq 0 \) if [1], [43], [44]:

i. \( N(0) = 0 \).
ii. \( \{N(t), t \geq 0\} \) has independent increments.
iii. \( P[N(t+\Delta t)-N(t) \geq 2] = o(\Delta t) \).
iv. \( P[N(t+k)-N(t) = 1] = \lambda(t)k + o(k) \).

Moreover, the MVF of \( N(t) \) is

\[
m(t) = \int_{0}^{t} \lambda(y)dy,
\]

and \( N(s+t) - N(s) \) has a Poisson distribution with the mean

\[
m(t) = \int_{0}^{t} \lambda(y)dy,
\]

It can be seen that Assumptions (i), (iii), and (iv) might not be difficult to hold. However, there is doubt about whether the independent increments and Poisson distribution hold well. To test these two assumptions, several statistical concepts are needed.

The properties of the Poisson process include independent increments; that is, the numbers of events occurring in non-overlapping intervals are independent. When predicting the occurrence of failures, we should focus on the increments in each individual time interval rather than on the cumulative events during the whole period, starting from the beginning of testing. On the whole, there is no doubt that Poisson processes are acknowledged to provide a good approximation of the occurrence of many real-world events, such as arrivals at a counter, telephone calls entering a switchboard, and so on [44]. Therefore, considering the use of the NHPP models, it is correct to focus on the increments in each individual time interval rather than on the cumulative events.

Hypothesis testing [45] can be set up to reflect a decision problem and is frequently used in many application domains. It typically consists of two types of hypotheses in the testing: One is called the null hypothesis (H0), and the other is the alternative hypothesis (H1). The goal of this process is to determine whether there is enough evidence to infer which of the hypotheses is true. Two possible decisions for hypothesis testing are to conclude that there is enough evidence to reject the null hypothesis or that there is not enough evidence to reject the null hypothesis.

The test statistic is the criterion based on decisions regarding the hypotheses. There are two methods to make the decision: the p-value method [45] and the rejection region method [1], [45]. The rejection region method requires the decision maker to select a significance level for a range of values, such that if the test statistic falls into that range, we decide to reject the null hypothesis. The p-value of a test is the probability of observing a test statistic at least as extreme as the one computed, given that the null hypothesis is true. The p-value is compared with the selected value of the significance level. If the p-value is less than the significance level, we judge the p-value to be small enough to reject the null hypothesis; otherwise, we do not reject the null hypothesis. For the rejection region and the p-value method, a significant level of 0.05 is typically suggested as a convenient cutoff level to reject the null hypothesis [46].

As mentioned earlier, two more types of tests are needed to test the Poisson distribution and independent increments. The first type is goodness of fit [47], which can be used to test how well an observation set fits the theoretical distribution, such as the Poisson distribution. Typically, the goodness-of-fit test summarizes the discrepancy between observed values and expected values under a certain probability distribution, that is, to measure the distance between the data and a certain probability distribution, and the computed distance is compared to a critical value. The fit is considered good if the distance (or the test statistic) is less than the critical value. There are numerous statistical tests that can measure whether a given distribution is suited to a dataset, and three famous tests are used in this paper: the Chi-Squared test [48], the Kolmogorov-Smirov test [48], and the Cramér-von Mises test [47], [49], since they are widely used in the field of reliability modeling [35], [50], [51].

The second type of test is the serial independence test, which can be used for testing the independent increments. The collected failure data during software testing is a time series, which is an ordered sequence of values of a variable at equally spaced time intervals [50] that time series analysis methods for testing serial independence can be performed. Testing independence is important for time series modeling in many statistical applications, such as in economic forecasting, census analysis, and stock market analysis [50], [52]. It verifies whether a sequence of consecutive observations is independent over time. In general, tests such as the autocorrelation function (ACF) [52], the Ljung–Box test [53], and the BDS test [54] can be the measures. It is common to use these three tests in time series analysis [55]–[57].

In this paper, the null hypothesis for testing Poisson distribution is “The selected failure dataset is Poisson distributed,” and for independence increments, it is “The sequence of the selected failure dataset is serial independent.” The alternative hypothesis is the opposite. Essentially, in this paper, we conducted our tests using R language [58] with certain R packages.
B. PROPOSED TWO-PHASE TEST PROCEDURE

In this section, we present the proposed test procedure to verify whether the failure data satisfies the characteristics and assumptions of the NHPP. While previous research typically divides datasets into tiny units and performs a uniform distribution test, which may take much effort to collect data, we plan to examine the issue in a top-down manner. That is, we simply examine the entire failure data in the proposed method. Notice that the tests we conduct are goodness of fit to Poisson distribution and serial independence. We combined these two tests and named them the NHPP characteristic test.

Our proposed method for testing the NHPP characteristics of the failure data is divided into two phases. The first phase includes two steps: One step involves testing the goodness of fit to Poisson distribution, and the other includes testing the existence of serial dependency. Note that the above two steps will be examined for the entire dataset. It should be particularly noted that we have to perform the second phase of the test if the selected dataset failed either the Poisson distribution test, serial independence test, or both. Otherwise, we can conclude that the selected failure data satisfies the characteristics and assumptions of NHPP. To eliminate the disadvantages of biased statistical tests, we used three different tests for each characteristic, and the majority result was accepted; that is, a dataset passed the test if the null hypothesis was not rejected by two or more tests.

The goodness-of-fit test examines how likely two variables can be used to test the similarity between an observation dataset and a specific theoretical probability distribution. Probability distribution describes the probability of obtaining values of a variable and provides a useful benchmark for evaluating data [41]. Thus, it can be used to test whether observed frequencies are consistent with expected frequencies based on the probability mass function. The probability mass function of the Poisson distribution is given by the following equation:

\[
P(X = k) = \frac{(\lambda^k e^{-\lambda})}{k!},
\]

where \(\lambda\) is the mean of the random variable. The transformed frequencies of the selected failure datasets will be compared to the expected frequencies. In the two-phase test, three goodness of fit tests are selected: the Chi-Squared test, the Kolmogorov-Smirnov test, and the Cramé-
von Mises test, due to their representativeness and wide usage [35], [50], [51]. Detailed information for the three goodness-of-fit tests is given in Appendix A.

As mentioned above, testing for independence is crucial in statistical applications [52], [54], [59]. For a serially independent time series, there are no relationships between current and past data points, and diagnostic testing verifies whether consecutive observations are independent. In this paper, we also chose three tests for serial independence: the ACF, the Ljung–Box test, and the BDS test, which are widely used in time series analysis. The details of the independent testing are provided in Appendix B.

Note that the second phase of the test is for datasets in which at least one characteristic test (i.e., Poisson distribution or independence property) has been rejected. Similarly, two steps are contained in the second phase. The first step includes observing and adjusting the selected data by partitioning it into several regions and/or removing some outliers. The second step is to further retest the adjusted data. To partition the failure data (i.e., generate a partition field that splits the data into separate subsets), we need to detect or determine the change points [60], [61]. R language community contributors have built some packages (see cpm [62] and change-point [63]) that can easily detect change points. To remove outliers, a boxplot can be used, as it can identify the middle 50% of the data, the median, and the extreme points [50], [64]. Additionally, a dataset may contain both change points and outliers. The second step in this phase is to perform the examination again for the NHPP characteristics of the selected dataset as the first phase, which tests for both Poisson distribution and serial independence.

Moreover, the Laplace trend test, which is generally performed in applying SRGM(s) [1], [11], will also be considered within the two-phase test. We combine the test results of both the two-phase test and the Laplace trend test to infer whether the failure data satisfies the characteristics and assumptions of NHPP and to determine their appropriate-ness for NHPP-based SRGM(s). If the Laplace factor values indicate decreasing reliability, it may not be an appropriate dataset for further software reliability modeling, even if the two-phase test concludes that the dataset is the NHPP type. If the failure rate decreases with time, it means that the developed software becomes stable and that its reliability grows. By contrast, increment in th failure rate may imply the decay of software reliability. Let time interval \((0, t)\) be divided into \(j\) equal units of time, and the Laplace factor (LF) is defined by [11]

\[
LF(j) = \frac{\sum_{i=1}^{j} (i - 1) x_i - \frac{\sum_{i=1}^{j} x_i}{2}}{\sqrt{\frac{1}{12} \sum_{i=1}^{j} x_i}},
\]

where \(x_i\) is the number of faults observed during time unit \(i\). Positive values of LF(j) indicate decreasing software reliability, whereas negative values indicate increasing software reliability. If the LF values vary between \(-2\) and \(+2\), it represents stable reliability. Note that the trend analyses allow periods of reliability growth and reliability decreases to be identified to apply SRGMs to data exhibiting trends in accordance with modeling assumptions.

To clarify the entire procedure, a flow chart is provided in Figure 4. We spent time discussing the characteristics of the failure datasets. Altogether, we first reviewed the NHPP assumption, and two of the assumptions may not intuitively hold in the failure dataset, that is, the Poisson distribution and independent increments. Thus, a two-phase test procedure is proposed to examine whether the failure data are NHPP type.

As mentioned in previous sections, if we accept the false assumption that failure data is the NHPP type and apply NHPP-based SRGM(s) when it is not, it may lead to doubts regarding the credibility of the software reliability modeling.
Two types of statistical tests, that is, the goodness-of-fit test, we will test the entire data of each set as an interval. As we mentioned in the preceding section, in the first phase that these 25 datasets are interval domain data. Data: interval domain data and time domain data [8]. Note characteristics of 25 datasets are listed in Table 4. In general, from famous open source software projects: the Eclipse JDT, and the other three datasets are 25 failure datasets are selected. Twenty-two of them are then present the data analysis results of the two-phase test. The Laplace trend test in Section IV.

The Laplace test is superior from an optimality point of view and is recommended for use when the NHPP assumption is made [65]. Next, we analyze the characteristics of 25 real failure datasets by performing a two-phase test with assumption is made [65]. Next, we analyze the characteristics of 25 real failure datasets by performing a two-phase test with the NHPP or not. We test the entire dataset in the first phase, and in the second phase, datasets that fail the first phase will be adjusted using region partitioning and/or outlier removal and tested again.

In this paper, three goodness-of-fit tests (i.e., the Chi-Squared test, the KS test, and the Cramér-von Mises test), which were performed in the proposed method for the Poisson distribution property of a dataset, are used. Later, three serial independence tests—the ACF, Ljung-Box, and the BDS tests—used for testing independent increments were also executed. Note that for each characteristic test, we accept the majority conclusion from the three respective tests. Additionally, we combine the Laplace trend test within the two-phase test to determine whether a failure dataset is properly applied to SRGM(s). A dataset is said to “fit the characteristics of SRGM(s)” only if it passes the two-phase test, and the Laplace trend test results in stable or increasing reliability. Actually, the Laplace test is superior from an optimality point of view and is recommended for use when the NHPP assumption is made [65]. Next, we analyze the characteristics of 25 real failure datasets by performing a two-phase test with the Laplace trend test in Section IV.

IV. DATA ANALYSIS AND DISCUSSION
In this section, we first describe the selected failure data and then present the data analysis results of the two-phase test.

A. FAILURE DATA SETS
In this paper, all selected datasets are failure-count data and 25 failure datasets are selected. Twenty-two of them are widely used datasets [11], and the other three datasets are from famous open source software projects: the Eclipse JDT core, Mozilla Firefox, and Oracle VirtualBox [66]–[68]. The characteristics of 25 datasets are listed in Table 4. In general, there are two common methods to collect software failure data: interval domain data and time domain data [8]. Note that these 25 datasets are interval domain data.

B. TEST RESULTS IN THE FIRST PHASE
As we mentioned in the preceding section, in the first phase test, we will test the entire data of each set as an interval. Two types of statistical tests, that is, the goodness-of-fit test for the Poisson distribution and the serial independence test for independent increments, will be conducted in this phase. Three tests are performed for each type of test, and the majority conclusion is taken. Furthermore, if the null hypothesis is rejected by two tests and is not rejected by one test, we conclude that the null hypothesis is rejected. Note that the null hypothesis for these two tests are “the dataset is Poisson distributed” and that “the dataset is serial independent,” respectively. Moreover, the significance level in this paper is set at 0.05. In cases where a majority test result for a dataset does not rejects the null hypothesis, this dataset is seen as “passing” the test; otherwise, it “fails” the test. Furthermore, a dataset is said to pass the first phase test if it passes both tests for Poisson distribution and serial independence. Moreover, issues regarding the combined results of the two tests (especially for those failed datasets) are discussed in Section IV.C.

1) POISSON DISTRIBUTION TEST
First, Table 5 presents the test results of the 25 selected datasets using the Chi-Squared test, the Kolmogorov-Smirnov test, and the Cramér-von Mises test for Poisson distribution in the first phase. Note that the test statistics of each dataset for the Chi-Squared test and the Kolmogorov-Smirnov test are listed in the first row, while the second test examines whether any data point pairs in the second row, with a significance level of 0.05. To illustrate the proposed method, let us take DS1. It can be seen that the Chi-Squared statistic is 26.650, which is greater than the critical value 14.067, and thus we reject the null hypothesis that DS1 is Poisson distributed. Additionally, the Kolmogorov-Smirnov statistic is 0.127, which is smaller than the critical value of 0.454; thus, we do not reject the null hypothesis. For the Cramér-von Mises statistic, the p-value of DS1 is 0.157, which is greater than the significance level 0.05; hence, the null hypothesis is not rejected. The null hypothesis was not rejected according to the two tests (i.e., the Kolmogorov-Smirnov test and the Cramér-von Mises test). As a result, we conclude that DS1 passed the test for Poisson distribution in this phase. Similarly, the remaining 24 datasets can be inferred. It can be seen that 88% (22/25) of the datasets passed the test in this phase, and only one dataset was rejected by all three tests. It is obvious that most of the datasets are Poisson distributed. However, whether the datasets have NHPP characteristics is further discussed in the following sections.

2) SERIAL INDEPENDENCE TEST
The second test examines whether any data point pairs in an interval of the data series are independent. Note that in the first phase, the entire dataset is seen as one interval, and the null hypothesis is that the dataset is serially independent. First, because the results of the ACF plot and the BDS test cannot be shown as one numeric value, we use some examples to explain how we interpret the results of the ACF plot and the BDS test to determine whether a failure dataset is properly applied to SRGM(s). A dataset is said to “fit the characteristics of SRGM(s)” only if it passes the two-phase test, and the Laplace trend test results in stable or increasing reliability.

FIGURE 4. The flowchart of the proposed two-phase test.

and prediction results. We therefore propose a two-phase test to check whether the selected failure data have the characteristic(s) of NHPP or not. We test the entire dataset in the first phase, and in the second phase, datasets that fail the first phase will be adjusted using region partitioning and/or outlier removal and tested again.
TABLE 4. The characteristics of selected datasets.

| DS    | Duration | Time Unit | Failure-Counts | Remarks                                                                 |
|-------|----------|-----------|----------------|-------------------------------------------------------------------------|
| DS[11],[69] | 14       | CPU minute | 27             | The software system is a compiler project in a university in Japan consisting of 1,000 lines of code. |
| DS2[11],[42] | 109      | day       | 535            | The software system is for a real-time control application consisting of 870,000 lines of code. |
| DS3[11],[70] | 111      | day       | 481            | The software system is a monitoring and real-time control system consisting of 200,000 lines of code. |
| DS4[11],[70] | 199      | day       | 55             | The software system is a railway interlocking system consisting of 14,500 assembly instructions. |
| DS5[11],[70] | 16       | day       | 198            | The software system is a real-time control system consisting of about 90 modules. The average size of the modules are about 1,000 lines of PL/I, Fortran, and Assembly languages. |
| DS6[11],[71] | 83       | hour      | 118            | The software system is a flight dynamic application consisting of 10,000 lines of code. |
| DS7[11],[71] | 163      | hour      | 180            | The software system is a flight dynamic application consisting of 22,500 lines of code. |
| DS8[11],[71] | 217      | hour      | 213            | The software system is a dynamic application consisting of 38,500 lines of code. |
| DS9[11],[42] | 46       | day       | 266            | The software system is a control application system. The test/debugging data consists of both modular test and combinatorial test. |
| DS10[11] | 63       | week      | 133            | DS10-14 are projects under US NASA Jet Propulsion Laboratory (JPL). DS10 is 14,000 lines of uncommented assembly language consisting of three subsystems. |
| DS11[11] | 181      | week      | 224            | DS11 is also a JPL project that was developed as a planetary orbiter carrying an atmospheric probe consisting of 7,000 uncommented source lines of HAL/S and 15,000 source lines of assembly language. |
| DS12[11] | 41       | week      | 351            | DS12 is another JPL project. |
| DS13[11] | 114      | week      | 188            | DS13 is roughly the same size as DS11. |
| DS14[11] | 73       | week      | 367            | DS14 is a facility for tracking and acquiring data from each resource satellite in high-inclination orbits consisting 103,000 uncommented lines of code. It is written in a mixture of C, Fortran, EQUEL, and OSL. |
| DS15[11],[72] | 22     | 5,000 seconds | 54             | The data is those collected on system 2 presented in [11]. |
| DS16[11],[73] | 81       | 10 days   | 461            | The software system is the Brazilian Electronic Switching System, TROPICO R-1500 for 1500 subscribers. The software size is about 300 kC written in Assembly Language. |
| DS17[2],[11] | 850     | day       | 34             | The software system is for the real-time multi-computer complex of the US Naval Fleet Computer Programming Center of the US Naval Tactical Data Systems. It consists of 38 different project modules. |
| DS18[11],[74] | 38       | day       | 231            | The software system provides the flight controller at the Johnson Space Center with processing support to exercise command and control over flight operations. |
| DS19[11],[75] | 21       | day       | 46             | The software system is an online data control software package that has been available since 1980 in Japan consisting of 40,000 lines of code. |
| DS20[11],[75] | 19       | week      | 328            | DS20 is from a PL/I database application program consisting of 1,317,000 lines of code. |
| DS21[16] | 17       | week      | 144            | DS21 was collected during the development of a middle-sized software project. |
| DS22[33] | 21       | week      | 136            | DS22 was collected from the Rome Air Development Center Project consisting of 21,700 object instructions, and it took 21 weeks and 9 programmers to test the system. |
| DS23[66] | 419      | day       | 218            | The failure data was collected from the open source software project Eclipse JDT core version 4.6 from 2015/7/22 to 2016/9/12. |
| DS24[68] | 160      | day       | 324            | The failure data was collected from the open source software project Mozilla Firefox version 49 from 2016/4/26 to 2016/10/02. |
| DS25[67] | 27       | day       | 99             | The failure data was collected from the open source software project Oracle VirtualBox version 5.1.4 from 2016/8/17 to 2016/9/12. |

BDS test as well as the Ljung-Box test. The ACF plots of all selected datasets are shown in Figure 5.

Basically, if few or no lags in the plot are over the bounds (i.e., the lags converge to zero), the null hypothesis is not rejected. That is, in this study, it passed the test for serial independence. Figure 6 shows a graphical representation of the ACF of DS1. It can be seen from Figure 6 that there are a few lags over the bounds. Thus, DS1 passed this test. The second test for serial independence is the Ljung-Box test, and the p-value of DS1 is 0.477, which is greater than 0.05; thus, the null hypothesis is not rejected. In this study, the null hypothesis is not rejected if at least five p-values are greater than 0.05, with no p-values for each dimension of a close point being equal to 0, which indicates dependency. Table 6 shows the BDS test results for DS1. Based on these conditions, the results show that for DS1, the null hypothesis is rejected for the BDS test. Table 7 also summarizes the results of the serial independence test for DS1. As can be seen, DS1 is not rejected by two tests for serial independence, and we would say that it passed the independence test.

Similarly, let us select DS3 and DS19 to explain how the proposed methods can be used for failure data analysis. It is worth noting that among the 25 selected failure datasets, DS3 is the only dataset rejected by all three tests, and DS19 is the only dataset not rejected by all three tests. Figure 7 demonstrates the ACF plot for DS3. Figure 7 shows that over 25% lags exceed the bounds. In this case, the null hypothesis is rejected. The p-value of the Ljung–Box test is smaller than 0.001, which is much smaller than 0.05; hence, we also reject the hypothesis. The BDS test for DS3 is provided in Table 8. It can be seen that there are many zero p-values, and this implies that DS3 is hardly serially independent. We reject
the null hypothesis in three tests, and as a result, we could say that DS3 failed the serial independence test in this phase. A summary of the serial independence test for DS3 is shown in Table 9.

On the other hand, Figure 8 and Table 10 show the results of the ACF plot and BDS test for DS19, respectively. Figure 8 is a typical pattern for serial independence: few lags over the bounds and lags not over the bounds converging to zero. The null hypothesis for ACF is thus not rejected for DS19, nor for the BDS test, since there are no p-values in both dimensions of a close point, and at least five p-values are greater than 0.05 in Table 10. In this case, the null hypothesis is not rejected. In addition, the p-value of the Ljung–Box test is 0.673, which is much greater than 0.05. The summary of the serial independence test for DS19 is presented in Table 11. It is obvious that DS19 passes the test.

Finally, Table 12 shows the test results for independence in the first phase for 25 datasets. From Table 12, we can see that 36% (9/25) of the datasets passed the test, which means 36% of them can be seen as being serially independent in the first phase. For those that did not pass the test, 40% (10/25) of the datasets were rejected by all three tests. With the combined result of Section IV.B Parts 1 and 2, we can conclude whether or not a dataset satisfies the characteristics of NHPP in this preliminary phase, which is discussed in the next section.

3) SUMMARY OF MAIN RESULTS IN THE FIRST PHASE

Test results of Section IV.B Parts 1 and 2 were combined, and overall results are summarized in Table 13. Datasets were marked as “passed” only if they passed both tests. In the first testing phase, in which all the data were treated as one interval, 4% (1/25) of the datasets failed both of the two tests, 8% (2/25) of the datasets failed the test for Poisson distribution but passed the test for serial independence, 60% (15/25) of the datasets failed the test for serial independence but passed the test for Poisson distribution, and 28% (7/25) of the datasets passed the test. For the 28% of the datasets that passed the test in the first phase, it can be said that these datasets satisfied the characteristics of NHPP; however, because we treat all the datasets as an interval, we cannot simply conclude that the remaining datasets do not follow NHPP properties. As mentioned in Section III.B, those that failed either test for Poisson distribution, serial independence, or both, needed further testing in the second phase.

Finally, the Laplace trend test can also be used to examine data stability after the NHPP characteristic test. The combination of the NHPP characteristic test and the trend test clarifies whether a dataset qualifies for application of NHPP-based SRGMs. The plots of the Laplace trend test for all of the datasets are provided in Figure 9. We also calculated the Laplace trend factor for 25 datasets at the end of testing, and the results are provided in Table 14. To summarize, DS1, DS5, DS15, DS18, DS19, and DS25 are appropriate datasets for NHPP-based SRGMs as they passed the NHPP characteristic test, while being stable or showing increasing reliability. DS22 has decreasing reliability although it passed the NHPP characteristic test. As a result, this system (DS22) should be continuously tested, while the other six datasets could immediately have NHPP-based SRGMs applied to them.
Further analysis should be conducted for datasets that have not passed the tests for Poisson distribution, serial independence, or both. There are two primary concerns in further analysis: One is the existence of more than one region (or the existence of change points) in the data sequence, and the other is the existence of outliers. If the above two cases happen or exist in the dataset, we might need to take some action to adjust...
First, we use the change-point detection method proposed by Ross [62] for region partitioning. This method has been implemented as an R package, so it could be easily utilized to analyze the data. Here, we briefly introduce this method. We first describe the detection of one change point. Suppose there is a fixed-length sequence containing the $n$ observations $x_1, \ldots, x_n$ which may or may not contain a change point. For ease of exposition, we assume that at most one change point can be present. If no change point exists, the observations are i.i.d. according to the distribution $F_0$. If a change point exists at some time $\tau$, then the observations have a distribution of $F_0$ prior to this point, and a distribution $F_1$ afterwards, where $F_0 \neq F_1$.

$$H_0: \quad X_0 \sim F_0(x; \theta_0), \quad i = 1, \ldots, n,$$

$$H_1: \quad X_i \sim \begin{cases} F_0(x; \theta_0) & i = 1, 2, \ldots, k, \\ F_1(x; \theta_1) & i = k + 1, k + 2, \ldots, n, \end{cases}$$

(11)

Consider the problem of testing whether a change occurs immediately after a specific observation, $k$. This leads to choosing between the following two hypotheses:

$$D_n = \max_{k=2,\ldots,n-1} D_{k,n} = \max_{k=2,\ldots,n-1} \left| \frac{D_{k,n} - \mu_{D_{k,n}}}{\sigma_{D_{k,n}}} \right|.$$  

(12)

where $D_n$ is the test statistic and $D_{k,n}$ is the two-sample test statistic, and $\theta_1$ represents the potentially unknown parameters of each distribution.

A two-sample hypothesis test can be used to solve this problem, such as the Cramér-von-Mises test, and the value of $D_{k,n}$ can thus be computed. If $D_{k,n}$ exceeds some appropriately chosen threshold $h_{k,n}$, then the null hypothesis that the two samples have identical distributions is rejected, and

| DS   | ACF  | Ljung-Box | BDS     | Result   |
|------|------|-----------|---------|----------|
| DS1  | Not Rejected | 0.477     | Rejected | Passed   |
| DS2  | Rejected      | <0.001    | Rejected | Failed   |
| DS3  | Rejected      | <0.001    | Rejected | Failed   |
| DS4  | Not Rejected  | 0.006     | Rejected | Failed   |
| DS5  | Not Rejected  | 0.077     | Rejected | Passed   |
| DS6  | Not Rejected  | 0.016     | Rejected | Failed   |
| DS7  | Not Rejected  | <0.001    | Rejected | Failed   |
| DS8  | Not Rejected  | <0.001    | Rejected | Failed   |
| DS9  | Not Rejected  | 0.978     | Not Rejected | Passed |
| DS10 | Rejected      | 0.014     | Not Rejected | Failed |
| DS11 | Rejected      | <0.001    | Rejected | Failed   |
| DS12 | Rejected      | <0.001    | Rejected | Failed   |
| DS13 | Rejected      | <0.001    | Rejected | Failed   |
| DS14 | Not Rejected  | 0.001     | Rejected | Failed   |
| DS15 | Not Rejected  | 0.533     | Rejected | Passed   |
| DS16 | Rejected      | <0.001    | Rejected | Failed   |
| DS17 | Rejected      | <0.001    | Rejected | Failed   |
| DS18 | Not Rejected  | 0.216     | Rejected | Passed   |
| DS19 | Not Rejected  | 0.674     | Not Rejected | Passed |
| DS20 | Not Rejected  | 0.823     | Rejected | Passed   |
| DS21 | Rejected      | <0.001    | Rejected | Failed   |
| DS22 | Not Rejected  | 0.063     | Rejected | Passed   |
| DS23 | Rejected      | <0.001    | Rejected | Failed   |
| DS24 | Rejected      | <0.001    | Rejected | Failed   |
| DS25 | Not Rejected  | 0.310     | Not Rejected | Passed |
TABLE 13. Test results of the first phase.

| DS   | Dist. | Indep. | Result |
|------|-------|--------|--------|
| DS1  | Passed| Passed | Passed |
| DS2  | Passed| Failed | Failed |
| DS3  | Failed| Failed | Failed |
| DS4  | Passed| Failed | Failed |
| DS5  | Passed| Passed | Passed |
| DS6  | Passed| Failed | Failed |
| DS7  | Failed| Failed | Failed |
| DS8  | Failed| Failed | Failed |
| DS9  | Failed| Passed | Failed |
| DS10 | Failed| Failed | Failed |
| DS11 | Passed| Failed | Failed |
| DS12 | Failed| Failed | Failed |
| DS13 | Passed| Failed | Failed |
| DS14 | Passed| Failed | Failed |
| DS15 | Passed| Passed | Passed |
| DS16 | Passed| Failed | Failed |
| DS17 | Failed| Failed | Failed |
| DS18 | Passed| Passed | Passed |
| DS19 | Passed| Passed | Passed |
| DS20 | Failed| Failed | Failed |
| DS21 | Failed| Failed | Failed |
| DS22 | Passed| Passed | Passed |
| DS23 | Failed| Failed | Failed |
| DS24 | Failed| Failed | Failed |
| DS25 | Passed| Passed | Passed |

TABLE 14. Result of the first phase with laplace trend test of passed datasets.

| No. | Dist. | Indep. | Result | LF |
|-----|-------|--------|--------|----|
| DS1 | Passed| Passed | Passed | -0.979 |
| DS2 | Passed| Failed | Failed | -13.664 |
| DS3 | Failed| Failed | Failed | -18.321 |
| DS4 | Failed| Failed | Failed | 0.225 |
| DS5 | Passed| Passed | Passed | 0.879 |
| DS6 | Failed| Failed | Failed | -4.031 |
| DS7 | Failed| Failed | Failed | -2.759 |
| DS8 | Failed| Failed | Failed | -3.851 |
| DS9 | Failed| Passed | Failed | -6.507 |
| DS10| Passed| Failed | Failed | -1.286 |
| DS11| Failed| Failed | Failed | -4.459 |
| DS12| Failed| Failed | Failed | -9.667 |
| DS13| Passed| Failed | Failed | 4.687 |
| DS14| Passed| Failed | Failed | -1.486 |
| DS15| Passed| Passed | Passed | -5.448 |
| DS16| Passed| Failed | Failed | -12.173 |
| DS17| Passed| Failed | Failed | -5.185 |
| DS18| Passed| Passed | Passed | -1.263 |
| DS19| Passed| Passed | Passed | 0.260 |
| DS20| Failed| Failed | Failed | -5.232 |
| DS21| Failed| Failed | Failed | -6.515 |
| DS22| Passed| Passed | Passed | 5.070 |
| DS23| Passed| Failed | Failed | 6.179 |
| DS24| Passed| Failed | Failed | -4.362 |
| DS25| Passed| Passed | Passed | -2.619 |

it is concluded that a change point has occurred immediately following observation $x_k$. Note that $D_{k,n}$ is evaluated at every value where $1 < k < n$, and the maximum value is used since it is not known in advance where the change point occurs. The test statistic is as follows:

$$\hat{\tau} = \arg \max_k D_{k,n}.$$  (13)

For multiple change points, this method can be extended as follows. Let $x_s$ denote the $s$-th observation that has been received, where $s \in \{1, 2, \ldots\}$. Whenever a new observation $x_s$ is received, this approach treats $x_1, \ldots, x_s$ as a fixed-length sequence and computes $D_s$ using the above batch methodology, where $D_s$ rather than $D_n$ is used to highlight the sequential nature of the procedure. A change is then flagged if $D_s > h_s$ for some appropriately chosen threshold. If no change is detected, the next observation $x_{s+1}$ is received, and the procedure therefore consists of a repeated sequence of hypothesis tests. In general, the $D_{k,n}$ statistics will have properties which allow $D_{s+1}$ to be computed from $D_s$ without incurring too much computational overhead. A more detailed description of the change-point detection method can be found in [61], [62].

Here, let us use DS3 as an illustration, since DS3 is the only dataset that failed both Poisson distribution and independence tests. To remove the outliers and observe the dataset trend easily, we plot the Boxplot of DS3, which is shown in Figure 12. A boxplot is a graphical representation of numerical data depicted through their quartiles, which identifies minimum, first quartile Q1, median, third quartile Q3, and maximum values. If the data point exceeds the maximum value, we will regard it as an outlier and remove it. Moreover, using the above-mentioned partition method, DS3 can be partitioned into four regions: [1], [31], [32], [49], [50], [75], [76], and [111]. We merge small regions that are less than 15 data points to their neighboring regions, since too small regions may be inappropriate for further modeling. Figures 10 and 11 show the plots of the failure-count vs. time plot and cumulative failure-count plot labeled with partitioning lines for DS3, respectively. As seen in Figure 11, there are four regions of the Laplace trend change for DS3.

In the first region, which exists until day 31, there is a more obvious increase in the total number of detected failures. Additionally, Figure 13 presents the plot of the detected failures for DS3’s first region, and Figure 14 shows the plot of the cumulative detected failures for DS3’s first region. Table 15 shows the test results of the first region, and Table 16 shows the test results of the first region with one outlier removed. As seen in Table 15, the first region failed the test for Poisson distribution. From the boxplot in Figure 12, we can also observe that the first region contains three outliers. It is worth noting that after removing one outlier (day 14), this region passed the characteristic test (see Table 16).

The second region, between days 32 and 49, still showed an obvious increase in the total number of detected failures.
The third region, between days 50 and 75, indicates a small increase in the total number of detected failures. However, it must be noted that the slowdown in the total number of detected failures in the third region (see Figure 10) can be explained by the level of software reliability that was almost achieved by that time. In the last region, between days 76 and 111, there was almost no big change in the total number of detected failures, which obviously indicates stable growth in reliability.

Similarly, the test results for regions 2–4 are presented in Table 17. As seen in Figure 12, there are no obvious outliers under these three regions. Thus, we do not need to deal with outliers. Moreover, after partitioning and removing one outlier for region 1, all four regions passed the NHPP.
characteristic test. Since all four regions passed the characteristic test after the second phase test, we may say that DS3 still has an NHPP characteristic. This indicates that the existence of regions and outliers has an influence on the NHPP characteristic test. Note that we did not discuss the Laplace trend test results here, but all past results should be analyzed, combined with the Laplace trend test, to clarify whether a dataset is proper for NHPP-based SRGMs.

V. CONCLUSION

Programs were tested, and fault(s) causing failure(s) were found and recorded throughout the SDLC process. From the collected failure data, software engineers and/or developers would be able to improve their programming, reduce the number of errors in their programs, and do their jobs more
effectively. Similarly, when using collected software failure data, the goals of software reliability prediction and modeling are to estimate the mean time to the next failure, predict the total number of (remaining) faults, predict the next five failures, decide when to stop testing, and so on. In practice, these tasks are the target of SRGMs. However, blindly applying SRGMs could not lead to meaningful prediction results when the failure data type does not fit the characteristics of selected SRGMs.

In this paper, we provided insight into the fundamental characteristics of the collected software failure data through a discussion of the underlying assumptions. We proposed a two-phase method and used the Laplace trend test to verify the hypothesis that failure data follow an NHPP. The first phase of our proposed method was mainly used to examine the whole dataset as an interval. The serial independence method was applied in this paper to test whether the NHPP’s independence property holds. Note that if the failure data does not pass the first phase of examination, it will be further adjusted by region partitioning and/or outlier removal, and then the failure data will be reexamined in the second phase of the proposed method. Through the combined test methods, software developers and project managers can determine whether the failure data can be properly (or definitely) used by NHPP-based SRGMs. Experiments were performed based on 25 sets of failure data collected from a wide variety of software development projects, and the results are discussed in detail.

Used correctly, our proposed methods will be a useful tool for quickly determining the characteristics of software failure data. Software engineers and managers will be able to correctly judge whether failure data are definitely NHPP type or not and will not have to accept false results saying that the collected failure dataset is NHPP type when it is not. Altogether, it is essential that software engineers and/or project managers are able to perform reliability analysis on properly collected data and use appropriate SRGMs.

APPENDIX A THREE GOODNESS OF FIT TESTS

A. THE CHI-SQUARED TEST

The Chi-Squared test [48] is a statistical test that can evaluate how likely observation and expected frequencies are. This test statistic is constructed from a sum of squared errors. The test statistic of the Chi-Squared test is defined by

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i},$$

(A.1)

where $O_i$ is the observed frequency of type $i$, $E_i$ is the expected frequency of type $i$, and $n$ is the total number of observations (and is denoted as the same meaning below in this section). The Chi-Squared statistic is compared to the Chi-Squared distribution with $n-1$ degrees of freedom, which is called the CV.

B. THE KOLMOGOROV-SMIRNOV TEST

The Kolmogorov-Smirnov test (or KS test) [48] compares an observation dataset with a reference probability distribution and can also be modified as a goodness-of-fit test. The Kolmogorov-Smirnov statistic evaluates a distance between the empirical distribution function of the sample data and the cumulative distribution function of the reference distribution. The empirical distribution function $F_n(x)$ for $n$ observations $x_i$ is defined as

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} I_{[-\infty, x]}(x_i),$$

(A.2)

where $I_{[-\infty, x]}(x_i)$ is the indicator function, equal to 1 if $x_i \leq x$ and equal to 0 otherwise. The Kolmogorov-Smirnov statistic for a given cumulative distribution function $F(x)$ is

$$T_{KS} = \sup_x |F_n(x) - F(x)|,$$

(A.3)

where sup is the supremum of the set of distances and the test statistic is compared to the Kolmogorov distribution [13].

C. THE CRAMÉR-VON MISES TEST

The Cramér-von Mises test (or CvM test) [47] judges the goodness of fit of a cumulative distribution function $F(x)$ compared to a given empirical distribution function. The test statistic of the Cramér-von Mises test is defined as

$$T_{CvM} = \frac{1}{12n} + \frac{n}{2} \sum_{i=1}^{n} \left[ \frac{2i-1}{2n} - F(x_i) \right]^2,$$

(A.4)

where $x_1, x_2, \ldots, x_n$ are the observed values in increasing order. This statistic is compared to the tabulated CVs of the Cramér-von Mises test.

APPENDIX B THREE TESTS FOR SERIAL INDEPENDENCE

A. THE AUTOCORRELATION

The autocorrelation function [52] can be used to detect non-randomness in a dataset. Instead of correlation between two different variables, autocorrelation is a correlation coefficient between two values of a data sequence at times $t_i$ and $t_{i+l}$. The lag $l$ autocorrelation function for given measurements $Y_1, Y_2, \ldots, Y_n$ at time $t_1, t_2, \ldots, t_n$ is defined as

$$r_l = \frac{\sum_{i=1}^{n-l} (Y_i - \bar{Y})(Y_{i+l} - \bar{Y})}{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}.$$  

(B.1)

B. THE BOX-LJUNG TEST

The Ljung–Box test [53] examines the overall randomness based on a number of lags rather than testing randomness at each distinct lag like autocorrelation. The test statistic is defined by

$$Q = n(n+2) \sum_{i=1}^{g} \frac{r_l}{n-l},$$

(B.2)

where $n$ is the sample size, $r_l$ is the sample autocorrelation at lag $l$, and $g$ is the number of lags being tested. The test
statistic is compared to the Chi-Squared distribution with \( g \) degrees of freedom.

**C. THE BDS TEST**

The BDS test [54], [59] is proposed by and named after Brock, Dechert, and Scheinkman. It checks whether a sequence of random variables are independent and identically distributed (i.i.d.) and can be used for testing serial independence. Let \( Y_u = (y_u, y_{u+1}, \ldots, y_{u+n-1}) \). In the BDS test, the correlation integral is a measure of the proportion that any pairs of \( u \)-vectors \((Y^u, Y^u)\) are within a certain distance. The BDS test is defined as follows with dimension \( n \) and distance \( \varepsilon \):

\[
C(u, \varepsilon) = \lim_{T \to \infty} \left( \frac{T - u}{2} \right)^{-1} \sum_{t=1}^{T-u} I_s(Y^u_t, Y^u_s),
\]

where \( I_s(Y^u_t, Y^u_s) = 1 \) if the maximal norm \( \|y^u_t - Y^u_s\| < \varepsilon \) and 0 otherwise.

If \( y_u \) are indeed i.i.d., \( C(u, \varepsilon) \) should equal to \( C(1, \varepsilon)^n \). The BDS test determines whether \( C(u, \varepsilon) \) and \( C(1, \varepsilon)^n \) are sufficiently close. The test statistic is

\[
B_T(u, \varepsilon) = \frac{\sqrt{T - u + 1}C_T(u, \varepsilon) - C_T(1, \varepsilon)^{1/n}}{\hat{\sigma}(u, \varepsilon)},
\]

where

\[
C_T(u, \varepsilon) = \left( \frac{T - u}{2} \right)^{-1} \sum_{t=1}^{T-u} I_s(Y^u_t, Y^u_s),
\]

and \( \hat{\sigma}^2(u, \varepsilon) \) is a consistent estimator of the asymptotic variance of \( \frac{\sqrt{T - u + 1}}{\hat{\sigma}(u, \varepsilon)} \). The asymptotic null distribution of the BDS test is the normal distribution \( N(0, 1) \).

**REFERENCES**

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

[2] A. L. Goel and K. Okumoto, “Time-dependent error-detection rate model for software reliability and other performance measures,” *IEEE Trans. Rel.*, vol. R-28, no. 3, pp. 206–211, Aug. 1979.

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

[4] A. Gana and S. T. Huang, “Statistical modeling applied to managing global SES—2000 switch software development,” *Bell Labs Tech. J.*, vol. 2, no. 1, pp. 144–153, Aug. 2002.

[5] T. K. Nayak, S. Bose, and S. Kundu, “On inconsistancy of estimators of parameters of non-homogeneous Poisson process models for software reliability,” *Statist. Probab. Lett.*, vol. 78, no. 14, pp. 2217–2221, Oct. 2008.

[6] F. Vonta, M. S. Nikulin, N. Limnios, and C. Huber, *Statistical Models and Methods for Biomedical and Technical Systems*, Boston, MA, USA: Birkhauser, 2007.

[7] S. Bittanti, *Software Reliability Modeling and Identification* (Lecture Notes in Computer Science). Berlin, Germany: Springer-Verlag, 1987.

[8] H. Pham, *System Software Reliability Reliability Engineering Series*. London, U.K.: Springer, 2006.

[9] S. S. Gokhale and K. S. Trivedi, “A time/structure based software reliability model,” *Astr. Softw. Eng.*, vol. 8, nos. 1–4, pp. 85–121, 1999.

[10] M. Xie, “Software reliability models—past, present and future,” in *Recent Advances in Reliability Theory: Methodology, Practice and Inference*, N. Limnios and M. Nikulin, Eds. Boston, MA, USA: Birkhauser, 2000, pp. 323–340.

[11] M. R. Lyu, *Handbook of Software Reliability Engineering*. New York, NY, USA: McGraw-Hill, 1996.

[12] N. F. Schneidewind and T. W. Keller, “Application of reliability models to the space shuttle,” *IEEE Softw.*, vol. 9, no. 4, pp. 28–33, Jul. 1992.
YI-FAN HOU received the bachelor’s degree in management from the National Tsing Hua University, Hsinchu, Taiwan, in 2014, through the Double Specialty Program of Management and Technology, and the M.S. degree from the Institute of Information Systems and Applications, National Tsing Hua University, in 2017. Her research interests include software reliability and statistics applications.

CHIN-YU HUANG (Member, IEEE) received the M.S. and Ph.D. degrees in electrical engineering from the National Taiwan University, Taipei, in 1994 and 2000, respectively. He was with the Bank of Taiwan, from 1994 to 1999, and was a Senior Software Engineer at Taiwan Semiconductor Manufacturing Company, from 1999 to 2000. Before joining NTHU, in 2003, he was a Division Chief of the Central Bank of China, Taipei. He is currently a Full Professor with the Department of Computer Science, Institute of Information Systems and Applications, National Tsing Hua University, Hsinchu, Taiwan. His research interests include software reliability engineering, software testing, software metrics, software testability, fault tree analysis, and system safety assessment. He received the ‘Tu You Wu Memorial Award of National Science Council, Taiwan, in 2008. He has been on the Editorial Board of Scientific Programming, since 2017, and the Editorial Board of Journal of Information Science and Engineering, since 2016. He is currently serving as an Associate Editor for the IEEE TRANSACTIONS ON RELIABILITY.

CHIH-CHIANG FANG received the B.E. and M.E. degrees from the National Yunlin University of Science and Technology, Yunlin, Taiwan, in 1999 and 2001, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan. His research interests include software testing, debugging, and software measurement.