The Journal of Systems and Software 216 (2024) 112123

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Long-term software fault prediction with wavelet shrinkage estimation ☆, ☆☆

Jingchi Wu *, Tadashi Dohi, Hiroyuki Okamura

Graduate School of Advanced Science and Engineering, Hiroshima University, 1-4-1 Kagamiyama, Higashi-Hiroshima City, Hiroshima, 739-8527, Japan

ARTICLE INFO

Keywords:
Software reliability
Fault prediction
Non-homogeneous Poisson processes
Wavelet shrinkage estimation
Predictive performance
Tool development

ABSTRACT

Wavelet shrinkage estimation received considerable attentions to estimate stochastic processes such as a non-homogeneous Poisson process in a non-parametric way, and was applied to software reliability estimation/prediction. However, it lacks the prediction ability for unknown future patterns in long term and penalizes assessing the software reliability in practice. In this paper, we focus on the long-term prediction of the number of software faults detected in the testing phase and propose many novel long-term prediction methods based on the wavelet shrinkage estimation. The fundamental idea is to adopt both the denoised fault-count data and prediction values, and to minimize several kinds of loss functions to make effective predictions. We also develop an automated wavelet-based software reliability assessment tool, W-SRAT2, which is a drastic extension of the existing tool, W-SRAT, by adding those prediction algorithms. In numerical experiments with 6 actual software development project data, we investigate the predictive performance of our long-term prediction approaches, which consist of 2,640 combinations, and compare them with the common software reliability growth models with the maximum likelihood estimation. It is shown that our wavelet shrinkage estimation/prediction methods outperform the existing software reliability growth models.

1. Introduction

Software reliability growth model (SRGM) is a simple but the most fundamental solution to monitor the software testing progress, and can be used for quantification of software product reliability and software release engineering (Lyu, 1996). It describes the relationship between software testing time and the cumulative number of software fault-counts. Software fault-count phenomenon is commonly modeled by a stochastic counting process with respect to the testing time measured by calendar time in many cases. Especially, non-homogeneous Poisson process (NHPP)-based SRGMs have been considered as the most popular solutions in the literature, where the parametric form of the intensity function in an NHPP is assumed in advance. Although the NHPP is one of the simplest stochastic counting processes, a series of the reliability assessment procedures including parameter estimation, model selection and prediction of reliability measures, are not so trivial. In the parameter estimation with the software fault-count data experienced in the testing, the commonly used technique is the maximum likelihood estimation. However, several drawbacks for the maximum likelihood estimation are pointed out. For instance, there may not always exist a unique and finite maximum likelihood estimate of the model parameter. Also, it has been known that the parametric model selection of NHPP-based SRGMs strongly depends on the kind of software fault-count data, so that the best SRGM which can fit every software fault-count data does not exist. Then the model selection of SRGMs is quite troublesome in practice, because a great number of parametric SRGMs have been proposed during the last almost 5 decades. The Bayesian estimation is an alternative option to predict the quantitative software reliability (Achcar et al., 1998; Kuo and Yang, 1996). However, the computation cost to calculate the posterior distribution of the number of software faults is rather expensive, and the interpretation of the resulting quantification is not always easy for practitioners (Okamura and Dohi, 2009), because the prior distributions and the hyperparameters affect the prediction results as well.

Hence, non-parametric methods to estimate the intensity function in NHPP-based SRGM without assuming any parametric form have been proposed by many authors. Barghout et al. (1998) considered a non-parametric order statistics SRGM to predict the software fault-detection time. Saito and Dohi (2015) applied a non-parametric maximum likelihood estimation with the fault-detection time data, and estimated the intensity function under a monotone constraint. Dohi et al. (2020) investigated several kernel-based non-parametric approaches, and showed that non-parametric SRGMs can fit the underlying fault-count data better than the representative parametric

☆ This paper is an extended version of the conference paper (Wu etal, 2021) presented by the same authors.
☆☆ Editor: Raffaela Mirandola.
* Corresponding author.
E-mail addresses: d220580@hiroshima-u.ac.jp (J. Wu), dohi@hiroshima-u.ac.jp (T. Dohi), okamu@hiroshima-u.ac.jp (H. Okamura).

https://doi.org/10.1016/j.jss.2024.112123
Received 15 February 2024; Received in revised form 7 May 2024; Accepted 4 June 2024
Available online 7 June 2024

NHPP-based SRGMs. The above references just focused on the statistical estimation of the software intensity function with the software fault-detection time domain data, but did not consider the time-interval (group) data of software fault-counts measured by calendar time. Saito and Dohi (2016) applied the same idea as Sofer and Miller (1991) to the group data analysis, and refined the non-parametric estimation/prediction scheme with the group data. Unfortunately, these non-parametric methods are still expensive in computation. In general, whether it is a parametric model or a non-parametric model, optimization algorithms need to be applied to obtain estimates. For parametric models, maximum likelihood estimation is used, while for non-parametric models, such as kernel estimation, bandwidth estimation requires the application of optimization algorithms. Therefore, they are expensive in computation. Xiao and Dohi (2013a,b) introduced the wavelet shrinkage methods to estimate the discrete intensity function of NHPP-based SRGM as another non-parametric estimation scheme with cheap in computation, where only simple mathematical operations are needed (without 'lht' threshold method). However, they do not development a long-term prediction method for wavelet-based SRGMs. In this paper, we focus on the long-term prediction of the number of software faults detected in the testing phase and propose many novel long-term prediction methods based on the wavelet shrinkage estimation. The fundamental idea is to adopt both the denoised fault-count data and prediction values, and to minimize several kinds of loss functions to make effective predictions.

Apart from non-parametric approaches with SRGMs, it is quite important to develop the computer-aided software reliability assessment tools in actual software reliability management. Since the early 80 s, several reliability assessment tools have been developed to promote the application of software reliability engineering. Emerald (Hudepohl et al., 1996), AT&T SRE Toolkit (Software, 1990), SoRel (Kanoun et al., 1993), SMERFS (Farr, 1993), CASRE (Lyu and Nikora, 1992), Robust (Li and Malaiya, 1995), SREPT (Ramani et al., 2000), CARATS (Huang and Lyu, 2011), SRMP (Reliability, 1988) are the legacy tools in software reliability engineering, where almost all tools deal with the parametric SRGMs. Okamura and Dohi (2013) developed SRATS with 11 parametric NHPP-based SRGMs, and implemented the EM (Expectation-Maximization) algorithms to enjoy the global convergence of model parameters in the maximum likelihood estimation. Shibata et al. (2007, 2015) developed PISRAT and M-SRAT to incorporate the software metrics such as the development effort, code size, code complexity, etc. in software reliability assessment. Cinque et al. (2017) and Carrozza et al. (2018) proposed a debugging-workflow-aware SRGM to utilize data from issue tracking systems, and an SVEVIA framework for software quality assessment, decision support and productivity management with SRGMs, respectively. Nagaaraju et al. (2019) developed SFRAT, which is an open source to incorporate the methods into the software testing workflows, and to contribute related statistical predictions. In this way, the tool development is necessary to apply the theoretical models to the practical software reliability management. It should be noted, however, that the above tools have also treated only the parametric SRGMs with specific intensity functions and different parameter estimation methods such as the least squares method. In other words, there is no software reliability assessment tool based on the non-parametric approach in the past.

This paper introduces 2640 wavelet prediction methods by applying various data transformation methods and threshold rules. To address the issue of determining the most effective combination of data transformation methods and threshold rules for different types of software failure data, we rank the prediction methods in terms of PMAE. We test the predictive performance of each combination across 6 different actual software development project datasets and 3 different software testing phases. The model with the strongest overall predictive performance implies the greatest versatility. This method can be applied to various types of software failure data and represents the most effective combination of data transformation methods and threshold rules. We compare our wavelet prediction method with existing parametric methods. In the field of software reliability, hundreds of software reliability growth models have been proposed. We compare with the most representative 11 software reliability models. The results show that our wavelet method significantly outperforms any of the eleven existing representative SRMs in terms of overall predictive performance. We also discuss the sensitivity about different prediction methods and described the threats to the validity of the experiment.

The remaining part of this paper is organized as follows. In Section 2, we briefly introduce the recently research work about wavelet in software reliability. In Section 3, we overview the NHPP-based SRGMs. Section 4 briefly summarizes the wavelet shrinkage estimation techniques (Xiao and Dohi, 2013a,b). Our contribution starts in Section 5. In Section 5.1, we summarize the short-term software reliability prediction method proposed by Xiao and Dohi (2013a,b) In Section 5.2, we propose two new long-term prediction algorithms and some variants of the loss function in Section 5.3. In Section 5.4, we further propose a weighted loss function that can be superimposed on the loss function proposed in Section 5.3. Overall, we propose 60 different long-term prediction methods in Section 5. The each long-term prediction method can combine with different wavelet shrinkage estimation (denoise) method, so we propose total 2640 wavelet-based long-term prediction methods. In Section 6, we introduce our developed wavelet-based software reliability assessment tool, W-SRAT2, from the viewpoints of the interface and functionality. Section 7 is devoted to numerical experiments. Finally, the paper is concluded with some remarks in Section 8.

2. Recently research

The wavelet shrinkage estimation (WSE) is a lightweight non-parametric estimation for the NHPP-based SRGM , which can be applied when the mean value function or intensity function is completely unknown. The fundamental idea is to apply the discrete Haar-wavelet (Daubechies, 1992). Xiao and Dohi (2013a,b) considered the data transform based WSE (DT-WSE) (Xiao and Dohi, 2013a) and non-data transform based WSE (NDT-WSE) (Xiao and Dohi, 2013b) to estimate the NHPP-based SRGM without specifying the mean value function. Since the wavelet shrinkage estimation is based on denoising of the underlying fault-count data, it does not need to solve any complex optimization problem in the maximum likelihood estimation, and to compute multiple integrals via the Markov chain Monte Carlo in the Bayesian estimation. In that sense, the wavelet-based approach is regarded as a lightweight non-parametric estimation method of the software intensity function. However, it has a serious drawback that not only the long-term prediction of the number of software faults but also the quantification of software reliability for an arbitrary testing/operational period is difficult. Xiao and Dohi (2013a) proposed an intuitive approach to predict sequentially the number of software faults in a fashion similar to Barghout et al. (1998) . This method indeed enables us to make the one-stage look-ahead prediction, but does not lead to the multi-stage look-ahead long-term prediction of quantitative software reliability, which is defined as the probability that software fault is fault-free for an arbitrary time period.

Recently, Wu et al. (2021) developed an automated wavelet-based software reliability assessment tool, W-SRAT, and implemented a long-term prediction algorithm for predicting the number of software fault-counts in future, by refining the one-stage look-ahead prediction by Xiao and Dohi (2013a) . Although it was the first and unique challenge for the long-term prediction in non-parametric NHPP-based SRGMs with the wavelet shrinkage estimation, it is important to make the prediction accuracy higher. More specifically, Wu et al. (2021) considered only one prediction algorithm with 40 wavelet shrinkage estimation methods.

In this paper, we focus on the long-term prediction of the number of software faults detected in the testing phase and propose many novel long-term prediction methods based on the Haar wavelet shrinkage estimation. In fact, both NDT-WSE and DT-WSE are based on the Haar wavelet function, which is the representative discrete wavelet, because we deal with the fault-count group data. The advantage of the Haar wavelet function over other wavelet functions is due to the low computational cost in estimation. The Haar wavelet transform can be easily implemented with simple algebra. Other wavelet functions, such as the Daubechies wavelet function, usually do not have a closed-form expression and can be derived numerically. Dissimilar to the maximum likelihood estimation, the wavelet shrinkage estimation (denoising) does not require to solve optimization problems, because the denoising is a simple operation in almost all cases expect 'lht' threshold method. Therefore, it is obvious that the computational cost of Haar wavelet estimation is less than that of other methods. The fundamental idea about our proposed long-term prediction methods is to adopt both the denoised fault-count data and prediction values, and to minimize several kinds of loss functions to make effective predictions.

We also develop an automated wavelet-based software reliability assessment tool, W-SRAT2, which is a drastic extension of the existing tool W-SRAT (Wu et al., 2021) by adding those prediction algorithms. W-SRAT2 provides, in total, 2640 combinations of prediction methods consisting of 60 wavelet shrinkage prediction algorithms under 44 wavelet shrinkage estimation methods with 36 data transform methods and 8 non-data transform methods in Xiao and Dohi (2013a,b), and enables us to use one of them for software reliability estimation/prediction. W-SRAT2 consists of two components; a user interface on the web in the front end, and a computation module written by Python language in the back end. It is a web-based freeware to predict the cumulative number of software faults and the quantitative software reliability.

3. Summary of NHPP-based SRGMs

3.1. Definition

Software reliability growth model (SRGM) describes the relationship between software testing time and the number of software faults detected in the software testing, and is regarded as a stochastic counting process. Non-homogeneous Poisson process (NHPP) is considered as an appropriate SRGM for describing the above relationship. Let $N(t)$ be the cumulative number of software faults detected during the time interval $(0, t]$, which is a random variable at a fixed time t, having the mean value function $\mathrm{E}[N(t)] = \Lambda(t)$ or intensity function $\lambda(t) = d\Lambda(t)/dt$. Then the probability mass function (p.m.f.) of $N(t)$ is given by

$$\Pr(N(t) = k) = \frac{(\Lambda(t))^k}{k!} e^{-\Lambda(t)}. \tag{1}$$

When the software testing time t is discretized as $t = t_0, t_1, t_2, \ldots (t \in \mathbb{Z})$ with $t_0 = 0$, we have

$$\Pr(N(t_i) - N(t_{i-1}) = k) = \frac{\left(\Lambda(t_i) - \Lambda(t_{i-1})\right)^k}{k!} \times e^{-\{\Lambda(t_i) - \Lambda(t_{i-1})\}}, \tag{2}$$

which denotes the p.m.f. of cumulative number of software faults detected during the testing time interval $(t_{i-1}, t_i]$ $(i = 1, 2, \ldots)$. From Eqs. (1) and (2), it is easy to know that the NHPP-based SRGM is determined uniquely by giving the mean value function $\Lambda(t)$. The main concerns in the past literature were to propose appropriate parametric forms of $\Lambda(t)$ so as to fit the underlying software fault-count data experienced before. In the discrete-time setting, Yamada et al. (1984), Yamada and Osaki (1985) and Okamura et al. (2004, 2005) introduced several parametric discrete NHPP-based SRGMs. The discrete-time SRGMs seem to be used for the above time scale at the first look. However, it is essentially same as the group data analysis with the continuous-time SRGMs. Hereafter, we will consider the discrete-time NHPP-based SRGM. When software is released at the nth testing time, t_n, the quantitative software reliability function is defined as the probability that no software fault is detected during the time period $(t_n, t_n + l]$, where l $(= 1, 2, \ldots)$ denotes the operational period after the release. Then the software reliability function is given by $R_n(l) = \exp\{-(\Lambda(t_n + l) - \Lambda(t_n))\}$ (Lyu, 1996).

3.2. Maximum likelihood estimation

Suppose that the mean value function $\Lambda(t) = \Lambda(t; \boldsymbol{\theta})$ and the intensity function $\lambda(t) = \lambda(t; \boldsymbol{\theta})$ are known, but the parameters $\boldsymbol{\theta}$ are unknown. The commonly used technique to estimate the software mean value function $\Lambda(t; \boldsymbol{\theta})$ is the maximum likelihood estimation. Let $\boldsymbol{\theta}$ be the model parameter (vector) in the mean value function in NHPP-based SRGMs. If the functional form of $\Lambda(t; \boldsymbol{\theta})$ is given with an arbitrary $\boldsymbol{\theta}$, then the statistical estimation problem is reduced to determine the parameter $\boldsymbol{\theta}$ from the underlying software fault-count data. Suppose that the fault-count data, which consist of the discrete time sequence, t_i $(i = 0, 1, 2, \ldots, n)$, and the cumulative number of software faults detected by time t_i, y_i, is available in the testing phase. Then the log likelihood function with the group data (t_i, y_i) $(i = 0, 1, 2, \ldots, n)$ is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}; (t_i, y_i), i = 0, 1, 2, \ldots, n) = &\sum_{i=1}^{n} (y_i - y_{i-1}) \ln\{\Lambda(t_i; \boldsymbol{\theta}) - \Lambda(t_{i-1}; \boldsymbol{\theta})\} \\ &- \sum_{i=1}^{n} \ln\{(y_i - y_{i-1})!\} - \Lambda(t_n; \boldsymbol{\theta}),\end{aligned} \tag{3}$$

where $(t_0, y_0) = (0, 0)$. The maximum likelihood estimate of $\boldsymbol{\theta}$, $\hat{\boldsymbol{\theta}}$, is obtained by maximizing $\mathcal{L}(\boldsymbol{\theta}; (t_i, y_i), i = 0, 1, 2, \ldots, n)$ with respect to $\boldsymbol{\theta}$. In this paper, we consider the representative NHPP-based SRGMs with $\Lambda(t; \boldsymbol{\theta}) = \omega F(t; \boldsymbol{\alpha})$ with $\boldsymbol{\theta} = (\omega, \boldsymbol{\alpha})$, where $F(t; \boldsymbol{\alpha})$ is the cumulative distribution function (c.d.f.) denoting the absolutely continuous fault-detection time distribution and $\boldsymbol{\alpha}$ is the model parameter vector. Okamura and Dohi (2013) developed SRATS, Software Reliability Assessment Tool on the Spreadsheet, and implemented 11 representative NHPP-based SRGMs (fault-detection time c.d.f.). SRATS carry out the maximum likelihood estimation with the EM algorithms (Okamura et al., 2002). Table 1 presents the 11 representative NHPP-based SRGMs implemented in SRATS (Okamura and Dohi, 2013).

4. Wavelet shrinkage estimation

4.1. Overview of wavelet shrinkage estimation

Next, we consider the case where the mean value function $\Lambda(t)$ or intensity function $\lambda(t)$ is completely unknown. In this case, the method of parametric maximum likelihood in Section 3.2 does not work under the incomplete knowledge on the intensity function and/or mean value function. Xiao and Dohi (2013a,b) considered the data transform-based WSE (DT-WSE) (Xiao and Dohi, 2013a) and non-data transform-based WSE (NDT-WSE) (Xiao and Dohi, 2013b) to estimate the NHPP-based SRGM without specifying the mean value function.

In what follows, we summarize the WSE of an NHPP-based SRGM. Define the *discrete intensity function*:

$$\Delta\Lambda(t_i) = \Lambda(t_i) - \Lambda(t_{i-1}) = \int_{t_{i-1}}^{t_i} \lambda(t) dt, \qquad i = 1, 2, \ldots, \tag{4}$$

which denotes the expected number of software faults during the ith testing interval $(t_{i-1}, t_i]$, $i = 1, 2, \ldots$. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ be the (non-cumulative) number of software faults detected at the ith testing interval and be the realizations of $N(t_i) - N(t_{i-1})$, where $y_i = \sum_{j=0}^{i} x_j$ and $x_0 = 0$. Our concern is to estimate $\Delta\Lambda(t_i)$ with the observation $\boldsymbol{x}$. The WSE is summarized in the following three steps:

Table 1
The representative parametric NHPP-based SRGMs.

Models	$\Lambda(t;\boldsymbol{\theta})$
Exponential dist. (exp) (Goel and Okumoto, 1979)	$\Lambda(t;\boldsymbol{\theta}) = \omega F(t;\boldsymbol{\alpha})$ $F(t;\boldsymbol{\alpha}) = 1 - e^{-bt}$
Gamma dist. (gamma) (Yamada et al., 1983; Zhao and Xie, 1996)	$\Lambda(t;\boldsymbol{\theta}) = \omega F(t;\boldsymbol{\alpha})$ $F(t;\boldsymbol{\alpha}) = \int_0^t \frac{c^b s^{b-1} e^{-cs}}{\Gamma(b)} ds$
Pareto dist. (pareto) (Abdel-Ghaly et al., 1986)	$\Lambda(t;\boldsymbol{\theta}) = \omega F(t;\boldsymbol{\alpha})$ $F(t;\boldsymbol{\alpha}) = 1 - (\frac{b}{t+b})^c$
Truncated normal dist. (tnorm) (Okamura et al., 2013)	$\Lambda(t;\boldsymbol{\theta}) = \omega \frac{F(t;\boldsymbol{\alpha}) - F(0;\boldsymbol{\alpha})}{1 - F(0;\boldsymbol{\alpha})}$ $F(t;\boldsymbol{\alpha}) = \frac{1}{\sqrt{2\pi}b} \int_{-\infty}^t e^{-\frac{(s-c)^2}{2b^2}} ds$
Log-normal dist. (lnorm) (Achcar et al., 1998; Okamura et al., 2013)	$\Lambda(t;\boldsymbol{\theta}) = \omega F(\ln t;\boldsymbol{\alpha})$ $F(t;\boldsymbol{\alpha}) = \frac{1}{\sqrt{2\pi}b} \int_{-\infty}^t e^{-\frac{(s-c)^2}{2b^2}} ds$
Truncated logistic dist. (tlogist) (Ohba, 1984)	$\Lambda(t;\boldsymbol{\theta}) = \omega F(t;\boldsymbol{\alpha})$ $F(t;\boldsymbol{\alpha}) = \frac{1-e^{-bt}}{1+ce^{-bt}}$
Log–logistic dist. (llogist) (Gokhale and Trivedi, 1998)	$\Lambda(t;\boldsymbol{\theta}) = \omega F(\ln t;\boldsymbol{\alpha})$ $F(t;\boldsymbol{\alpha}) = \frac{(bt)^c}{1+(bt)^c}$
Truncated extreme-value max dist. (txvmax) (Ohishi et al., 2009)	$\Lambda(t;\boldsymbol{\theta}) = \omega \frac{F(t;\boldsymbol{\alpha}) - F(0;\boldsymbol{\alpha})}{1 - F(0;\boldsymbol{\alpha})}$ $F(t;\boldsymbol{\alpha}) = e^{-e^{-\frac{t-c}{b}}}$
Log-extreme-value max dist. (lxvmax) (Ohishi et al., 2009)	$\Lambda(t;\boldsymbol{\theta}) = \omega F(\ln t;\boldsymbol{\alpha})$ $F(t;\boldsymbol{\alpha}) = e^{-(\frac{t}{b})^{-c}}$
Truncated extreme-value min dist. (txvmin) (Ohishi et al., 2009)	$\Lambda(t;\boldsymbol{\theta}) = \omega \frac{F(0;\boldsymbol{\alpha}) - F(t;\boldsymbol{\alpha})}{F(0;\boldsymbol{\alpha})}$ $F(t;\boldsymbol{\alpha}) = e^{-e^{-\frac{t-c}{b}}}$
Log-extreme-value min dist. (lxvmin) (Goel, 1985)	$\Lambda(t;\boldsymbol{\theta}) = \omega (1 - F(-\ln t;\boldsymbol{\alpha}))$ $F(t;\boldsymbol{\alpha}) = e^{-e^{-\frac{t-c}{b}}}$

$(\omega > 0, b > 0, c > 0)$.

- **Step 1:** Apply the wavelet transform to the realization $\boldsymbol{x}$ to obtain some *scaling coefficients* $\boldsymbol{c}_j$ and *wavelet coefficients* $\boldsymbol{d}_j$, which are real vectors with scale j and contain the information on the time series $\boldsymbol{x}$ at different scales. For convenience of notation, we refer to $\boldsymbol{d}_j$ as $\boldsymbol{d}$, if not specifically stated.
- **Step 2 (Thresholding):** Determine a threshold level τ. The estimates of wavelet coefficients $\boldsymbol{d}$ are obtained by $\widetilde{\boldsymbol{d}} = T(\boldsymbol{d}, \tau)$, where $T(\cdot,\cdot)$ denotes an operator to apply the threshold level τ in order to remove the noise in $\boldsymbol{d}$. There are two thresholding rules; *hard thresholding rule* and *soft thresholding rule* in general. In the hard and soft thresholding rules, the operators are given by $T(\boldsymbol{d},\tau) = \boldsymbol{d}1_{\{|\boldsymbol{d}|>\tau\}}$ and $T(\boldsymbol{d},\tau) = \mathrm{sgn}(\boldsymbol{d}) \times \max\{|\boldsymbol{d}| - \tau, 0\}$, respectively, where $1_{\{\cdot\}}$ and $\mathrm{sgn}(\cdot)$ are the indicator function and the sign function, respectively.
- **Step 3:** Apply the denoised wavelet and scaling coefficients to obtain an (denoised) estimate of $\Delta\Lambda(t_i)$.

4.2. Haar wavelet transform

For the purpose of denoising, we represent the time series $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ as a linear combination of the *Harr scaling function* and *Harr wavelet function*. Suppose $n = 2^j$ $(j = 1, 2, \ldots)$ at the moment. Define the Haar scaling and wavelet functions by

$$\phi(t) = \begin{cases} 1 & (0 \le t < 1) \\ 0 & (\text{otherwise}) \end{cases} \tag{5}$$

and

$$\psi(t) = \begin{cases} 1 & (0 \le t < 1/2) \\ -1 & (1/2 \le t < 1) \\ 0 & (\text{otherwise}), \end{cases} \tag{6}$$

respectively. By introducing the *scaling parameter* j and the *shifting parameter* k, we also define:

$$\phi(t)_{j,k} = \phi(2^j t - k), \tag{7}$$

$$\psi(t)_{j,k} = \psi(2^j t - k). \tag{8}$$

The scaling parameter j means the reduction of the scale of the function $\phi(t)$ or $\psi(t)$ to $1/2^j$, and the shifting parameter k represents the shifting effect of the rescaled $\phi(t)$ or $\psi(t)$ to the right of the number axis k units, where $1/2^j$ is regarded as one scale unit.

Next, we give the mathematical preliminary of the Haar scaling and wavelet functions.

Property 1. *For $\forall j \in \mathbb{N}^+$ and $0 \le k \le 2^j - 1$, the Haar scaling and wavelet functions with different scales satisfy*

$$\phi(t)_{j,2k} = \frac{1}{2}\left[\phi(t)_{j-1,k} + \psi(t)_{j-1,k}\right], \tag{9}$$

$$\phi(t)_{j,2k+1} = \frac{1}{2}\left[\phi(t)_{j-1,k} - \psi(t)_{j-1,k}\right]. \tag{10}$$

To simplify the analysis, let $t_i = i$ $(= 0, 1, 2, \ldots, n)$ with equity interval length $t_{i+1} - t_i = 1$. From the realization $\boldsymbol{x}$, we define the following naive estimate as a segmented function:

$$\Delta\hat{\Lambda}(t) = \begin{cases} x_1 & (0 \le t < 1) \\ x_2 & (1 \le t < 2) \\ \quad \cdots & \\ x_n & (n-1 \le t < n). \end{cases} \tag{11}$$

The wavelet transform can deal with an arbitrary function $f(\cdot) \in L^2([0,1])$, so we rescale $\Delta\hat{\Lambda}(t)$ with $t' = t/n$ as

$$\Delta\hat{\Lambda}(t') = \begin{cases} x_1 & (0 \le t' < 1/n) \\ x_2 & (1/n \le t' < 2/n) \\ \quad \cdots & \\ x_n & ((n-1)/n \le t' < 1). \end{cases} \tag{12}$$

From $\phi_{j,k}(t') = 1$ for $t' \in [k/2^j, (k+1)/2^j)$, $\Delta\hat{\Lambda}(t')$ can be expressed as

$$\Delta\hat{\Lambda}(t') = \begin{cases} x_1 \cdot \phi_{J,0}(t') & (0 \le t' < 1/n) \\ x_2 \cdot \phi_{J,1}(t') & (1/n \le t' < 2/n) \\ \qquad \cdots & \\ x_n \cdot \phi_{J,n-1}(t') & ((n-1)/n \le t' < 1) \end{cases} \tag{13}$$

$$= \sum_{k=0}^{2^J-1} c_{J,k}\phi_{J,k}(t'), \tag{14}$$

where $J = \max\{j;\ 2^j \le n,\ j \in \mathbb{N}^+\}$ is the *highest resolution level*, and $\boldsymbol{c}_J = \boldsymbol{x} = (c_{J,0}, c_{J,1}, \ldots, c_{J,2^J-1})$ are the scaling coefficients.

In general, we define the $j-$level function space V_j:

$$V_j = \left\{ \sum_{k=0}^{2^j-1} c_{j,k}\phi(t')_{j,k} | c_{j,k} \in \mathbb{R}, 0 \le t' \le 1 \right\}, \quad j \in \mathbb{N}. \tag{15}$$

It holds that $\Delta\hat{\Lambda}(t') \in V_J$ for an arbitrary discrete function $\Delta\hat{\Lambda}(t')$ with length 2^j. Next, we define the j-level function space W_j:

$$W_j = \left\{ \sum_{k=0}^{2^j-1} d_{j,k}\psi(t')_{j,k} | d_{j,k} \in \mathbb{R}, 0 \le t' \le 1 \right\}, \quad j \in \mathbb{N}, \tag{16}$$

where $\boldsymbol{d}_j = (d_{j,0}, d_{j,1}, \ldots, d_{j,2^j-1})$ are the wavelet coefficients.

Property 2. *V_j and W_j are function spaces which are spanned by the Haar scaling function and wavelet function. Then, it holds that $V_{j+1} = W_j \oplus V_j$ for $\forall j \in \mathbb{N}$, where $\oplus$ means the direct sum.*

The above property leads to the fact that

$$V_j = W_{j-1} \oplus V_{j-1} = W_{j-1} \oplus \cdots \oplus W_{j_0} \oplus V_{j_0} \tag{17}$$

for $\forall j \in \mathbb{N}^+$, where j_0 (≥ 0) is the *primary resolution level*. It is common to set $j_0 = 0$ in many applications. From Eq. (17), an arbitrary function $f_j \in V_j$ can be described as

$$f_j = w_{j-1} + \cdots + w_{j_0} + f_{j_0}, \tag{18}$$

where $f_{j_0} \in V_{j_0}$ and $w_j \in W_j$.

From Properties 1 and 2, $\Delta\hat{\Lambda}(t')$ in Eq. (14) can be obtained by

$$\begin{aligned}
\Delta\hat{\Lambda}(t') &= \sum_{k=0}^{2^J-1} c_{J,k}\phi_{J,k}(t') \\
&= \sum_{k=0}^{2^{(J-1)}-1} \left[d_{(J-1,k)}\psi(t')_{J-1,k} + c_{(J-1,k)}\phi(t')_{J-1,k} \right] \\
&= w_{J-1} + f_{J-1} \\
&= w_{J-1} + w_{J-2} + \cdots + w_{j_0} + f_{j_0} \\
&= \sum_{j=j_0}^{J-1} \sum_{k=0}^{2^j-1} d_{j,k}\psi_{j,k}(t') + \sum_{k=0}^{2^{j_0}-1} c_{j_0,k}\phi_{j_0,k}(t'),
\end{aligned} \tag{19}$$

where the scaling coefficients $c_{j_0,k}$ and wavelet coefficients $d_{j,k}$ are given by

$$c_{j-1,k} = \frac{1}{2}\left(c_{j,2k} + c_{j,2k+1}\right), \tag{20}$$

$$d_{j-1,k} = \frac{1}{2}\left(c_{j,2k} - c_{j,2k+1}\right) \tag{21}$$

with $\boldsymbol{c}_J = \boldsymbol{x} = (c_{J,0}, \ldots, c_{J,2^J-1})$. This decomposition is called the *Harr wavelet transformation*, and transforms the original time series $\boldsymbol{x}$ to the scaling and wavelet coefficients. From a simple algebra, the inverse Harr transforms are given by

$$c_{j,2k} = c_{j-1,k} + d_{j-1,k}, \tag{22}$$

$$c_{j,2k+1} = c_{j-1,k} - d_{j-1,k}. \tag{23}$$

For more details on the wavelet transform, see Xiao and Dohi (2013a,b).

4.3. Gaussian thresholding

Thresholding plays a central role for the wavelet shrinkage estimation. The rationale for thresholding is due to the fact that the wavelet coefficients should be sparse, so we suppose that the wavelet coefficients, which are close to zero, are caused by the estimation error. Then, it is appropriate to remove the estimation noise to improve an estimate $\Delta\hat{\Lambda}(t')$. Donoho and Johnstone (1994) and Nason (1996) proposed the *universal threshold* (ut) and the *leave-out-half cross-validation threshold* (lht), for the noise caused by a Gaussian process, which are given by

$$\tau = \sqrt{2\log n}, \tag{24}$$

$$\tau = \left(1 - \frac{\log 2}{\log n}\right)^{-\frac{1}{2}} \sigma(n/2), \tag{25}$$

respectively. In Eqs. (24) and (25), n is the length of input vector $\boldsymbol{x}$ and the mathematical definition of parameter $\sigma(n/2)$ is referred in Nason (1996). Since the wavelet coefficients have a multiresolution feature, a different *level-dependent universal threshold* (lut) for the sake of Eq. (24) is given by

$$\tau_j = \sqrt{2\log n_j} \qquad j = 1, 2, \ldots, J, \tag{26}$$

under the Gaussian assumption, where n_j are the lengths of jth level wavelet coefficients.

It should be noted that 'ut', 'lht' and 'lut' in the above can be used to reduce the estimation errors caused by a Gaussian process. Because the underlying software fault-count data are considered as realizations of the Poisson random variables in Eq. (1), we need the data transform from the Poisson count data to the Gaussian data before applying the Gaussian noise thresholding.

In this paper, we employ 6 kinds of data transform methods. For the Poisson count data $\boldsymbol{x}$, we apply the following representative data transform formulae.

- **Bartlett transforms** (Bartlett, 1936, 1947):

$$\begin{aligned}
s_i &= f_{B1}(x_i) = 2\sqrt{x_i - \frac{1}{2}}, \\
s_i &= f_{B2}(x_i) = \sqrt{x_i - \frac{1}{2}},
\end{aligned} \tag{27}$$

  can be regarded as samples from the normal distributions with variances of 1 and 1/4, respectively. We call these data transform methods Bartlett1 (B1) and Bartlett2 (B2), respectively with the inverse data transforms $x_i = (s_i/2)^2 + 1/2$ and $x_i = (s_i)^2 + 1/2$.
- **Anscombe transforms** (Anscombe, 1948) :

$$\begin{aligned}
s_i &= f_{A1}(x_i) = 2\sqrt{x_i - \frac{3}{8}}, \\
s_i &= f_{A2}(x_i) = \sqrt{x_i - \frac{3}{8}},
\end{aligned} \tag{28}$$

  can be regarded as samples from the normal distributions with variances of 1 and 1/4, respectively. We call these data transform methods Anscombe1 (A1) and Anscombe2 (A2), respectively. Their inverse data transforms are given by $x_i = (s_i/2)^2 + 3/8$ and $x_i = (s_i)^2 + 3/8$.
- **Fisz transforms** (Fisz, 1955): Let $c_i = \frac{1}{2}(x_{2i} + x_{2i+1})$ and $d_i = \frac{1}{2}(x_{2i} - x_{2i+1})$. Then,

$$\begin{aligned}
s_i &= f_{F1}(c_i, d_i) = \frac{d_i}{\sqrt{c_i}}, \\
s_i &= f_{F2}(c_i, d_i) = \frac{d_i}{2\sqrt{c_i}},
\end{aligned} \tag{29}$$

  can be regarded as samples from the normal distributions with variances of 1 and 1/4, respectively. We call these data transform methods Fisz1 (F1) and Fisz2 (F2), respectively, where their inverse data transforms are given by $x_{2i} = c_i + d_i$ and $x_{2i+1} = c_i - d_i$ with $d_i = s_i\sqrt{c_i}$ and $d_i = 2s_i\sqrt{c_i}$.

The WSEs with the above Gaussian noise thresholding are referred to as DT-WSEs in this paper (see Xiao and Dohi (2013a)).

4.4. Poissonian thresholding

Kolaczyk (1997) developed a specific threshold level for the Poisson noise and called it the *level-dependent threshold* (ldt):

$$\tau_j = 2^{\frac{J-j+2}{2}} \left\{ 2\log(2^j) + \sqrt{\left(4\log(2^j)\right)^2 + 8\lambda_0 \log(2^j) 2^{(J-j)}} \right\}, \tag{30}$$

where λ_0 is the sample mean of the underlying data $\boldsymbol{x}$, and j is the *current resolution level*. We can apply 'ldt' directly to wavelet coefficients $\boldsymbol{d}$ for denoising without transforming from the Poisson to the Gaussian data.

Another thresholding technique for the Poisson noise by Kolaczyk (1997) is the *translation-invariant Poisson smoothing using Haar Wavelets* (TIPSH) to overcome the 'staircase'-like appearance in the estimate of $\Delta\hat{\Lambda}(t')$ in Eq. (19). Xiao and Dohi (2013b) employed the TIPSH algorithm to denoise the Poisson count data without applying the data transform. From Eqs. (5) and (6), it is seen that the Haar wavelets are not continuous, but have a block-like shape. This staircase-like structure sometimes produces the 'artifacts' (Kolaczyk, 1997). A typical impact of artifacts is observed in image processing. An example of a potential artifact is the sharp edges or unnatural abrupt changes that may occur in the transformed image due to the step-like characteristics of the Haar wavelet. These abrupt changes do not reflect the actual characteristics of the original image, but are introduced by the transformation process. This type of artifact may cause confusion in subsequent image analysis or interpretation, thus measures need to be taken to identify and eliminate these artifacts.

To resolve these artifact problems, TIPSH algorithm works well by averaging estimates of all circularly shifted versions of the fault-count data.

From Eq. (19), we confirm again that $\Lambda(t')$ can be uniquely represented by vector $\boldsymbol{c}_J = \boldsymbol{x}$. Let $W_\tau(\boldsymbol{c}_J)$ be the wavelet shrinkage estimate of $\boldsymbol{c}_J$. By expanding $\boldsymbol{c}_J$ to scaling coefficients $\boldsymbol{c}_{j,k}$ and wavelet coefficients $\boldsymbol{d}_{j,k}$, and denoising the wavelet coefficients as $\widetilde{\boldsymbol{d}}_{j,k} = T(\boldsymbol{d}_{j,k}, \tau)$, we obtain the resulting estimate $\widetilde{\boldsymbol{c}}_J$ of $\boldsymbol{c}_J$. Let $\boldsymbol{S}(\cdot)$ be an operator to generate a matrix, where $S_i(\boldsymbol{x})$ represents to move the elements of $\boldsymbol{c}_J$, and i positions cyclically. Then, the elements of $\boldsymbol{c}_J$ are shifted such as

$$\boldsymbol{S}(\boldsymbol{c}_J) = \left[S_1(\boldsymbol{x}), S_2(\boldsymbol{x}), \dots, S_n(\boldsymbol{x})\right]^T = \begin{bmatrix} x_2 & x_3 & \dots & x_n & x_1 \\ x_3 & x_4 & \dots & x_1 & x_2 \\ & & \vdots & & \\ x_1 & x_2 & \cdots & x_{n-1} & x_n \end{bmatrix}, \tag{31}$$

where T is transpose. Define the matrix

$$\widetilde{\boldsymbol{S}}(\boldsymbol{c}_J) = \left[W_\tau\left(S_1(\boldsymbol{x})\right), W_\tau\left(S_2(\boldsymbol{x})\right), \dots, W_\tau\left(S_n(\boldsymbol{x})\right)\right]^T \tag{32}$$

and the shift-inverse operator $\boldsymbol{S}_{-1}(\cdot)$:

$$\boldsymbol{S}_{-1}(\boldsymbol{S}(\boldsymbol{c}_J)) = \begin{bmatrix} x_1 & x_2 & \dots & x_{n-1} & x_n \\ x_1 & x_2 & \dots & x_{n-1} & x_n \\ & & \vdots & & \\ x_1 & x_2 & \cdots & x_{n-1} & x_n \end{bmatrix}. \tag{33}$$

Then the TIPSH estimate, $\widetilde{\boldsymbol{c}}_J$, is given by

$$\widetilde{\boldsymbol{c}}_J = \boldsymbol{G} \cdot \boldsymbol{S}_{-1}(\widetilde{\boldsymbol{S}}(\boldsymbol{c}_J)) = (\widetilde{c}_{J,0}, \widetilde{c}_{J,1}, \dots, \widetilde{c}_{J,n-1}), \tag{34}$$

where $\boldsymbol{G} = (\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n})$ is an n-dimensional vector. Finally, we get an estimate of the mean value function by

$$\widetilde{\Lambda}(t') = \sum_{k=0}^{2^J-1} \widetilde{c}_{J,k} \phi_{J,k}(t'). \tag{35}$$

4.5. Moving average

In the above subsection, we considered only the case where the data length is $n = 2^j$ $(j = 1, 2, \dots)$. This is due to the fact that the Haar wavelet transform can only handle the data of length $n = 2^j$ $(j = 1, 2, \dots)$ in nature. Hence we need a way to apply the Haar wavelet transform to data with arbitrary length. Xiao and Dohi (2013a,b) applied the moving average method to solve this problem. For the data with arbitrary length $n \in \mathbb{N}^+$, we first obtain the maximum resolution $J = \max\left\{j;\ 2^j \le n,\ j \in \mathbb{N}^+\right\}$. If $2^J = n$, then we can directly apply the Haar wavelet transform, otherwise, say $2^J < n$, we partition the function $\Delta\hat{\Lambda}(t')$ into sub-functions $\Delta\hat{\Lambda}_l(t')$, $l = 0, 1, \dots, n - 2^J$:

$$\Delta\hat{\Lambda}_l(t') = \begin{cases} x_{1+l} & \left(\frac{l}{n} \le t < \frac{l+1}{n}\right) \\ x_{2+l} & \left(\frac{l+1}{n} \le t < \frac{l+2}{n}\right) \\ \dots & \\ x_{2^J+l} & \left(\frac{2^J+l-1}{n} \le t < \frac{2^J+l}{n}\right) \\ 0 & \text{otherwise.} \end{cases} \tag{36}$$

The length of each sub-function $\Delta\hat{\Lambda}_l(t')$ is given by 2^J, so that the Haar wavelet transform can be applied. Then, we calculate the estimate of each sub-function $\Delta\widetilde{\Lambda}_l(t')$ individually:

$$\begin{aligned} \Delta\widetilde{\Lambda}_l(t') &= W_\tau(\Delta\hat{\Lambda}_l(t')) \\ &= \begin{cases} \widetilde{x}^l_{1+l} & \left(\frac{l}{n} \le t < \frac{l+1}{n}\right) \\ \widetilde{x}^l_{2+l} & \left(\frac{l+1}{n} \le t < \frac{l+2}{n}\right) \\ \dots & \\ \widetilde{x}^l_{2^J+l} & \left(\frac{2^J+l-1}{n} \le t < \frac{2^J+l}{n}\right) \\ 0 & \text{otherwise,} \end{cases} \end{aligned} \tag{37}$$

where $W_\tau(\cdot)$ denotes the Haar wavelet estimation. Finally, we obtain the estimate $\Delta\widetilde{\Lambda}(t')$ by averaging the $\Delta\widetilde{\Lambda}_l(t')$ as

$$\Delta\widetilde{\Lambda}(t) = \frac{\sum_{l=0}^{n-2^J} \Delta\widetilde{\Lambda}_l(t')}{n - 2^J}. \tag{38}$$

5. Wavelet shrinkage prediction

As motivated in Section 1, there is no satisfactory method for the long-term prediction of the cumulative number of software faults when WSE is applied. Wu et al. (2021) proposed a long-term prediction algorithm for predicting the number of software fault-counts in future, by repeating the one-stage look-ahead prediction. Note that the prediction algorithm depends on the estimation method of the discrete intensity function in the NHPP-based SRGMs. More specifically, Wu et al. (2021) considered a multi-stage look-ahead prediction under 40 estimation methods. In this paper, we develop 60 prediction algorithms under 44 estimation methods, so the total number of these combinations is 2640. The 44 estimation methods consist of 36 data transform algorithms and 8 non-data transform algorithms, where 6 data transform methods (Bartlett1, Bartlett2, Ascome1, Anscome2, Fisz1, Fisz2), 2 thresholding rules (soft and hard rules), and 3 threshold values (ut, lut, lht) are considered in the data transform algorithms, and 2 thresholding rules (soft and hard rules) and 2 threshold values (ldt, lht) are used in the non-data transform algorithms under each case where TIPSH is used or not.

5.1. One-stage look-ahead prediction

Xiao and Dohi (2013a) introduced a short-term prediction method (Barghout et al., 1998) for the wavelet shrinkage prediction. Their idea is to use the observed data $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ and a arbitrary constant q

for constructing a loss function $M_0(\boldsymbol{x}, q)$. The constant minimizing the value of $M_0(\boldsymbol{x}, q)$, $\hat{q} = q^*$, is regarded as a prediction of the cumulative number of software faults detected at the subsequent testing day. We use the observed data $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and an arbitrary constant q to make a new sequence $\boldsymbol{x}_q = (x_1, x_2, \ldots, x_n, x_{n+1} = q)$. We apply a wavelet shrinkage estimation to obtain an estimate $\tilde{\boldsymbol{x}}_q = (\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{x}_{n+1})$ by denoising the updated sequence $\boldsymbol{x}_q$. Then the prediction value q^* is given by

$$q^* = \arg\min_q M_0(\boldsymbol{x}, q), \tag{39}$$

$$M_0(\boldsymbol{x}, q) = \left\| \tilde{\boldsymbol{x}}_q - \boldsymbol{x}_q \right\|_2 = \sqrt{\sum_{i=1}^{n+1} (\tilde{x}_i - x_i)^2}. \tag{40}$$

Algorithm 1: Long-Term Prediction Algorithms 1

Data: Observed software faults group data $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, Prediction term k.
Result: Prediction value $(q_1^*, q_1^*, \ldots, q_k^*)$

```
# Predict k term value;
i = 1;
while i ≤ k do
    # Construct prediction sequence;
    if i == 1 then
        s_{q̂_{i-1}} = x
    else
        s_{q̂_{i-1}} = x + (q*_1, ..., q*_{i-1})
    end
    # Calculate the optimal value;
    q_i^- = arg min_{q_i} M(s_{q̂_{i-1}}, q_i);
    # Obtain the predictive value;
    q*_i = q_i^-;
    # Counter increment;
    i = i + 1;
end
```

5.2. Multi-stage look-ahead prediction

Based on the one-stage look-ahead prediction method, Wu et al. (2021) proposed the long-term prediction algorithm, Algorithm 1, by repeating the short-term prediction. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{q} = (q_1^*, q_2^*, \ldots, q_{k-1}^*)$ be the observed data and $(k-1)$ one-stage look-ahead short-term predictions by substituting $x_{n+i} = q_i^*$ $(i = 1, 2, k-1)$, respectively. We generate a new sequence $\boldsymbol{s}_{q_{k-1}^*} = (x_1, x_2, \ldots, x_n, x_{n+1} = q_1^*, x_{n+2} = q_2^*, \ldots, x_{n+k-1} = q_{k-1}^*)$ by combining $\boldsymbol{x}$ and $\boldsymbol{q}$. Then, we obtain the optimal value q_k^- by

$$q_k^- = \arg\min_{q_k} M_1(\boldsymbol{s}_{q_{k-1}^*}, q_k), \tag{41}$$

$$M_1(\boldsymbol{s}_{q_{k-1}^*}, q_k) = \left\| \tilde{\boldsymbol{s}}_{q_k} - \boldsymbol{s}_{q_k} \right\|_2 = \sqrt{\sum_{i=1}^{n+k} \left(\tilde{x}_i - x_i\right)^2}, \tag{42}$$

where $\boldsymbol{s}_{q_k} = (x_1, \ldots, x_n, x_{n+1} = q_1^*, \ldots, x_{n+k-1} = q_{k-1}^*, x_{n+k} = q_k)$, and $\tilde{\boldsymbol{s}}_{q_k} = (\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{x}_{n+1} = \tilde{q}_1^*, \ldots, \tilde{x}_{n+k-1} = \tilde{q}_{k-1}^*, \tilde{x}_{n+k} = \tilde{q}_k)$, which is a denoised sequence of $\boldsymbol{s}_{q_k}$, i.e., $\tilde{q}_{k-1}^*$ is defined by denoising q_{k-1}^*. The kth prediction value q_k^* is given by q_k^- in Algorithm 1. Wu et al. (2021) investigated Algorithm 1 under the loss function $M_1(\boldsymbol{s}_{q_{k-1}^*}, q_k)$. In the latter discussion we propose different loss functions to develop the other prediction algorithms. In other words, Algorithm 1 implicitly assumes to apply the updated data sequence to obtain the next kth prediction value.

We further propose Algorithm 2 by denoising q_k^- in Eq. (41) to get the prediction value q_k^*. More specifically, after obtaining the optimal value q_k^- in Eq. (41), we construct the $\boldsymbol{s}_{q_k^-} = (x_1, \ldots, x_n, x_{n+1} = q_1^*, \ldots, x_{n+k-1} = q_{k-1}^*, x_{n+k} = q_k^-)$. Next, we denoise the $\boldsymbol{s}_{q_k^-}$ by the wavelet shrinkage estimation which be used in Eq. (42) to obtain $\tilde{\boldsymbol{s}}_{q_k^-} = (\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{x}_{n+1} = \tilde{q}_1^*, \ldots, \tilde{x}_{n+k-1} = \tilde{q}_{k-1}^*, \tilde{x}_{n+k} = \tilde{q}_k^-)$. Then, The kth prediction value q_k^* is given by $\tilde{q}_k^-$ in Algorithm 2.

Algorithm 2: Long-Term Prediction Algorithms 2

Data: Observed software faults group data $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, Prediction term k.
Result: Prediction value $(q_1^*, q_1^*, \ldots, q_k^*)$

```
# Predict k term value;
i = 1;
while i ≤ k do
    # Construct prediction sequence;
    if i == 1 then
        s_{q*_{i-1}} = x
    else
        s_{q*_{i-1}} = x + (q*_1, ..., q*_{i-1})
    end
    # Calculate the optimal value;
    q_i^- = arg min_{q_i} M(s_{q*_{i-1}}, q_i);
    # Construct the estimation sequence;
    s_{q_i^-} = s_{q*_{i-1}} + (q_i^-);
    # Denoise the estimation sequence by wavelet shrinkage
      estimation (WSE);
    s̃_{q_i^-} = WSE(s_{q_i^-});
    # Obtain the predictive value ;
    # (the last element of the sequence s̃_{q_i^-});
    q*_i = q̃_i^- = s̃_{q_i^-}[-1];
    # Counter increment;
    i = i + 1;
end
```

5.3. Some variations

In Eq. (42), the loss function $M_1(\boldsymbol{s}_{q_{k-1}^*}, q_k)$ minimizes the sum of squared errors between $\tilde{x}_i$ and x_i for $i = 1, 2, \ldots, n+k$. However, note that $M_1(\boldsymbol{s}_{q_{k-1}^*}, q_k)$ contains the prediction $q_k = x_{n+k}$ and does depend on q_k which should be irrelevant to the errors at the kth prediction point. We define another loss function;

$$M_2(\boldsymbol{s}_{q_{k-1}^*}, q_k) = \sqrt{\sum_{i=1}^{n+k-1} \left(\tilde{x}_i - x_i\right)^2}, \tag{43}$$

This minor modification with $n+k-1$ components would enable us to remove an effect caused by the kth prediction.

In the one-stage look-ahead prediction, the loss function $M_0(\boldsymbol{x}, q)$ is based on n observations. If one is interested in the prediction with the most recent data sequence, then $n+k$ or $n+k-1$ components in Eqs. (42) and (43) may not be needed to calculate the sum of squared errors. Dissimilar to $M_2(\boldsymbol{s}_{q_{k-1}^*}, q_k)$ in Eq. (43), we define

$$M_3(\boldsymbol{s}_{q_{k-1}^*}, q_k) = \sqrt{\sum_{i=k}^{n+k} \left(\tilde{x}_i - x_i\right)^2}, \tag{44}$$

keeping the number of components as n. As a natural extension, we combine $M_2(\boldsymbol{s}_{q_{k-1}^*}, q_k)$ and $M_3(\boldsymbol{s}_{q_{k-1}^*}, q_k)$ by the following loss function;

$$M_4(\boldsymbol{s}_{q_{k-1}^*}, q_k) = \sqrt{\sum_{i=k}^{n+k-1} \left(\tilde{x}_i - x_i\right)^2}. \tag{45}$$

In the loss function $M_3(\boldsymbol{s}_{q_{k-1}^*}, q_k)$, we apply $\boldsymbol{s}_{q_k} = (x_1, \ldots, x_n, x_{n+1} = q_1^*, \ldots, x_{n+k-1} = q_{k-1}^*, x_{n+k} = q_k)$. Instead of denoising $\boldsymbol{s}_{q_k}$ and obtaining

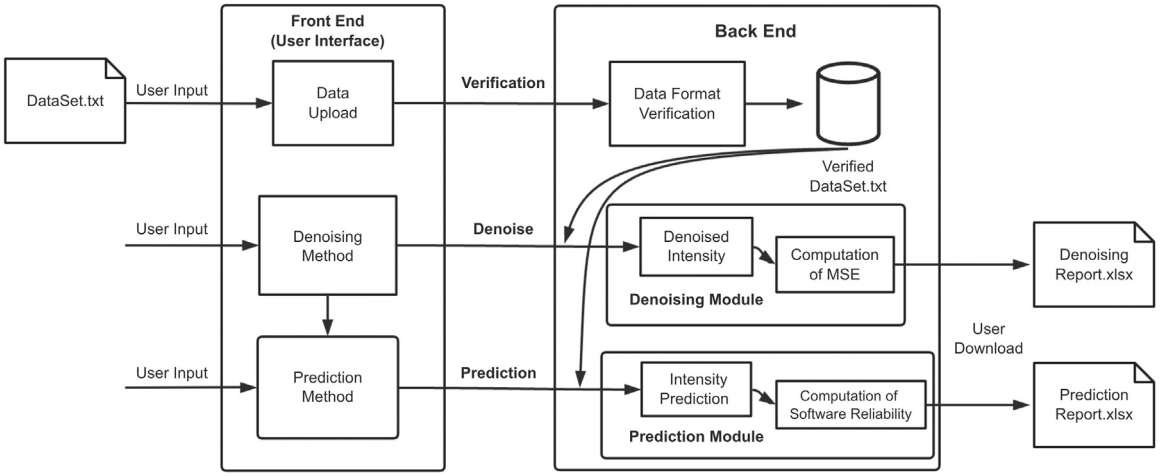


Fig. 1. System architecture.

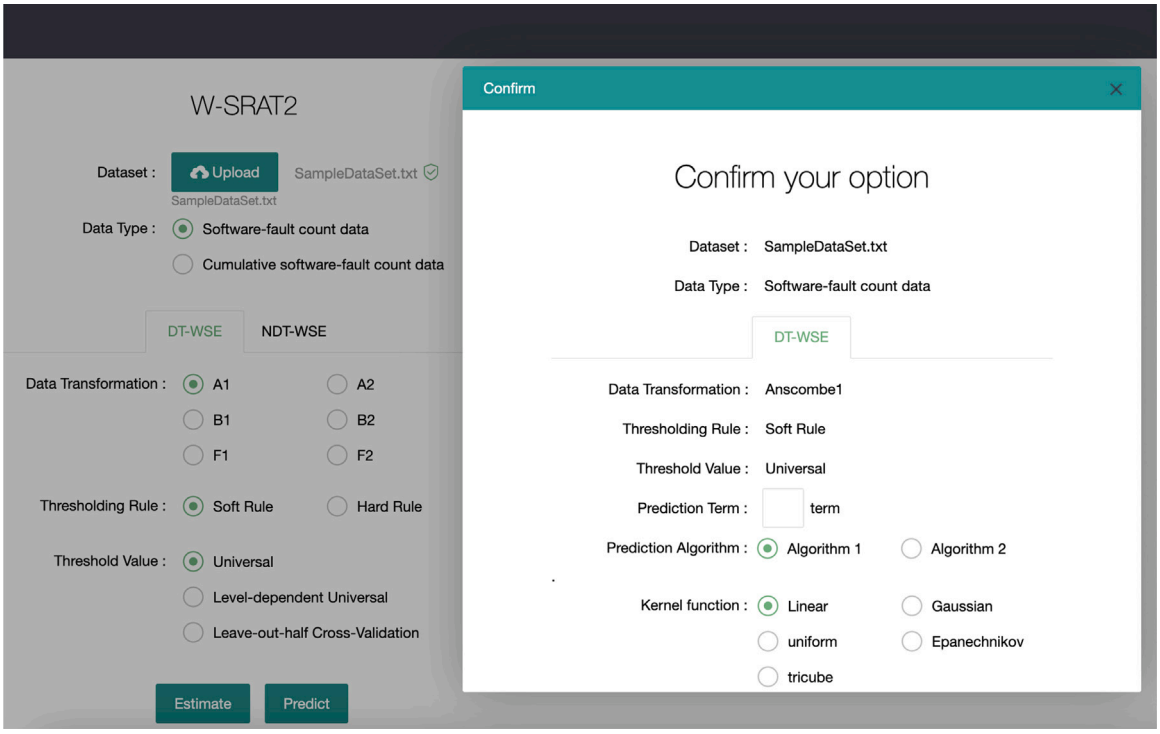


Fig. 2. Interface of W-SRAT2 (screenshot).

$\tilde{\boldsymbol{s}}_{q_k} = (\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{x}_{n+1} = \tilde{q}_1^*, \ldots, \tilde{x}_{n+k-1} = \tilde{q}_{k-1}^*, \tilde{x}_{n+k} = \tilde{q}_k)$, we deal with $\boldsymbol{s}_{q_{k-1}^*} = (x_1, \ldots, x_n, x_{n+1} = q_1^*, \ldots, x_{n+k-1} = q_{k-1}^*)$ by removing the last prediction $x_{n+k} = q_k$. Denoising $\boldsymbol{s}_{q_{k-1}^*}$ yields $\check{\boldsymbol{s}}_{q_{k-1}^*} = (\check{x}_1, \ldots, \check{x}_n, \check{x}_{n+1} = \hat{\tilde{q}}_1, \ldots, \check{x}_{n+k-1} = \hat{\tilde{q}}_{k-1})$, where $\check{x}_n$ $(\neq \tilde{x}_n)$ is the denoised data with different sequence $\boldsymbol{s}_{q_{k-1}^*}$. Hence, we define a different loss function by

$$M_5(\boldsymbol{s}_{q_{k-1}^*}, q_k) = \sqrt{\sum_{i=k}^{n+k-1} \left(\tilde{x}_i - \check{x}_i\right)^2}. \tag{46}$$

Further combining the loss function $M_1(\boldsymbol{s}_{q_{k-1}^*}, q_k)$ with $M_5(\boldsymbol{s}_{q_{k-1}^*}, q_k)$, we obtain

$$M_6(\boldsymbol{s}_{q_{k-1}^*}, q_k) = \sqrt{\sum_{i=1}^{n+k-1} \left(\tilde{x}_i - \check{x}_i\right)^2}. \tag{47}$$

In the above ways, we can define 6 loss functions in each algorithm, Algorithm 1 or Algorithm 2, and propose $6 \times 2 = 12$ prediction methods.

5.4. Weighted loss functions

In Eq. (42) we apply a simple quadratic form for the loss function. However, it is well known that such a simple loss function does not work to make a good prediction in non-linear complex phenomena. Then, we introduce several weighted loss functions. Let k_i be the weights for estimates $\tilde{\boldsymbol{s}}_{q_k} = (\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{x}_{n+k})$ in Algorithms 1 and 2. Then we define the weighted loss function;

$$M_2(\boldsymbol{x}_{\boldsymbol{q}}, q_k) = \sqrt{\sum_{i=1}^{n+k} \left(k_{i,n+k}\tilde{x}_i - x_i\right)^2},$$

$$k_{i,n+k} = \frac{K\left(\frac{i-(n+k+1)}{h}\right)}{\sum_{j=1}^{n+k} K\left(\frac{j-(n+k+1)}{h}\right)} \quad i = 1, 2, \ldots, n+k, \tag{48}$$

Table 2
Estimation methods in DT-WSE.

DT-WSE		
Data transform	Anscome1 (A1) Bartlett1 (B1) Fisz1 (F1)	Anscome2 (A2) Bartlett2 (B2) Fisz2 (F2)
Thresholding rule	Soft rule (S)	Hard rule (H)
Threshold value	Universal threshold (ut) Level-dependent universal threshold (lut) Leave-out-half' cross-validation threshold (lht)	

Table 3
Estimation methods in NDT-WSE.

NDT-WSE		
Estimation method	Haar-wavelet shrinkage estimation (HS) Translation-invariant estimation (TI)	
Thresholding rule	Soft rule (S)	Hard rule (H)
Threshold value	Level-dependent threshold (ldt) Leave-out-half' cross-validation threshold (lht)	

where h (>0) is the bandwidth, and the function $K(\cdot)$ is the kernel function. In this paper we assume the following 4 kernel functions excluding the linear kernel $K(x) = x$;

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \quad \text{(Gaussian)}, \tag{49}$$

$$K(x) = \frac{3}{4}\left(1 - x^2\right) I(x) \quad \text{(Epanechnikov)}, \tag{50}$$

$$K(x) = \frac{70}{81}\left(1 - |x|^3\right)^3 I(x) \quad \text{(Tricube)}, \tag{51}$$

$$K(x) = \frac{1}{2} I(x) \quad \text{(Uniform)}, \tag{52}$$

$$I(x) = \begin{cases} 1 & (|x| \le 1) \\ 0 & (|x| > 1) \end{cases}. \tag{53}$$

In our pre-experiments, we investigated the dependency of the bandwidth in prediction and observed empirically that $h = 1.1$ is an appropriate choice. In the latter discussion, we assume $h = 1.1$.

Including the fundamental linear kernel function, we have 5 weighted loss functions and propose $12 \times 5 = 60$ prediction methods under one wavelet shrinkage estimation. Finally, we develop the predictive performance of our long-term prediction approaches, which consist of $44 \times 60 = 2640$ combinations for the prediction methods.

6. W-SRAT2

In the early paper (Wu et al., 2021) we developed W-SRAT, Wavelet-based Software Reliability Assessment Tool. In this paper, we release the new version W-SRAT2[1] by improving the prediction functionality of the cumulative number of software faults and the quantitative software reliability. W-SRAT2 is a web-based freeware without cumbersome installation and complicated deployment, and is a unique solution to support the wavelet-shrinkage estimation/prediction which is applicable to not only the software fault prediction but also some signal processing problems. It runs in a cloud computing environment and does not depend on the kind of operating system.

6.1. System architecture

The system architecture of W-SRAT2 is depicted in Fig. 1. It consists of two software components; the user interface (front end) written by HTML and the work end (back end) written by Python language. In the front end we prepared three human–computer interaction interfaces as follows.

[1] https://wsrat.wujingchi.com.

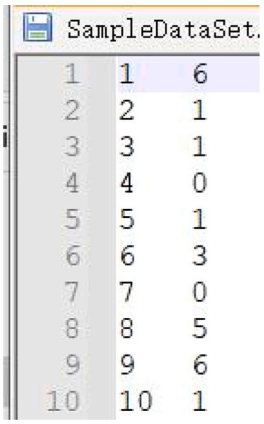


Fig. 3. Data format (screenshot).

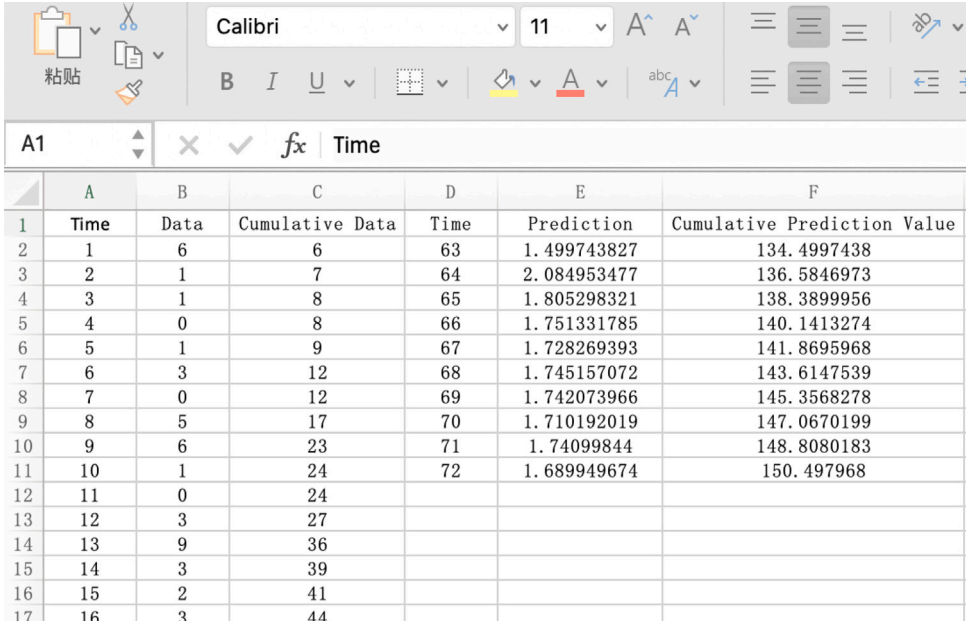
- `Data Upload Interface` - The data upload interface is responsible for uploading the software fault-count dataset and is used to send it to the 'Verification Module' in the back end. If the dataset is legal, then it is temporarily saved on a server and is used for denoising/prediction.
- `Denoising Interface` - The denoising interface is responsible for accepting denoising method input, where the 'Denoising/Prediction Module' in the back end shares the same interface for denoising method input.
- `Prediction Interface` - The prediction interface is responsible for receiving the prediction method and prediction term (length) from the current observation point and forwarding these information to the 'Prediction Module' in the back end.

In the back end, we process the data sent from the front end and output the Denoising Report/Prediction Report in the xlsx form, where three modules are used for the purposes.

- `Verification Module` - The 'Verification Module' is responsible for ensuring that the data uploaded by the user can be used for denoising/prediction. When the data is legal, it is temporarily saved on the server for the use of denoising/prediction.
- `Denoising Module` - The 'Denoising Module' is responsible for denoising the discrete intensity function based on the selected denoising method. The denoising result is saved in an xlsx file.
- `Prediction Module` - The 'Prediction Module' is responsible for predicting the discrete intensity function based on the selected denoising and prediction method with a given prediction term. The prediction result is saved in an xlsx file.

6.2. Data format

In W-SRAT2, we deal with only the group data, because almost all software fault-count data in actual development projects are observed in calendar time. Again, let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ be the number of software faults detected at testing time t_i and the cumulative value of software fault-counts, where $y_i = \sum_{j=1}^{i} x_j$. It can handle either of two kinds of data; $\boldsymbol{x}$ and $\boldsymbol{y}$. Fig. 3 presents the data format for $\boldsymbol{x}$, where the first column denotes the testing time (day, week, etc.), and the second column presents the number of software faults detected at each testing time. Once the data set was uploaded after selecting the data type, the format is checked with a verification function to ensure whether the data format is correct or not. In Fig. 2, we show the interface of W-SRAT2.



	A	B	C	D	E	F
1	Time	Data	Cumulative Data	Time	Prediction	Cumulative Prediction Value
2	1	6	6	63	1.499743827	134.4997438
3	2	1	7	64	2.084953477	136.5846973
4	3	1	8	65	1.805298321	138.3899956
5	4	0	8	66	1.751331785	140.1413274
6	5	1	9	67	1.728269393	141.8695968
7	6	3	12	68	1.745157072	143.6147539
8	7	0	12	69	1.742073966	145.3568278
9	8	5	17	70	1.710192019	147.0670199
10	9	6	23	71	1.74099844	148.8080183
11	10	1	24	72	1.689949674	150.497968
12	11	0	24			
13	12	3	27			
14	13	9	36			
15	14	3	39			
16	15	2	41			
17	16	3	44			

Fig. 4. Prediction report sample of W-SRAT2 (part 1/2) (screenshot).

Table 4
Comparison of the prediction methods based on $PMAE_1$.

Model name	DS1	DS2	DS3
	First testing phase		
Parametric (Best)	5.25 (gamma)	32.98 (exp)	7.17 (pareto)
Wavelet (Best)	**4.61** Method(2/Linear/M_3/lut/S/A1)	**21.40** Method(1/Gaussian/M_1/lht/S/B2)	**7.06** Method(2/Linear/M_2/lut/S/F2)
	Middle testing phase		
Parametric (Best)	3.59 (lxvmin)	46.16 (lxvmax)	10.09 (exp)
Wavelet (Best)	**3.05** Method(2/Linear/M_5/ldt/S/TI)	**7.62** Method(2/Gaussian/M_1/lht/S/B1)	**3.32** Method(2/Gaussian/M_6/ldt/S/HS)
	Last testing phase		
Parametric (Best)	11.40 (gamma)	6.76 (txvmin)	**1.51** (pareto)
Wavelet (Best)	**2.59** Method(1/Linear/M_5/lut/S/F2)	**3.34** Method(1/Gaussian/M_3/lht/S/F1)	1.52 Method(2/Gaussian/M_6/ldt/S/HS)
Model Name	DS4	DS5	DS6
	First testing phase		
Parametric (Best)	54.81 (lxvmax)	9.88 (lxvmax)	29.61 (gamma)
Wavelet (Best)	**7.94** Method(1/Gaussian/M_3/ldt/H/HS)	**4.36** Method(1/uniform/M_1/lut/S/A1)	**2.68** Method(2/Gaussian/M_1/ldt/H/TI)
	Middle testing phase		
Parametric (Best)	7.66 (gamma)	22.53 (lxvmax)	4.56 (gamma)
Wavelet (Best)	**7.51** Method(1/Gaussian/M_6/ldt/S/HS)	**4.70** Method(2/Linear/M_5/ut/S/A2)	**1.70** Method(2/Gaussian/M_1/ut/S/B1)
	Last testing phase		
Parametric (Best)	8.08 (lxvmax)	5.09 (lxvmax)	1.55 (txvmin)
Wavelet (Best)	**4.93** Method(2/Gaussian/M_3/lht/S/A1)	**3.26** Method(1/Gaussian/M_3/lut/S/A1)	**1.07** Method(1/Linear/M_3/lut/S/F2)

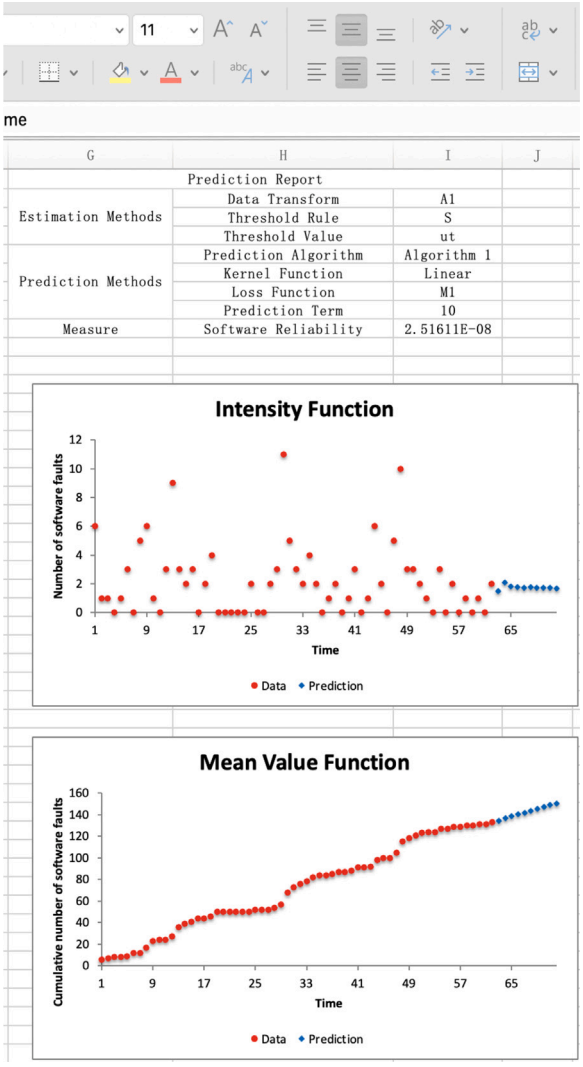


Fig. 5. Prediction report sample of W-SRAT2 (part 2/2) (screenshot).

6.3. Functionality

The discrete intensity estimation is the most fundamental function in W-SRAT2. First, we choose one of the estimation schemes; DT-WSE (Xiao and Dohi, 2013a) or NDT-WSE (Xiao and Dohi, 2013b). When DT-WSE is selected, we need to determine (i) data transformation method, (ii) thresholding rule, (iii) threshold value as a component of the estimation method. In NDT-WSE, instead of the data transform, we choose one of the estimation algorithms; Haar-wavelet shrinkage estimation (HS) or translation-invariant estimation (TI) (Xiao and Dohi, 2013b). Tables 2 and 3 summarize the estimation methods in DT-WSE and NDT-WSE, respectively. When the leave-out-half cross-validation threshold (lut) or the translation-invariant estimation (TI) is used in the discrete intensity estimation, it tends to take longer computation time, because the optimization procedure in 'lut' is more complex than the other thresholds. In contrast, in TI, the data set is expanded into n shift-data sets, which result in an n-times increase on the computation cost. Once the estimation method is selected, the discrete intensity function is estimated with the software fault-count data experienced in the past.

In `Denoising Module` of Fig. 1, the software intensity function is estimated by means of one of wavelet shrinkage estimation methods on all the past observation points. In this module we implement 44 estimation methods which consist of 36 DT-WSE and 8 NDT-WSE. The DT-WSE contains 6 data transform methods, 2 thresholding rules and 3 threshold values in Table 2. On one hand, NDT-WSE involves two patterns (H and TI) and 2 thresholding rules in Table 3. We plot not only the estimated discrete intensity function with the observation $\boldsymbol{x}$ and the mean value function with the cumulative observation $\boldsymbol{y}$ for better understanding the fault-count behavior. To investigate the goodness-of-fit of the NHPP-based SRGM with a selected estimation method, we calculate two kinds of mean squares errors (MSEs):

$$\mathrm{MSE}_1 = \frac{\sqrt{\sum_{i=1}^{n}\{\hat{\Lambda}(t_i) - y_i\}^2}}{n}, \tag{54}$$

$$\mathrm{MSE}_2 = \frac{\sqrt{\sum_{i=1}^{n}\{\Delta\hat{\Lambda}(t_i) - x_i\}^2}}{n}, \tag{55}$$

where n is the testing length. By checking the MSE value, the user can assess the goodness-of-fit of the NHPP-based SRGM under consideration, and may try to select a different combination of the system parameters so as to fit the underlying data. Note that the information criteria such as Akaike information criterion (AIC) (Akaike, 1998) does not work for our non-parametric approach.

In `Prediction Module` of Fig. 1, we predict the future value of the discrete intensity function with a given prediction time length l. For the estimation result based on one estimation method out of 44 methods, we apply one prediction algorithm out of 60 methods.

The commonly used metrics to assess the predictive performance are the predictive mean absolute errors (PMAEs):

$$\mathrm{PMAE}_1 = \frac{\sum_{i=n+1}^{n+l}|\hat{\Lambda}(t_i) - y_i|}{l}, \tag{56}$$

$$\mathrm{PMAE}_2 = \frac{\sum_{i=n+1}^{n+l}|\Delta\hat{\Lambda}(t_i) - x_i|}{l}, \tag{57}$$

where n is the data length, l is the prediction length, x_i (y_i) is the (cumulative) number of software faults. Because W-SRAT2 can be used to predict the number of software faults detected at an arbitrary time in the future, the ex-post facto evaluation on the predictive performance is not made on the tool. Based on our long-term prediction algorithms, we calculate $q_1^*, q_2^*, \ldots, q_l^*$ and return them as a series of prediction points in an excel sheet. The visual function is also used to plot the predictive behaviors of the discrete intensity function and the mean value function for the future. Finally, from Eq. (4), the quantitative software reliability $R_n(l) = \exp\{-(\Lambda(t_n + l) - \Lambda(t_n))\} = \exp\{-\sum_{i=n}^{n+l}\Delta\Lambda(t_i)\}$ is calculated with the observation point $t_n = n$ and the prediction length l.

In summary, we provide an actual example of a predictive report in Figs. 4 and 5. Fig. 4 presents the first half of the predictive report. Here, we depict the software fault data uploaded by the user along with the prediction number of software faults. Fig. 5 showcases the second half of the predictive report. This slide describes the prediction options selected by the user along with software reliability from release to prediction point. Moreover, it features the graphical representation of the intensity function and the mean value function.

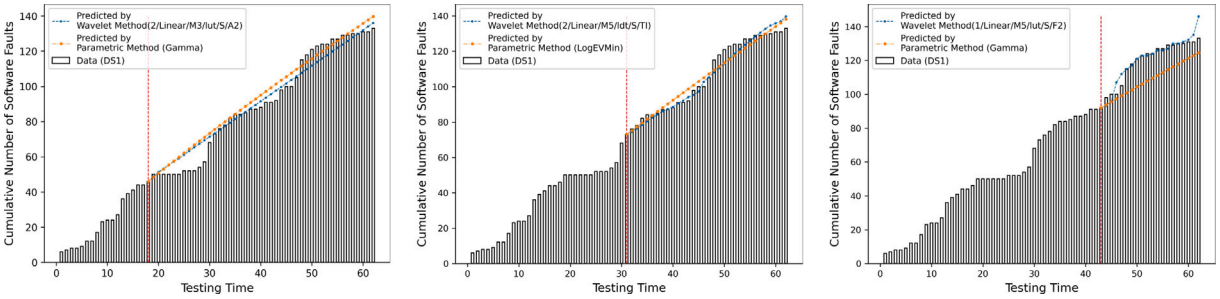
7. Experiments

7.1. Preliminary

We examine the prediction performance of all combinations of wavelet shrinkage prediction. More specifically, we investigate the predictive performance of 2640 prediction models on three phases of 6 data sets. For comparison with the representative parametric NHPP-based SRGMs, we assume 11 parametric models in Table 1, where the maximum likelihood estimation is applied with SRATS (Okamura and Dohi, 2013).

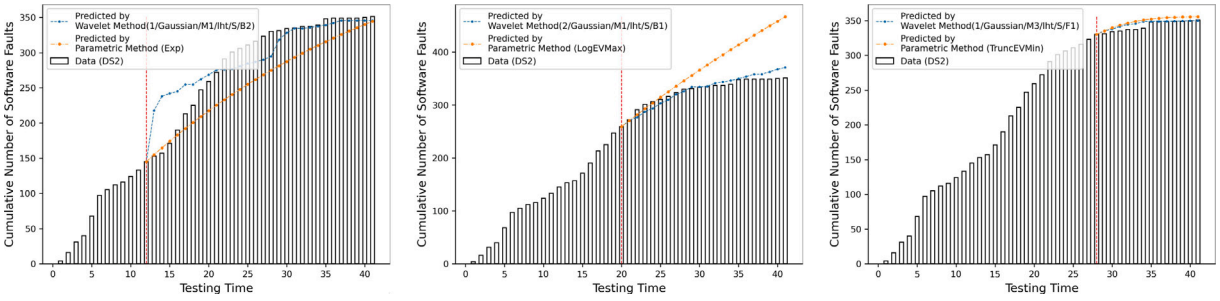
To clarify the prediction method via the wavelet shrinkage prediction, we introduce the notation Method(C1/C2/C3/C4/C5/C6), where

C1: algorithm (Algorithm 1, Algorithm 2),



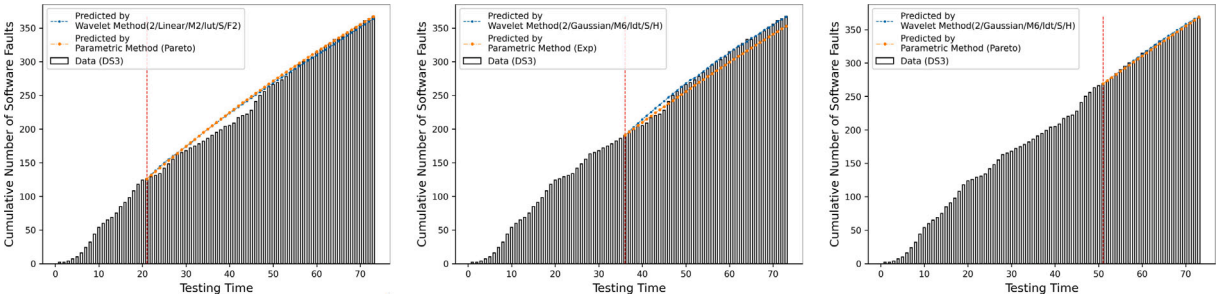
(a) Early testing phase (b) Middle testing phase (c) Late testing phase

Fig. 6. Prediction behaviors of the cumulative number of software faults in DS1.



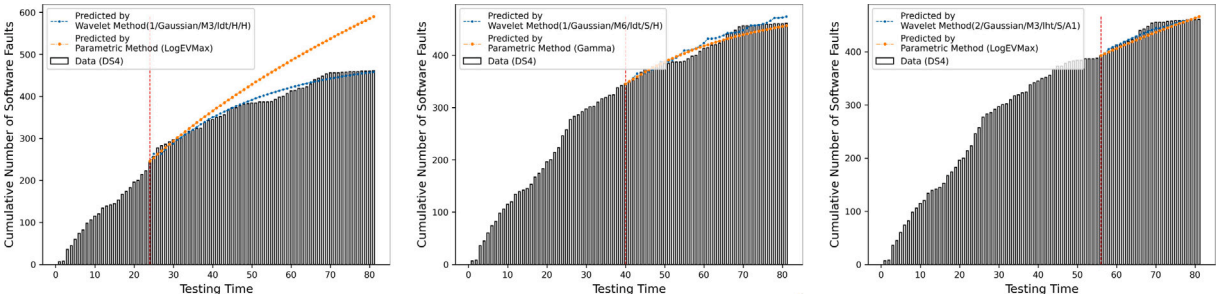
(a) Early testing phase (b) Middle testing phase (c) Late testing phase

Fig. 7. Prediction behaviors of the cumulative number of software faults in DS2.



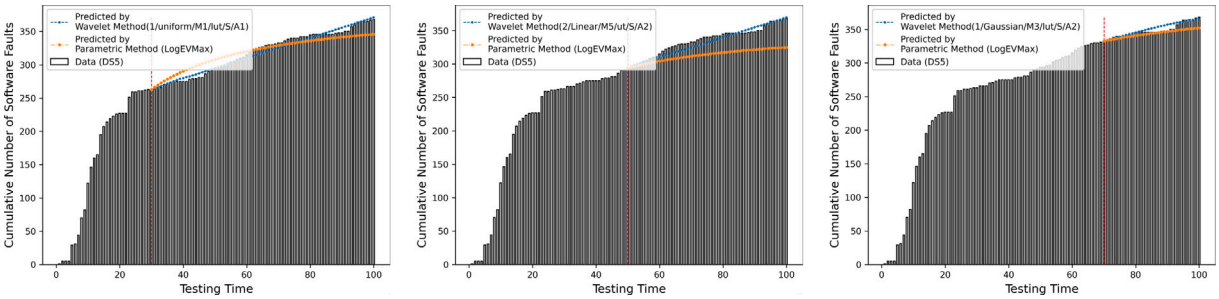
(a) Early testing phase (b) Middle testing phase (c) Late testing phase

Fig. 8. Prediction behaviors of the cumulative number of software faults in DS3.



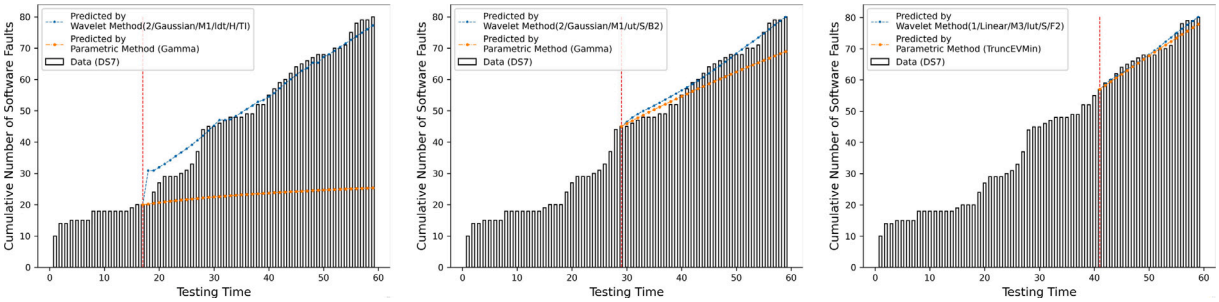
(a) Early testing phase (b) Middle testing phase (c) Late testing phase

Fig. 9. Prediction behaviors of the cumulative number of software faults in DS4.



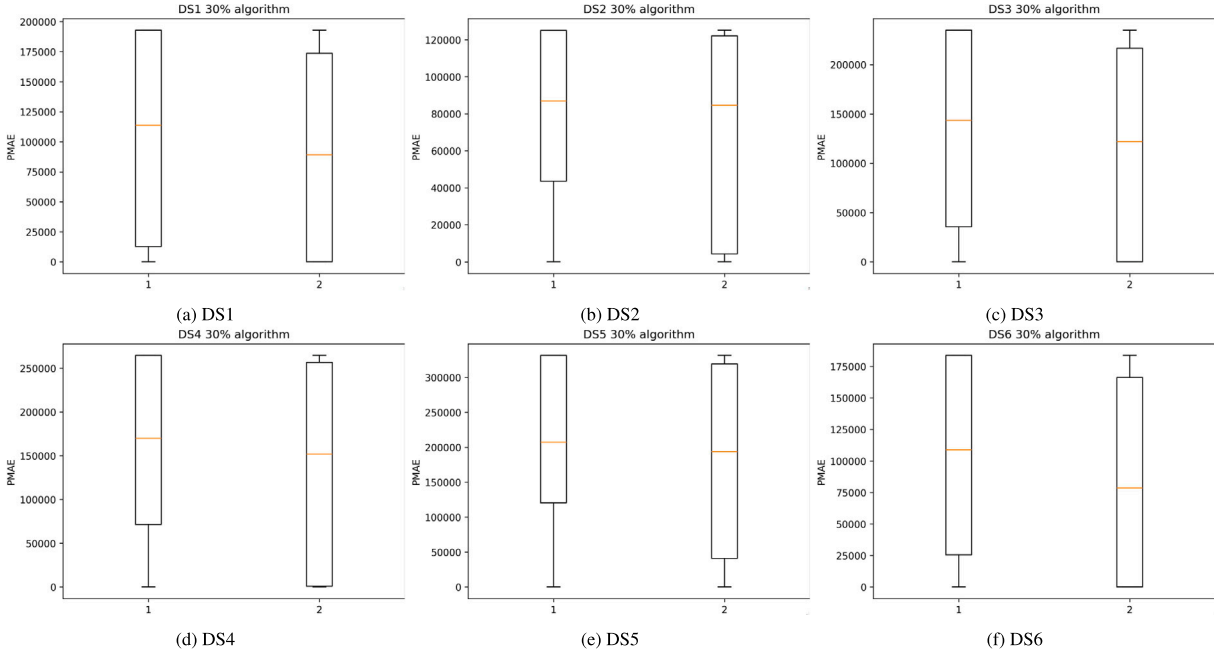
(a) Early testing phase (b) Middle testing phase (c) Late testing phase

Fig. 10. Prediction behaviors of the cumulative number of software faults in DS5.



(a) Early testing phase (b) Middle testing phase (c) Late testing phase

Fig. 11. Prediction behaviors of the cumulative number of software faults in DS6.



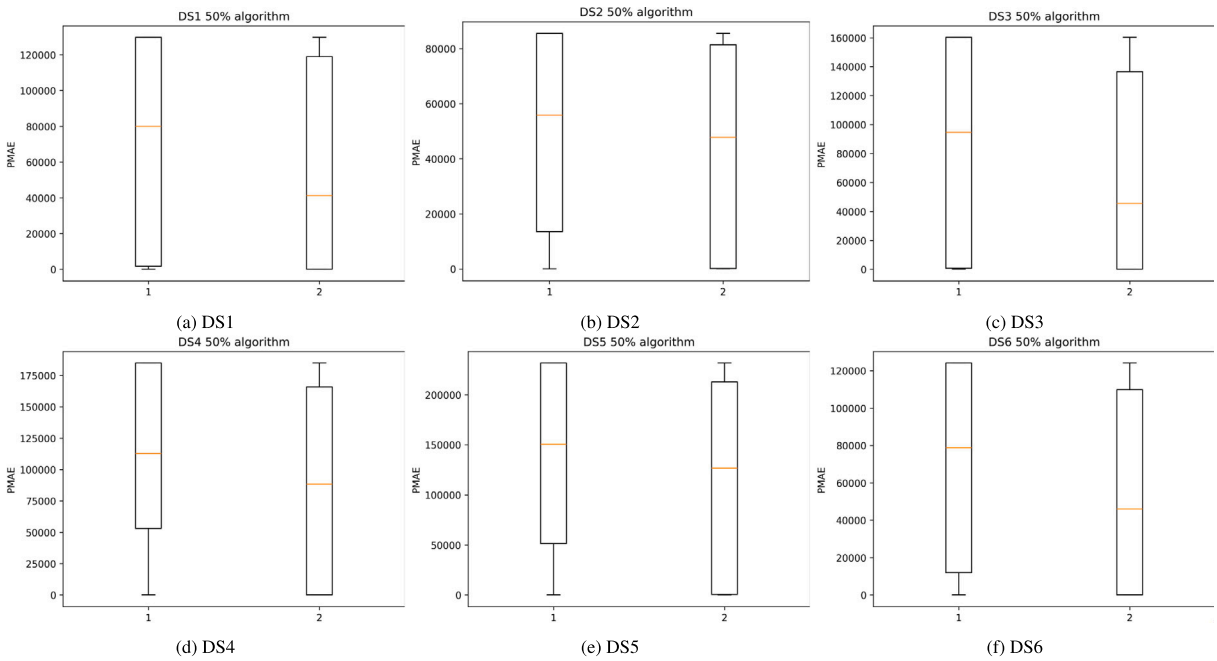
(a) DS1 (b) DS2 (c) DS3

(d) DS4 (e) DS5 (f) DS6

Fig. 12. Sensitivity analysis about prediction algorithm (C1) in early testing phase.

Table 5
Information of software fault-count data.

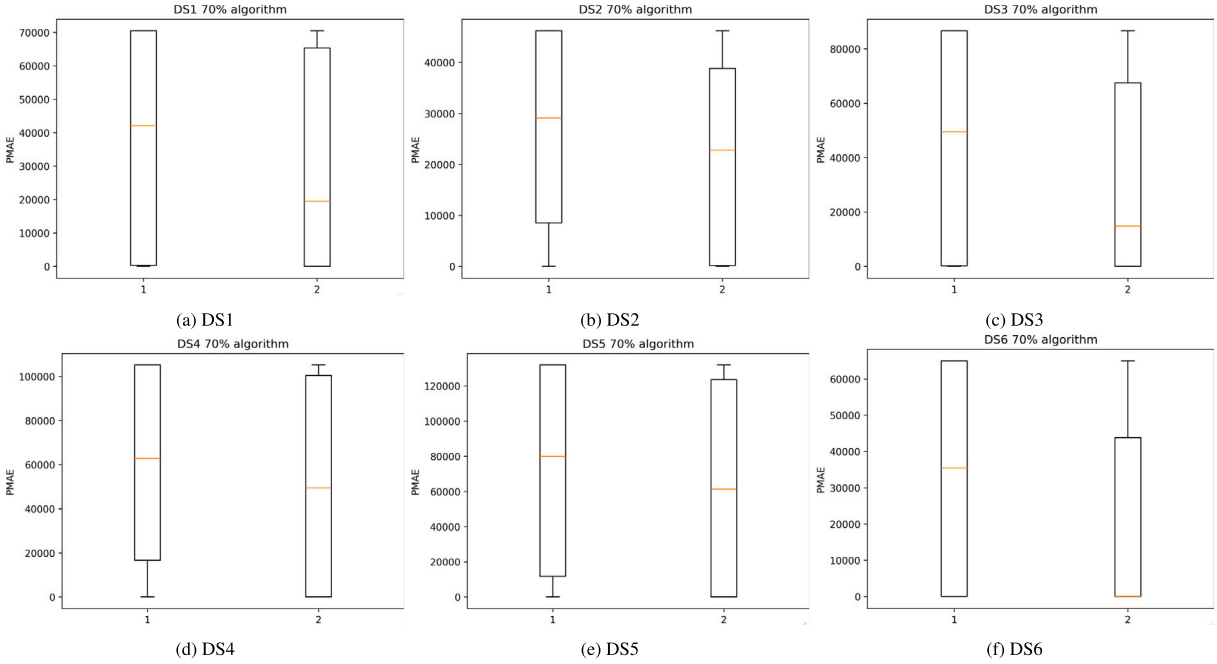
Data set	Application	Language	Period	Total No.
DS1 (Lyu, 1996)	Spacecraft system	Assembly	62 weeks	133
DS2 (Lyu, 1996)	Spacecraft system	HAL/S	41 weeks	351
DS3 (Lyu, 1996)	Spacecraft system	C, Fortran, EQUEI, OSL	73 weeks	367
DS4 (Lyu, 1996)	Brazilian electronic switching system	Assembly	81 days	461
DS5	Retro video game emulation for macOS	Objective-C	100 months	368
DS6	A web-based tool for Spriting and Pixel art	Javascript	59 months	80



(a) DS1 (b) DS2 (c) DS3

(d) DS4 (e) DS5 (f) DS6

Fig. 13. Sensitivity analysis about prediction algorithm (C1) in middle testing phase.



(a) DS1 (b) DS2 (c) DS3

(d) DS4 (e) DS5 (f) DS6

Fig. 14. Sensitivity analysis about prediction algorithm (C1) in late testing phase.

C2: kernel function (linear, Gaussian, Epanechnikov, tricube, uniform),

C3: loss function (M_1, M_2, M_3, M_4, M_5, M_6),

C4: threshold value (ut, lut, lht, ldt),

C5: thresholding rule (S, H),

C6: data transform (A1 in DT, A2 in DT, B1 in DT, B2 in DT, F1 in DT, F2 in DT, HS in NDT, TI in NDT).

7.2. Data sets

In this paper, we deal with the time series data. We use 6 actual software development project data sets. The data sets DS1∼ DS4 are referenced in the well-known literature (Lyu, 1996), where they were named as J1, J3, J5 and SS1. The data sets DS5 and DS6 come from GitHub.[2,3] All software fault-count data above are recorded as the group data, i.e., the number of software faults detected at each testing

[2] https://github.com/OpenEmu/OpenEmu, from Feb. 2012 to Sept. 2020.
[3] https://github.com/piskelapp/piskel/, from Sept. 2012 to Jul. 2018.

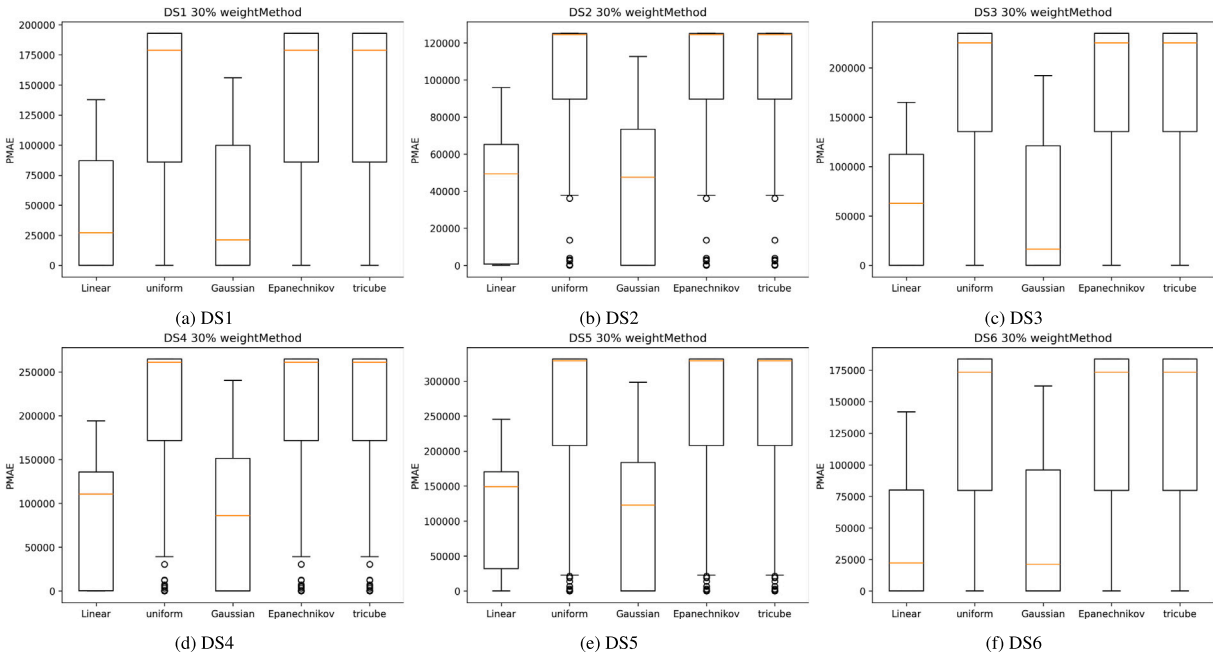


Fig. 15. Sensitivity analysis about prediction weight method (C2) in early testing phase.

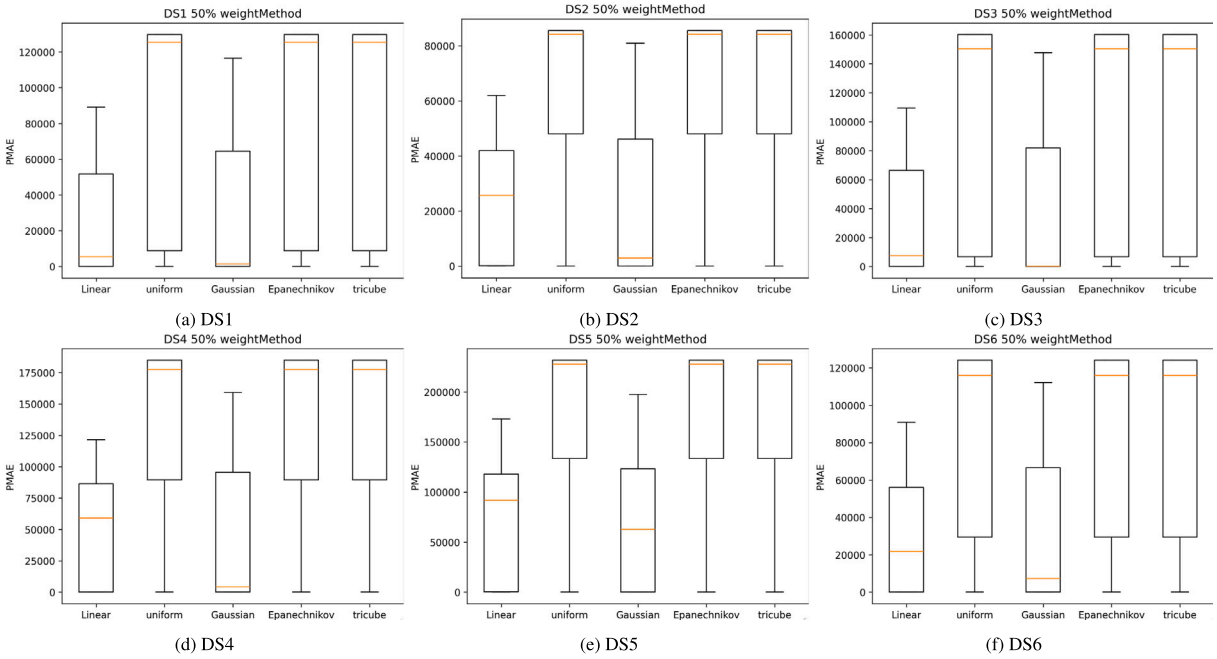


Fig. 16. Sensitivity analysis about prediction weight method (C2) in middle testing phase.

time (month). Table 5 summarizes the information of the software fault-count data used for the analysis.

To investigate the predictive performances, we divide the entire data into the training data and the validation data, where three cases with 30%, 50% and 70% of the entire data are considered for the training data, and are corresponding to the early testing phase, the middle testing phase and the late testing phase, respectively. Note that we do not draw random samples as the training data in our experiments. So, we make the prediction for the remaining periods of 70%, 50% and 30% in respective data sets.

7.3. Prediction

In this paper, we conducted $3 \times 6 = 18$ prediction experiments for each prediction method (i.e., a total of $2640 \times 18 = 47\,520$ prediction experiments) to thoroughly analyze the predictive performance of each method. Figs. 6–11 illustrate the prediction behaviors of the cumulative number of software faults in DS1∼DS6, where we plot the best prediction results by the parametric NHPP-based SRGMs and the wavelet shrinkage prediction methods. The horizontal axis represents time. The vertical axis represents the cumulative number of software faults

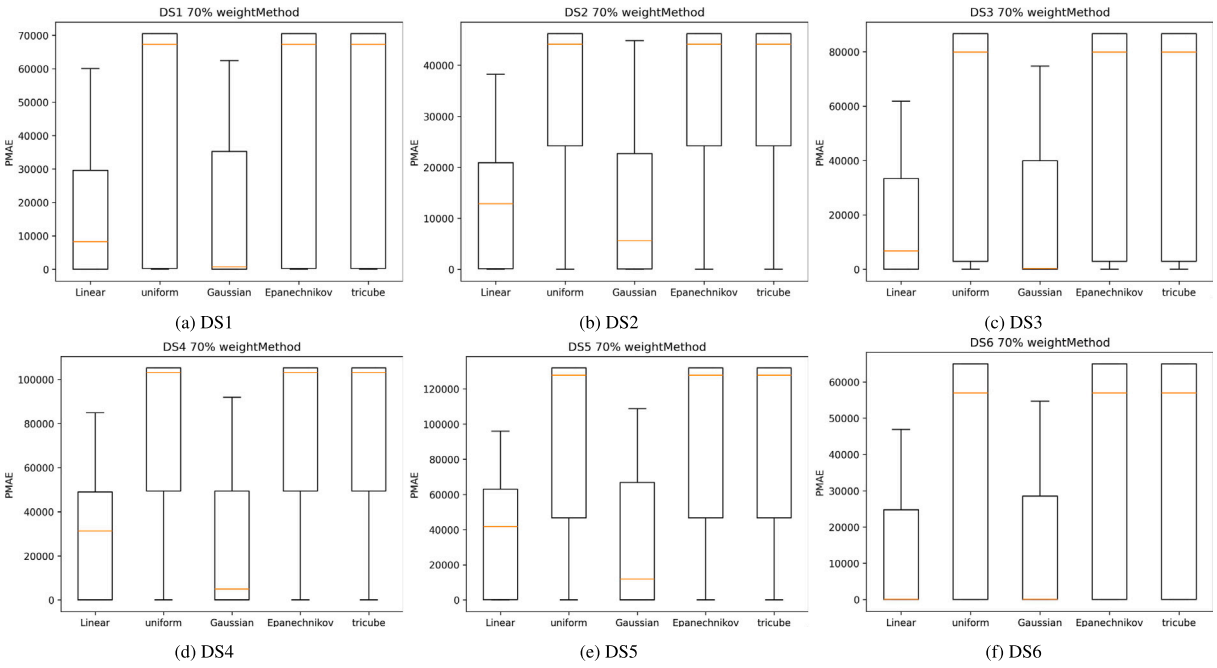


Fig. 17. Sensitivity analysis about prediction weight method (C2) in late testing phase.

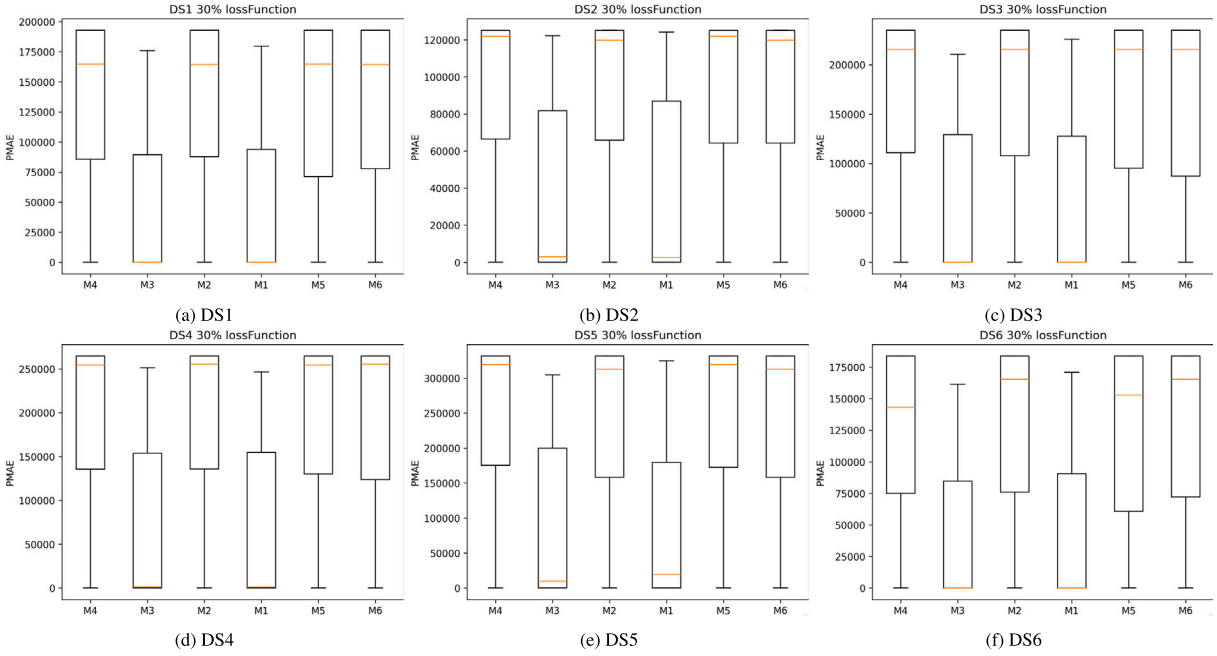


Fig. 18. Sensitivity analysis about prediction loss function (C3) in early testing phase.

detected. In the early testing phase, it is seen that the parametric SRGM with the gamma type c.d.f. indicated a flat prediction result and could not predict the future behavior of the cumulative number of software faults accurately. However, the wavelet method could catch the real data trend even in the remaining testing period. In the middle testing phase, we find that both parametric SRGM and the wavelet method showed the similar prediction, but as the prediction length becomes longer, the parametric SRGM was worse than the wavelet method. In the late testing phase, the predictive performances with both models were almost similar. In our experiments, we observed that when the future trend of the cumulative number of software faults rapidly grows, the parametric SRGM could not often catch up such a change point, because a deterministic mean value function was assumed in advance. On the other hand, it is seen that our wavelet shrinkage prediction could fit such data with a change point in the trend.

Next, we quantify the predictive performances of both the parametric SRGMs and the wavelet shrinkage methods. Table 4 presents the minimum $PMAE_1$ in three testing phases of all data sets, where we compared 11 parametric SRGMs and 2640 wavelet shrinkage prediction methods. In the Table 4, the red-colored value denotes the minimum $PMAE_1$ in each testing phase of each data set. Although the parametric SRGM with pareto c.d.f. gave a slightly smaller value than M(2/Gaussian/M_6/ldt/S/HS), it can be seen that the wavelet shrinkage prediction methods could provide the better prediction results in the remaining data sets. From the results in Table 4, we can conclude that

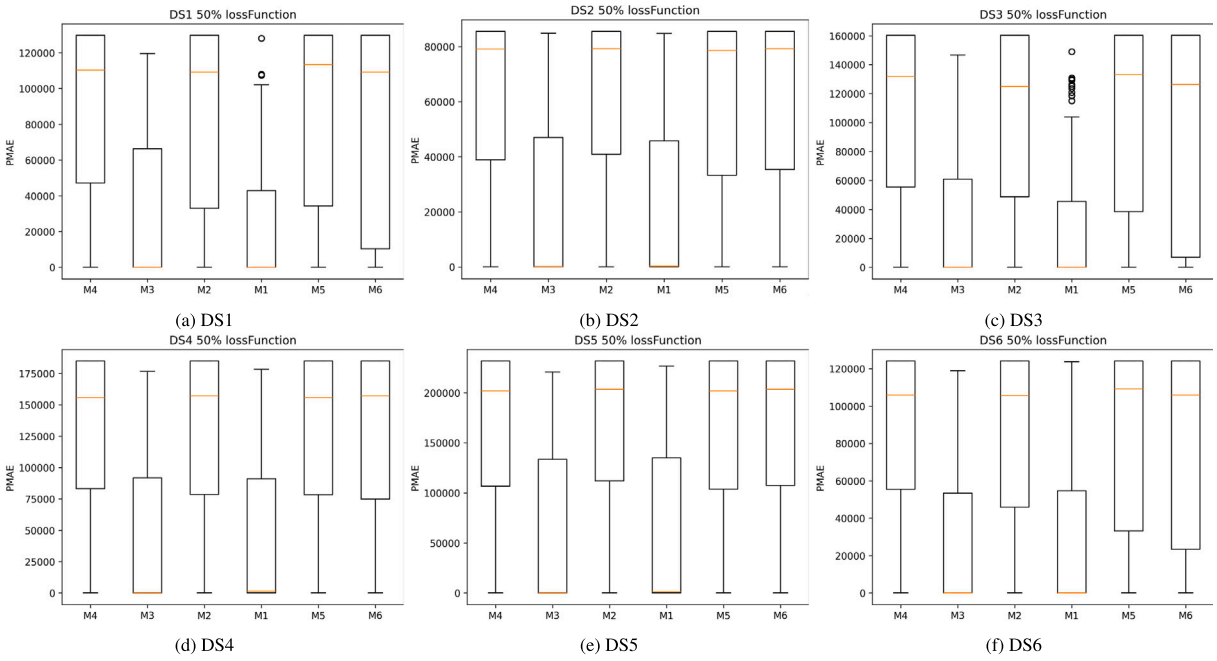


Fig. 19. Sensitivity analysis about prediction loss function (C3) in middle testing phase.

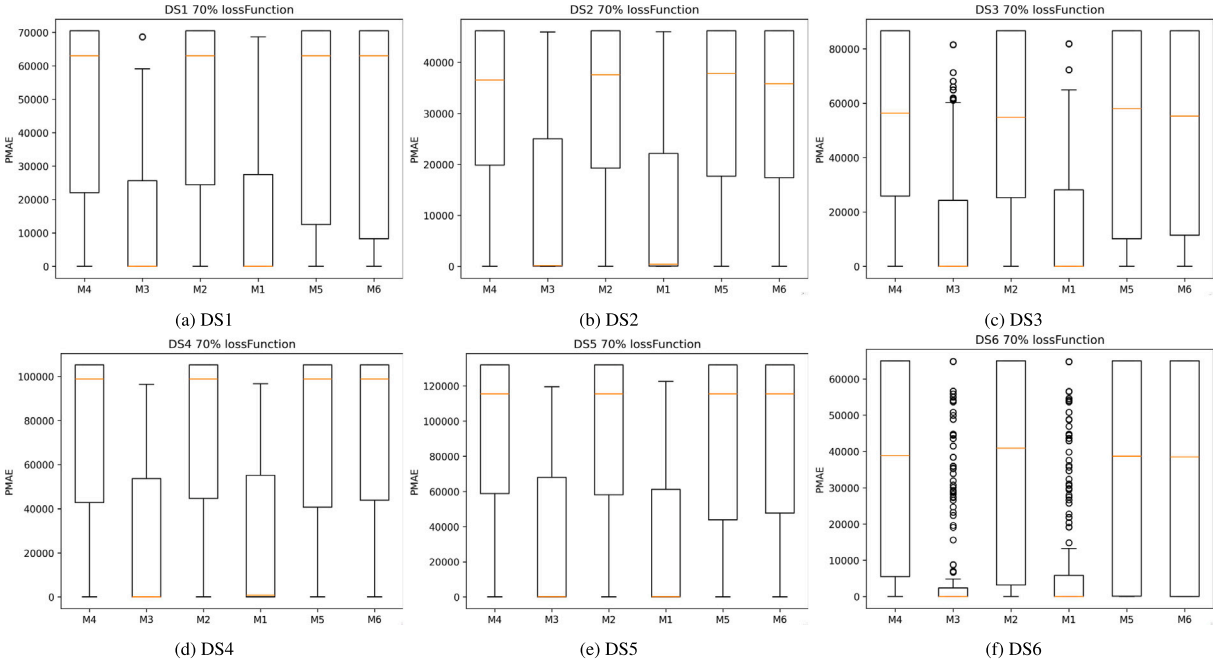


Fig. 20. Sensitivity analysis about prediction loss function (C3) in late testing phase.

the wavelet shrinkage methods have excellent potentials to make an accurate prediction of the cumulative number of software faults.

7.4. Ranking prediction methods

In Table 4 we compared the best prediction models in both parametric and non-parametric approaches. However, it is worth mentioning that the best model depends on the kind of data sets in each testing phase. Especially, it is quite hard to choose the best wavelet shrinkage prediction method from 2640 candidates. The one of our purpose in this paper is to investigate which combination methods in the wavelet approach are appropriate for choosing the better prediction models. The references in Xiao and Dohi (2013a,b) carefully examined the best goodness-of-fit methods in wavelet shrinkage estimation. On the other hand, we force on the prediction methods in terms of PMAE. It has been shown through Xiao and Dohi (2013a,b) and our numerical experiments that best goodness-of-fit model is not always equivalent to the best prediction model. This fact implies that our ranking results can provide a usefully information to predict the cumulative number of software faults by means of the wavelet approaches to software engineering practitioners. For the purpose of recommendation to practitioners, we rank all the parametric and non-parametric models in terms of the $PMAE_1$. Tables 6 and 7 present the ranking results based on the average $PMAE_1$, where the best NHPP-based SRGMs and the best wavelet shrinkage prediction methods are listed up in the ascending order.

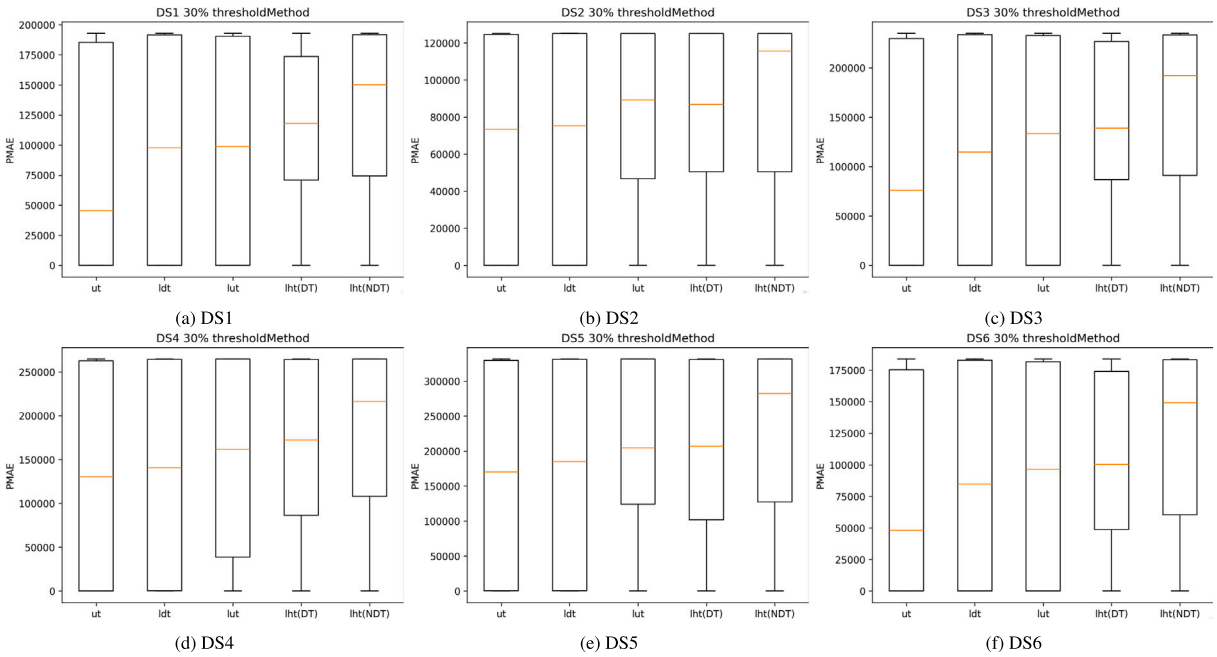


Fig. 21. Sensitivity analysis about prediction threshold method (C4) in early testing phase.

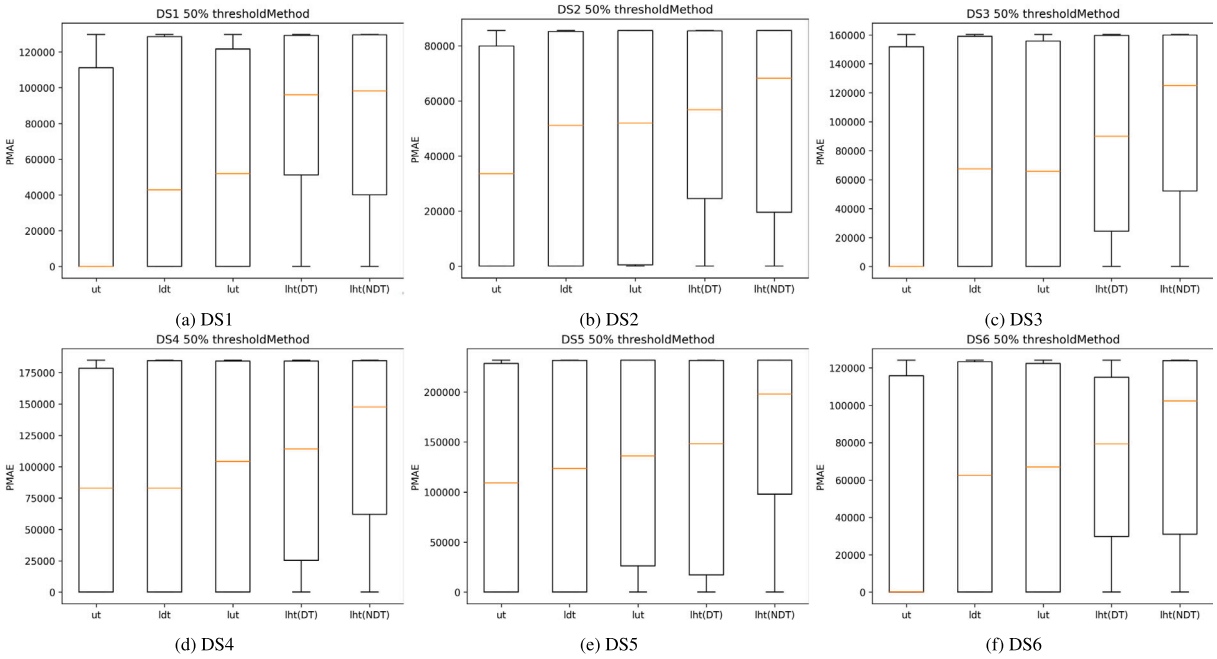


Fig. 22. Sensitivity analysis about prediction threshold method (C4) in middle testing phase.

From Table 6, it can be seen that pareto, exp and lxvmax provided the better predictive performances in both early and middle testing phases. On one hand, in the late testing phase, two extreme-value models, txvmin and lxvmax, gave the first and second places in the ranking. If one is interested in the average $PMAE_1$ to select an appropriate NHPP-based SRGM, then we strongly recommend to use the lxvmax type NHPP-based SRGM for the software fault prediction.

In Table 7 we rank the best 10 ranked wavelet shrinkage prediction methods out of 2640 methods by comparing the average $PMAE_1$, where the higher ranked wavelet shrinkage prediction methods are all based on the Gaussian kernel function in the weighted loss function. In the early and late testing phases, Algorithm 2 is better than Algorithm 1 in the top two ranked methods, but in the middle testing phase Algorithm 1 could work better in the top 10 ranked methods than Algorithm 2. In the variations of loss functions, only M_1 and M_3 gave the better results in average. On the wavelet shrinkage estimation method, it is found that the translation-invariant estimation (TI) in NDT-WSE provided the top ranked prediction results in all cases. In DT-WSE, the universal threshold 'ut' and the leave-out-half cross-validation threshold 'ldt' tended to give the better results. Comparing the average PMAEs in Table 6 with those in Table 7, we could recognize the significant difference on the average PMAEs. Hence, although it is troublesome to select the appropriate wavelet shrinkage prediction methods, our practical suggestion is to apply the top 5 ranked methods for the accurate predictions.

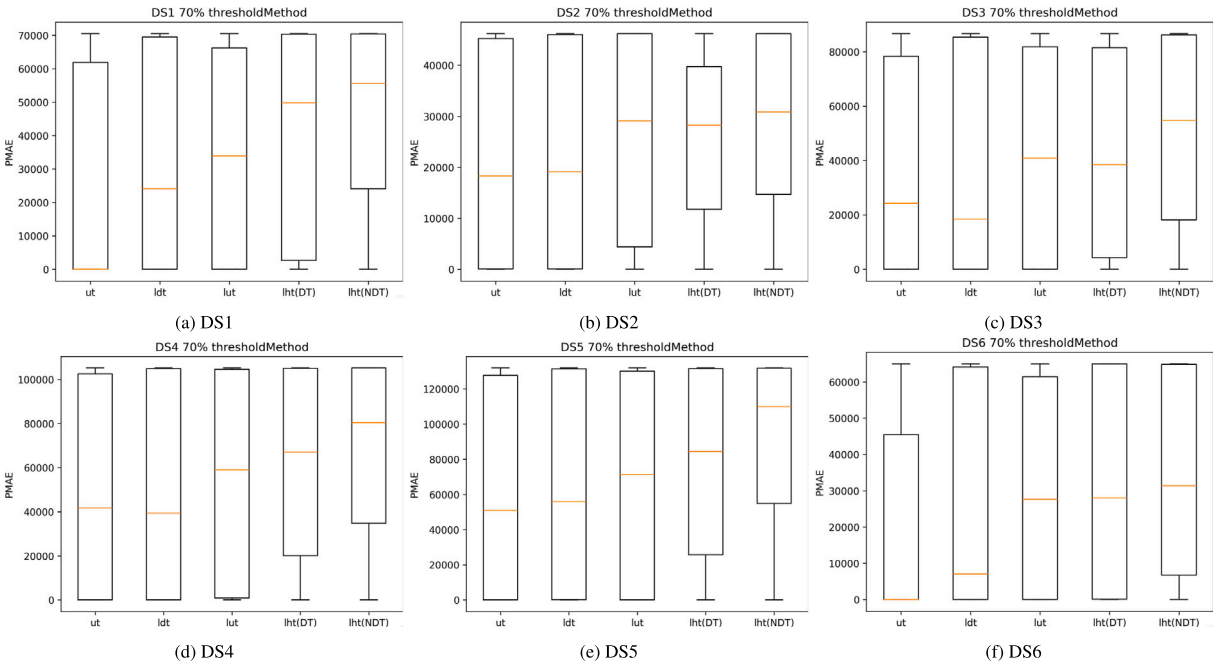


Fig. 23. Sensitivity analysis about prediction threshold method (C4) in late testing phase.

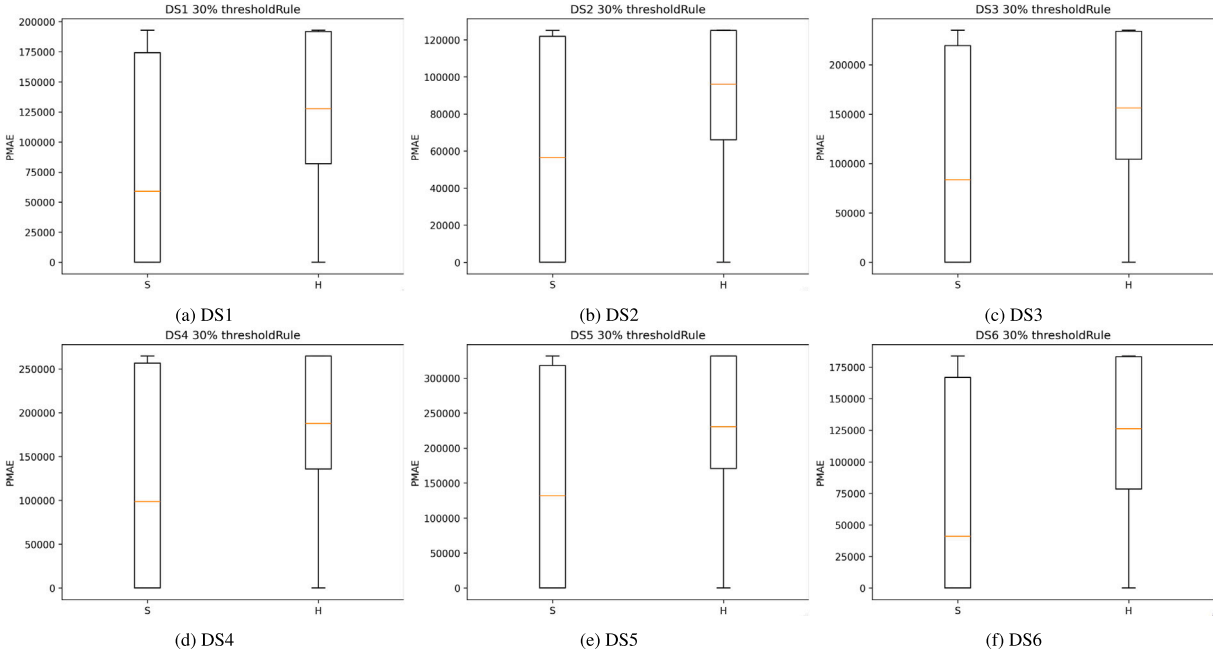


Fig. 24. Sensitivity analysis about prediction threshold rule (C5) in early testing phase.

7.5. Sensitivity analysis

As 7.3 section mentioned, we examine 2640 method in 18 prediction experiments for determining the best generalization performance method. In this sub-section, we are going to investigate how effect the prediction performance with different prediction method variations. From 7.1 section, it is clear that one our wavelet-based prediction method consists by 6 prediction method part : C1 ∼ C6. We fix one part of them and observer the PMAE distribution by changing other part.

Figs. 12–29. presents the box plot about different prediction method parts with 6 data set and 3 testing phases. It is seen that for one prediction method part, a specific prediction method option has similar PMAE distributions with different testing phases and data sets.

For C1 part, we can see that the different prediction algorithm shows similar prediction performance in sense of PMAE distribution across different testing phases and different data sets. For C2 part, Gaussian kernel-based weighted method and Linear-based weighted method show the better prediction performance. For C3 part, M1 and M3 loss function show the better prediction performance. For C4 part, lht shows worse PMAE distribution in both DT-based wavelet prediction and NDT-based wavelet prediction. the other threshold method show similar PMAE distribution. For C5 part, soft threshold rule show better PMAE distribution than hard threshold rule. For C6 part, the different

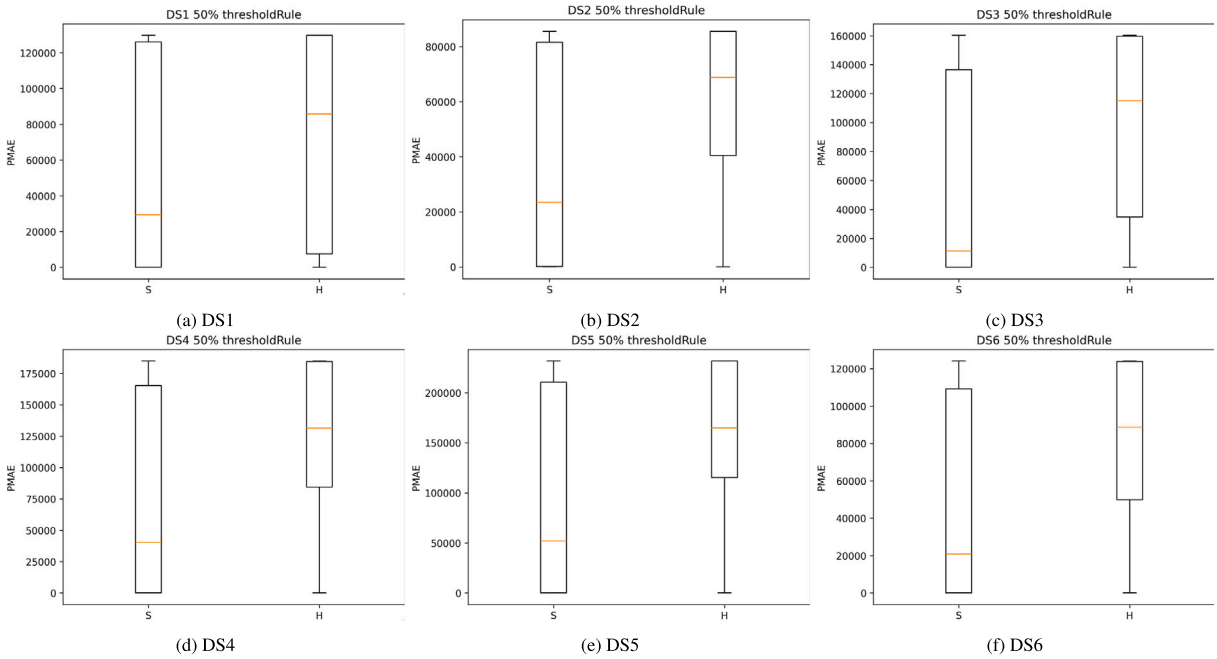


Fig. 25. Sensitivity analysis about prediction threshold rule (C5) in middle testing phase.

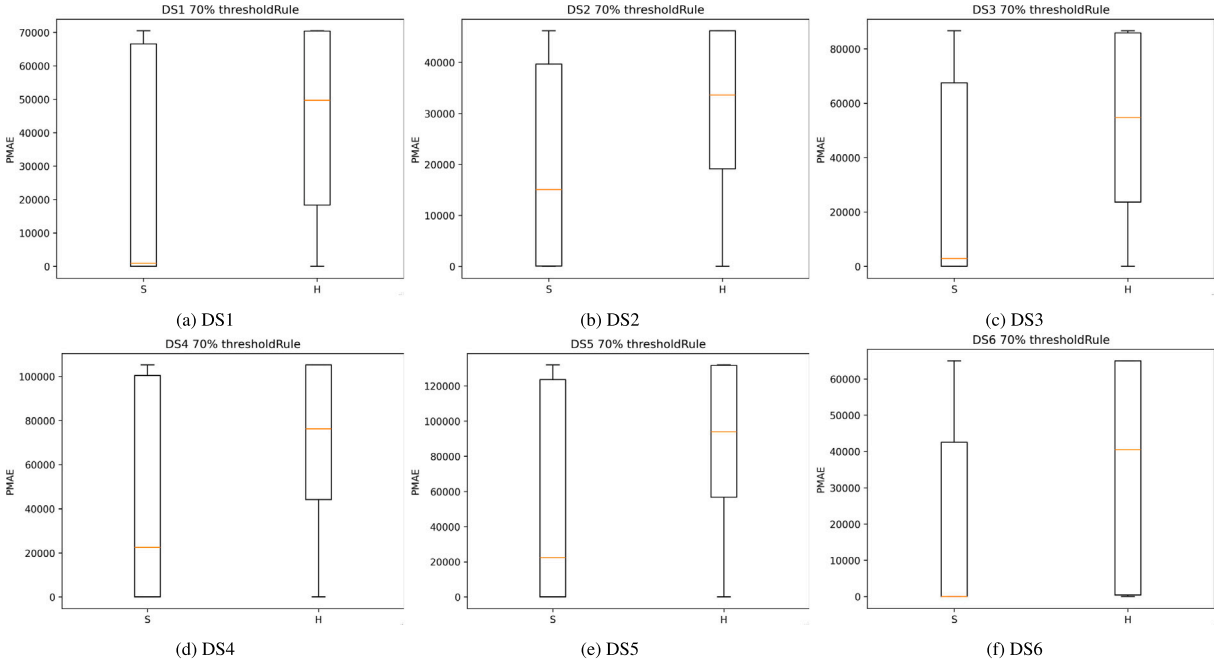


Fig. 26. Sensitivity analysis about prediction threshold rule (C5) in late testing phase.

data transform method show similar PMAE distribution in different data set and testing phases.

As a conclusion for sensitivity analysis, by selecting Gaussian kernel-based weighted method and Linear-based weighted method. M1 and M3 loss function can meaningfully improve the predictive accuracy of the wavelet prediction method. The prediction ranking in Section 7.4 also corroborated our conclusion.

7.6. Threats to validity

The degree of the validity of the experimental design is as important as the experimental results (Hyman, 1982). In this paper, we are concerned to the internal, external, and construct validity of the experiment, respectively.

Internal validity: Internal validity describes whether there are uncontrolled factors in the experiment that we have not considered. Collecting representative software fault data is an important threat to internal validity. We utilized 3 sets of software fault testing data from Jet Propulsion Laboratory (JPL) space station subsystems and a set of software data from Brazilian electronic switching system, TROPICO R-1500, Lyu (1996). In addition, considering the recent development of open-source software, we also used two sets of software failure data from well-known projects on GitHub.

Furthermore, the choice of minimization algorithm is also an important threat to internal validity. In this paper, our wavelet forecasting

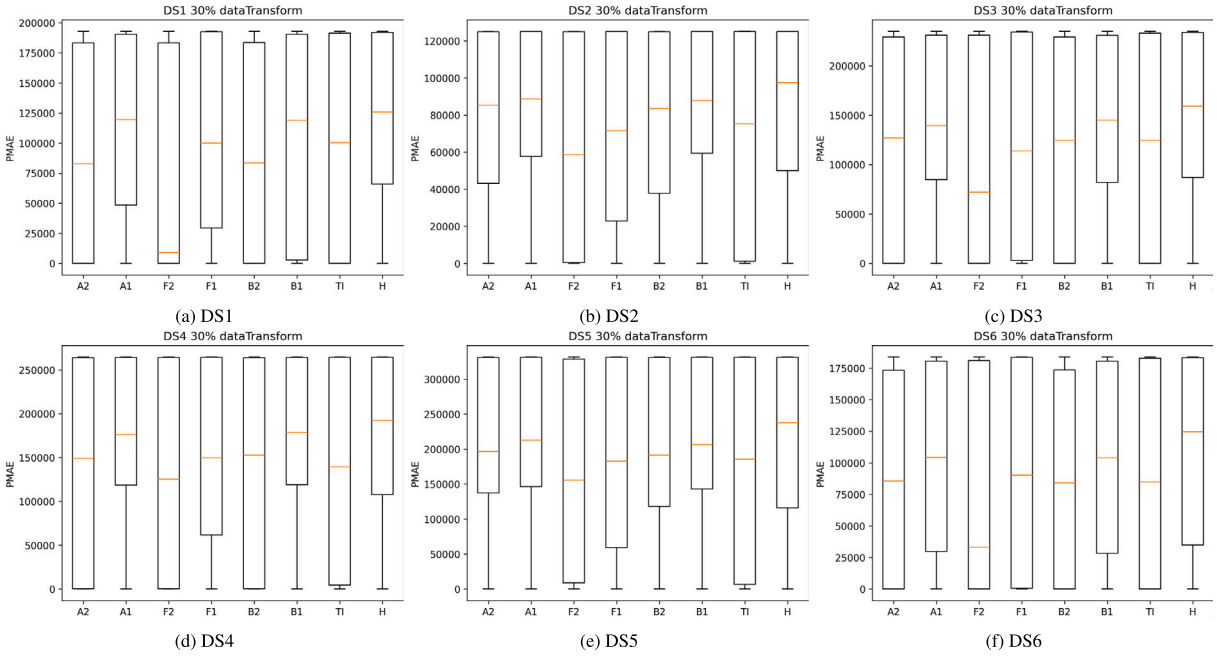


Fig. 27. Sensitivity analysis about prediction data transform (C6) in early testing phase.

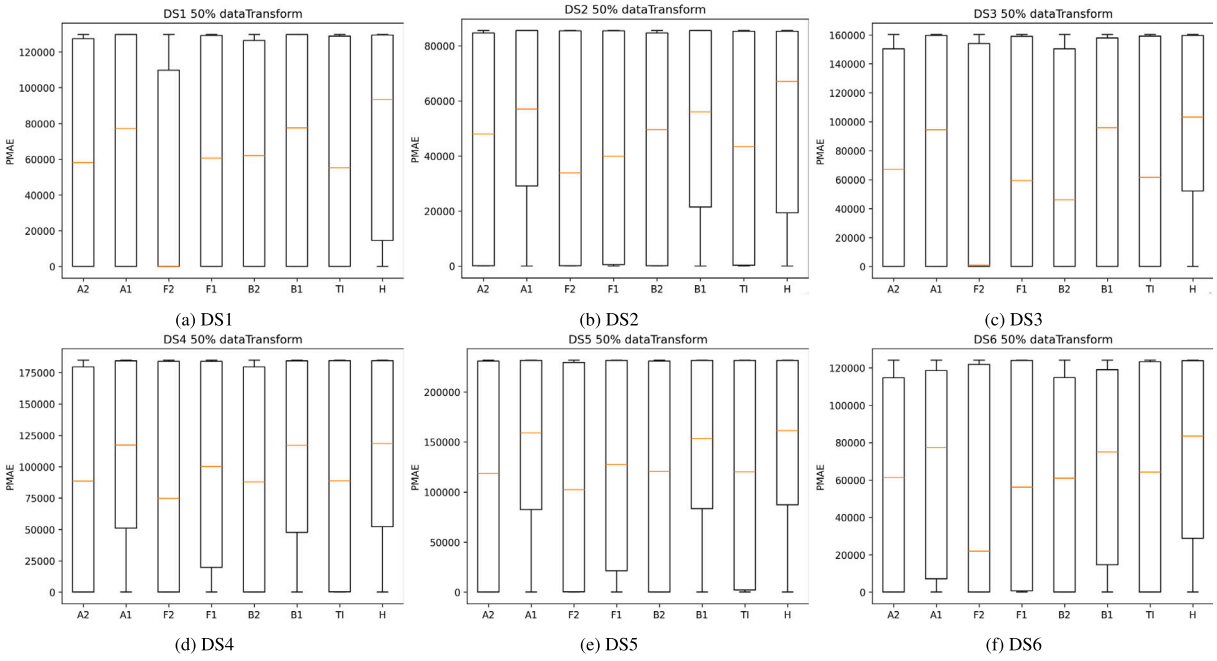


Fig. 28. Sensitivity analysis about prediction data transform (C6) in middle testing phase.

method applies Newton's method to solve the optimization problem. Here, the initial value is set to the number of software failures from the previous day, as we assume, without loss of generality, that the number of software faults detected on two consecutive days is close.

In addition to this, the traditional software reliability models we selected for comparison with the wavelet prediction method also constitute an aspect of the threat to internal validity. In our experiments, we applied 11 of the most representative SRGMs in the field of software reliability and fairly tested their predictive performance. In our experiments, the EM algorithm (Okamura et al., 2004) was used for parameter estimation of the parametric model. This algorithm has been implemented in the SRATS tool, which is widely used in the field of software reliability.

External validity: External validity discusses whether the same experimental results can be generalized to other situations or settings. Considering the recent development of open-source software, we also used two sets of software failure data from well-known projects on GitHub to improve the external validity of our experiments. The size of the dataset may also affect the experimental results. To address this issue, we applied six sets of software failure data of varying sizes.

Construct validity: Construct validity is the degree to which a test measures what it claims, or purports, to be measuring. To verify that our proposed model is able to predict the number of faults more accurately, we adopted PMAE as the evaluation metric that most intuitively demonstrates the accuracy of predictions. Although other metrics such as PMSE are also widely used in the field of software reliability,

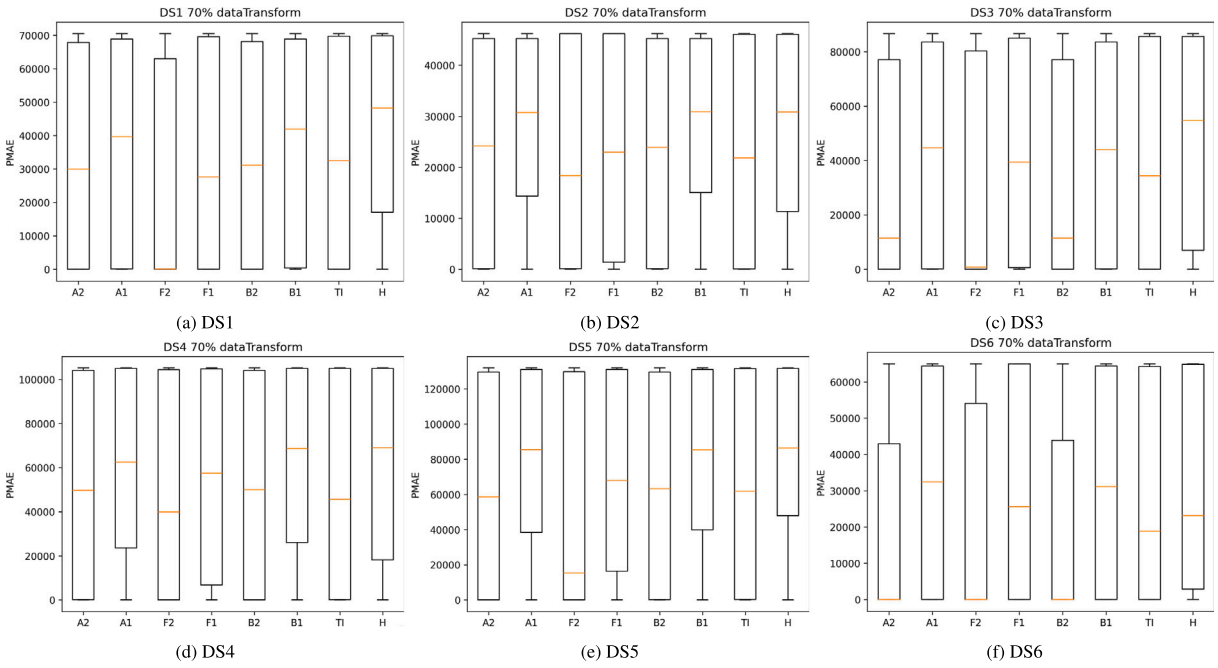


Fig. 29. Sensitivity analysis about prediction data transform (C6) in late testing phase.

Table 6
Ranking of NHPP-based SRGMs based on the average $PMAE_I$.

Early testing phase							
Model	Average $PMAE_I$	DS1	DS2	DS3	DS4	DS5	DS6
pareto	31.65	5.35	34.29	7.17	98.52	14.22	30.37
lxvmax	36.87	13.57	79.14	33.54	54.81	9.88	30.31
exp	40.19	7.10	32.98	27.75	126.70	13.82	32.79
lnorm	45.65	9.08	102.49	13.23	73.80	45.11	30.18
llogist	62.97	5.30	117.29	61.53	117.21	46.27	30.22
gamma	66.54	5.25	125.70	63.24	122.45	52.95	29.61
lxvmin	73.15	5.49	131.68	92.24	124.33	55.12	30.05
txvmax	77.69	14.91	133.37	89.58	142.70	52.77	32.80
tlogist	83.83	31.61	135.86	102.82	145.21	54.69	32.80
tnorm	85.45	32.63	138.08	107.99	145.48	55.65	32.84
txvmin	89.30	42.12	136.96	116.43	151.56	55.91	32.81
Middle testing phase							
Model	Average $PMAE_I$	DS1	DS2	DS3	DS4	DS5	DS6
pareto	21.66	3.63	55.77	14.90	22.52	27.57	5.60
exp	22.55	3.83	66.62	10.09	20.99	27.64	6.14
lxvmax	24.04	8.75	46.16	29.37	30.38	22.53	7.07
lnorm	28.90	6.21	54.71	44.19	25.53	36.55	6.18
llogist	31.08	3.60	71.87	58.96	10.25	37.05	4.75
gamma	31.98	3.60	72.47	63.60	7.66	40.00	4.56
lxvmin	33.99	3.59	74.90	70.63	10.03	40.06	4.70
txvmax	39.22	6.06	95.63	72.90	13.01	40.76	6.95
tlogist	41.28	6.80	99.17	75.18	18.85	40.31	7.35
tnorm	43.59	7.66	97.44	79.42	28.46	39.81	8.75
txvmin	46.02	7.39	98.77	81.87	41.50	38.45	8.13
Late testing phase							
Model	Average $PMAE_I$	DS1	DS2	DS3	DS4	DS5	DS6
txvmin	13.51	19.51	6.76	3.70	39.76	9.76	1.55
lxvmax	14.33	14.46	44.45	7.64	8.08	5.09	6.28
tnorm	14.96	16.87	19.10	5.80	36.65	9.57	1.79
tlogist	15.02	16.14	24.00	6.22	32.80	9.27	1.69
pareto	15.19	12.58	51.84	1.51	17.70	5.46	2.07
lnorm	15.71	13.22	43.11	9.45	12.90	9.91	5.65
exp	15.78	12.64	51.88	2.94	17.82	7.50	1.87
llogist	16.22	11.54	37.47	13.69	18.78	11.18	4.65
txvmax	16.55	15.35	29.76	11.35	30.32	10.83	1.72
lxvmin	16.81	11.47	31.29	12.69	29.30	11.50	4.62
gamma	17.05	11.40	34.21	13.45	26.33	12.39	4.55

however, considering the measurement of the average prediction error across different datasets, PMAE is a more appropriate choice.

8. Concluding remarks

In this paper, we have proposed several long-term prediction methods of the cumulative number of software faults detected in the testing phase via the wavelet shrinkage prediction. We have also implemented an automated wavelet-based software reliability assessment tool, W-SRAT2, which is a drastic extension of the existing tool, W-SRAT, by adding those prediction algorithms. In numerical experiments with 6 actual software development project data, we have investigated the predictive performance of our long-term prediction approaches, which consist of 2640 combinations, and compared them with the common software reliability growth models with the maximum likelihood estimation. It has been shown that our wavelet shrinkage estimation/prediction methods outperformed the existing software reliability growth models in almost all cases.

Note that the superiority of the wavelet methods was known in the goodness-of-fit in the estimation of the number of software faults in Xiao and Dohi (2013a,b). However, some views point out that, the wavelet shrinkage estimation may leads the overfitting to the particular datasets used in the study. In face, Xiao and Dohi (2013a,b) has discussed about the overfitting, where the threshold method with 'lht' showed the best goodness-of-fit. In Figs. 3∼4. Xiao and Dohi (2013a), it is evident to confirm the overfitting. On the other hand, in this paper, throughout our sensitivity analysis, it can be seen that the predictive performance of the threshold method with 'lht' and concluded that it has weaker predictive performance. Our lesson learned in this paper is that the wavelet-based approach is still useful in the prediction of the number of software faults. Further, through the empirical works with 2640 wavelet shrinkage prediction methods and 18 cases for 6 data sets, we have recommended a few prediction methods and shown that the common NHPP-based SRGM with parametric mean value functions did not work better than our wavelet shrinkage methods. Since we have already released W-SRAT2 as a freeware, the research community in software reliability engineering and the practitioners will confirm the predictive performances with different software development project data.

Table 7
Ranking of wavelet methods based on the average $PMAE_1$.

Early testing phase							
Model	Average $PMAE_1$	DS1	DS2	DS3	DS4	DS5	DS6
Method(2/Gaussian/M_3/ldt/H/TI)	24.62	6.78	86.68	14.93	24.37	10.49	4.48
Method(2/Gaussian/M_1/ldt/H/TI)	25.45	5.90	88.35	12.15	20.58	23.05	2.68
Method(1/Gaussian/M_1/ldt/S/TI)	31.93	16.54	89.33	25.37	18.48	13.12	28.73
Method(1/Gaussian/M_3/ut/S/F1)	32.06	17.18	86.40	36.45	7.97	15.84	28.50
Method(1/Gaussian/M_3/ut/S/F2)	32.06	17.18	86.40	36.45	7.97	15.84	28.50
Method(1/Gaussian/M_3/ut/H/F2)	32.06	17.18	86.40	36.45	7.97	15.84	28.50
Method(1/Gaussian/M_1/ut/S/A1)	33.17	18.11	59.63	21.05	25.49	45.57	29.18
Method(1/Gaussian/M_1/ut/S/A2)	33.17	18.11	59.63	21.05	25.49	45.57	29.18
Method(1/Gaussian/M_1/ut/S/B1)	33.38	17.72	59.54	20.96	25.67	47.39	28.99
Method(1/Gaussian/M_1/ut/S/B2)	33.38	17.72	59.54	20.96	25.67	47.39	28.99
Middle testing phase							
Model	Average $PMAE_1$	DS1	DS2	DS3	DS4	DS5	DS6
Method(1/Gaussian/M_3/ldt/S/TI)	17.87	14.21	17.02	41.02	8.06	23.83	3.09
Method(1/Gaussian/M_1/ut/S/B2)	18.39	15.50	26.98	31.13	22.87	8.40	5.44
Method(1/Gaussian/M_1/ut/H/B2)	18.39	15.50	26.98	31.13	22.87	8.40	5.44
Method(1/Gaussian/M_1/ut/S/B1)	18.39	15.50	26.98	31.13	22.87	8.40	5.44
Method(1/Gaussian/M_3/ut/S/F1)	18.61	8.83	27.92	29.05	28.64	13.84	3.36
Method(1/Gaussian/M_3/ut/S/F2)	18.61	8.83	27.92	29.05	28.64	13.84	3.36
Method(1/Gaussian/M_3/ut/H/F2)	18.61	8.83	27.92	29.05	28.64	13.84	3.36
Method(1/Gaussian/M_1/ut/S/A1)	18.62	15.99	26.92	31.22	22.63	9.27	5.71
Method(1/Gaussian/M_1/ut/S/A2)	18.62	15.99	26.92	31.22	22.63	9.27	5.71
Method(1/Gaussian/M_1/ut/H/A2)	18.62	15.99	26.92	31.22	22.63	9.27	5.71
Late testing phase							
Model	Average $PMAE_1$	DS1	DS2	DS3	DS4	DS5	DS6
Method(2/Gaussian/M_3/lht/S/TI)	12.89	21.70	7.14	34.42	7.49	4.19	2.38
Method(2/Gaussian/M_3/ldt/H/TI)	12.94	12.88	25.26	6.65	22.98	8.55	1.35
Method(1/Gaussian/M_3/ldt/H/TI)	13.51	17.51	12.72	17.54	23.57	6.50	3.24
Method(2/Gaussian/M_1/lht/S/TI)	13.72	21.66	12.64	34.22	6.89	4.49	2.41
Method(2/Gaussian/M_1/ldt/H/TI)	13.94	12.79	33.61	6.20	21.95	7.93	1.19
Method(2/Gaussian/M_3/ldt/S/TI)	14.13	13.02	46.33	2.33	13.36	8.56	1.20
Method(1/Gaussian/M_3/lut/S/F1)	14.14	17.14	16.10	13.36	28.92	4.91	4.40
Method(1/Gaussian/M_3/lut/S/F2)	14.14	17.14	16.10	13.36	28.92	4.91	4.40
Method(2/Gaussian/M_3/lut/S/F2)	14.32	11.17	32.70	8.41	18.13	14.30	1.17
Method(2/Gaussian/M_3/lut/S/F1)	14.32	11.17	32.70	8.41	18.13	14.30	1.17

For scenarios where software failure data exhibit highly irregular behavior, or where the fault discovery rate suddenly changes due to external factors (such as significant code restructuring or the introduction of new technologies), it is often necessary to determine whether there is a change point in the data. In future, we are going to propose an change point monitor-based long-term prediction method. We also have a plan to open the source code of W-SRAT2 in GitHub for the practical use. We will generalize the wavelet shrinkage estimation/prediction methods to apply more informative data with software metrics.

CRediT authorship contribution statement

Jingchi Wu: Writing – review & editing, Writing – original draft, Visualization, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Tadashi Dohi:** Validation, Supervision. **Hiroyuki Okamura:** Validation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

This work was supported by JST, the establishment of university fellowships towards the creation of science technology innovation, Grant Number JPMJFS2129.

References

Abdel-Ghaly, A.A., Chan, P.Y., Littlewood, B., 1986. Evaluation of competing software reliability predictions. IEEE Trans. Softw. Eng. SE-12, 950—967.
Achcar, J.A., Dey, D.K., Niverthi, M., 1998. A Bayesian approach using nonhomogeneous Poisson processes for software reliability models. Front. Reliab. 1—18, World Scientific.
Akaike, Hirotogu, 1998. Information theory and an extension of the maximum likelihood principle. In: Selected Papers of Hirotugu Akaike. Springer New York, New York, NY, pp. 199–213.
Anscombe, F.J., 1948. The transformation of Poisson, binomial and negative-binomial data. Biometrika 35 (3/4), 246–254.
Barghout, M., Littlewood, B., Abdel-Ghaly, A., 1998. A non-parametric order statistics software reliability model. Softw. Test. Verif. Reliab. 8 (3), 113–132.
Bartlett, M.S., 1936. The square root transformation in analysis of variance. J. R. Stat. Soc. 3 (1), 68–78.
Bartlett, M.S., 1947. The use of transformations. Biometrics 3 (1), 39–52.
Carrozza, G., Pietrantuono, R., Russo, S., 2018. A software quality framework for large-scale mission-critical systems engineering. Inf. Softw. Technol. 102, 100–116.
Cinque, M., Cotroneo, D., Pecchia, A., Pietrantuono, R., Russo, S., 2017. Debugging-workflow-aware software reliability growth analysis. Softw. Test. Verif. Reliab. 27 (7), e1638.
Daubechies, I., 1992. Ten Lectures on Wavelets. SIAM.
Dohi, T., Zheng, J., Okamura, H., 2020. Data-driven software reliability evaluation under incomplete knowledge on fault count distribution. Qual. Eng. 32 (3), 421–433.
Donoho, D.L., Johnstone, J.M., 1994. Ideal spatial adaptation by wavelet shrinkage. Biometrika 81 (3), 425–455.
Farr, W., 1993. Statistical modeling and estimation of reliability functions for software (SMERFS) user's guide. Revision 3, PN.

Fisz, M., 1955. The limiting distribution of a function of two independent random variables and its statistical application. Colloq. Math. 3, 138–146.
Goel, A.L., 1985. Software reliability models: Assumptions, limitations, and applicability. IEEE Trans. Softw. Eng. 12, 1411–1423.
Goel, A.L., Okumoto, K., 1979. Time-dependent error-detection rate model for software reliability and other performance measures. IEEE Trans. Reliab. R-28, 206—211.
Gokhale, S.S., Trivedi, K.S., 1998. Log–logistic software reliability growth model. In: Proceedings of the Third IEEE International High-Assurance Systems Engineering Symposium. HASE-1998, IEEE CPS, pp. 34—41.
Huang, C.-Y., Lyu, M., 2011. Estimation and analysis of some generalized multiple change-point software reliability models. IEEE Trans. Reliab. 60 (2), 498—514.
Hudepohl, J., Aud, S., Khoshgoftaar, T., Allen, E., Mayrand, J., 1996. Emerald: Software metrics and models on the desktop. IEEE Softw. 13 (5), 56–60.
Hyman, R., 1982. Quasi-experimentation: Design and analysis issues for field settings. J. Pers. Assess. 46 (1), 96–97.
Kanoun, K., Kaaniche, M., Laprie, J., Metge, S., 1993. Sorel: A tool for reliability growth analysis and prediction from statistical failure data. In: Proceeding of the 23rd IEEE International Symposium on Fault-Tolerant Computing. FTCS-23, pp. 654–659.
Kolaczyk, E.D., 1997. Nonparametric estimation of gamma-ray burst intensities using haar wavelets. Astrophys. J. 483 (1), 340–349.
Kuo, L., Yang, T.Y., 1996. Bayesian computation for nonhomogeneous Poisson processes in software reliability. J. Am. Stat. Assoc. 91, 763–773.
Li, N., Malaiya, Y., 1995. ROBUST: A next generation software reliability engineering tool. In: Proceedings of the 6th International Symposium on Software Reliability Engineering. ISSRE-1995, IEEE CPS, pp. 375–380.
Lyu, M. (Ed.), 1996. Handbook of Software Reliability Engineering. McGraw-Hill.
Lyu, M., Nikora, A., 1992. CASRE: A computer-aided software reliability estimation tool. In: IEEE International Workshop on Computer-Aided Software Engineering. IEEE CPS, pp. 264–275.
Nagaaraju, V., Shekar, V., Steakelum, J., Luperon, M., Shi, Y., Fiondella, L., 2019. Practical software reliability engineering with the software failure and reliability assessment tool (SFRAT). SoftwareX 10, 100357.
Nason, G.P., 1996. Wavelet shrinkage using cross-validation. J. R. Stat. Soc.: Ser. B (Methodol.) 58 (2), 463–479.
Ohba, M., 1984. Inflection S-shaped software reliability growth model. In: Stochastic Models in Reliability Theory. Springer, Berlin/Heidelberg, pp. 144—162.
Ohishi, K., Okamura, H., Dohi, T., 2009. Gompertz software reliability model: Estimation algorithm and empirical validation. J. Syst. Softw. 82, 535–543.
Okamura, H., Dohi, T., 2009. A Bayesian inference tool for NHPP-based software reliability assessment. In: Future Generation Information Technology 2009. FGIT 2009, In: Lecture Notes in Computer Science, vol. 5899, Springer-Verlag, pp. 226–237.
Okamura, H., Dohi, T., 2013. SRATS: Software reliability assessment tool on spreadsheet. In: Proceedings of the 24th International Symposium on Software Reliability Engineering. ISSRE-2013, IEEE CPS, pp. 100–107.
Okamura, H., Dohi, T., Osaki, S., 2013. Software reliability growth models with normal failure time distributions. Reliab. Eng. Syst. Saf. 116, 135–141.
Okamura, H., Murayama, A., Dohi, T., 2004. EM algorithm for discrete software reliability models: A unified parameter estimation method. In: Proceedings of the 8th IEEE International Symposium on High Assurance Systems Engineering. HASE-2004, IEEE CPS, pp. 219–228.
Okamura, H., Murayama, A., Dohi, T., 2005. A unified parameter estimation algorithm for discrete software reliability models. Opsearch 42 (4), 355–377.
Okamura, Hiroyuki, Watanabe, Yasuhiro, Dohi, Tadashi, 2002. Estimating mixed software reliability models based on the EM algorithm. In: Proceedings International Symposium on Empirical Software Engineering. IEEE.
Ramani, S., Gokhale, S., Trivedi, K., 2000. SREPT: Software reliability estimation and prediction tool. Perform. Eval. 39 (1/4), 37–60.
1988. Software Reliability Modeling Programs, Version 1.0. Technical Report, Reliability and Statistical Consultants Ltd..
Saito, Y., Dohi, T., 2015. Software reliability assessment via non-parametric maximum likelihood estimation. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. (A) E98-A (10), 2042–2050.
Saito, Y., Dohi, T., 2016. Predicting software reliability via completely monotone nonparametric estimator with grouped data. J. Syst. Softw. 117, 296–306.
Shibata, K., Rinsaka, K., Dohi, T., 2007. PISRAT: Proportional intensity-based software reliability assessment tool. In: Proceedings of 13th Pacific Rim International Symposium on Dependable Computing. PRDC 2007, IEEE CPS, pp. 43–52.
Shibata, K., Rinsaka, K., Dohi, T., 2015. M-SRAT: Metrics-based software reliability assessment tool. Int. J. Perform. Eng. 11 (4), 369–379.
Sofer, A., Miller, D.R., 1991. A nonparametric software-reliability growth model. IEEE Trans. Reliab. 40 (3), 329–337.
1990. Draft Software Reliability Engineering: Reliability Estimation Tools Reference Guide, Version 3.7. Technical Report, AT & T Bell Laboratories.
Wu, J., Dohi, T., Okamura, H., 2021. W-SRAT: Wavelet-based software reliability assessment tool. In: Proceedings of the 21st IEEE International Conference on Software Quality, Reliability, and Security. QRS 2021, IEEE CPS, pp. 564–573.
Xiao, X., Dohi, T., 2013a. Wavelet shrinkage estimation for non-homogeneous Poisson process based software reliability models. IEEE Trans. Reliab. 62 (1), 211–225.
Xiao, X., Dohi, T., 2013b. Estimating software intensity function based on translation-invariant Poisson smoothing approach. IEEE Trans. Reliab. 62 (4), 930–945.
Yamada, S., Ohba, M., Osaki, S., 1983. S-shaped reliability growth modeling for software error detection. IEEE Trans. Reliab. R-32, 475–484.
Yamada, S., Osaki, S., 1985. Discrete software reliability growth models. Appl. Stoch. Models Data Anal. 1 (1), 65–77.
Yamada, S., Osaki, S., Narihisa, H., 1984. Software reliability growth modeling with number of test runs. IEICE Trans. E67 (2), 79–83.
Zhao, M., Xie, M., 1996. On maximum likelihood estimation for a general non-homogeneous Poisson process. Scand. J. Stat. 23, 597–607.

Jingchi Wu obtained his B.S.E. degree in Software Engineering from Harbin University of Science and Technology in Harbin, China, in 2019. He then pursued his M.S. degrees in Graduate School of Advanced Science and Engineering from Hiroshima University in Higashi-hiroshima, Japan, completing them in 2022. Starting in 2022, he embarked on his Ph.D. journey also in the Graduate School of Advanced Science and Engineering from Hiroshima University.

Tadashi Dohi received the B.S.E., M.S., and D.Eng. degrees in engineering from Hiroshima University, Higashi-hiroshima, Japan, in 1989, 1991 and 1995, respectively. In 1992 he joined the Department of Industrial and Systems Engineering, Hiroshima University, as an Assistant Professor. In 1992, and 2000, he was a Visiting Research Fellow in the Faculty of Commerce and Business Administration, University of British Columbia, Canada, and Hudson School of Engineering, Duke University, USA, respectively, on the leave absent from Hiroshima University. Since 2002, he has been working as a Full Professor in the Department of Information Engineering, Hiroshima University. He is now a Vice Dean of School of Informatics and Data Science, Hiroshima University. Dr. Dohi's research interests include Reliability Engineering, Software Reliability, Dependable Computing, Performance Evaluation, Computer Security and Operations Research. He is a Regular Member of ORSJ, IEICE, IPSJ, REAJ, IEEE Computer Society, and IEEE Reliability Society. He published over 550 peer-reviewed articles, 30 book chapters, and edited 20 books/proceedings in the above research areas. Dr. Dohi served as the General Chair of 12 international conferences including ISSRE 2011 and DASC 2019, and as the Program Committee Chair of 3 international events. He has worked as a Program Committee Member in many international conferences such as DSN, ISSRE, SRDS, SAC, COMPSAC, QRS, EDCC, PRDC, HASE, among many others. He is an Associate Editor/Editorial Board Member of over a dozen international journals including IEEE Transactions on Reliability. He is now the President of REAJ (Reliability Engineering Association of Japan) and the Executive Committee Member of ORSJ (Operations Research Society of Japan).

Hiroyuki Okamura (Member, IEEE) received the B.S.E., M.S., and D.Eng. degrees in engineering from Hiroshima University, Higashi-hiroshima, Japan, in 1995, 1997, and 2001, respectively. In 1998, he joined Hiroshima University as an Assistant Professor, where he has been an Associate Professor with the Department of Information Engineering, Graduate School of Engineering, since 2003. He has been a Professor with the Department of Information Engineering, Graduate School of Engineering, since 2018. His current research interests include performance evaluation, dependable computing, and applied statistics. Prof. Okamura is a member of the Operations Research Society of Japan, the Institute of Electrical, Information and Communication Engineers, the Japan Society for Industrial and Applied Mathematics, the Information Processing Society of Japan, and the Association for Computing Machinery.