Annals of Operations Research (2022) 311:525–546
https://doi.org/10.1007/s10479-020-03732-3

S.I. : RELIABILITY MODELING WITH APPLICATIONS BASED ON BIG DATA

A generalized multiple environmental factors software reliability model with stochastic fault detection process

Mengmeng Zhu[1] · **Hoang Pham**[2]

Published online: 23 July 2020

Abstract
Software systems have been widely applied in numerous safety–critical domains; however, large-scale software development is still considered as a complicated and expensive activity. As the latest trends in software industry accelerate the complexity and dependency of software development, such complicated and human-centered process needs to be addressed well. Meanwhile, recent survey investigations (Zhu et al. in J Syst Softw 109:150–160, 2015; Zhu and Pham in J Syst Softw 132:72–84, 2017) revealed that environmental factors, defined from software development, have significant impacts on software reliability. Considering such significant impacts, we first propose a generalized multiple-environmental-factors software reliability growth model with multiple environmental factors and the associated randomness under the martingale framework. The randomness is reflected on the process of detecting software fault. Indeed, this is a stochastic fault detection process. As an illustration, a specific multiple-environmental-factors software reliability growth model incorporating two specific environmental factors, *percentage of reused modules* and *frequency of program specification change*, is further developed. Lastly, we employ two real-world data sets to demonstrate the prediction performance of the proposed generalized multiple-environmental-factors software reliability growth model.

Keywords Software reliability growth model · Environmental factors · Martingale framework

✉ Mengmeng Zhu
mzhu7@ncsu.edu

[1] Department of Textile Engineering, Chemistry and Science, North Carolina State University, Raleigh, NC 27606, USA

[2] Department of Industrial and Systems Engineering, Rutgers University, Piscataway, NJ 08854, USA

1 Introduction

Software system is one of the essential elements in our modern society, which has widely applied in numerous safety–critical domains (Ivanov et al. 2018; De Melo and Sanchez 2008; Fiondella et al. 2013; Özdamar and Alanya 2001; Zachariah 2015). The emphasis of modern software development has changed significantly over the years. As identified by Bosch and Bosch-Sijtsema (2010), the latest trends in software industry accelerate the complexity and dependency of software development. The first trend is software product lines build-up. Bosch and Bosch-Sijtsema (2010) defined software product lines consist of platforms which can be used by many products in the organization. Each development team will select and configure components from the platforms in order to build a reliable and consistent product based on the individual functionality. The adoption of software production lines can be helpful on cost and time management, but it also brings extra dependency into the product and organization, which could cause the added complexity (Bosch and Bosch-Sijtsema 2010; Clements and Northrop 2002). The second trend is software global development within several organizations across different countries. Many software companies have placed several sites globally or partnered with remoted companies, mostly located in India and China (Garcia-Crespo et al. 2010; Garg et al. 2014; Carmel and Agarwal 2001; Herbsleb and Moitra 2001; Sangwan et al. 2006). Software global development has its overwhelming advantages but also faces many challenges such as culture differences, time zone and maturity of software engineering, all of which contribute to the elevation of the complexity of dependent management to a new level (Bosch and Bosch-Sijtsema 2010; Cascio and Shurygailo 2003). The third trend is the establishment of software ecosystems. In recent years, software development has transformed from a solo activity within the organization to a highly collaborative ecosystems, which can be placed globally (Storey et al. 2017). Such ecosystems allow the development of new functionality occurs outside of the organization, however blur projects/tasks boundaries (Singer et al. 2013; Harman et al. 2012; Ghazawneh and Henfridsson 2013; Basole and Karla 2011). Hence, software ecosystems also contribute to elevate the dependency level between products and organizations (Bosch and Bosch-Sijtsema 2010).

Since software systems are included in most areas of human activities so that software quality assurance and software reliability prediction are very critical in various industries (Condori-Fernandez and Lago 2018; El-Sebakhy 2009). However, delivering high-quality and reliable software products is not easy (Ponnurangam and Uma 2005; Zhu and Pham 2018a, b). Despite of being widely studied and of interested to global market, software quality is still a complex and costly task for researchers and practitioners. One of the fundamental software quality characteristics is reliability. Software reliability described in Lyu (2007) as "the probability of failure-free software operation for a specified period of time in a specified environment".

To estimate the remaining software faults in software program, predict software reliability and software failure rate given the time of interest, and plan release time, a great number of nonhomogeneous Poisson process (NHPP) software reliability growth models (SRGMs) were developed. However, most of SRGMs have not addressed the random effect of application environments. Only a few studies have incorporated the random effect of environments or fault reduction factor in SRGMs. For instance, Teng and Pham (2006) assumed that the random effects were represented by a unit-free environment factor. A generalized SRGM with the unit-free environment factor was developed to represent both software testing and operation phase. This unit-free environment factor was

modelled as a beta or gamma distribution in order to propose two specific SRGMs. Fault reduction factor (FRF) is the number of the removed faults corresponding to the failures (Musa 1980), which could be affected by other factors, e.g., imperfect debugging, delay debugging, etc. Hsu et al. (2011) considered the FRF as a time-variable function and further incorporated it in the SRGM to improve the accuracy of failure prediction. Pham (2014) incorporated the uncertainty of the operation environment into a software Vtub-shaped fault detection rate model. Specifically, software fault detection rate follows a Vtub-shape function and the uncertainty of the operation environments is represented by a random variable, modeled by gamma distribution. Chang et al. (2014) incorporated the idea of uncertainty of operating environments into the testing-coverage SRGM. Minamino et al. (2017) proposed a two-dimensional SRGM based on a CES type time function, which is a generalized form of Cobb–Douglas function with testing-time and testing-effort factor. Inoue et al. (2016) proposed a bivariate SRGM with the uncertainty of the change of software failure-occurrence phenomenon at the change-point. Zhu and Pham (2018c) incorporated a single factor and the impact of this factor in the SRGM. Recently, Qiu et al. (2019) proposed the stress testing method with influencing factors that cause systems work under certain stress and explored the mathematical relationship between mean time to failure and influencing factors.

The motivations of this paper are described as follows. First, even some studies incorporated the uncertainty of environments or a single factor in the model development, however they cannot represent a generalized SRGM with random application environments. Given the great changes in software development, such complicated and human-centered software development process needs to be addressed more appropriately. Second, environmental factors (EFs), such as amount of programming effort, programmer organization, human nature, testing environment, program complexity, design methodology, were firstly defined by Zhang and Pham (2000) from the perspectives of software complexity, human nature, team collaboration and the interaction with hardware systems. Recent survey investigations (Zhu et al. 2015; Teng and Pham 2017) have also revealed the significant impacts of EFs on software reliability and provided the latest rank of the importance level of EFs in software development. Thus, how to incorporate multiple EFs and the associated randomness induced by these EFs into the development of SRGM is essential yet challenging.

Therefore, we aim to propose a generalized SRGM incorporating multiple EFs and the associated randomness induced by these EFs under the martingale framework, in which researchers and practitioners are able to obtain a specific SRGM according to the individual application environments. Martingale framework, specifically, Brownian motion, is introduced to reflect the associated randomness. We consider the associated randomness is reflected on the process of software fault detection. Section 2 discusses the importance of EFs and introduces two specific EFs from recent studies (Zhu et al. 2015; Teng and Pham 2017), *percentage of reused modules* (PoRM) and *frequency of program specification change* (FoPSC). Section 3 first introduces the martingale framework and reviews the related work. Next, we propose a generalized framework of multiple-environmental-factors NHPP (MEF-NHPP) SRGM and further develop a specific MEF-NHPP SRGM incorporating two specific EFs, PoRM and FoPSC. Sections 4 first discusses parameter estimation and comparison criteria and then illustrates two numerical examples with the real-world Open Source Software (OSS) project data sets to demonstrate the prediction powder of the proposed generalized framework of MEF-NHPP SRGM. Section 5 draws the conclusion and describes the future research directions.

2 Environmental factors

Thirty-two EFs were first identified by Zhang and Pham (2000) from four phases of software development and the interactions with hardware subsystems. For example, one of the EFs, named program complexity, is defined to measure the program size in terms of the kiloline of code. Other EFs, such as requirements analysis is used to verify the understanding of the requirements generating from customers. Testing environment is the specific environment set up in testing phase in order to simulate the operational environment and detect software faults. Testing effort can be identified by testing expenditures, testing causes or the years of working. The definitions and detailed discussion of all EFs can be found in references (Zhang and Pham 2000; Zhu and Pham 2017).

Fifteen years later, Zhu et al. (2015) reinvestigated the impact of these EFs on software reliability and aimed to provide the latest ranking of the EFs, the correlation between factors, reduce the dimension of the EFs and compare the findings with the previous studies (Zhang and Pham 2000; Zhang et al. 2001). Most EFs on the top ten group in the previous studies (Zhang and Pham 2000; Zhang et al. 2001) still list on the top ten group in the latest investigation (Zhu et al. 2015). The latest top ten EFs in developing single-release software are FoPSC, testing effort, relationship of detailed design to requirement, testing environment, testing coverage, program complexity, programmer skill, PoRM, testing methodologies and domain knowledge. Later, Zhu and Pham (2017) launched another survey study to examine the impact of the EFs on software reliability in developing multiple-releases software. The top ten EFs in developing multiple-releases software are PoRM, amount of programming effort, requirement analysis, FoPSC, level of programming technologies, testing effort, relationship of detailed design to requirement, testing coverage, program workload and program complexity.

As demonstrated from the previous studies (Zhang and Pham 2000; Zhu et al. 2015; Zhu and Pham 2017; Zhang et al. 2001), EFs have significant impacts on reliability in software development; hence, it is plausible to incorporate multiple EFs in software reliably model to improve software reliability prediction accuracy. In order to illustrate the effectiveness of the proposed generalized framework of MEF-NHPP SRGM in considerations of the ranking and significance levels of EFs and practical applications, we thus develop a specific MEF-NHPP SRGM in Sect. 3 with two specific EFs, PoRM and FoPSC. In the following Sects. 2.1 and 2.2, we express the reasons of selecting these two EFs and their corresponding distributions based on the collected data.

2.1 PoRM

PoRM (Zhu and Pham 2018c; Zhang and Pham 2000) is defined as follows

$$PoRM = \frac{S_0}{S_N + S_0} \tag{1}$$

where S_0 represents the kiloline of code in the existing modules. S_N denotes the kiloline of code in the new modules (Zhu and Pham 2018c).

The PoRM data was collected from various industries such as manufacturing, high technology, online retailing, IT service and research institution (Zhu and Pham 2018c). The participants had different positions including managers, testing engineers, programmers and other roles contributed to software development. To provide valid and reliable responses, survey participants were working on software development-related

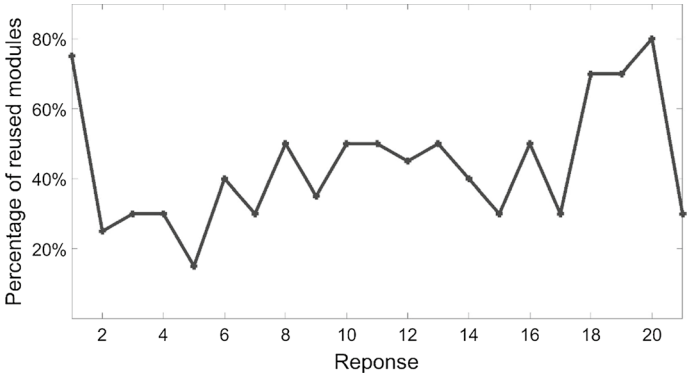

Fig. 1 The collected PoRM data

area or IT department during the data collection period. The collected PoRM (Zhu and Pham 2018c) is shown in Fig. 1. As the research results obtained in reference (Zhu and Pham 2018c), Gamma distribution is employed to model PoRM with parameters γ_1 and θ_1, expressed as PoRM $\sim$ Gamma(6.487, 14.726).

2.2 FoPSC

Lehman (1980) summarized the Program Evolution Laws. The first law of Program Evolution is continuing change, which expresses that large program is never completed and will continue evolving. Changes of specifications occur since the initial development until product delivery, which increases the risk of adding extra software cost but could add more values and improve software reliability (McGee and Greer 2010). Changes of specifications, studied by Harker et al. (1993), mostly due to the reasons such as fluctuations within the organization or market, consequence of system-usage, customer migratory issues, the increased understanding of requirements and adaption issues. Later, many studies have also discussed the importance of the changes of specifications from the perspectives such as product strategy, hardware/software environment/interaction, testability and functionality enhancement (Nurmuliani et al. 2004a, b; Carlshamre 2002; Shi et al. 2013).

Meanwhile, FoPSC is one of the significant EFs on the top ten list affecting software reliability in both survey investigations (Zhu et al. 2015; Zhu and Pham 2017). We define FoPSC as the total times of all the specifications have been changed in all the historical versions in software development. In this study, we will use the percentage of all the changes in a project to estimate the parameters. We employ the data sets provided in references (Shi et al. 2013; Loconsole and Borstler 2005) to estimate the distribution of FoPSC. The collected FoPSC data is illustrated in Fig. 2.

Considering the definition of FoPSC, gamma distribution or beta distribution is appropriate for modeling FoPSC. We compare the log-likelihood value of gamma distribution and beta distribution for the collected data. It concludes that beta distribution is a better fit for FoPSC. Parameter estimation of the beta distribution is also obtained from the collected data, stated as follows: FoPSC $\sim$ *Beta*(1.411, 7.409).

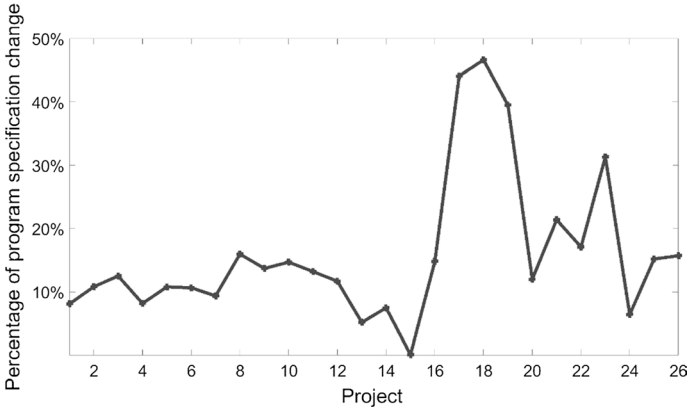

Fig. 2 The collected FoPSC data

3 A generalized MEF-NHPP SRGM

3.1 Some related work

The underlying assumptions of NHPP SRGM are the detection of software fault is a NHPP and the software failure intensity is proportional to the software fault detection rate and the remaining fault in the program. Most NHPP SRGMs are proposed in terms of the equation given as follows (Zhu and Pham 2018a, b, c; Teng and Pham 2006; Musa 1980; Hsu et al. 2011; Pham 2014; Goel and Okumoto 1979; Pham 2007).

$$\frac{d}{dt}m(t) = h(t)[N(t) - m(t)] \tag{2}$$

where $m(t)$ is the expected number of software failures detected by time t, $N(t)$ represents the fault content function, $h(t)$ is the software fault detection rate per unit of time. Depending on the model consideration, $N(t)$ and $b(t)$ are modeled as a constant or a time-dependent function, respectively. For example, Goel-Okumoto model (Goel and Okumoto 1979) assumed that $h(t) = b$ and $N(t) = a$. Inflection S-shaped model (Pham 2007) assumed that $h(t) = \frac{b}{1+\beta e^{-bt}}$ and $N(t) = a$. PNZ model (Pham 2007) assumed that $h(t) = \frac{b}{1+\beta e^{-bt}}$ and $N(t) = a(1 + \alpha t)$.

In order to identify the random development/application environments or the impact of the EF on software reliability, a stochastic software fault detection is adopted with applying $h(t)$ to be $h(t, \eta)$, where η represents the random environment effect or the EF. Equation (2) will be reformulated as follows

$$\frac{d}{dt}m(t, \eta) = h(t, \eta)[N(t) - m(t, \eta)] \tag{3}$$

As an illustration, Pham (2014) considered $h(t, \eta) = h(t)\eta$, named dynamic multiplicative noise model, and $N(t) = N$ in the model, in which η is a random variable. Pham and Pham (2019) considered $h(t, w) = h(t) + \dot{M}(t, w)$ as a dynamic additive noise model in the software reliability model. $\dot{M}(t, w)$ denotes the derivative of M as regards time t. $M(t)$ is defined as a martingale as regards the filtration $(\mathcal{F}_t, t \geq 0)$.

Later, inspired by references (Pham 2014; Pham and Pham 2019), Zhu and Pham (2018c) considered the dynamic multiplicative model as well as the additive noise model in the SRGM, which is described as $h(t,\eta) = h(t) + \lambda_0 G(t,\eta) + \dot{B}(t)$. A software reliability model considering a single EF and its impact was developed by Zhu and Pham (2018c), given as follows

$$\frac{d}{dt}m(t,\eta) = \left[h(t) + \lambda_0 G(t,\eta) + \dot{B}(t)\right][N(t) - m(t,\eta)] \tag{4}$$

where η represents the EF, PoRM. $G(t,\eta)$ is a time-dependent function. λ_0 is the coefficient associated with the $G(t,\eta)$. Standard Gaussian white noise is represented by $\dot{B}(t)$, in which

$$\frac{d}{dt}B(t) = \dot{B}(t) \tag{5}$$

where $B(t)$ denotes Brownian motion.

Brownian motion is a martingale as well (Mikosch 1998; Mörters and Peres 2010). Mikosch (1998) indicated that $\{ B(t) : t \geq 0 \}$ and $\{ B^2(t) - t : t \geq 0 \}$ both are martingale in regard to the nature filtration $(\mathcal{F}_t, t \geq 0)$ given $\{ B(t) : t \geq 0 \}$ denotes as Brownian motion. One of the martingale properties can be applied to Eq. (4) is

$$E\left[\int_v^t h(s,w)ds\right] = \int_v^t h(s)ds \tag{6}$$

Meanwhile, $\dot{B}(t)$ is a standard Gaussian process with the covariance structure shown as follows

$$E\left[\dot{B}(t)\dot{B}(u)\right] = \delta(u-t), 0 < t < u \tag{7}$$

where δ is the Dirac Delta measure. Thus, the general solution of Eq. (4) obtained by reference (Zhu and Pham 2018c) is given as follows

$$m(t,\eta) = N(t) - N(0)e^{-\int_0^t \left[h(s)+\lambda_0 G(s,\eta)+\dot{B}(s)\right]ds} - \int_0^t e^{-\int_u^t \left[h(s)+\lambda_0 G(s,\eta)+\dot{B}(s)\right]ds} N'(u)du \tag{8}$$

However, reference (Zhu and Pham 2018c) only considered a single EF in the software reliably model, while many other EFs also have the significant impacts on software reliability in software development (Zhu et al. 2015; Zhu and Pham 2017). Thus, we propose a theoretic framework of MEF-NHPP SRGM incorporating multiple EFs and the associated randomness in the following section.

3.2 A generalized MEF-NHPP SRGM

Considering the significant impacts of EFs on software reliability in software development in recent survey investigations (Zhu et al. 2015; Zhu and Pham 2017) and the great changes in the complicated and human-centered software development process, we develop a generalized MEF-NHPP SRGM incorporating multiple EFs and the associated randomness induced by these EFs under the martingale framework.

The assumptions of the proposed MEF-NHPP SRGM are described below.

(1) Software fault detection is a NHPP.
(2) Software failure intensity is proportional to the remaining faults in the program.
(3) The manifestation of software failures is due to the remaining faults in the program.
(4) Software faults are independent.
(5) Multiple EFs are considered in the proposed model. All EFs are independent. We do not consider correlation between EFs in this study.
(6) The randomness induced by the impact of EFs, is imposed on software fault detection rate and modeled by martingale process; specifically, Brownian motion.

Hence, a theoretic MEF-NHPP software reliability model is proposed as follows

$$\frac{d}{dt}m\left(t,\underbrace{\eta_1,\eta_2,\ldots,\eta_n}\right)=\left[h(t)+\sum_{i=1}^{n}\lambda_i G_i\left(t,\eta_i\right)+\dot{B}(t)\right]\left[N(t)-m\left(t,\underbrace{\eta_1,\eta_2,\ldots,\eta_n}\right)\right] \quad (9)$$

where $m\left(0,\underbrace{\eta_1,\eta_2,\ldots,\eta_n}\right)=0.\underbrace{\eta_1,\eta_2,\ldots,\eta_n}$ represents the n-dimensional vector. η_i is a random variable and represents $\mathrm{EF}_i, i=1,2,\ldots,n$. $m\left(t,\underbrace{\eta_1,\eta_2,\ldots,\eta_n}\right)$ represents the expected number of software failures detected by time t considering multiple EFs. $h(t)$ represents software fault detection rate per unit of time without the impact of EFs. $G_i\left(t,\eta_i\right)$ is a time-dependent function, which also denotes the effect brought by $\mathrm{EF}_i, i=1,2,\ldots,n$, on software fault detection rate per unit of time. λ_i denotes the coefficient associated with $G_i\left(t,\eta_i\right)$. $N(t)$ is the fault content function. $\dot{B}(t)$ is a standard Gaussian white noise, as presented in Eq. (5).

With the applications of the martingale property and the general solution from references (Zhu and Pham 2018c; Pham and Pham 2019), the mean value function of the proposed MEF-NHPP SRGM is obtained

$$m\left(t,\underbrace{\eta_1,\eta_2,\ldots,\eta_n}\right)=N(t)-N(0)e^{-\int_0^t\left(h(s)+\sum_{i=1}^{n}\lambda_i G_i\left(s,\underbrace{\eta_1,\eta_2,\ldots,\eta_n}\right)+\dot{B}(s)\right)ds}-\int_0^t e^{-\int_u^t\left(h(s)+\sum_{i=1}^{n}\lambda_i G_i\left(s,\underbrace{\eta_1,\eta_2,\ldots,\eta_n}\right)+\dot{B}(s)\right)ds}N'(u)du \quad (10)$$

By applying Eqs. (6) and (7) on $h(t)+\sum_{i=1}^{n}\lambda_i G_i\left(t,\underbrace{\eta_1,\eta_2,\ldots,\eta_n}\right)+\dot{B}(t)$, we can obtain the following equation

$$\int_0^t\left[h(s)+\sum_{i=1}^{n}\lambda_i G_i\left(s,\underbrace{\eta_1,\eta_2,\ldots,\eta_n}\right)+\dot{B}(s)\right]ds=\int_0^t\left[h(s)+\sum_{i=1}^{n}\lambda_i G_i\left(s,\underbrace{\eta_1,\eta_2,\ldots,\eta_n}\right)\right]ds+B(t) \quad (11)$$

By substituting Eq. (11) to Eq. (10), the mean value function is thus obtained

$$\bar{m}\left(t,\underbrace{\eta_1,\eta_2,\ldots,\eta_n}\right)=N(t)-N(0)e^{-\int_0^t h(s)ds}e^{\frac{t}{2}}e^{-\int_0^t\sum_{i=1}^n\lambda_i G_i(s,\eta_i)ds}-\int_0^t e^{-\int_u^t h(s)ds}e^{\frac{t-u}{2}}e^{-\int_u^t\sum_{i=1}^n\lambda_i G_i(s,\eta_i)ds}N'(u)du \tag{12}$$

Let

$$G_i\left(t,\eta_i\right)=\eta_i v_i(t) \tag{13}$$

where $v_i(t)$ is a time-dependent function, which also represents the effect of time on EF_i, $i=1,\ldots,n$.

As discussed in the model assumptions, all EFs are independent in this study. Each EF will be represented by a random variable. To present an explicit solution of Eq. (12), we apply the expectation on both sides of Eq. (12) with respect to $\eta_1,\eta_2,\ldots,$ and η_n. Hence, the mean value function of the proposed MEF-NHPP SRGM can be expressed as

$$\begin{aligned}\bar{m}_{\underbrace{\eta_1,\eta_2,\ldots,\eta_n}}(t)=N(t)-N(0)e^{-\int_0^t h(s)ds}e^{\frac{t}{2}}\left[\prod_{i=1}^n\int_0^\infty e^{-\int_0^t\lambda_i\eta_i v_i(s)ds}f\left(\eta_i\right)d\eta_i\right]\\-\int_0^t N'(u)e^{-\int_u^t\left(h(s)-\frac{1}{2}\right)ds}\left[\prod_{i=1}^n\int_0^\infty e^{-\int_u^t\lambda_i\eta_i v_i(s)ds}f\left(\eta_i\right)d\eta_i\right]du\end{aligned} \tag{14}$$

Equation (14) is the generalized mean value function in consideration of multiple EFs and the associated randomness. If the distribution of each EF is known, by applying the Laplace transform of each probability density function, we have high possibility to obtain a closed-form expression of Eq. (14).

3.3 A specific MEF-NHPP SRGM

As discussed above, to demonstrate the performance of the proposed MEF-NHPP SRGM, we apply two specific EFs, PoRM and FoPSC, into the proposed model. The mean value function of the specific MEF-NHPP SRGM is thus obtained as follows

$$\begin{aligned}\bar{m}_{\underbrace{\eta_1,\eta_2}}(t)=N(t)-N(0)e^{-\int_0^t h(s)ds}e^{\frac{t}{2}}\left[\prod_{i=1}^2\int_0^\infty e^{-\int_0^t\lambda_i\eta_i v_i(s)ds}f\left(\eta_i\right)d\eta_i\right]\\-\int_0^t N'(u)e^{-\int_u^t\left(h(s)-\frac{1}{2}\right)ds}\left[\prod_{i=1}^2\int_0^\infty e^{-\int_0^t\lambda_i\eta_i v_i(s)ds}f\left(\eta_i\right)d\eta_i\right]du\end{aligned} \tag{15}$$

where η_1 denotes PoRM, η_2 denotes FoPSC. λ_1 and λ_2 are the coefficients associated with the function $G_1\left(t,\eta_1\right)$ and $G_2\left(t,\eta_2\right)$, respectively. $v_1(t)$ and $v_2(t)$ represent the time-dependent function and reflect the effect of time on PoRM and FoPSC, respectively.

Gamma distribution, with parameters γ_1 and θ_1, is applied to model PoRM · The probability density function (PDF) of PoRM is given as follows

$$f(\eta_1) = \frac{\theta_1^{\gamma_1}\eta_1^{\gamma_1-1}e^{-\theta_1\eta_1}}{\Gamma(\gamma_1)} \tag{16}$$

Beta distribution, with parameters β_1 and β_2, is applied to model FoPSC. The PDF of FoPSC is given as follows

$$f(\eta_2) = \frac{\Gamma(\beta_1+\beta_2)\eta_2^{\beta_1-1}(1-\eta_2)^{\beta_2-1}}{\Gamma(\beta_1)\Gamma(\beta_2)} \tag{17}$$

The Laplace transform is given as follows

$$\int_0^\infty xe^{-sx}f(x)dx = -\frac{dF^*(s)}{ds} \tag{18}$$

By applying Eq. (18), the Laplace transform of Eq. (16) is given as follows

$$F_{\eta_1}^*(s) = \left[\frac{\theta_1}{\theta_1+s}\right]^{\gamma_1} \tag{19}$$

By applying Eq. (18), the Laplace transform of Eq. (17) is given as follows (Teng and Pham 2006)

$$F_{\eta_2}^*(s) = e^{-s} \times HG\big(\big[\beta_2\big],[\beta_1+\beta_2\big],s\big) \tag{20}$$

where $\mathrm{HG}\big(\big[\beta_2\big],[\beta_1+\beta_2\big],s\big)$ is the generic hypergeometric function such as

$$HG\big(\big[a_1,a_2,\ldots,a_m\big],[b_1,b_2,\ldots,b_n\big],s\big) = \sum_{j=0}^{\infty}\left[\frac{s^j\prod_{i=1}^m\frac{\Gamma(a_i+j)}{\Gamma(a_i)}}{\prod_{i=1}^n\frac{\Gamma(b_i+j)}{\Gamma(b_i)}j!}\right] \tag{21}$$

Therefore, the Laplace transform of beta distribution can be further written as

$$\begin{aligned} F_{\eta_2}^*(s) &= e^{-s}\left[\sum_{j=0}^{\infty}\frac{\Gamma(\beta_1+\beta_2)\Gamma(\beta_2+j)s^j}{\Gamma(\beta_2)\Gamma(\beta_1+\beta_2+j)j!}\right] \\ &= \sum_{j=0}^{\infty}\frac{\Gamma(\beta_1+\beta_2)\Gamma(\beta_2+j)\times Poisson(j,s)}{\Gamma(\beta_2)\Gamma(\beta_1+\beta_2+j)} \end{aligned} \tag{22}$$

where $Poisson(j,s) = \frac{s^je^{-s}}{j!}$.

Substituting Eqs. (19) and (22) into Eq. (15), the mean value function of the specific MEF-NHPP SRGM is thus obtained as follows

$$\bar{m}\underbrace{_{\eta_1,\eta_2}}(t) = N(t) - \left[\frac{\theta_1}{\theta_1 + \int_0^t \lambda_1 v_1(s)ds}\right]^{\gamma_1} \left[\sum_{j=0}^{\infty} \frac{\Gamma(\beta_1+\beta_2)\Gamma(\beta_2+j) \times Poisson\left(j, \int_0^t \lambda_2 v_2(s)ds\right)}{\Gamma(\beta_2)\Gamma(\beta_1+\beta_2+j)}\right] \left[N(0)e^{-\int_0^t h(s)ds} e^{\frac{t}{2}} + \int_0^t N'(u)e^{-\int_u^t \left(h(s)-\frac{1}{2}\right)ds} du\right] \tag{23}$$

where $Poisson\left(j, \int_0^t \lambda_2 v_2(s)ds\right) = \frac{\left(\int_0^t \lambda_2 v_2(s)ds\right)^j e^{-\int_0^t \lambda_2 v_2(s)ds}}{j!}$.

Different formulations of $h(t)$, $v_i(t)$ and $N(t)$ considering different testing scenarios assumptions can be substituted into Eq. (23) to obtain the final solution. As an example, let $h(t) = \frac{b}{1+ce^{-bt}}$, $v_1(t) = e^{-a_1 t}$, $v_2(t) = e^{-a_2 t}$ and $N(t) = \frac{1}{k}e^{kt}$, where b, c, a_1, a_2, and k are the coefficient of $h(t), v_1(t), v_2(t)$, and $N(t)$, respectively. Substituting $h(t), v_1(t), v_2(t)$ and $N(t)$ into Eq. (23), the mean value function of the proposed specific MEF-NHPP SRGM with the selected functions is obtained as follows

$$\bar{m}\underbrace{_{\eta_1,\eta_2}}(t) = \frac{1}{k}e^{kt} - \frac{e^{\frac{t}{2}}}{c+e^{bt}}\left[\frac{\theta_1}{\theta_1 + \frac{\lambda_1}{a_1}(1-e^{-a_1 t})}\right]^{\gamma_1} \left[\sum_{j=0}^{\infty} \frac{\Gamma(\beta_1+\beta_2)\Gamma(\beta_2+j) \times Poisson\left(j, \frac{\lambda_2}{a_2}(1-e^{-a_2 t})\right)}{\Gamma(\beta_2)\Gamma(\beta_1+\beta_2+j)}\right] \left[\frac{c+1}{k} - \frac{c}{k-\frac{1}{2}}e^{\left(k-\frac{1}{2}\right)t} - \frac{1}{b+k-\frac{1}{2}}e^{\left(b+k-\frac{1}{2}\right)t} + \frac{c}{k-\frac{1}{2}} + \frac{1}{b+k-\frac{1}{2}}\right] \tag{24}$$

where $Poisson\left(j, \frac{\lambda_2}{a_2}(1-e^{-a_2 t})\right) = \frac{\left[\frac{\lambda_2}{a_2}(1-e^{-a_2 t})\right]^j e^{-\frac{\lambda_2}{a_2}(1-e^{-a_2 t})}}{j!}$.

4 Applications

Recently, the increasing adoption of OSS by individuals, software companies and government-supported organizations has promoted the wide application of OSS in our modern society. We employ two Apache[1] OSS project data sets, named Whirr and Juddi, to elucidate the effectiveness and performance of the proposed generalized MEF-NHPP SRGM. As an illustration, we compare the performance of failure prediction of the specific MEF-NHPP SRGM with other SRGMs given in Table 1. Note that only the single-environmental-factor (SEF) model (Zhu and Pham 2018c) considers EF.

[1] https://www.apache.org

Table 1 NHPP SRGMs

NHPP SRGM	Mean value function
Goel-Okumoto model (Goel and Okumoto 1979)	$m(t) = a\left(1 - e^{-bt}\right)$
Inflection S-shaped model (Pham 2007)	$m(t) = \frac{a\left(1-e^{-bt}\right)}{1+\beta e^{-bt}}$
Delayed S-shaped model (Pham 2007)	$m(t) = a\left[1 - (1 + bt)e^{-bt}\right]$
Yamada imperfect debugging model (Pham 2007)	$m(t) = a\left(1 - e^{-bt}\right)\left(1 - \frac{\alpha}{b}\right) + \alpha at$
PNZ model (Pham 2007)	$m(t) = \frac{a\left[\left(1-e^{-bt}\right)\left(1-\frac{\alpha}{b}\right)+\alpha t\right]}{1+\beta e^{-bt}}$
IFD model (Pham 2007)	$m(t) = a - ae^{-bt}\left[1 + (b + d)t + bdt^2\right]$
SEF model (Zhu and Pham 2018c)	$m(t) = \frac{1}{k}e^{kt} - \frac{e^{\frac{t}{2}}}{c+e^{bt}}\left[\frac{\theta}{\theta+\frac{\lambda_0}{a}\left(1-e^{-at}\right)}\right]^{\gamma}$ $\left(\frac{c+1}{k} - \frac{c}{k-\frac{1}{2}}e^{\left(k-\frac{1}{2}\right)t} - \frac{1}{b+k-\frac{1}{2}}e^{\left(b+k-\frac{1}{2}\right)t} + \frac{c}{k-\frac{1}{2}} + \frac{1}{b+k-\frac{1}{2}}\right)$

4.1 Parameter estimation and comparison criteria

Least squares estimation (LSE) and maximum likelihood estimation are commonly applied to estimate the unknown parameters. LSE finds the optimal parameter values by minimizing S, described as follows

$$S = \sum_{i=1}^{n}\left[m\left(t_i\right) - y_i\right]^2 \tag{25}$$

where y_i is the observed number of software failures at time t_i, $i = 1, 2, \ldots, n$. $m\left(t_i\right)$ is the expected number of software failures at time t_i, $i = 1, 2, \ldots, n$. LSE is employed to estimate the unknown parameters in this study. We consider the parameters for PoRM and FoPSC are estimated from the real data, as seen in Sect. 2; hence, we will only need to estimate other seven unknown parameters seen in Eq. (24), which are k, b, c, λ_1, λ_2, a_1, and a_2. The genetic algorithm is applied to solve the Eq. (25) and estimate the unknown parameters.

Four comparison criteria (Pham 2007; Huang and Kuo 2002; Li et al. 2012), mean squared error (MSE), predictive-ratio risk (PRR), predictive power (PP) and Variation, are employed to evaluate the model performance, listed as follows.

$$MSE = \frac{\sum_{i=1}^{n}\left[m\left(t_i\right) - y_i\right]^2}{n - N} \tag{26}$$

$$PRR = \sum_{i=1}^{n}\left[\frac{m\left(t_i\right) - y_i}{m\left(t_i\right)}\right]^2 \tag{27}$$

$$PP = \sum_{i=1}^{n}\left[\frac{m\left(t_i\right) - y_i}{y_i}\right]^2 \tag{28}$$

Table 2 DS-I

Time unit t_i	Failures between t_{i-1} and t_i	Cumulative failures by t_i	Time unit t_i	Failures between t_{i-1} and t_i	Cumulative failures by t_i	Time unit t_i	Failures between t_{i-1} and t_i	Cumulative failures by t_i
1	6	6	12	4	66	23	6	102
2	6	12	13	0	66	24	22	124
3	6	18	14	4	70	25	3	127
4	8	26	15	5	75	26	1	128
5	13	39	16	5	80	27	1	129
6	6	45	17	2	82	28	0	129
7	8	53	18	10	92	29	0	129
8	2	55	19	1	93	30	0	129
9	3	58	20	1	94	31	4	133
10	3	61	21	2	96	32	3	136
11	1	62	22	0	96	-	-	-

$$Variation = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left[y_i - m(t_i) - Bias\right]^2} \tag{29}$$

where $Bias = \frac{1}{n}\sum_{i=1}^{n}\left[m(t_i) - y_i\right]$. n denotes the total number of observations. N denotes the number of unknown parameters in each model. MSE evaluates the distance of the failure prediction to the observed data. PRR evaluates the distance of the predicted failures to the observed failures against the predicted failures; while PP evaluates the distance of the predicted failures to the observed failures against the observed failures.

We understand the evaluated model tends to give better prediction as the number of unknown parameters increases. That is the reason we employ the above four criteria to compare the models from different perspectives. For all the four comparison criteria, the evaluated model has better prediction power as the criteria become smaller.

4.2 Numerical example I

In this first numerical example, Whirr OSS project data, collected from September 2010 to April 2013, is applied to demonstrate the model performance. Whirr is a group of libraries to run cloud service. Whirr OSS project data is denoted as data set I (DS-I) in this paper. Table 2 describes time unit t_i, software failures between t_{i-1} and t_i, denoted as $y_i - y_{i-1}$, and cumulative software failures by t_i, denoted as y_i, as seen in Eqs. (25–29). For example, the observed software failures between time unit t_1 and t_2 is 6. The cumulative software failures by time unit t_2 is 12. The observed software failures between time unit t_2 and t_3 is 6. The cumulative software failures by time unit t_3 is 18. The observed software failures between time unit t_{31} and t_{32} is 3. The cumulative software failures by time unit t_{32} is 136.

The first 24 observations of DS-I, $y_1, \ldots, y_{24}$, are considered as training set to estimate the parameters of the selected SRGMs. The observations, $y_{25}, \ldots, y_{32}$, are thus considered as testing set. Table 3 presents the selected criteria comparison and the estimated parameters of SRGMs based on the training set by applying LSE. Comparing with other SRGMs

Table 3 Parameter estimates and model comparisons of DS-I

NHPP model	Parameter estimates	MSE	PRR	PP	Variation
Goel-Okumoto model (Goel and Okumoto 1979)	$\hat{a} = 201.250$ $\hat{b} = 0.033$	36.561	0.315	0.228	6.076
Inflection S-shaped model (Pham 2007)	$\hat{a} = 150.030$ $\hat{b} = 0.096$ $\hat{\beta} = 1.830$	68.326	0.789	0.483	7.992
Delayed S-shaped model (Pham 2007)	$\hat{a} = 131.400$ $\hat{b} = 0.144$	110.047	20.902	2.123	10.544
Yamada imperfect debugging model (Pham 2007)	$\hat{a} = 185.180$ $\hat{b} = 0.033$ $\hat{\alpha} = 0.010$	60.193	0.401	0.300	74.941
PNZ model (Pham 2007)	$\hat{a} = 161.010$ $\hat{b} = 0.069$ $\hat{\alpha} = 0.001$ $\hat{\beta} = 0.930$	54.515	0.523	0.333	6.795
IFD model (Pham 2007)	$\hat{a} = 143.045$ $\hat{b} = 0.129$ $\hat{d} = 0.001$	143.253	40.461	2.665	11.076
SEF model (Zhu and Pham 2018c)	$\hat{k} = 0.014$ $\hat{b} = 0.589$ $\hat{c} = 0.039$ $\hat{a} = 75.000$ $\hat{\lambda_0} = 2.001$	35.173	0.254	0.311	5.390
Proposed MEF-NHPP SRGM	$\hat{k} = 0.013$ $\hat{b} = 0.574$ $\hat{c} = 0.0004$ $\hat{a_1} = 231.432$ $\hat{\lambda_1} = 1.920$ $\hat{a_2} = 161.203$ $\hat{\lambda_2} = 3.291$	35.065	0.021	0.016	5.091

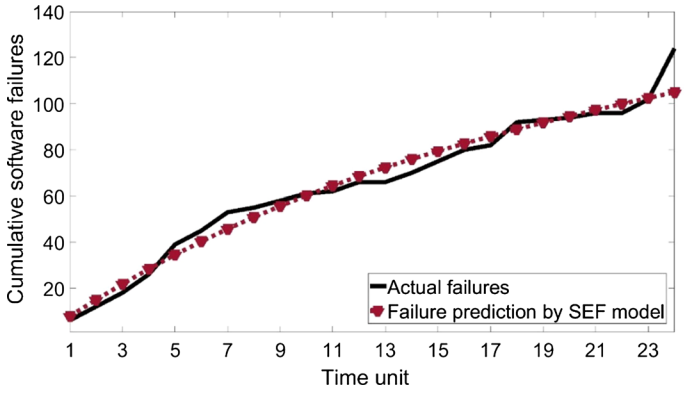

Fig. 3 DS-I comparison of actual failures with failure prediction by SEF model

without considering EF (Pham 2007) and considering a single EF, SEF model (Zhu and Pham 2018c), the proposed MEF-NHPP SRGM has the smallest values of all four criteria,

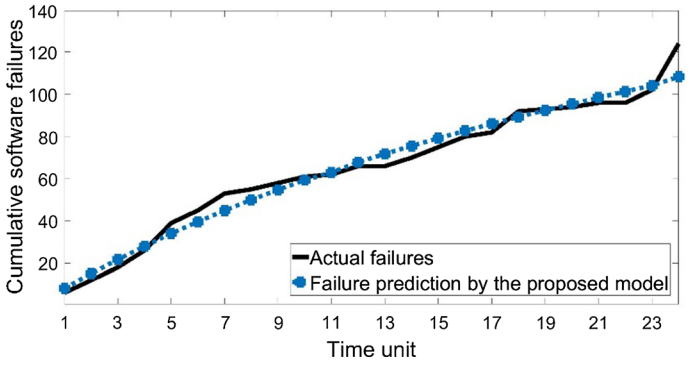

Fig. 4 DS-I comparison of actual failures with failure prediction by the proposed MEF-NHPP SRGM

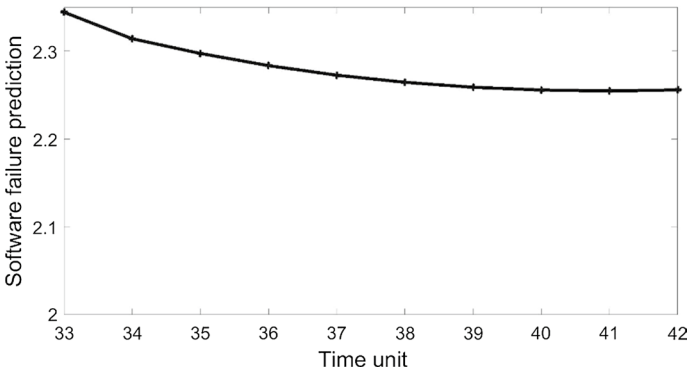

Fig. 5 DS-I software failure prediction between time unit t_i to $t_{i+1}, i = 32, 33, \ldots, 41$

as seen in Table 3. As an illustration of model comparison, we only present Figs. 3 and 4 in this section. Figure 3 illustrates the comparison of the actual failures with failure prediction by SEF model. Figure 4 illustrates the comparison the predicted software failures based on the proposed MEF-NHPP SRGM and the actual failures.

Based on the research outcomes obtained in references (Teng and Pham 2006; Musa 1980; Hsu et al. 2011; Pham 2014; Zhu and Pham 2018c), software reliability model with considering the random environments has better performance in terms of failure prediction and reliability estimation. Hence, we only present the criteria comparison of SEF model and the proposed MEF-NHPP SRGM for the testing set. The values of MSE, PRR, PP and Variation of SEF model for the testing set are 273.750, 0.176, 0.131 and 34.903, respectively. The values of MSE, PRR, PP and Variation of the proposed MEF-NHPP SRGM for the testing set are 161.750, 0.099, 0.078 and 26.445, respectively. The proposed MEF-NHPP SRGM has smaller values of all four criteria for the testing set, as compared with SEF model. We thus conclude that the proposed MEF-NHPP SRGM has the best fitting performance.

The given dataset, as seen in Table 2, describes software failures from time unit t_1 to t_{32}. One of the great advantages of SRGMs is to estimate software failures based on the time of interest and determine the optimal release time of the software product. Software

Table 4 DS-II

Time unit t_i	Failures between t_{i-1} and t_i	Cumulative failures by t_i	Time unit t_i	Failures between t_{i-1} and t_i	Cumulative failures by t_i	Time unit t_i	Failures between t_{i-1} and t_i	Cumulative failures by t_i
1	7	7	12	22	88	23	11	140
2	2	9	13	11	99	24	4	144
3	8	17	14	8	107	25	0	144
4	9	26	15	2	109	26	5	149
5	2	28	16	7	116	27	0	149
6	5	33	17	3	119	28	9	158
7	5	38	18	4	123	29	13	171
8	7	45	19	1	124	30	1	172
9	10	55	20	0	124	31	1	173
10	9	64	21	0	124	32	12	185
11	2	66	22	5	129	33	0	185

practitioners and researchers are also interested in the software failure prediction after time unit t_{32}. Thus, we estimate the software failures after time unit t_{32} based on the proposed MEF-NHPP SRGM. Figure 5 shows the predicted software failures between time unit t_i to $t_{i+1}, i = 32, 33, \ldots, 41$, which provides a practical reference for software development team to decide the time to stop testing and how much testing resource will be allocated to the project.

4.3 Numerical example II

The second numerical example applies Apache Juddi OSS project data, collected from February 2009 to February 2014. Juddi OSS project data is denoted as data set II (DS-II) in this paper. Table 4 describes time unit t_i, software failures between t_{i-1} and t_i, denoted as $y_i - y_{i-1}$, and cumulative software failures by t_i, denoted as y_i, as seen in Eqs. (25–29). For example, the observed software failures between time unit t_1 and t_2 is 2. The observed software failures between time t_2 and t_3 is 8. The cumulative software failures by time unit t_2 is 9. The observed software failures between time unit t_{32} and t_{33} is 0. The cumulative software failures by time unit t_{33} is 185.

The first 28 observations of DS-II,$y_1, \ldots, y_{28}$, are considered as training set to estimate the parameters of the selected SRGMs in the second numerical example. The observations, $y_{29}, \ldots, y_{33}$, are thus considered as testing set. Based on LSE method, we estimate the unknown parameters of the selected SRGMs by the training set. Accordingly, the model comparisons between the proposed MEF-NHPP SRGM and other SRGMs based on the selected criteria can be obtained. As listed in Table 5, the proposed MEF-NHPP SRGM has the smallest values of the selected criteria, such as MSE, PP and Variation, compared with other SRGMs. In term of the comparison criterion PRR, PNZ model carries the smallest PRR value, however PNZ model also has much larger MSE value compared with the proposed MEF-NHPP SRGM. PRR is the criterion that evaluates the distance of the predicted failures to the observed failures against the predicted failures; in other words, it assigns

Table 5 Parameter estimates and model comparisons of DS-II

NHPP model	Parameter estimates	MSE	PRR	PP	Variation
Goel-Okumoto model (Goel and Okumoto 1979)	$\hat{a} = 181.250$ $\hat{b} = 0.056$	136.477	1.285	3.462	11.892
Inflection S-shaped model (Pham 2007)	$\hat{a} = 179.230$ $\hat{b} = 0.193$ $\hat{\beta} = 13.159$	176.235	5.292	1.513	12.794
Delayed S-shaped model (Pham 2007)	$\hat{a} = 200.090$ $\hat{b} = 0.112$	67.922	27.778	1.536	8.221
Yamada imperfect debugging model (Pham 2007)	$\hat{a} = 230.250$ $\hat{b} = 0.034$ $\hat{\alpha} = 0.008$	69.510	0.625	1.180	103.480
PNZ model (Pham 2007)	$\hat{a} = 300.130$ $\hat{b} = 0.048$ $\hat{\alpha} = 0.001$ $\hat{\beta} = 1.321$	90.902	0.372	0.418	8.809
IFD model (Pham 2007)	$\hat{a} = 189.960$ $\hat{b} = 0.134$ $\hat{d} = 0.010$	74.124	528.248	2.576	8.240
SEF model (Zhu and Pham 2018c)	$\hat{k} = 0.009$ $\hat{b} = 0.626$ $\hat{c} = 1.078$ $\hat{a} = 51.725$ $\widehat{\lambda_0} = 25.346$	63.989	4.895	1.224	7.576
Proposed MEF-NHPP SRGM	$\hat{k} = 0.008$ $\hat{b} = 0.597$ $\hat{c} = 0.900$ $\widehat{a_1} = 100.000$ $\widehat{\lambda_1} = 40.148$ $\widehat{a_2} = 113.644$ $\widehat{\lambda_2} = 29.289$	44.780	0.990	0.327	5.421

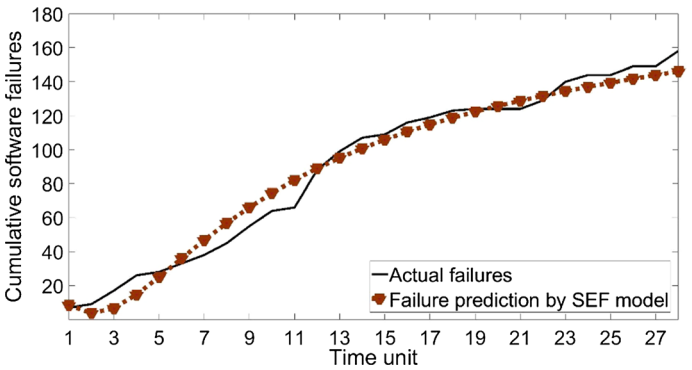

Fig. 6 DS-II comparison of actual failures with failure prediction by SEF model

larger penalty to the model which underestimates the failures. MSE is generally considered as the priority criterion since it penalizes larger prediction errors more than others. Hence,

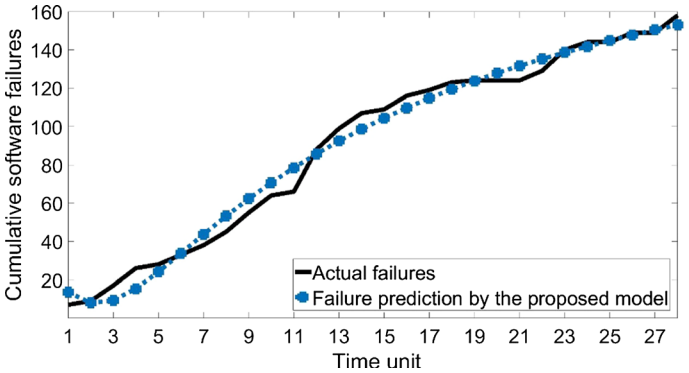

Fig. 7 DS-II comparison of actual failures with failure prediction by the proposed MEF-NHPP SRGM

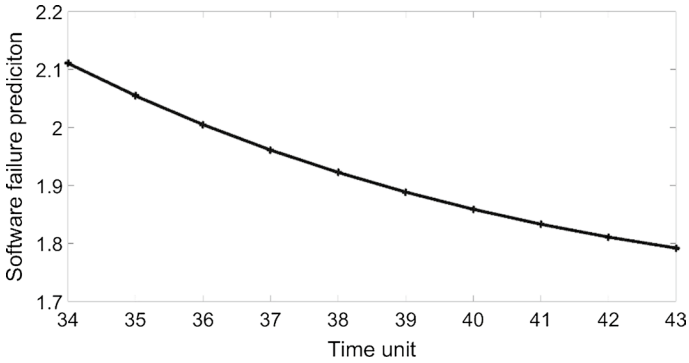

Fig. 8 DS-II software failure prediction between time unit t_i to $t_{i+1}, i = 33, 34, \ldots, 42$

the proposed model is concluded to be the best fit since it has the lowest value of MSE, PP and Variation for the training set, compared with other SRGMs incorporating a single EF (Zhu and Pham 2018c) and without considering EF (Pham 2007). As an illustration, Fig. 6 displays the comparison between the failures predicted by SEF model and the actual failures. Figure 7 displays the comparison between the failures predicted by the proposed generalized MEF-NHPP SRGM and the actual failures.

Moreover, the criteria comparison of SEF model and the proposed MEF-NHPP SRGM for the testing set are described as follows. The values of criteria such as MSE, PRR, PP and Variation of SEF model for the testing set are 736.800, 0.162, 0.115 and 50.793, respectively. The values of criteria such as MSE, PRR, PP and Variation of the proposed MEF-NHPP SRGM for the testing set are 289.800, 0.056, 0.045 and 37.357, respectively. The proposed MEF-NHPP SRGM has smaller values of all four criteria for the testing set. Therefore, the proposed MEF-NHPP SRGM is concluded as the best fit.

Software failure prediction can be calculated based on the proposed MEF-NHPP SRGM. Indeed, we provide software failure prediciton after time unit t_{33} for failure DS-II as well. Figure 8 shows software failure predition between time unit t_i to $t_{i+1}, i = 33, 34, \ldots, 42$ based on the proposed generalized MEF-NHPP SRGM. Software failure prediction can be

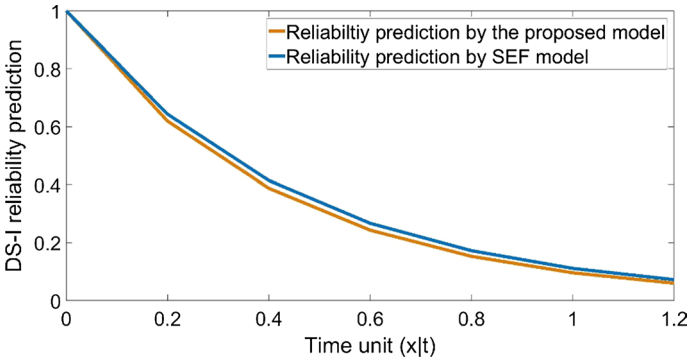

Fig. 9 Reliability prediction comparison of DS-I

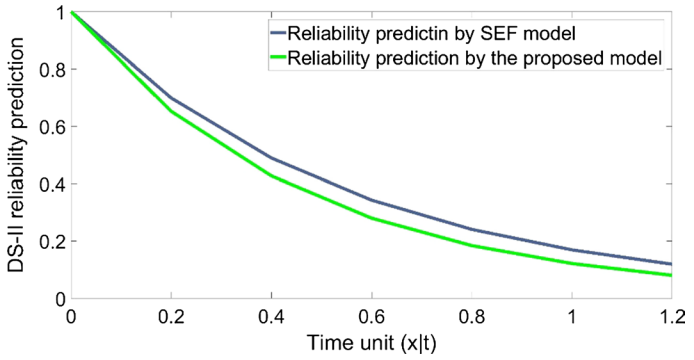

Fig. 10 Reliability prediction comparison of DS-II

a great help for testing resource allocation, software multiple releases planning and the determiantion of software optimal release time.

4.4 Reliability prediction

Software reliability within $(t, t + x)$ can be determined after the unknown parameters estimated by LSE. Software reliability is calculated by the equation stated as follows

$$R(x|t) = e^{-[m(t+x)-m(t)]} \tag{30}$$

All other models have not considered EFs, except SEF model. Since OSS projects are significantly impacted by the EFs (Zhu and Pham 2017), indeed, we only compare the reliability prediction predicted by SEF model and the proposed MEF-NHPP SRGM. Given time unit $t = 32$ and $t = 33$ for DS-I and DS-II and varying x from time unit 0 to 1.2 in Eq. (30), Figs. 9 and 10 illustrate the comparison of the reliability prediction calculated by SEF model and the proposed MEF-NHPP SRGM for these two data sets, respectively. As seen from Figs. 9 and 10, the reliability value predicted by the proposed MEF-NHPP SRGM is less than SEF model for both data sets.

5 Conclusions and future research

Given the great changes in software development, such complicated and human-centered software development process needs to be addressed well. Meanwhile, recent survey investigations (Zhu et al. 2015; Zhu and Pham 2017) have revealed the significant impacts of EFs on software reliability and provided the latest rank of the importance level of EFs in software development. Hence, how to incorporate multiple EFs and the randomness caused by these EFs into the development of software reliability model is essential yet challenging.

We firstly develop a generalized MEF-NHPP SRGM with multiple EFs and the associated randomness. Each EF is modeled as a random variable. The randomness induced by the EFs is elucidated by the martingale framework. We then incorporate a stochastic software fault detection process in the model due to the associated randomness. Software practitioners and researchers are able to obtain a specific MEF-NHPP SRGM according to the individual application environments from the proposed generalized MEF-NHPP SRGM. In order to elucidate the effectiveness of the proposed MEF-NHPP SRGM, we select two EFs, PoRM and FoPSC, from recent studies (Zhu et al. 2015; Zhu and Pham 2017) to further develop a specific MEF-NHPP SRGM. Lastly, two OSS data sets are employed to demonstrate the predictive power in terms of software failure and reliability of the proposed generalized MEF-NHPP SRGM.

Future research can be drawn from many directions. First, the dependencies between EFs and the impact of such dependencies on software reliability can be further investigated in the next step. Secondly, the impact of EFs is reflected on software fault detection process in this study. The investigation of such impact on the total fault content can be conducted.

References

Basole, R. C., & Karla, J. (2011). On the evolution of mobile platform ecosystem structure and strategy. *Business & Information Systems Engineering, 3*(5), 313.

Bosch, J., & Bosch-Sijtsema, P. (2010). From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software, 83*(1), 67–76.

Carlshamre, P. (2002). Release planning in market-driven software product development: Provoking an understanding. *Requirements Engineering, 7*(3), 139–151.

Carmel, E., & Agarwal, R. (2001). Tactical approaches for alleviating distance in global software development. *IEEE Software, 18*(2), 22–29.

Cascio, W. F., & Shurygailo, S. (2003). E-leadership and virtual teams. *Organizational Dynamics, 31*(4), 362–376.

Chang, I. H., Pham, H., Lee, S. W., & Song, K. Y. (2014). A testing-coverage software reliability model with the uncertainty of operating environments. *International Journal of Systems Science: Operations & Logistics, 1*(4), 220–227.

Clements, P., & Northrop, L. (2002). *Software product lines: Practices and patterns*. Reading: Addison-Wesley.

Condori-Fernandez, N., & Lago, P. (2018). Characterizing the contribution of quality requirements to software sustainability. *Journal of Systems and Software, 137,* 289–305.

De Melo, A. C. V., & Sanchez, A. J. (2008). Software maintenance project delays prediction using Bayesian Networks. *Expert Systems with Applications, 34*(2), 908–919.

El-Sebakhy, E. A. (2009). Software reliability identification using functional networks: A comparative study. *Expert Systems with Applications, 36*(2), 4013–4020.

Fiondella, L., Rajasekaran, S., & Gokhale, S. S. (2013). Efficient software reliability analysis with correlated component failures. *IEEE Transactions on Reliability, 62*(1), 244–255.

Garcia-Crespo, A., Colomo-Palacios, R., Soto-Acosta, P., & Ruano-Mayoral, M. (2010). A qualitative study of hard decision making in managing global software development teams. *Information Systems Management, 27*(3), 247–252.

Garg, H., Rani, M., & Sharma, S. P. (2014). An approach for analyzing the reliability of industrial systems using soft-computing based technique. *Expert Systems with Applications, 41*(2), 489–501.
Ghazawneh, A., & Henfridsson, O. (2013). Balancing platform control and external contribution in third-party development: The boundary resources model. *Information Systems Journal, 23*(2), 173–192.
Goel, A. L., & Okumoto, K. (1979). Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability, 28*(3), 206–211.
Harker, S. D., Eason, K. D. & Dobson, J. E. (1993). The change and evolution of requirements as a challenge to the practice of software engineering. In *Proceedings of IEEE international symposium on requirements engineering* (pp. 266–272). IEEE, San Diego, California.
Harman, M., Jia, Y. & Zhang, Y. (2012). App store mining and analysis: MSR for app stores. In *Proceedings of the 9*th *IEEE working conference on mining software repositories (MSR)* (pp. 108–111). IEEE, Zurich, Switzerland.
Herbsleb, J. D., & Moitra, D. (2001). Global software development. *IEEE Software, 18*(2), 16–20.
Hsu, C. J., Huang, C. Y., & Chang, J. R. (2011). Enhancing software reliability modeling and prediction through the introduction of time-variable fault reduction factor. *Applied Mathematical Modelling, 35*(1), 506–521.
Huang, C. Y., & Kuo, S. Y. (2002). Analysis of incorporating logistic testing-effort function into software reliability modeling. *IEEE Transactions on Reliability, 51*(3), 261–270.
Inoue, S., Ikeda, J., & Yamada, S. (2016). Bivariate change-point modeling for software reliability assessment with uncertainty of testing-environment factor. *Annals of Operations Research, 244*(1), 209–220.
Ivanov, V., Reznik, A., & Succi, G. (2018). Comparing the reliability of software systems: A case study on mobile operating systems. *Information Sciences, 423,* 398–411.
Lehman, M. M. (1980). Programs, life cycles, and laws of software evolution. In *Proceedings of the IEEE* (pp. 1060–1076).
Li, H., Zeng, M., Lu, M., Hu, X., & Li, Z. (2012). Adaboosting-based dynamic weighted combination of software reliability growth models. *Quality and Reliability Engineering International, 28*(1), 67–84.
Loconsole, A., & Borstler, J. (2005). An industrial case study on requirements volatility measures. In *Proceedings of the 12*th *Asia-Pacific software engineering conference* (pp. 8). IEEE, Taipei, Taiwan.
Lyu, M. R. (2007) Software reliability engineering: A roadmap. In *2007 future of software engineering* (pp. 153–170). IEEE, Minneapolis, Minnesota.
McGee, S., & Greer, D. (2010). Sources of software requirements change from the perspectives of development and maintenance. *International Journal on Advances in Software, 3*(1–2), 186–200.
Mikosch, T. (1998). *Elementary stochastic calculus with finance in view*. Singapore: World Scientific.
Minamino, Y., Inoue, S. & Yamada, S. (2017). Two-dimensional software reliability growth modeling based on a CES type time function. In *Proceedings of 2017 international conference on infocom technologies and unmanned systems* (pp. 120–125). Dubai, United Arab Emirates.
Mörters, P., & Peres, Y. (2010). *Brownian motion*. Cambridge: Cambridge University Press.
Musa, J. D. (1980). The measurement and management of software reliability. In *Proceedings of the IEEE* (pp. 1131–1143).
Nurmuliani, N., Zowghi, D. & Powell, S. (2004a). Analysis of requirements volatility during software development life cycle. In *Proceedings of 2004 Australian software engineering conference* (pp. 28–37). IEEE, Melbourne, Australia.
Nurmuliani, N., Zowghi, D. & Williams, S. P. (2004b). Using card sorting technique to classify requirements change. In *Proceedings of the 12*th *IEEE international conference on requirements engineering* (pp. 240–248). IEEE, Kyoto, Japan.
Özdamar, L., & Alanya, E. (2001). Uncertainty modelling in software development projects (with case study). *Annals of Operations Research, 102*(1–4), 157–178.
Pham, H. (2007). *System software reliability*. Berlin: Springer.
Pham, H. (2014). A new software reliability model with Vtub-shaped fault-detection rate and the uncertainty of operating environments. *Optimization, 63*(10), 1481–1490.
Pham, T., & Pham, H. (2019). A generalized software reliability model with stochastic fault-detection rate. *Annals of Operations Research, 277*(1), 83–93.
Ponnurangam, D., & Uma, G. V. (2005). Fuzzy complexity assessment model for resource negotiation and allocation in agent-based software testing framework. *Expert Systems with Applications, 29*(1), 105–119.
Qiu, K., Zheng, Z., Trivedi, K. S., & Yin, B. (2019). Stress testing with influencing factors to accelerate data race software failures. *IEEE Transactions on Reliability*. https://doi.org/10.1109/TR.2019.2895052.
Sangwan, R., Bass, M., Mullick, N., Paulish, D. J., & Kazmeier, J. (2006). *Global software development handbook*. Boca Raton: Auerbach Publications.

Shi, L., Wang, Q. & Li, M. (2013). Learning from evolution history to predict future requirement changes. In *Proceedings of the 21*st *IEEE international on requirements engineering conference* (pp. 135–144). IEEE, Rio de Janeiro, Brazil.

Singer, L., Figueira Filho, F., Cleary, B., Treude, C. Storey, M. A. & Schneider, K. (2013). Mutual assessment in the social programmer ecosystem: An empirical investigation of developer profile aggregators. In *Proceedings of 2013 conference on computer supported cooperative work* (pp. 103–116). ACM, San Antonio, Texas.

Storey, M. A., Zagalsky, A., Figueira Filho, F., Singer, L., & German, D. M. (2017). How social and communication channels shape and challenge a participatory culture in software development. *IEEE Transactions on Software Engineering, 43*(2), 185–204.

Teng, X., & Pham, H. (2006). A new methodology for predicting software reliability in the random field environments. *IEEE Transactions on Reliability, 55*(3), 458–468.

Zachariah, B. (2015). Optimal stopping time in software testing based on failure size approach. *Annals of Operations Research, 235*(1), 771–784.

Zhang, X., & Pham, H. (2000). An analysis of factors affecting software reliability. *Journal of Systems and Software, 50*(1), 43–56.

Zhang, X., Shin, M. Y., & Pham, H. (2001). Exploratory analysis of environmental factors for enhancing the software reliability assessment. *Journal of Systems and Software, 57*(1), 73–78.

Zhu, M., & Pham, H. (2017). Environmental factors analysis and comparison affecting software reliability in development of multi-release software. *Journal of Systems and Software, 132,* 72–84.

Zhu, M., & Pham, H. (2018a). A two-phase software reliability modeling involving with software fault dependency and imperfect fault removal. *Computer Languages, Systems & Structures, 53,* 27–42.

Zhu, M., & Pham, H. (2018b). A multi-release software reliability modeling for open source software incorporating dependent fault detection process. *Annals of Operations Research, 269*(1–2), 773–790.

Zhu, M., & Pham, H. (2018c). A software reliability model incorporating martingale process with gamma-distributed environmental factors. *Annals of Operations Research*. https://doi.org/10.1007/s10479-018-2951-7.

Zhu, M., Zhang, X., & Pham, H. (2015). A comparison analysis of environmental factors affecting software reliability. *Journal of Systems and Software, 109,* 150–160.