The Journal of Systems and Software 115 (2016) 102–110

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Modeling and analysis of reliability of multi-release open source software incorporating both fault detection and correction processes

Jianfeng Yang[a], Yu Liu[b,*], Min Xie[b], Ming Zhao[c]

[a] Faculty of Information Engineering, Guizhou Institute of Technology, Guiyang, China
[b] Department of Systems Engineering and Engineering Management, City University of Hong Kong, Hong Kong, China
[c] Faculty of Engineering and Sustainable Development, University of Gävle, Gävle, Sweden

ARTICLE INFO

Article history:
Received 6 September 2015
Revised 22 December 2015
Accepted 14 January 2016
Available online 6 February 2016

Keywords:
Software reliability
Multiple upgrading
Fault correction process

ABSTRACT

Large software systems require regular upgrading that tries to correct the reported faults in previous versions and add some functions to meet new requirements. It is thus necessary to investigate changes in reliability in the face of ongoing releases. However, the current modeling frameworks mostly rely on the idealized assumption that all faults will be removed instantaneously and perfectly. In this paper, the failure processes in testing multi-release software are investigated by taking into consideration the delays in fault repair time based on a proposed time delay model. The model is validated on real test datasets from the software that has been released three times with new features. A comprehensive analysis of optimal release times based on cost-efficiency is also provided, which could help project managers to determine the best time to release the software.

1. Introduction

The software industry is growing rapidly and has become very competitive. As a result, many software developers are cutting back their schedules to ensure prompt delivery and developing new features to keep their products competitive. This is especially true for large and complex software. After a release, reported faults in previous versions will be removed and new functions may be designed to meet new requirements in new versions. Developers generally pay greater attention to balancing competition in the market, and thus risk quality because of the short software lifecycle. The upgradation process constitutes a challenge for software companies looking to produce highly reliable software and ensure the release time is on schedule.

Over the past four decades, researchers have studied a variety of methods to assess software reliability. One of the most widely investigated and applied of those methods is the software reliability growth model (SRGM) (Lyu, 2007; Amin et al., 2013; Febrero et al., 2014; Yamada, 2014). Most SRGMs utilize the fault data collected during the test process to describe the stochastic behavior of the software fault detection process (FDP) with respect to time, and it is reasonable to assume that the fault counts in each time interval are mutually independent of each other (Amin et al., 2013). Non-homogeneous Poisson process (NHPP) model is considered as one of the most effective models (Goel and Okumoto, 1979; Lyu, 1996; Ohishi et al., 2009). They have been successfully applied in many software projects to manage tests and predict operational reliability (Jeske and Zhang, 2007; Lin and Huang, 2008; Rana et al., 2014). They have also been utilized in making critical decisions, such as those involved in cost-benefit analysis, resource allocation, and release-time determination (Peng et al., 2013; Park and Baik, 2015; Wang et al., 2015).

Furthermore, a number of specific SRGMs have been proposed for investigating the reliability of Open Source software (OSS), which is a growing area of software development and applications. For example, Tamura and Yamada (2013) propose a method of software reliability assessment for the embedded OSS with flexible hazard rate modeling. Pachauri et al. (2013) blended fuzzy set theory with software reliability measurement and total cost analysis, and Gratus and Pratibha (2013) proposed an approach for carrying out pre-statistical data analyses based on assessment of software's reliability metrics. Luan and Huang (2014) proposed an improved Pareto distribution model for analyzing the failure process, although their method is confined to ungrouped data.

In this paper, the failure process in testing multi-release software is further explored by taking into consideration a delay in the fault repair time based on the time-delay model proposed by Wu et al. (2007). Both fault-correction and detection processes are considered. It is assumed that the faults in a new version comprise both undetected faults in a previous version and new faults

* Corresponding author. Tel.: +852 56286295.
E-mail address: lyu.12@my.cityu.edu.hk (Y. Liu).

http://dx.doi.org/10.1016/j.jss.2016.01.025
0164-1212/

introduced during the development process of the new version. A framework for assessing the expected number of remaining faults in each version is proposed and the optimal release time for each version is also investigated.

The remainder of this paper is organized as follows. Section 2 outlines the proposed framework for multi-release software modeling. In Section 3, the parameter estimation with Least Square Estimation (LSE) is developed and the optimal release strategy for such software is discussed. Section 5 demonstrates the application of the proposed models with a three-release dataset collected from a practical OSS test process and presents the results of optimal release time analysis. Finally, the conclusion is given in Section 6.

2. Literature review and further discussion of the multi-release problems

2.1. Modeling the multi-release situation

Most of the existing SRGMs focus on the software development process of only a single version. It is thus necessary to investigate changes in reliability arising from ongoing releases, which is a rather complex problem as usually there are many reasons for a new release. Several studies have been carried out in this regard in the literature. For example, Smidts et al. (1998) applied software failure data from a previous release to perform reliability estimation on a current release, and developed an early prediction model with a proposed Bayes framework using subjective and/or objective data from older projects. Hu et al. (2011) considered a scenario in which a software development team develops, tests, and releases software version by version, and proposed a number of practical assumptions. Li et al. (2011) later proposed a model that focuses on OSS and regards changes in testing effort with time as a hump-shaped curve. Recently, Pachauri et al. (2015) proposed a modeling framework considering the inflection S-shaped fault reduction factor and extended this model into multi-release software.

Many other factors, such as fault severities and test resources, are also incorporated into the modeling of multi-release software. Different severities describing the difficulty of correcting faults are considered during the upgrade process by Garmabaki et al. (2011), who assumed that the severity of the dormant faults in previous versions may change in subsequent versions. Kapur et al. (2012) discovered that some dormant faults in previously released versions can be removed in the tests of subsequent versions, and proposed a chain of SGRMs that take into account testing resources with a Cobb Douglas production function to optimize upgrade modeling and release time prediction.

2.2. Modeling fault correction delay

Most of the aforementioned modeling frameworks operate under the idealized assumptions that all faults are removed instantaneously and perfectly and that the expected number of removed faults is the same as the expected number of detected faults. In fact, time is always required for removal, and the expected number of removed faults at any given time is smaller than the expected number of detected faults (Gokhale et al., 2004). Accordingly, some researchers also take into account the fault correction process (FCP) and use corrected fault data to represent the correction time delay. Modeling both FDP and FCP requires more information from software testing records but improves estimation and prediction results. Schneidewind proposed an approach to FCP modeling that uses a constant delayed FDP (Schneidewind, 2001). He assumed that the rate of fault correction is proportional to the rate of failure detection.

However, because the FCP is heavily dependent on the FDP and there are many faults that have been detected but are still waiting for correction in some applications, the model usually underestimates the remaining faults in the code. Lo and Huang (2006) proposed an integrative method for analyzing the detection and correction processes using a differential equation. Wu et al. (2007) extended Schneidewind's model to a continuous version by substituting a time-dependent delay function for constant delay. Based on the aforementioned NHPP-based FDP and FCP modeling framework, both LSE and MLE (maximum likelihood estimation) approaches have been proposed. In addition, Hu et al. (2007) developed a neural networks configuration approach with an extra factor characterizing the dispersion of prediction repetitions used to simultaneously model the FDP and FCP. Huang and Hung (2010) later applied queuing models to describe the two processes with multiple change points. Incorporating a testing effort function and imperfect debugging, Peng et al. (2014) recently proposed a framework for analyzing both processes. Recently, Gaver and Jacobs (2014) proposed a queue model based on different failure mode assumptions.

3. Multi-release modeling framework for FDP and FCP

3.1. Single-release modeling framework for FDP and FCP

For single-version software, the method of modeling FDP is like the traditional NHPP SRGM in which the cumulative number of detected faults, $N(t)$, is assumed to follow a Poisson distribution with mean value function (MVF) $m_d(t)$, i.e.,

$$P\{N(t)=n\}=\frac{m_d^n(t)}{n!}e^{-m_d(t)}. \tag{1}$$

According to the basic assumption of fault removal, the MVF can be given by

$$\begin{cases}\dfrac{dm_d(t)}{dt}=\lambda_d(t)=\dfrac{F'(t)}{1-F(t)}[a-m_d(t)] \\ m_d(0)=0\end{cases}, \tag{2}$$

where $\lambda_d(t)$ refers to the failure rate during the test process and $F(t)$ is a cumulative distribution function. In solving the above differential equation, the MVF can be written as

$$m_d(t)=aF(t). \tag{3}$$

When $F(t)$ is assigned to an experiential distribution, it becomes the well-known GO model (Goel and Okumoto, 1979):

$$m_d(t)=a[1-\exp(-\gamma t)]. \tag{4}$$

The fault correcting process can be modeled as a stochastic time delay (obeys a random distribution of $G(t)$) of the FDP, and then delayed failure rate (and fault correcting rate) λ_c^* and delayed MVF m_c^* are as follows:

$$\lambda_c^*=\begin{cases}\lambda_d(t-\Delta t), & \Delta t\le t\\ 0, & \Delta t>t\end{cases} \tag{5}$$

$$m_c^*=\begin{cases}m_d(t-\Delta t), & \Delta t\le t\\ 0, & \Delta t>t\end{cases} \tag{6}$$

According to the approach proposed by Dai et al. (2007), $\lambda_c(t)$ can be the expectation of the delayed failure rate, that is,

$$\lambda_c(t)=E[\lambda_c^*]=\int_0^t\lambda_d(t-x)\cdot g(x)dx \tag{7}$$

$$m_c(t) = \int_0^t \lambda_c(\tau)d\tau = \int_0^t \int_0^\tau \lambda_d(\tau - x)\cdot g(x)dx\,d\tau$$
$$= \int_0^t \int_x^t \lambda_d(\tau - x)\cdot g(x)d\tau\,dx$$
$$= \int_0^t m_d(t-x)\cdot f(x)dx = E[m_c^*]. \tag{8}$$

It can be shown that $m_c(t) = E[m_c^*]$.

As the fault correction time delay can be approximated as an experiential distribution, Musa et al. (1987), thus, the MVF equation of the FCP is derived as follows:

$$m_c(t) = \begin{cases} a\left[1-(1+\gamma t)e^{-rt}\right], & \mu=\gamma \\ a\left[1-\dfrac{\mu}{\mu-\gamma}e^{-\gamma t}+\dfrac{\gamma}{\mu-\gamma}e^{-\mu t}\right], & \mu\neq\gamma \end{cases} \tag{9}$$

In addition, the more flexible gamma distribution ($\Delta t \sim$Gamma(α,β)) can be used to describe the time delay, that is,

$$m_c(t) = a\Gamma\left(t,\alpha,\frac{1}{\beta}\right) - \frac{ae^{-\gamma t}}{\Gamma(\alpha)\beta^\alpha} \times \int_0^t e^{-(\beta-\gamma)\cdot(t-x)}(t-x)^{\alpha-1}dx, a,\alpha,\beta,\gamma > 0. \tag{10}$$

3.2. Multi-release modeling framework for FDP and FCP

In multi-release circumstances, when new functions are added to the software, new faults are also embedded into the new version. As a result, the fault count in the new version can be decomposed into two groups: newly embedded faults and uncorrected faults in the previous version. The failure rate also changes. Accordingly, the FDP and FCP MVF of n-version software can be expressed as follows:

$$\begin{cases} m_{d1}(t) = a_1F_1(t), & 0<t\le T_1 \\ m_{d2}(t) = [a_2 + (m_{d1}(\text{inf}) - m_{d1}(T_1))]F_2(t-T_1), & T_1<t\le T_2 \\ \vdots & \vdots \\ m_{dn}(t) = \left[a_2 + \left(m_{d(n-1)}(\text{inf}) - m_{d(n-1)}(T_{n-1})\right)\right]F_n(t-T_{n-1}), & T_{n-1}<t\le T_n \end{cases} \tag{11}$$

$$\begin{cases} m_{c1}(t) = E[m_{d1}(t-\Delta t)], & \Delta t\sim G_1(t), & 0<t\le T_1 \\ m_{c2}(t) = E[m_{d2}(t-\Delta t)], & \Delta t\sim G_2(t), & T_1<t\le T_2 \\ \vdots & \vdots & \vdots \\ m_{cn}(t) = E[m_{dn}(t-\Delta t)], & \Delta t\sim G_n(t), & T_{n-1}<t\le T_n \end{cases} \tag{12}$$

where time division points $T_1, T_2, \ldots, T_n$ are the end times of the tests of each version. In the test records, however, these division points are often substituted for the time of the last fault detected.

If the multi-release process is considered as a successive process, the whole MVF should also be a successive function. Furthermore, due to the correction delay, some detected faults will remain suspended and wait for correction in next round of test (Huang and Hung, 2010). Thus, the following initial conditions are added to Eqs. (11) and (12):

$$\begin{cases} m'_{d1}(t) = m_{d1}(t), & 0<t\le T_1 \\ m'_{d2}(t) = m'_{d1}(T_1) + m_{d2}(t), & T_1<t\le T_2 \\ \vdots & \vdots \\ m'_{dn}(t) = m'_{d(n-1)}(T_{n-1}) + m_{dn}(t), & T_{n-1}<t\le T_n \end{cases} \tag{13}$$

and

$$\begin{cases} m'_{c1}(t) = m_{c1}(t), & 0<t\le T_1 \\ m'_{c2}(t) = m'_{c1}(T_1) + m_{c1}(t-T_1) \\ \quad +[m'_{d1}(T_1) - m'_{c1}(T_1)]G'_2(t-T_1), & T_1<t\le T_2 \\ \vdots & \vdots \\ m'_{cn}(t) = m'_{c(n-1)}(T_{n-1}) + m_{cn}(t-T_{n-1}) \\ \quad +[m'_{d(n-1)}(T_{n-1}) - m'_{c(n-1)}(T_{n-1})]G'_{(n-1)}(t-T_{n-1}), & T_{n-1}<t\le T_n \end{cases} \tag{14}$$

In the above, $G'_n(t)$ refers to the distribution of time delay for correcting the detected faults in previous versions. Because of the debugging effort in form test and priority schedule, the efforts in correcting those faults will be less than the efforts in correcting faults in current version. Thus, $G'_n(t)$ is assumed to be different from $G_n(t)$.

4. Parameter estimation and optimal release planning

4.1. Parameter estimation

Nonlinear LSE is applied to estimate the model parameters, and those estimates are the optimal solution to the following nonlinear programming problem:

$$\min_\theta S(\theta) = \sum_{j=1}^{n}\left[\sum_{i=1}^{k_j}(m'_{dj}(t_i,\theta) - n_{ji})^2 + \sum_{i=1}^{k_j}(m'_{cj}(t_i,\theta) - m_{ji})^2\right], \tag{15}$$

where n_{ij} and m_{ij} refer to the observed cumulative fault counts in the FDP and FCP of version j by testing time t_i, and θ constitutes the model parameters. The foregoing extreme problem can be solved with the build-in function in MATLAB.

4.2. Optimal upgrading planning problem for multi-release software

When a company is planning to develop software, one of the most significant issues it needs to consider is when to release a new version of that software. The decision depends on which criteria the company is most concerned with. For commercial software, stability and economic benefits are the main criteria, whereas reliability is the sole criterion for safety-critical software. The model for determining the optimal timing of a software generation release can be formulated based on different goals set by management. Because the optimization model for commercial software is always aimed at minimizing the total cost, such model is also called the cost model. Cost models often incorporate budget, warranty cost, and reliability requirements (Koch and Kubat, 1983; Pham and Zhang, 1999; Huang and Lyu, 2005). Wu et al. (2007) proposed a cost model involving the FCP, and Kapur et al. (2014) developed another for multi-release software. The basic cost model can be divided into three parts:

$$C_{\text{Total}} = C_{\text{Cost per time unit}} + C_{\text{Debugging cost in test phase}} + C_{\text{Debugging cost after release}}. \tag{16}$$

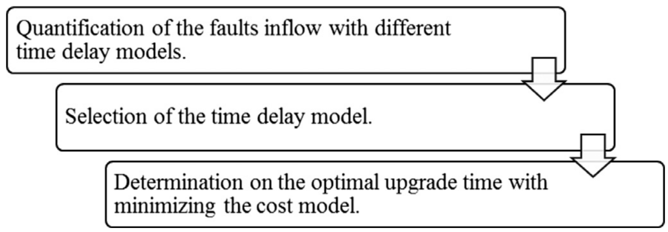

Fig. 1. The structure of the decision model for the determination of optimal version-update time.

In each version, the debugging cost may change due to alterations in personnel. Thus, the total cost of the test process for each version can be written as follows:

$$C_{\text{Total}}^{(i)} = \begin{cases} c_{11}T_1 + c_{12}^{d}m_{d1}(T_1) + c_{12}^{c}m'_{c1}(T_1) & i = 1 \\ \quad +c_{14}\left[m'_{c1}(\inf) - m'_{c1}(T_1)\right] & \\ c_{i1}(T_i - T_{i-1}) + c_{i2}^{d}m_{di}(T_i) & \\ \quad +c_{i2}^{c}\left[m'_{ci}(T_i) - m'_{c(i-1)}(T_{i-1})\right] & i \geq 2 \\ \quad +c_{i4}\left[m'_{ci}(\inf) - m'_{ci}(T_i)\right] & \end{cases}, \tag{17}$$

where i is the version order. The first term denotes the testing cost per unit time, the second term is related to the fault correction and detection effort, and the third is the warranty cost introduced by correcting a fault after release. The total cost of n versions is

$$\begin{aligned} C_{\text{Total}} &= c_{11}T_1 + \sum_{i=1}^{n} c_{i2}^{d}m_{di}(T_i) + c_{12}^{c}m'_{c1}(T_1) \\ &\quad + \sum_{j=2}^{n}\left\{c_{j1}(T_j - T_{j-1}) + c_{j2}^{c}\left[m'_{cj}(T_j) - m'_{c(j-1)}(T_{j-1})\right]\right\} \\ &\quad + c_{n4}\left[m'_{cn}(\inf) - m'_{cn}(T_n)\right]. \end{aligned} \tag{18}$$

According to the cost model above, optimal-release time decisions can be made by minimizing the cost model in Eq. (17) during the test period of each version in sequence. The final cost is calculated with Eq. (18).

4.3. Summary of the procedure

The procedure of the decision on optimal upgrade time is summarized in Fig. 1. The first step of the implementation is to quantify the faults inflow, which are the expected detection and correction fault number, i.e. $m_d(t)$, $m_c(t)$. The following step is the comparison of the models with different type of time delay. The comparing criteria used in this paper are the mean square error (MSE) and R^2, the square of multiple correlation coefficient that measures the goodness-of-fit of the model considered. The model with smaller MSE or higher R^2 is selected. The adjusted R^2 is also considered to measure the influence of extra parameters. Finally, based on the cost model proposed above, the optimal upgrade time is obtained by minimizing the cost model in Eq. (17).

5. Numerical illustration

In this section, two numerical application examples are given for the illustration purpose. The applied datasets are from online bug tracking systems (Mozilla and Gnome). Many organizations, particularly enterprises using OSS, utilize such systems in managing their software development processes.

In the numerical example, only those faults that are not duplicate and are reproducible for others are selected. The detection time of each fault is the opening time of this fault record, and the correction time is the time that the fault was marked as ‘FIXED’. Then, those faults are grouped according to the detection or correction week, as shown in Appendix.

5.1. Datasets

5.1.1. Mozilla

The fault tracking data are collected and organized on three successive versions Firefox 3.0, 3.5, and 3.6 from Bugzilla (https://bugzilla.mozilla.org/), as listed in Appendix. By the time of releasing a new version, 53, 48, and 26 faults bad been detected and there were 17, 15 and 9 faults respectively had been detected without correction. Compare to the total number of detected faults, the numbers of such faults account for 26.15%, 29.41% and 31.03% respectively, which are quite large and should not be ignored.

5.1.2. Gnome

The faults data are collected from the product of gnome-control-center (https://bugzilla.gnome.org/); and 4 successive versions from 2.0 to 2.3 are selected (the fault data are listed in Appendix).

5.2. Model application

In this subsection, the proposed models are tested with the aforementioned three-version dataset form Mozilla. Using the nonlinear least squares method in MATLAB, the estimates are obtained as reported in Table 1. Both the exponential time-delay model and gamma time-delay model are trained. To compare the performance of the proposed models, the data are also estimated with three single-release models without consideration of correcting the faults in previous version.

Both the MSE and R^2 criteria in the foregoing results show that the proposed models display a much better goodness-of-fit than the single-release models in terms of both FDP and FCP. The goodness-of-fit of the multi-release models increases with each version upgrade. Since the exponential distribution is a special form of gamma distribution with $\alpha = 1$, the proposed model with more flexible gamma time delay provides a better regression of the dataset, also indicated by the lower MSE and larger R^2, than the exponential time delay model (Figs. 2 and 3).

Fig. 4 depicts the relative errors of the foregoing models in terms of the test week. It can be seen that the exponential time delay model tend to be underestimation in predicting the FDP before first release. After first release, both of the exponential and Gamma time delay models tend to be similar in predicting the FDP. In early period of FCP, the exponential time delay model gives a more reliable prediction and the Gamma time delay tends to be more accuracy in predicting the later part of FCP.

5.3. Optimal upgrading planning problem for Firefox

For the sake of simplicity, it is assumed that an equal debugging cost and per unit time cost for each version: cost per unit time $c_{i1} = 5$, debugging cost in testing phase $cd\ i2 = 5$, $cd\ i2 = 5$, and post-release debugging $c_{i4} = 5$.

Using the cost model in Eq. (18), the optimal upgrading time can be estimated as shown in Table 3 based on the parameters in Table 1(2) and compared with the observed release time.

It can be seen from Table 2 that under different assumption of correcting time delay, the optimal test period and estimated total cost for each version is similar with each other. While the optimal values are quite different from the observed data due to the assumption of cost parameters.

Table 1
Comparison of various models.

(1) Exponential time delay					
Model	$\Delta t \sim \text{Exp}(\mu)$				
	Single-release models				*Multi-release model*
Version	*I*	*II*	*III*	*Total*	–
LSE	$a_1 = 57.2746$ $\gamma_1 = 0.0862$ $\mu_1 = 0.0490$	$a_1 = 52.4442$ $\gamma_1 = 0.0930$ $\mu_1 = 0.0930$	$a_1 = 31.6647$ $\gamma_1 = 0.0459$ $\mu_1 = 0.0569$		$a_1 = 57.2316\, \gamma_1 = 0.0865\, \mu_1 = 0.0489$ $a_2 = 59.9005\, \gamma_2 = 0.1086\, \mu_2 = 0.0561\, \mu'_2 = 0.1440$ $a_3 = 38.5533\, \gamma_3 = 0.0430\, \mu_3 = 0.0186\, \mu'_3 = 0.0795$
MSE_d	17.9124	2.8988	3.3402	9.5480	8.8065
MSE_c	22.5427	4.4236	3.7767	11.6458	10.2941
R_d^2	0.9527	0.9913	0.9772	0.9835	0.9976
R_c^2	0.9786	0.9950	0.9727	0.9791	0.9974
$adj\text{-}R_d^2$		0.9824 ($p = 9$)			0.9974 ($p = 11$)
$adj\text{-}R_c^2$		0.9777 ($p = 9$)			0.9972 ($p = 11$)
(2) Gamma time delay					
Model	$\Delta t \sim \text{Gamma}(\alpha,\ \beta)$				
	Single-release models				*Multi-release model*
Version	*I*	*II*	*III*	*Total*	–
LSE	$a_1 = 60.6022$ $\gamma_1 = 0.0711$ $\alpha_1 = 0.2511$ $\beta_1 = 0.0063$	$a_2 = 54.6681$ $\gamma_2 = 0.0848$ $\alpha_2 = 0.4552$ $\beta_2 = 0.0329$	$a_2 = 34.9429$ $\gamma_2 = 0.0382$ $\alpha_2 = 0.1933$ $\beta_2 = 0.0033$		$a_1 = 60.5407\, \gamma_1 = 0.0714\, \alpha_1 = 0.2418\, \beta_1 = 0.0057$ $a_2 = 58.6963\, \gamma_2 = 0.0981\, \alpha_2 = 0.7371\, \beta_2 = 0.0569$ $a_3 = 38.4262\gamma_3 = 0.0418\alpha_3 = 0.7067\beta_3 = 0.4357$ $\alpha'_2 = 0.2148\beta'_2 = 0.0023\alpha'_3 = 23.1943\beta'_3 = 0.1084$
MSE_d	16.1689	2.6885	3.0592	8.3578	8.0629
MSE_c	6.0826	1.7014	1.0308	3.2438	3.0345
R_d^2	0.9661	0.9931	0.9772	0.9854	0.9978
R_c^2	0.9807	0.9936	0.9849	0.9899	0.9991
$adj\text{-}R_d^2$		0.9840 ($p = 12$)			0.9975 ($p = 16$)
$adj\text{-}R_c^2$		0.9890 ($p = 12$)			0.9990 ($p = 16$)

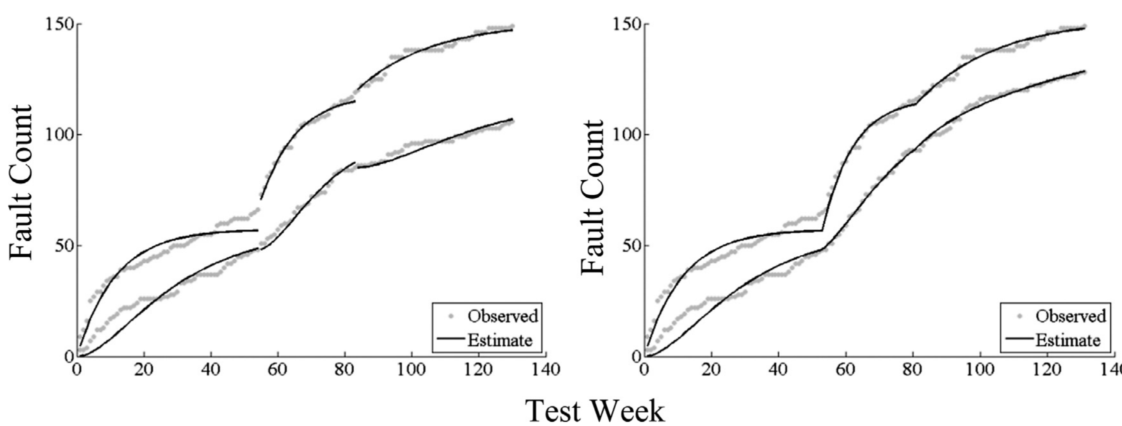

Fig. 2. Proposed model with exponential time delay.

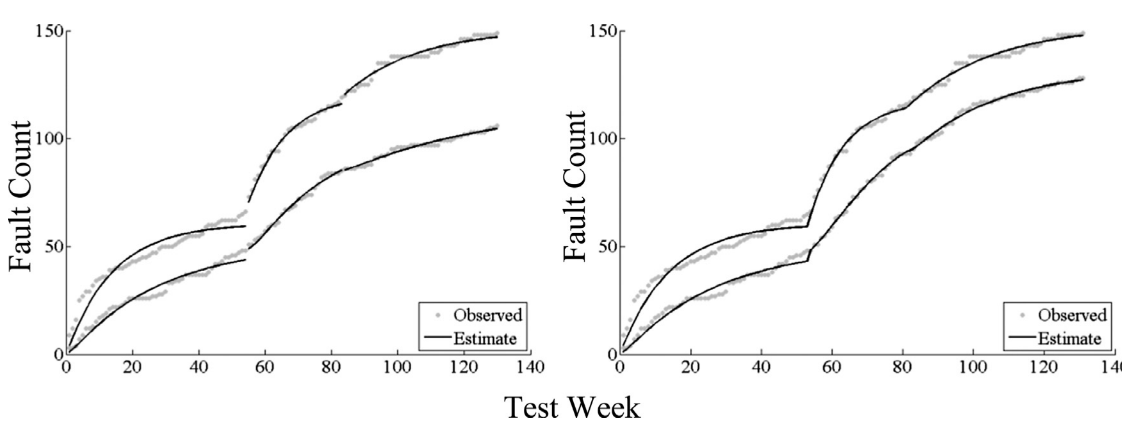

Fig. 3. Proposed model with gamma time delay.

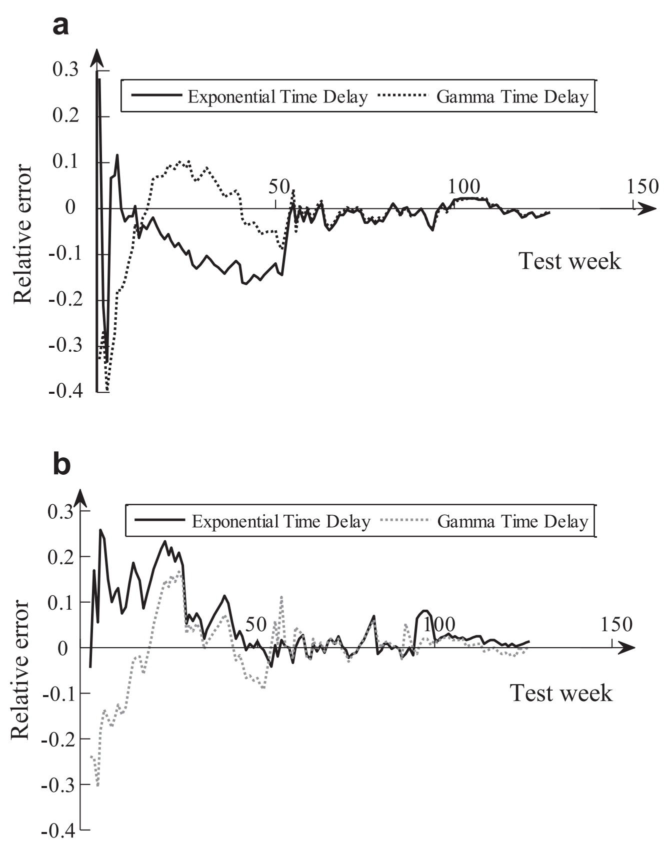

Fig. 4. (a) Relative errors of predicted $m_d(t)$ of proposed models. (b) Relative errors of predicted $m_c(t)$ of proposed models.

Table 2
Original data and optimal test time and cost of each version.

Model			Version I	Version II	Version III	Total
Exponential delay	Optimal value	Test period	78	65	90	233
		Expected cost	780.8	724.5	580.7	1786.0
Gamma delay	Optimal value	Test period	82	63	92	237
		Expected cost	838.5	703.8	568.6	1610.9

5.4. Optimal upgrading planning problem for Gnome2 control-center

Based on the procedure discussed in Section 4, the determination of the optimal version-update time for Gnome2 Control-center is presented as in the following steps:

Step 1: Quantification of the faults inflow with different time delay models.
Based on the failure data of Gnome2 Control-center, the model parameters can be estimated as shown in Table 3 and Fig. 5.
Step 2: Selection of the time delay model.
From the results listed in Table 3, the time delay model with Exponential distribution has a larger MSE_c but its R^2 and adj-R^2 are closer to 1. Thus, the time delay model with Exponential time delay is selected.
Step 3: Determination on the optimal upgrade time with minimizing the cost model.

Finally, based on the selected model, the cost function is evaluated and the optimal upgrade plan is shown in Table 4 with the assumption that an equal debugging cost and per unit time cost for each version: cost per unit time $c_{i1} = 5$, debugging cost in testing phase $c_{i2}^d = 6$, $c_{i2}^d = 5$, and post-release debugging $c_{i4} = 20$. The results mean that if the software is upgraded as this plan, then it can provide the greatest overall benefits for management.

6. Discussion

In this paper, a framework for modeling multi-release software reliability is first developed and parameter estimation problem in this situation is also studied. The developed approach takes into consideration delays in fault repair time based on the time delay model. The research incorporates software upgrading that has been an efficient approach to dealing with the increasing competition in today's market. It is assumed that the faults in a new software version comprise both undetected faults in the previous version and new faults introduced during the development process of the new version. The detected but not corrected faults before last release are also considered.

The proposed model has been applied to real test datasets collected from popular types of OSS. The results have shown that the proposed multi-release model provides better parametric estimates and reliable stochastic modeling than using the single-release model in each version. Cost analysis based on the proposed model is also carried out by solving the optimization problem on the release time.

It should be pointed out that a number of interesting research questions related to our proposed approach remain outstanding for

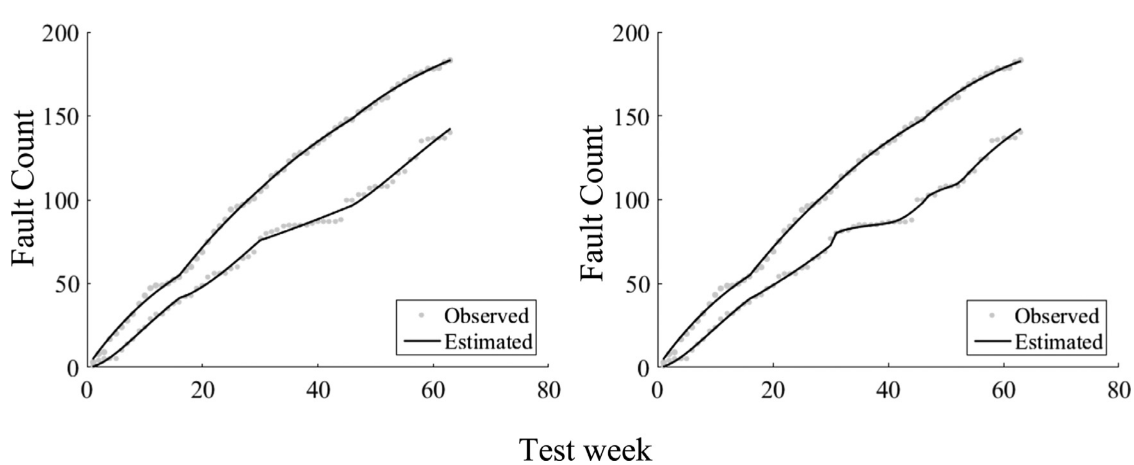

Fig. 5. Proposed model with different types of time delay.

Table 3
Comparison of various models.

Model	$\Delta t \sim \mathrm{Exp}(\mu)$	$\Delta t \sim \mathrm{Gamma}(\alpha, \beta)$
LSE	$a_1 = 98.8265\ \gamma_1 = 0.0507\ \mu_1 = 0.1996$ $a_2 = 169.3778\ \gamma_2 = 0.0259\ \mu_2 = 0.1026\ \mu'_2 = 0.0792$ $a_3 = 119.6673\ \gamma_3 = 0.0270\ \mu_3 = 0.0215\ \mu'_3 = 0.0379$ $a_4 = 70.1511\ \gamma_4 = 0.0405\ \mu_4 = 0.1044\ \mu'_4 = 0.0405$	$a_1 = 97.1932\ \gamma_1 = 0.0518\ \alpha_1 = 1.0381\ \beta_1 = 0.2083$ $a_2 = 148.8898\ \gamma_2 = 0.0301\ \alpha_2 = 9.1581\ \beta_2 = 0.9697$ $a_3 = 91.0224\gamma_3 = 0.0383\alpha_3 = 94.0139\beta_3 = 7.4084$ $a_4 = 57.1587\gamma_3 = 0.0557\alpha_3 = 101.5929\beta_3 = 16.0038$ $\alpha'_2 = 7.0867\beta'_2 = 2.0184\alpha'_3 = 0.3059\ \beta'_3 = 0.0030$ $\alpha'_4 = 0.5640\ \beta'_4 = 0.0088$
MSE_d	3.2752	3.3255
MSE_c	7.2173	3.8019
R_d^2	0.9994	0.9994
R_c^2	0.9928	0.9920
adj-R_d^2	0.9992	0.9991
adj-R_c^2	0.9905	0.9876

Table 4
Optimal upgrade time and cost of each version.

Model			Version I	Version II	Version III	Version IV	Total
Exponential delay	Optimal value	Test period	49	90	122	69	330
		Expected cost	1497.4	2813.1	2923.1	631.2	4473.1

further study. They include how to efficiently extract and classify data collected from the error reporting systems for OSS systems, how to build a multi-release software reliability model incorporating stochastic differential equation theory, and how to integrate masked failure data into the proposed modeling approach.

Acknowledgments

The authors sincerely thank the editor and the reviewers of an earlier version of this paper. The comments have led to significant improvement in the presentation and additional research carried out.

Appendix. Fault count record for Firefox 3.0, 3.5, and 3.6

Firefox 3.0

Week	No. of detected faults	No. of corrected faults	Week	No. of detected faults	No. of corrected faults
1	9	3	28	49	28
2	12	3	29	50	28
3	16	4	30	50	29
4	25	7	31	50	33
5	27	9	32	50	33
6	29	12	33	51	34
7	29	12	34	52	34
8	32	13	35	53	35
9	34	15	36	54	37
10	35	17	37	55	37
11	36	18	38	55	37
12	36	19	39	55	37
13	39	21	40	55	37
14	39	22	41	56	37
15	40	22	42	59	37
16	40	22	43	60	38
17	40	23	44	60	40
18	41	24	45	60	42
19	42	26	46	61	42
20	43	26	47	62	43
21	43	26	48	62	45
22	44	26	49	62	45
23	45	26	50	62	46
24	45	26	51	62	46
25	46	26	52	64	47
26	47	27	53	65	48
27	47	27			

Firefox 3.5

Week	No. of detected faults	No. of corrected faults	Week	No. of detected faults	No. of corrected faults
1	66	48	15	105	76
2	73	51	16	105	77
3	76	51	17	106	80
4	81	54	18	106	80
5	83	55	19	107	81
6	87	57	20	108	83
7	88	59	21	108	83
8	92	63	22	109	86
9	94	64	23	112	88
10	94	65	24	113	91
11	94	66	25	113	92
12	99	70	26	115	93
13	102	73	27	115	93
14	104	73	28	116	93

Firefox 3.6

Week	No. of detected faults	No. of corrected faults	Week	No. of detected faults	No. of corrected faults
1	117	93	26	138	118
2	119	95	27	138	119
3	119	98	28	138	119
4	120	99	29	138	120
5	122	100	30	140	120
6	122	100	31	140	120
7	122	100	32	140	120
8	124	101	33	141	120
9	125	102	34	143	122
10	125	103	35	143	122

(continued on next page)

Week	No. of detected faults	No. of corrected faults	Week	No. of detected faults	No. of corrected faults
11	125	106	36	143	122
12	127	107	37	143	122
13	131	107	38	144	123
14	135	110	39	146	124
15	135	112	40	146	124
16	135	113	41	146	125
17	135	113	42	146	125
18	138	114	43	148	126
19	138	116	44	148	126
20	138	116	45	148	126
21	138	117	46	148	126
22	138	117	47	148	126
23	138	117	48	148	127
24	138	117	49	148	128
25	138	118	50	149	128

Gnome2 Control-center

Week	No. of detected faults	No. of corrected faults	Week	No. of detected faults	No. of corrected faults
2.0x			32	114	81
1	3	1	33	116	82
2	8	4	34	118	84
3	9	5	35	123	85
4	17	5	36	126	85
5	20	5	37	128	85
6	24	10	38	128	85
7	28	14	39	132	86
8	32	17	40	134	87
9	38	22	41	136	87
10	43	24	42	139	87
11	47	29	43	143	87
12	49	32	44	145	88
13	49	32	45	148	100
14	50	35	46	148	100
15	52	38		2.3x	
16	54	39	47	152	103
17	58	42	48	154	103
	2.1x		49	4242	49
18	60	43	50	158	108
19	65	47	51	160	108
20	69	49	52	161	108
21	75	54	53	166	111
22	81	56	54	169	116
23	84	56	55	171	117
24	88	56	56	173	124
25	94	59	57	175	125
26	96	60	58	176	135
27	97	65	59	178	136
28	99	66	60	178	137
29	101	69	61	179	137
30	105	77	62	182	137
	2.2x		63	6651	63
31	108	80			

References

Amin, A., Grunske, L., Colman, A., 2013. An approach to software reliability prediction based on time series modeling. J. Syst. Softw. 86 (7), 1923–1932.

Dai, Y., Xie, M., Long, Q., Ng, S.-H., 2007. Uncertainty analysis in software reliability modeling by Bayesian approach with maximum-entropy principle. IEEE Trans. Softw Eng. 33 (11), 781–795.

Febrero, F., Calero, C., Angeles Moraga, M., 2014. A systematic mapping study of software reliability modeling. Inf. Softw. Technol. 56 (8), 839–849.

Garmabaki, A.H.S., Aggarwal, A.G., Kapur, P.K., 2011. Multi up-gradation software reliability growth model with faults of different severity. In: Proceedings of 2011 IEEE International Conference on Industrial Engineering and Engineering Management. New York. IEEE, pp. 1539–1543.

Gaver, D.P., Jacobs, P.A., 2014. Reliability growth by failure mode removal. Reliab. Eng. Syst. Saf. 130, 27–32.

Goel, A.L., Okumoto, K., 1979. Time-dependent error-detection rate model for software reliability and other performance measures. IEEE Trans. Reliab. R-28 (3), 206–211.

Gokhale, S.S., Lyu, M.R., Trivedi, K.S., 2004. Analysis of software fault removal policies using a non-homogeneous continuous time Markov chain'. Softw. Qual. J. 12 (3), 211–230.

Gratus, V., Pratibha, X., 2013. Multi-release software: an approach for assessment of reliability metrics from field data. In: Prasath, R., Kathirvalavakumar, T. (Eds.), Mining Intelligence and Knowledge Exploration SE-48. Lecture Notes in Computer Science, Springer International Publishing, pp. 475–486.

Hu, Q.P., Peng, R., Xie, M., Ng, S.H., Levitin, G., 2011. Software reliability modelling and optimization for multi-release software development processes. In: Proceedings of *2011* IEEE International Conference on Industrial Engineering and Engineering Management. New York. IEEE, pp. 1534–1538.

Hu, Q.P., Xie, M., Ng, S.H., Levitin, G., 2007. Robust recurrent neural network modeling for software fault detection and correction prediction. Reliab. Eng. Syst. Saf. 92 (3), 332–340.

Huang, C.Y., Hung, T.Y., 2010. Software reliability analysis and assessment using queueing models with multiple change-points. Comput. Math. Appl. 60 (7), 2015–2030.

Huang, C.Y., Lyu, M.R., 2005. Optimal release time for software systems considering cost, testing-effort, and test efficiency. IEEE Trans. Reliab. 54 (4), 583–591.

Jeske, D.R., Zhang, Q., 2007. Assessing the validity of one-part software reliability models using likelihood ratio and early detection tests. J. Syst. Softw. 80 (6), 805–816.

Kapur, P.K., Pham, H., Aggarwal, A.G., Kaur, G., 2012. Two dimensional multi-release software reliability modeling and optimal release planning. IEEE Trans. Reliab. 61 (3), 758–768.

Koch, H.S., Kubat, P., 1983. Optimal release time of computer software. IEEE Trans. Softw. Eng. SE-9 (3), 323–327.

Li, X., Li, Y.F., Xie, M., Ng, S.H., 2011. Reliability analysis and optimal version-updating for open source software. Inf. Softw. Technol. 53 (9), 929–936.

Lin, C.T., Huang, C.Y., 2008. Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models. J. Syst. Softw. 81 (6), 1025–1038.

Lo, J., Huang, C., 2006. An integration of fault detection and correction processes in software reliability analysis. J. Syst. Softw. 79 (9), 1312–1323.

Luan, S.P., Huang, C.Y., 2014. An improved Pareto distribution for modelling the fault data of open source software. Softw. Test. Verif. Reliab. 24 (6), 416–437.

Lyu, M.R., 1996. Handbook of Software Reliability Engineering. Wiley, New York.

Lyu, M.R., 2007. Software reliability engineering: a roadmap. In: Briand, L.C., Wolf, A.L. (Eds.), Proceedings of international Conference on Future of Software Engineering, FoSE 2007, pp. 153–170.

Musa, J.D., Iannino, A., Okumoto, K., 1987. Software Reliability: Measurement, Prediction, Application. McGraw-Hill, Inc., New York.

Ohishi, K., Okamura, H., Dohi, T., 2009. Gompertz software reliability model: estimation algorithm and empirical validation. J. Syst. Softw. 82 (3), 535–543.

Pachauri, B., Dhar, J., Kumar, A., 2015. Incorporating inflection S-shaped fault reduction factor to enhance software reliability growth. Appl. Math. Model. 39 (5–6), 1463–1469.

Pachauri, B., Kumar, A., Dhar, J., 2013. Modeling optimal release policy under fuzzy paradigm in imperfect debugging environment. Inf. Softw. Technol. 55 (11), 1974–1980.

Park, J., Baik, J., 2015. Improving software reliability prediction through multi-criteria based dynamic model selection and combination. J. Syst. Softw. 101, 236–244.

Peng, R., Li, Y.F., Zhang, J.G., Li, X., 2013. A risk-reduction approach for optimal software release time determination with the delay incurred cost. Int. J. Syst. Sci. 46 (9), 1628–1637.

Peng, R., Li, Y.F., Zhang, W.J., Hu, Q.P., 2014. Testing effort dependent software reliability model for imperfect debugging process considering both detection and correction. Reliab. Eng. Syst. Saf. 126, 37–43.

Pham, H., Zhang, X., 1999. A software cost model with warranty and risk costs. IEEE Trans. Computers 48 (1), 71–75.

Rana, R., Staron, M., Berger, C., Hansson, J., Nilsson, M., Törner, F., Meding, W., Höglund, C., 2014. Selecting software reliability growth models and improving their predictive accuracy using historical projects data. J. Syst. Softw. 98, 59–78.

Schneidewind, N.F., 2001. Modelling the fault correction process. In: Proceedings of the 12th International Symposium on Software Reliability Engineering, pp. 185–190.

Smidts, C., Stutzke, M., Stoddard, R.W., 1998. Software reliability modeling: an approach to early reliability prediction. IEEE Trans. Reliab. 47 (3), 268–278.

Tamura, Y., Yamada, S., 2013. Reliability assessment based on hazard rate model for an embedded OSS porting-phase. Softw. Test. Verif. Reliab. 23 (1), 77–88.

Wang, J., Wu, Z., Shu, Y., Zhang, Z., 2015. An imperfect software debugging model considering log-logistic distribution fault content function. J. Syst. Softw. 100, 167–181.

Wu, Y.P., Hu, Q.P., Xie, M., Ng, S.H., 2007. Modeling and analysis of software fault detection and correction process by considering time dependency. IEEE Trans. Reliab. 56 (4), 629–642.

Yamada, S., 2014. Software Reliability Modeling: Fundamentals and Applications. Springer, Tokyo, Japan.

Jianfeng Yang is an Associate Professor in Guizhou University. He received his Ph.D. degree in software engineering and reliability analysis in Guizhou University. His research interests include reliability engineering and web system reliability and availability, and applied statistics.

Yu Liu is a Ph.D. student in the department of Systems Engineering and Engineering Management, City University of Hong Kong. His research interests include software reliability growth model, reliability prediction for software/hardware system. Liu received a MS in nuclear engineering from Tsinghua University.

Min Xie is an elected fellow of IEEE and a Chair Professor of Industrial Engineering at City University of Hong Kong. He received his PhD in Quality Technology in 1987 from Linkoping University in Sweden. He was awarded the prestigious LKY research fellowship in 1991 and he is a Professor at National University of Singapore.

Ming Zhao is a Ph.D. supervisor at Guizhou University. He received his B.S. degree in mathematics from Guizhou University in 1982, M.S. degree in statistics from Wuhan University, China, in 1986, and Ph.D. degree in industrial engineering from Linkoping University, Sweden, 1994. Before he became a Ph.D. candidate at Linkoping University, he had served three years as a reliability engineer at China Academy of Launch Vehicle Technology, Beijing, China. In 1997, he joined the Department of Technology, University of Gavle, Sweden. Since 2002, his research interests include reliability engineering, software quality engineering, and applied statistics.