Software Reliability Analysis using Queuing-based Model with Testing Effort

Nan Zhang
Department of Computer Science and Technology, Harbin Institution of Technology, Harbin, China
Email: zn_ftcl@hit.edu.cn

Gang Cui and Hongwei Liu
Department of Computer Science and Technology, Harbin Institution of Technology, Harbin, China
Email: cg@ftcl.hit.edu.cn

***Abstract*—In this paper, we investigate software fault detection and fault correction processes based on infinite server queuing model which incorporate testing effort functions. Some researches proposed in the literature to study fault detection and fault correction processes. However, most of them do not consider the amount of resources consumed during fault detection and fault correction processes. The consumption amount of resources is usually depicted by testing effort functions which can largely influence fault detection speed and the time to correct a fault detected. Therefore, we will show that new models incorporate testing effort functions into the fault detection and fault correction processes. In additional, we study how to use queuing models to explain the fault detection and fault correction processes during software development. Parameters are estimated and experiments on actual fault data sets are illustrated. The results show that the proposed models in this paper can estimated the number of initial faults better than the model without testing effort functions.**

***Index Terms*—software reliability, fault detection process, fault correction process, testing effort function, queuing model**

I. INTRODUCTION

Software reliability represents a user-oriented view of software quality, which is defined as the probability of fault-free software operation for a specified period of time in a specified environment [1]. To evaluate and predict software reliability, many mathematical models called software reliability growth models (SRGMs) have been proposed to help software engineers to manage software debugging process quantitatively [2-5]. Non-homogeneous Poisson process (NHPP) models, as a class of SRGMs, are extensively used. Moreover, NHPP SRGMs have been quite successful tools in practical software reliability engineering [2].

These SRGMs were developed based on the common assumptions that faults detected are corrected immediately and no new faults are introduced during software debugging process [2-5]. In practice, it is not the case. Each fault is reported, diagnosed, corrected, and then verified [6]. There is a considerable time delay between fault detection and fault correction processes. The time to correct a fault depends on the complexity of the fault, the skill of the debugging team, the available man power, and the software development environment and so on. Therefore, the time delay between the fault detection and fault correction is not negligible. In the past, some research activities showed how to use queuing theory to explain software debugging process [7-11]. For example, Wallace and Coleman [8] modeled the time delay between fault detection and correction by the concept of a fault correction queuing service with exponentially distributed delay - a highly statistically significant empirical result based on Shuttle data. Dohi et al. [9] presented an approach to treat both finite and infinite software reliability models in a unified modeling framework. By introducing an Infinite Server Queuing (ISQ) model to describe the software debugging behavior, they showed that it can involve representative NHPP models with a finite, and an infinite number of faults. Gokhale et al. [10] proposed a single-server queue to evaluate the fault correction activity and showed the benefits of applying multi-priority queuing models to the software defect resolution process. In addition, Huang et al [11] proposed an extended infinite server queuing model with multiple change-points to help managers and developers measure software reliability.

These queuing-based models have achieved a great improvement in the accuracy of assessment of software reliability, however, the amount of resources consumed were ignored during software fault detection and fault correction processes. The consumption amount of resources is the key elements for developers and managers during software debugging phase, which is described by testing effort function (TEF) curve. The testing effort is measured by the man-power, the number of CPU hours, and the number of executed test cases and so on [12]. In recent years, researchers have proposed SRGMs to describe the relationship among the testing time, the amount of testing effort expended, and the

This work is supported by the National High Technology Research and Development Plan of China under Grant No.2008AA01A201
Corresponding author: Nan Zhang

doi:10.4304/jsw.8.6.1301-1307

number of software faults detected [12-16]. However, the influence of testing effort on fault correction process did not consider in these models discussed above [17]. Musa et al. [2] pointed out the resources that governed the pace of software testing were fault identification personnel, fault correction personnel and computer time, and the fault correction personnel resource had the greatest effect on calendar time prediction. Thus, it is reasonable to incorporate the amount of resources consumed into queuing-based models when describing fault detection and fault correction processes.

In this paper, new SRGMs are derived with consideration of TEF during the fault detection and fault correction processes. ISQ models are used to describe software debugging process, where faults detected and fault correction resources are represented as arrival customers and infinite servers, respectively. Moreover, we assume that fault correction commenced with fault detection, i.e., the time delay between fault detection and fault correction is equal to fault correction time. The proposed models are initially formulated for the generally models, and then special models are given to simplify the forms of the general models. The new models relax the unrealistic assumptions of conventional SRGMs, and thus are capable of improving the quality of software reliability prediction and assessment. The remainder of this paper is organized as follows. Section 2 gives models of testing effort functions. Section 3 presents the formulation of ISQ with TEFs. Section 4 provides numerical examples to illustrate the application of the proposed model using a software fault data set. The conclusion is summarized in Section 5.

II. MODELS OF TESTING EFFORT FUNCTION

During software testing phase, it consumes much testing effort, such as man power, CPU time, and number of test cases [13]. The consumed testing effort can indicate how effective software faults are detected. Therefore, resources consumption can be modeled by various distributions.

Let $W(t)$ be the cumulative amount of testing effort expenditures in the testing time interval $(0,t]$ and $f(t)$ be the consumption rate of the testing effort expenditures. According to the assumptions [13], we get the different equation [13]:

$$\frac{dW(t)}{dt} = f(t)\times[\alpha - W(t)] \tag{1}$$

where α is the total amount of testing effort to be eventually consumed. Solving (1), we get

$$W(t) = \alpha\left(1-\exp\left(-\int_0^t f(v)dv\right)\right) \tag{2}$$

and $W(t)$ is the defined as follow [13]:

$$W(t) = \int_0^t w(x)dx \tag{3}$$

where $w(t)$ is the current testing effort consumption at time t. By assigning different values to $w(t)$, we obtain different TEF models. In this paper, we will briefly review two types of testing effort functions (TEFs): Exponentiated Weibull TEF and Logisitic TEF. The Exponentiated Weibull TEF has a great flexibility in accommodating the forms of the consumption rate function and can be used with a wide variety of possible expenditure patterns in actual software projects. Although the Weibull type curve can fit the data well under the general software development environment, it will have an apparent peak phenomenon when the shape parameter m>3. Therefore, Huang [15] used the testing effort consumption using a Logistic curve.

A. Exponentiated Weibull TEF

In recent years, Bokhari and Ahmad used an Exponentiated Weibull curve to describe the amount of effort spent on testing [14]. The Exponentiated Weibull curve includes the Exponential, Rayleigh, Weibull, generalized Exponential and generalized Rayleigh (Burr Type X) curves. The current testing effort consumption at time t is

$$w(t) = \alpha\beta m\theta t^{m-1}\exp\left(-\beta t^m\right)\left[1-\exp\left(-\beta t^m\right)\right]^{\theta-1} \tag{4}$$

where α is the total amount of testing effort expenditures; β is the scale parameter, m and θ are shape parameters. The cumulative testing effort consumed in $(0,t]$ is

$$W(t) = \int_0^t w(x)dx = \alpha\left[1-\exp\left(-\beta t^m\right)\right]^{\theta} \tag{5}$$

We have the following special cases:

For $\theta = 1$ && $m = 1$, there is an Exponential TEF, and the current testing effort consumption at time t is

$$w_{ex}(t) = \alpha\beta\exp(-\beta t) \tag{6}$$

The cumulative testing effort consumed in $(0,t]$ is

$$W_{ex}(t) = \alpha[1-\exp(-\beta t)] \tag{7}$$

For $\theta = 1$ && $m = 2$, there is a Rayleigh TEF, and the current testing effort consumption at time t is

$$w_r(t) = 2\alpha\beta t\exp\left(-\beta t^2\right) \tag{8}$$

The cumulative testing effort consumed in $(0,t]$ is

$$W_r(t) = \alpha\left[1-\exp\left(-\beta t^2\right)\right] \tag{9}$$

For $\theta = 1$, there is a Weibull TEF, and the current testing effort consumption at time t is

$$w_{we}(t) = \alpha\beta m t^{m-1}\exp\left(-\beta t^m\right) \tag{10}$$

The cumulative testing effort consumed in $(0,t]$ is

$$W_{we}(t) = \alpha\left[1-\exp\left(-\beta t^m\right)\right] \tag{11}$$

For $m = 1$, there is a generalized Exponential TEF, and the current testing effort consumption at time t is

$$w_{ge}(t) = \alpha\beta\theta\exp(-\beta t)(1-\exp(-\beta t))^{\theta-1} \tag{12}$$

The cumulative testing effort consumed in $(0,t]$ is

$$W_{ge}(t) = \alpha[1-\exp(-\beta t)]^{\theta} \tag{13}$$

For $m = 2$, there is a generalized Rayleigh TEF, and the current testing effort consumption at time t is

$$w_{gr}(t) = 2\alpha\beta\theta t\exp\left(-\beta t^2\right)\left[1-\exp\left(-\beta t^2\right)\right]^{\theta-1} \tag{14}$$

The cumulative testing effort consumed in $(0,t]$ is

$$W_{gr}(t) = \alpha\left[1 - \exp\left(-\beta t^2\right)\right]^{\theta} \quad (15)$$

B. Logistic TEF

Huang et al. [15] found that although a Weibull-type curve can well fit the data often used in the field of software reliability modeling, it display a "peak" phenomenon when the shape parameter $m > 3$. Hence, they modeled the testing effort consumption using a Logistic curve [15]. Logistic TEF was originally proposed by Parr [16] and it exhibits similar behavior to the Rayleigh curve except for the initial stage of the project. The Logistic current testing effort consumed at time t is given by

$$w_l(t) = \frac{\alpha A\beta \exp(-\beta t)}{[1 + A\exp(-\beta t)]^2} \quad (16)$$

where α is the total amount of testing effort expenditures, β is the consumption rate of testing effort expenditures, and A is a constant. The cumulative testing effort consumption in $(0, t]$ is

$$W_l(t) = \frac{\alpha}{1 + A\exp(-\beta t)} \quad (17)$$

C. Comparisons among Different TEF Models

To compare the performance of different TEFs, the actual software fault data set was used to these TEFs. The data set, reported by Musa et al. [2], is from the System T1 of the Rome Air Development Center (RADC) Project, and shown in Table Ⅰ. The compassion criteria for evaluation are described as following [15, 18]:

(1) $\Pr edictionError_i(PE_i) = (Actual_i - \Pr edicted_i)$

(2) $Bias = \sum_{i=1}^{n} \frac{PE_i}{n}$

(3) $Variation = \sqrt{\frac{\sum_{i=1}^{n}(PE_i - Bias)^2}{n-1}}$

(4) $Root\ Mean\ Square\ \Pr edictionError(RMSPE) = \sqrt{Bias^2 + Variation^2}$

The comparisons among Logistic TEF、Exponential Weibull TEF and Generalized Exponential TEF are illustrated in Table Ⅱ and Figs.1-2. Figs.1-2 illustrate the comparisons between the observed actual software fault data and the data estimated by the Exponential Weibull TEF and Logistic TEF. The computed the PE_i, Bias, Variation, and RMSPE based on the actual data set are shown in Table Ⅱ. Figs 1-2 and Table Ⅱ show that (1) the Logistic TEF and Exponential Weibull TEF yield a batter fit for the data set chosen; (2) the Logistic TEF and Exponential Weibull TEF are adopted for further analysis.

TABLE I.
SYSTEM T1 OF THE ROME AIR DEVELOPMENT CENTER PROJECT

Weeks	CPU hour	Identified Faults	Corrected faults
1	0.00917	2	1
2	0.010	0	1
3	0.003	0	0
4	0.023	1	1
5	0.041	1	1
6	0.004	2	0
7	0.025	1	1
8	0.302	9	2
9	0.973	13	6
10	0.020	2	4
11	0.450	11	1
12	0.250	2	14
13	0.94	11	5
14	1.34	14	19
15	3.32	18	19
16	3.56	12	10
17	2.66	12	12
18	3.77	15	20
19	3.40	6	12
20	2.40	3	2
21	1.80	1	5

TABLE II.
COMPARISON RESULTS FOR DIFFERENT TEF BASED ON SYSTEM T1

TEF	PE_{21}	*Bias*	*Variation*	*RMSPE*
Logistic	0.2035	-0.0746	0.4354	0.4417
Exponential Weibull	0.4666	-0.0431	0.4553	0.4573
Generalized Exponential	0.6431	0.0800	0.5229	0.5290

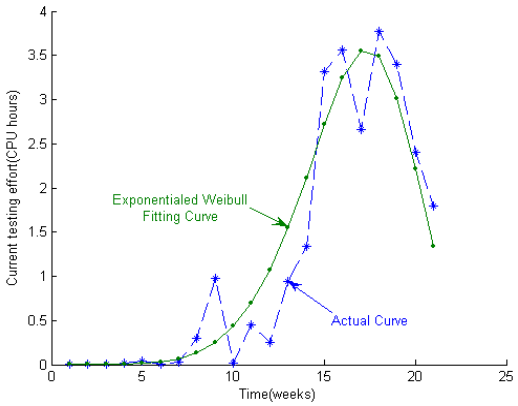

Figure 1. Observed/estimated current testing effort by Exponentiated Weibull TEF vs. time

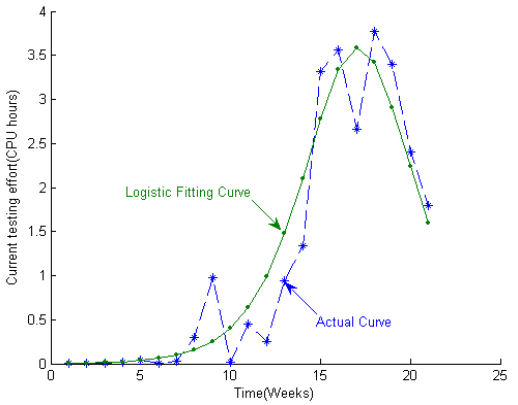

Figure 2. Observed/estimated current testing effort by Logistic TEF vs.time

III. PROPOSED MODELS FRAMEWORK

This section studies how to integrate testing effort into the fault detection and fault correction processes. The formulation of the proposed model is based on the following assumptions [2-5, 7, 11, 15]:

(1) The fault detection process follows a NHPP.
(2) The software is subject to faults at random times, caused by the manifestation of remaining faults in the system.
(3) Testing effort expenditures are described by an EW curve or a Logistic curve.
(4) The mean number of faults detected in the time interval $(t, t+\Delta t]$ by the current testing effort is proportional to the mean number of remaining faults in the system.
(5) The fault correction time is non-negligible so that the number of corrected faults lags behind total number of detected faults.
(6) The proposed queuing model for describing fault detection and correction processes is an ISQ with NHPP arrival and general service time distribution.
(7) No new faults are introduced, when faults are corrected.

Based on assumptions (1)-(3), we have

$$\frac{dm_d(t)}{dt}\times\frac{1}{w(t)} = b(t)\times[a - m_d(t)] \tag{18}$$

where $m_d(t)$ is the expected number of faults detected by time t, a is the expected total number of faults, $b(t)$ is the fault detection rate per unit detection effort, and $w(t)$ is the current effort expenditure at time t. Furthermore we have

$$\frac{d\left(m_d(t)\exp\left(\int_0^t b(s)w(s)ds\right)\right)}{dt} = \frac{d\left(a\exp\left(\int_0^t b(s)w(s)ds\right)\right)}{dt} \tag{19}$$

Solving Eq. (19) under boundary condition $m_d(0)=0$, we get

$$m_d(t) = a\left[1-\exp\left(-\int_0^t b(s)w(s)ds\right)\right] \tag{20}$$

- $b(s)=b$, $w_{ew}(t)=\alpha\beta m\theta t^{m-1}\exp\left(-\beta t^m\right)\left[1-\exp\left(-\beta t^m\right)\right]^{\theta-1}$

$$m_d(t)=a\left(1-\exp\left[-bW_{ew}(t)+bW_{ew}(0)\right]\right)=a\left(1-\exp\left[-bW^*_{ew}(t)\right]\right) \tag{21}$$

- $b(s)=b$, $w_l(t)=\dfrac{\alpha A\beta\exp(-\beta t)}{[1+A\exp(-\beta t)]^2}$

$$m_d(t)=a\left(1-\exp\left[-bW_l(t)+bW_l(0)\right]\right)=a\left(1-\exp\left[-bW_l^*(t)\right]\right) \tag{22}$$

Let $N_d(t)$, $N_c(t)$ be the cumulative number of faults detected and faults corrected by time t, respectively. We assume that there are n faults detected and k faults completely corrected in $(0,t]$. We have

$$\begin{aligned}
&\Pr\{N_c(t)=k, N_d(t)=n\}\\
&=\Pr\{N_c(t)=k \mid N_d(t)=n\}\Pr\{N_d(t)=n\}\\
&=\frac{n!}{k!(n-k)!}p(t)^k(1-p(t))^{n-k}\times\frac{[m_d(t)]^n\exp[-m_d(t)]}{n!}\\
&=\frac{[m_d(t)p(t)]^k\exp[-m_d(t)p(t)]}{k!}\times\frac{[m_d(t)(1-p(t))]^{n-k}\exp[-m_d(t)(1-p(t))]}{(n-k)!}
\end{aligned} \tag{23}$$

So, we obtain

$$\Pr\{N_c(t)=k\} = \frac{[m_d(t)p(t)]^k\exp[-m_d(t)p(t)]}{k!} \tag{24}$$

The mean value function of $N_c(t)$ is given by

$$E[N_c(t)] = m_c(t) = m_d(t)\times p(t) \tag{25}$$

where $m_c(t)$ is the expected number of faults corrected by time t and $p(t)$ is the probability that a fault detected will be completely corrected in $(0,t]$. Let T_1, T_2 denote fault detection time and fault correction time in $(0,t]$, respectively. Moreover, let $G(\bullet)$ is the cumulative distribution function of the service time. We obtain

$$\begin{aligned}
p(t) &= \int_0^t P\{T_2 \le t-\tau \cap T_1=\tau\}d\tau\\
&= \int_0^t P\{T_2 \le t-\tau\}\Pr\{T_1=\tau\}d\tau\\
&= \int_0^t G(t-\tau)\Pr\{T_1=\tau\}d\tau
\end{aligned} \tag{26}$$

The probability that a fault is detected at time τ is

$$P\{T_1=\tau\} = \frac{\lambda_d(\tau)}{m_d(t)} \tag{27}$$

Substituting (26) and (27) into (25), we can get

$$m_c(t) = \int_0^t G(t-\tau)\lambda_d(\tau)d\tau \tag{28a}$$

$$= \int_0^t g(t-\tau)m_d(\tau)d\tau \tag{28b}$$

where $g(\bullet)$ is the probability density function of the service time.

It is well known that human queue service time can be approximated with an exponential distribution [7]. Moreover, Musa et al. [2] pointed out that a real software fault data to illustrate that the exponential distribution is a reasonable model for the distributions of fault correction times. Hence, we assume that the service time is exponential distribution [11]. That is

$$G(t) = 1-\exp(-\eta t) \tag{29a}$$

$$g(t) = \eta\exp(-\eta t) \tag{29b}$$

where η is the service rate. That is to say, η is fault correction rate, and is a constant. In reality, fault correction rate strongly depends on the skill of debuggers or programmers, the amount of faults detected, and the number of correction resource consumed and so on. Particularly, the influence of resource on fault correction should be considered in correction process. Thus, we treat the fault correction rate as the time-dependent behavior of testing effort expenditures to interpret. So, the service rate can be obtained

$$\eta(t) = \eta w(t) \tag{30}$$

where $\eta(t)$ is time-dependent fault correction rate function per unit effort at time t. Substituting (28b) and (21), and (22), we can obtain respectively

$$m_c(t) = \int_0^t m_d(x)\eta w_{ew}(x)\exp(-\eta W_{ew}(t) + \eta W_{ew}(x))dx \tag{31}$$

$$m_c(t) = \int_0^t m_d(x)\eta w_l(x)\exp(-\eta W_l(t) + \eta W_l(x))dx \tag{32}$$

IV. NUMERICAL EXAMPLE

A. Data Description and Comparison Criteria

The real data set is from the System T1 of the RADC Project in [2] and shown in Table Ⅰ. System T1 was used for a real time command and control application. The size of software was about 21700 object instructions. It took 21 weeks' testing, 25.3 CPU hours, 9 programmers, and 136 software faults corrected.

To give quantitative comparisons, some criteria were used to judge the performance of the proposed model. The comparison criteria are the MSE and R-squared which are defined as follows [2].

(1) Mean Squared Errors (*MSE*)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(m_i - \hat{m}(t_i))^2 \tag{33}$$

(2) R-squared

$$R-squared = \frac{\sum_{i=1}^{n}(\hat{m}(t_i) - \overline{m})^2}{\sum_{i=1}^{n}(m_i - \overline{m})^2}, \quad \overline{m} = \frac{1}{n}\sum_{i=1}^{n} m_i \tag{34}$$

where m_i is the total number of fault detected by t_i, and $\hat{m}(t_i)$ is the estimated cumulative number of faults by time t_i obtained from the fitted mean value function. The lower MSE value means the better goodness-of-fit of the mode. The R-squared can take on any value between 0 and 1, with a value closer to 1 indicating a better fit.

B. Performance Analysis

This section evaluates the performance of proposed models and several existing models. First of all, parameters of selected models are estimated and the related mean value functions are obtained. Secondly, all the selected models are compared with each other based on objective criteria. In the past, fault detection and correction resources are not considered as separate resources, and the data sets of effort are not treated as separate. So, we assumed the detection effort and the correction effort were called testing effort. According to the comparisons among various TEFs are illustrated in section 2, the Exponentiated Weibull TEF and Logistic TEF are chosen to fit the actual software fault data set curve. The parameters α, β, m, θ and A in TEFs can be estimated by least squares estimation. The estimators for $\hat{\alpha}, \hat{\beta}, \hat{m}, \hat{\theta}$ and $\hat{A}$ are investigated for testing effort w_i spent at testing time $t_i (i = 1,2,\cdots,21)$ and are listed in Table Ⅲ.

TABLE III.
THE ESTIMATED PARAMETERS OF EW TEF AND LOGISTIC TEF

TEF	Estimated Parameters
Exponentiated Weibull TEF[14]	$\hat{a}$ =26.83, $\hat{\beta}$ =6.165e-9, $\hat{\theta}$ =0.95, $\hat{m}$ =6.5508
Logistic TEF[13]	$\hat{a}$ =29.1065, $\hat{\beta}$ =0.493515, $\hat{A}$ =4624.89

Using the estimated parameters $\hat{\alpha}, \hat{\beta}, \hat{m}, \hat{\theta}$ and $\hat{A}$, the other parameters a, b and η can be estimated by the method of least squares estimation. Table Ⅳ compares the performance of various SRGMs for the actual fault data set investigated in this paper. Due to the limitations of paper size, only 2 models [13, 19] are used for detailed discussions. In [13], Huang et al. proposed a SRGM incorporated the Logistic TEF. However, the time to correct a fault is negligible. In [19], Lo developed a general framework of the fault detection and fault correction processes, but the amount of resources consumed was ignored during these two processes. These have better performance as shown in Table Ⅳ. As shown in Table Ⅳ, the proposed models in this paper provide the lower value of MSE and the highest value of R-square than other models. Figs.3-4 plotted the actual fault data and the estimated mean value function up to time t. Fig.3 show that the Exponentiated Weibull TEF incorporates into ISQ SRGM. Fig.4 show that the Logistic TEF incorporates into ISQ SRGM. From Figs. 3-4, we can see that these models fit the observed data better, and predict the future behavior well for this actual fault data set. Thus, from these simulation/comparison results, the proposed model performs appreciably better than the others.

V. CONCLUSION

Many resources are consumed by a software development project. Most papers assumed that the consumption rate of testing resource expenditures during the testing and debugging phase is a constant or even do not consider such testing effort. In reality, software reliability models should be developed by incorporating different testing-effort functions. In this paper, we have shown how to apply infinite server queuing models with TEFs to model the fault detection and correction processes. New SRGMs with an Exponentiated Weibull TEF and a Logistic TEF are proposed that can consider the influence of resources on software debugging phase, and enhance the prediction and assessment of software reliability. Experiments are performed based on real software fault data set. Comparing with some selected SRGMs, experimental results show that the proposed models give a better fit to the observed data.

ACKNOWLEDGMENT

This work is supported by and the National high Tech Research and Development Plan of China under Grant No.2008AA01A201

TABLE IV.
COMPARISON RESULTS OF DIFFERENT SRGMS FOR DATA SET

Models	Estimated Parameters	MSE	R-square
$m_d(t)$ in Equation (21) $m_c(t)$ in Equation (31)	$\hat{a}$ =139.6 $\hat{b}$ =0.1297 $\hat{\eta}$ =0.8794	MSE_d= 42.94 MSE_c= 60.4762	R-square$_d$= 0.9826 R-square$_c$= 0.9745
$m_d(t)$ in Equation (22) $m_c(t)$ in Equation (32)	$\hat{a}$ =141.3 $\hat{b}$ =0.1293 $\hat{\eta}$ =0.8628	MSE_d= 32.17 MSE_c= 58.6667	R-square$_d$= 0.9869 R-square$_c$= 0.9753
Huang et al.[13]	$\hat{a}$ =138.026 $\hat{b}$ =0.1451	MSE= 62.41	R-square= 0.9758
Lo [19]	$\hat{a}$ =133.119 $\hat{b}$ =0.1597 $\hat{c}$ =0.7825	MSE= 58.70	R-square= 0.9812

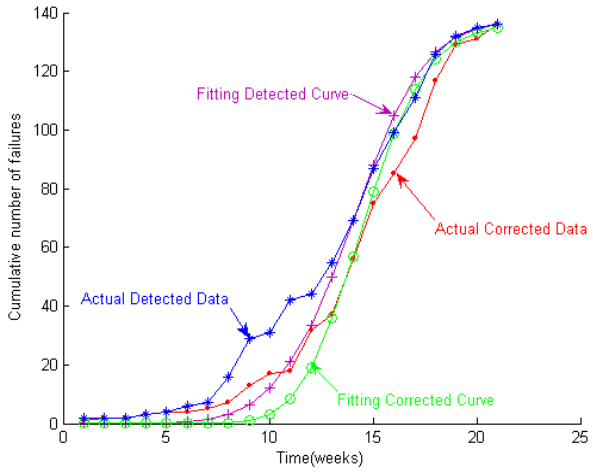

Figure 3. Observed/estimated cumulative number of faults using the Exponentiated Weibull TEF

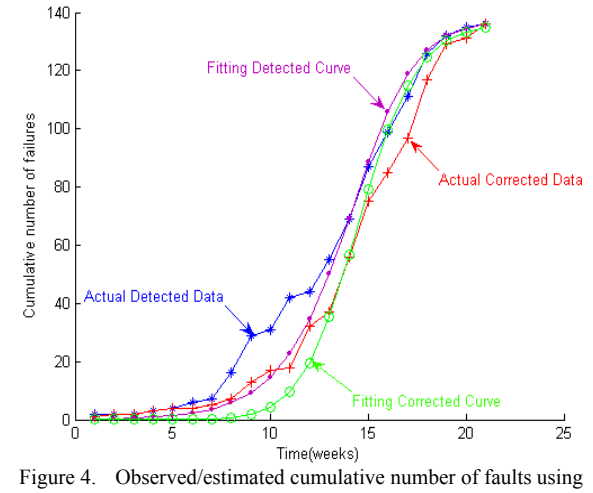

Figure 4. Observed/estimated cumulative number of faults using the Logistic TEF

REFERENCES

[1] American Institute of Aeronautics and Astronautics, *Recommended Practice for Software Reliability*. ANSI/AIAA R-013-1992, Feb 1993.

[2] J. D. Musa, A. annino, and K. Okumoto, *Software Reliability, Measurement, Prediction and Application*. New York: McGram Hill, 1987.

[3] M. Xie, Software *Reliability Modeling*. World Scientific Publishing Company, 1991.

[4] M. R. Lyu, *Handbook of Software Reliability Engineering*. New York: McGram Hill, 1996.

[5] H. Pham, *Software Reliability*. New York: Springer-Verlag, 2000.

[6] M. Haug, E. W. Olsen, and L. Consolini, *Software quality approaches: testing, verification and validation*. Berlin: Springer, 2001.

[7] K. S. Trivedi, *Probability and statistics with reliability, queuing, and computer science application*. 2nd ed. John Wiley and Sons, 2002.

[8] D. Wallace and C. Coleman, "Application and improvement of software reliability models," Technical Report, Software Assurance Technology Center, NASA Goddard Space Flight Center, 2001. http://satc.gsfc.nasa.gov/support/index.php.

[9] T. Dohi, S. Osaki, and K. S. Trivedi, "An infinite server queuing approach for describing software reliability growth: unified modeling and estimation framework," Proc. of the 11th Asia-Pacific Software Engineering Conference (APSEC 04), Korea, pp.110-119, 2004.

[10] S. S. Gokhale and R. E. Mullen, "Queuing models for field defect resolution process," Proc. of the 17th IEEE International Symposium on Software Reliability Engineering (ISSRE 06), USA, pp.353-362, 2006.

[11] C. Y. Huang and T. Y. Huang, "Software reliability analysis and assessment using queuing models with multiple change-points," Computers and Mathematics with Applications, vol. 60, pp. 2015-2030, 2010. doi:10.1016/j.camwa.2010.07.039

[12] S. Yamada, J. Hishitani, and S. Osaki, "Software reliability growth with a Weibull test-effort: a model & application," IEEE Transactions on Reliability, vol. 42, pp.100-105, 1993. doi: 10.1109/24.210278

[13] C. Y. Huang, S. Y. Kuo, and I. Y. Chen, "Analysis of a software reliability growth model with logistic testing-effort function," Proc. of the 8th International Symposium on Software Reliability Engineering (ISSRE 1997), USA, pp. 378-388, 1997.

[14] N. Ahmad, M.U. Bokhari, S.M.K. Quadri, and M.G.M. Khan, "The Exponentiated Weibull software reliability growth model with various testing-efforts and optimal release policy," International Journal of Quality &Reliability Management; vol. 25, pp. 211-235, 2008. doi: 10.1108/02656710810846952

[15] C. Y. Huang and S. Y. Kuo, "Analysis of incorporating Logistic testing-effort function into software reliability modeling," IEEE Transactions on Reliability; vol. 51, pp. 261-270, 2002. doi: 10.1109/32.624305

[16] F. N. Parr, "An alternative to the Rayleigh curve for software development effort," IEEE Transactions on Software Engineering, vol. 6, pp. 291-296, 1980. doi:10.1109/TSE.1980.230475

[17] R. Peng, Q. P. Hu, S. H. Ng, and M. Xie, "Testing effort dependent software FDP and FCP models with consideration of imperfect debugging," Proc. of the 4th International Conference on Secure Software Integration

and Reliability Improvement (SSIRI 2010), Singapore, pp.141-146, 2010.
[18] K. Pillai and V.S.S. Nair, "A model for software development effort and cost estimation," IEEE Transactions Software Engineering, vol. 23, pp. 485-497, 1997. doi:10.1109/32.624305
[19] J.H. Lo, "An integrated framework of the modeling of fault-detection and fault-correction processes in software reliability analysis," Proc. of IEEE International Conference on Industrial Informatics (INDIN 2008), Korea, pp. 557-562, 2008.

Nan Zhang was born in 1980 in China. In 2003, she received her B.S. degree in Department of Computer Science and Technology from Harbin Institute of Technology at Harbin. She earned her M.S. degree in 2008 in Computer Application for Northeast Forestry University at Harbin. She is currently a Ph.D. candidate in Department of Computer Science and Technology at Harbin Instituted of Technology. Her main research interests include software testing, software reliability evaluation, fault tolerance computing and mobile computing.

Gang Cui was born in 1947 in China. He earned his M.S. degree in 1989 and B.S. degree in 1976, both in Computer Science and Technology from Harbin Institute of Technology at Harbin. He is currently a professor and Ph.D. supervisor in School of Computer Science and Technology at Harbin Institute of Technology. He is a member of technical committee of fault tolerant computing of the computer society of China. His main research interests include fault tolerance computing, mobile computing, software testing. Prof. Cui has implemented several projects from the National 863 High-Tech Project and has won 1 First Prize, 2 Second Prizes and 3 Third Prizes of the Ministry Science and Technology Progress. He is a senior member of the CCF. He has published over 50 papers and one book.

Hongwei Liu was born in 1971 in China. He received the B.Sc., M.Sc., and Ph.D. degrees of engineering of computer architecture from the Harbin Institute of Technology, Heilongjiang China, in 1994, 2000, and 2004, respectively. He is currently a professor and Ph.D. supervisor in School of Computer Science and Technology at Harbin Institute of Technology. He is a member of technical committee of fault tolerant computing of the computer society of China. His main research interests are fault-tolerant computing, software reliability evaluation, software testing. He is a senior member of the CCF. He has published over 60 papers.