Available online at www.sciencedirect.com

SCIENCE DIRECT®

The Journal of Systems and Software 79 (2006) 1312–1323

The Journal of Systems and Software

www.elsevier.com/locate/jss

An integration of fault detection and correction processes in software reliability analysis

Jung-Hua Lo [a,*], Chin-Yu Huang [b]

[a] Department of Information Management, Lan Yang Institute of Technology, ILan, Taiwan
[b] Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan

Received 2 August 2004; received in revised form 6 December 2005; accepted 10 December 2005
Available online 19 January 2006

Abstract

Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment and is widely recognized as one of the most significant aspects of software quality. Over the past 30 years, many software reliability growth models (SRGMs) have been proposed and they can greatly help us to estimate some important measures such as the mean time to failure, the number of remaining faults, defect levels, and the failure intensity, etc. Besides, SRGMs can also help to determine person power needed to support the desired reliability requirements. However, from our studies, most of SRGMs only focus on describing the behavior of fault detection process and assume that faults are fixed immediately upon detection. In fact, this assumption may not be realistic. Thus, in this paper, we will propose a general framework for modeling the software fault detection and correction processes. We will also show that the proposed approaches cover a number of well-known SRGMs. Two numerical examples based on two real software failure data sets are presented and discussed in detail.
© 2005 Published by Elsevier Inc.

Keywords: Software fault detection and correction processes; Non-homogeneous Poisson process (NHPP); Mean value function (MVF); Software reliability growth model (SRGM)

1. Introduction

Over the past decade, the deployment of computer systems has grown dramatically. People in the modern society are increasingly dependent on both hardware and software systems. Since software is embedded in everything and permeates our daily life, the correct performance of a software system becomes an important issue of many critical systems. Software reliability can be viewed as a good measure of quantifying software failures and is defined as the probability of failure-free software operation for a specified period of time in a specified environment (Lyu, 1996). Numerous software reliability growth models (SRGMs) have been developed to measure software reliability, and some of them are based on NHPP (Lyu, 1996; Musa et al., 1987; Pham, 2000; Xie, 1991). These SRGMs are very useful to describe the error-detection process as a discrete or continuous process at a time-dependent error-detection rate (Goel and Okumoto, 1979; Lo et al., 2001, 2003; Yamada et al., 1983, 1993). The common assumption of conventional SRGMs is that the detected faults will be immediately removed. However, this assumption may not be very realistic, that is, it is rare to see that the detected faults are immediately corrected (Gokhale et al., 1998; Schneidewind, 1975, 2003; Ohba, 1984; Xie and Zhao, 1992).

Schneidewind (1975) first modeled the fault-correction process by using a constant delayed fault-detection process. Later, Xie and Zhao (1992) extended the Schneidewind model to a continuous version by substituting a time-dependent delay function for the constant delay. A key factor of the continuous version of Schneidewind model

* Corresponding author. Tel.: +886 3 9771997x845; fax: +886 3 9771069.
E-mail address: losir@mail.fit.edu.tw (J.-H. Lo).

Nomenclature

Acronyms

AE	accuracy of estimation
G–O	Goel–Okumoto
KS	Kolmogorov–Smirnov statistic
MLE	maximum likelihood estimation
MRE	mean of relative errors
MSF	mean of square faults
MVF	mean value function
NHPP	non-homogeneous Poisson process
RE	relative errors
SRGM	software reliability growth model

Notations

a	expected number of initial faults
$d(t)$	error detection rate per error
$E(\cdot)$	expected value function
M_a	actual cumulative number of detected faults during the testing process and after the test
$m(t)$	expected mean number of faults detected in time $(0,t)$
$m_c(t)$	expected mean number of faults corrected in time $(0,t)$
$N(t)$	cumulative number of errors detected by time t
Pr()	probability function
sup()	the least upper bound
$\lambda(t)$	fault detection rate per remaining fault, $m'(t)/(a-m(t))$
$\mu(t)$	fault correction rate per detected but not corrected fault, $m_c'(t)/(m(t)-m_c(t))$

is the time-dependent delay function, which measures the expected time lag to correct a detected fault. Actually, software debugging is a science. Fault correction personnel have to formulate a hypothesis and make predictions based on the hypothesis. Furthermore, they should run the software, observe its output, and confirm the hypothesis. We know that the time to remove a fault depends on the complexity of the detected faults, the skills of the debugging team, the available manpower, and the software development environment, etc. (Lyu, 1996; Musa et al., 1987; Musa, 1998). Therefore, it is very important for us to have different software reliability models for modeling the fault detection and correction processes. In this paper, we will propose a new software reliability model considering both the fault detection and correction processes. Some numerical examples are performed based on two real software failure data sets. Experimental results show that the proposed framework to incorporate debugging time lag for SRGM has a fairly accurate prediction capability.

This paper is organized as follows. In Section 2, the properties of the related models are reviewed and a description of characteristics of the NHPP models is discussed. An integration model of fault detection and correction processes is proposed in Section 3. Also, we show how some existing NHPP models are re-evaluated from the viewpoint of correction process and make some observations between the original NHPP models and the integrated models. The numerical examples and comparison results are presented in Section 4. Finally, the conclusions are made in Section 5.

2. Related works

2.1. A brief review of some SRGMs based on NHPP

Let $\{N(t),\ t \geqslant 0\}$ denote a counting process representing the cumulative number of faults detected by time t, $m(t)$ be the mean value function (MVF) of the expected number of faults detected in time $(0,t]$, and $\lambda(t)$ denote the failure intensity at testing time t. That is, they satisfy the following:

$$m(t) = E[\{N(t), t \geqslant 0\}] \tag{1}$$

and

$$\lambda(t) = \frac{\mathrm{d}m(t)}{\mathrm{d}t}. \tag{2}$$

Thus, an SRGM based on NHPP with mean value function $m(t)$ can be formulated as (Yamada et al., 1993)

$$P\{N(t)=n\} = \frac{m(t)^n}{n!}\exp(-m(t)), \quad n = 0,1,2,\ldots \tag{3}$$

From our studies (Lo et al., 2001, 2002; Lyu, 1996; Musa et al., 1987; Musa, 1998; Xie, 1991, 2000; Pham, 2000; Ohba, 1984), most existing SRGMs based on NHPP have the following basic assumptions concerning the software fault detection process:

(1) The cumulative number of failures follows a NHPP with mean value function, $m(t)$. The software fault detection rate at any time is proportional to the expected number of undetected faults at that time.
(2) The proportionality may change with time.
(3) The number of detected faults in each of the respective intervals is independent.
(4) Each time a failure occurs, the fault that caused it is perfectly removed with no new faults being introduced.
(5) The detected faults are immediately removed with certainty and correction of faults takes only negligible time.

According to above assumptions, we have

$$m(t+\Delta t) - m(t) = d(t) \times [a - m(t)]\Delta t, \tag{4}$$

where a is the expected number of faults to be eventually detected (i.e., $m(\infty) = a$) and $d(t)$ is the fault detection rate. Solving Eq. (4) and using Eq. (2), we obtain

$$\lambda(t) = d(t) \times (a - m(t)) \tag{5}$$

and

$$m(t) = a + (m(0) - a)\exp\left(-\int_0^t d(u)\,\mathrm{d}u\right). \tag{6}$$

In fact, we can have different SRGMs based on NHPP by using different fault detection rate (Pham, 2000).

2.2. Some SRGMs based on NHPP

In this section, we will show that some classical SRGMs based on NHPP can be easily derived from Eq. (6) with various fault detection rate.

2.2.1. The Goel–Okumoto model

The most well-known SRGM based on NHPP was the model proposed by Goel and Okumoto (1979; Xie, 1991; Yamada et al., 1993). The Goel–Okumoto (G–O) model assumed that the fault detection rate in the testing phase is constant. Thus, it is identical to take $d(t) = b$ in Eq. (6) and the MVF is

$$m(t) = a[1 - \exp(-bt)], \quad a > 0,\ b > 0. \tag{7}$$

2.2.2. The Yamada delayed S-shaped model

The Yamada delayed S-shaped (DSS) model (Yamada et al., 1983) is an S-shaped curve for the cumulative number of detected faults such that the failure rate initially increases and later decreases. Yamada et al. assumed that the fault detection rate is a time-dependent function described by an S-shaped curve because the testers' skills will gradually improve as time goes by (Xie, 1991). That is,

$$d(t) = \frac{b^2 t}{1 + bt}.$$

Thus, the MVF is

$$m(t) = a[1 - (1 + bt)\exp(-bt)], \quad a > 0,\ b > 0. \tag{8}$$

2.2.3. The inflected S-shaped model

This inflected S-shaped (ISS) model assumed that faults in a program are mutually dependent so that some faults are not detectable before some others are removed (Ohba, 1984; Xie, 1991). If the fault detection rate is $\frac{bc}{(1+c\exp(-bt))(1+c)}$, then the MVF is given by

$$m(t) = a\left(\frac{1 - \exp(-bt)}{1 + c\exp(-bt)}\right), \quad a > 0,\ b > 0,\ c > 0. \tag{9}$$

Note that the parameter c is the inflection factor that indicates the ratio of the number of detectable faults to the total number of faults in the program.

2.2.4. The Yamada Weibull model

Yamada et al. (1993, Xie, 1991) proposed a SRGM incorporating the amount of Weibull-type testing-effort expended during the software-testing phase. If the fault detection rate is $b\alpha\beta\gamma t^{\gamma-1}\exp(-\beta t^{\gamma})$, then the MVF is

$$a\{1 - \exp[-b\alpha(1 - \exp(-\beta t^{\gamma}))]\},$$
$$a > 0,\ b > 0,\ \alpha > 0,\ \beta > 0,\ \gamma > 0, \tag{10}$$

where α is the total amount of testing-effort required by software testing, β is the scale parameter, and γ is the shape parameter.

2.2.5. The Yamada exponential model

For the Yamada Weibull model (i.e., Eq. (10)), when $\gamma = 1$, we obtain the Yamada exponential model (Yamada et al., 1993). Actually, the exponential model is a special case of the Weibull model. Furthermore, if the fault detection rate is $b\alpha\beta\exp(-\beta t)$, the MVF can be derived as follows:

$$a\{1 - \exp[-b\alpha(1 - \exp(-\beta t))]\},$$
$$a > 0,\ b > 0,\ \alpha > 0,\ \beta > 0. \tag{11}$$

2.2.6. The Yamada Rayleigh model

Similarly, the Yamada Rayleigh model is a special case of the Yamada Weibull model if $\gamma = 2$ (Yamada et al., 1993). If the fault detection rate is $2b\alpha\beta t\exp(-\beta t^2)$, then MVF is given by

$$a\{1 - \exp[-b\alpha(1 - \exp(-\beta t^2))]\},$$
$$a > 0,\ b > 0,\ \alpha > 0,\ \beta > 0. \tag{12}$$

3. An integrated fault detection and correction model

In the past, much research on software reliability models has concentrated on modeling and predicting failure occurrence and has not given equal priority to modeling the fault correction process (Schneidewind, 2003). However, most latent software faults may remain uncorrected for a long time even after they are detected, which increases their impact. The remaining software faults are often one of the most unreliable reasons for software quality. Therefore, from the practical viewpoint, we may assume that the mean number of faults corrected in the time interval $(t, t + \Delta t]$ is proportional to the mean number of detected but not yet corrected faults remaining in the system. In general, the proportionality, $\mu(t)$, is time dependent (Xie and Zhao, 1992). Thus, the MVF of fault detection process, $m(t)$ and fault correction process, $m_c(t)$, examined up to time t, can be expressed in terms of the following differential equations:

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t} = \lambda(t)(a - m(t)), \quad a > 0 \tag{13}$$

and

$$\frac{\mathrm{d}m_c(t)}{\mathrm{d}t} = \mu(t)[m(t) - m_c(t)]. \tag{14}$$

Therefore, we see that $\lambda(t)$ can be rewritten as

$$\lambda(t) = \frac{m'(t)}{a - m(t)} \tag{15}$$

and can be interpreted as the fault detection rate per remaining fault. In particular, solving Eq. (13) with $\lambda(t) = b$ under the initial condition $m(0) = 0$, we have

$$m(t) = a[1 - \exp(-bt)], \quad a > 0, \ b > 0.$$

Obviously, it is the G–O model. Similarly, $\mu(t)$ in Eq. (14) can be given by

$$\mu(t) = \frac{m_c'(t)}{m(t) - m_c(t)}. \tag{16}$$

Thus, $\mu(t)$ can be treated as the fault correction rate per detected but not corrected fault.

Proposition 1. *If* $D(t) = \int_0^t \lambda(s)\,\mathrm{d}s$, $C(t) = \int_0^t \mu(s)\,\mathrm{d}s$, *and the differential equations for the MVF* $m(t)$ *and* $m_c(t)$ *of fault detection and correction processes are*

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t} = \lambda(t)(a - m(t))$$

and

$$\frac{\mathrm{d}m_c(t)}{\mathrm{d}t} = \mu(t)[m(t) - m_c(t)],$$

then we have

$$m(t) = a[1 - \exp(-D(t))], \tag{17}$$

and

$$m_c(t) = \exp(-C(t))\left\{\int_0^t ac(s)\exp C(s)[1 - \exp(-D(s))]\,\mathrm{d}s\right\}. \tag{18}$$

Note that the initial condition $m(0) = m_c(0) = 0$, *i.e., no failure at the beginning.*

Proof. Solving Eqs. (13) and (14) by multiplying both sides with $\exp(D(t))$, we get

$$\exp(D(t))\frac{\mathrm{d}}{\mathrm{d}t}(m(t)) + \lambda(t)m(t)\exp(D(t)) = a\lambda(t)\exp(D(t)).$$

Therefore,

$$\frac{\mathrm{d}}{\mathrm{d}t}(m(t)\exp(D(t))) = a\frac{\mathrm{d}}{\mathrm{d}t}(\exp(D(t))).$$

Since $D(0) = 0$, we have

$$m(t) = \exp(-D(t))[a\exp(D(t)) - a] = a(1 - \exp(-D(t))).$$

On the other hand, for Eq. (14), we multiply both sides with $\exp(C(t))$. In this case, we have

$$\begin{aligned}&\exp(C(t))\frac{\mathrm{d}}{\mathrm{d}t}(m_c(t)) + \mu(t)m_c(t)\exp(C(t))\\ &\quad = a\mu(t)\exp(C(t)) - a\mu(t)\exp(C(t) - B(t)).\end{aligned}$$

Thus,

Table 1
Various models corresponding to different $d(t)$ and $\mu(t)$

Models	$d(t)$	$\mu(t)$ (if $\lambda(t) = b$)
Model in Proposition 3	$\frac{bc(\exp(-ct)-\exp(-bt))}{b\exp(-bt)-c\exp(-bt)}$	c
G–O model	b	N/A
Yamada delayed S-shaped model	$\frac{b^2t}{1+bt}$	b
Inflected S-shaped model	$\frac{bc}{(1+c\exp(-bt))(1+c)}$	$\frac{b(1+c)}{c(1+c\exp(-bt))(1-\exp(-bt))}$
Yamada exponential model	$b\alpha\beta\exp(-\beta t)$	$\frac{b\alpha\beta\exp(-\beta t)\exp(-b\alpha(1-\exp(-\beta t)))}{\exp(-b\alpha(1-\beta t))-\exp(-bt)}$
Yamada Rayleigh model	$2b\alpha\beta t\exp(-\beta t^2)$	$\frac{2b\alpha\beta t\exp(-\beta t^2)\exp(-b\alpha(1-\exp(-\beta t^2)))}{\exp(-b\alpha(1-\beta t^2))-\exp(-bt)}$
Yamada Weibull model	$b\alpha\beta\gamma t^{\gamma-1}\exp(-\beta t^{\gamma})$	$\frac{b\alpha\beta\gamma t^{\gamma-1}\exp(-\beta t^{\gamma})\exp(-b\alpha(1-\exp(-\beta t^{\gamma})))}{\exp(-b\alpha(1-\beta t^{\gamma}))-\exp(-bt)}$

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}t}(m_c(t)\exp(C(t))) = a\int_0^t c(s)(\exp(C(s)))\\ &\quad(1 - \exp(-D(s)))\,\mathrm{d}s.\end{aligned}$$

Finally, we have the result,

$$m_c(t) = \exp(-C(t))\left\{\int_0^t ac(s)\exp C(s)[1 - \exp(-D(s))]\,\mathrm{d}s\right\}.$$

Proposition 2. *If* $\frac{\mathrm{d}m(t)}{\mathrm{d}t} = b(a - m(t))$ *and* $\frac{\mathrm{d}m_c(t)}{\mathrm{d}t} = b[m(t) - m_c(t)]$, *then we have*

$$m(t) = a[1 - \exp(-bt)]$$

and

$$m_c(t) = a[1 - (1 + bt)\exp(-bt)]. \tag{19}$$

Proposition 3. *If* $\frac{\mathrm{d}m(t)}{\mathrm{d}t} = b(a - m(t))$ *and* $\frac{\mathrm{d}m_c(t)}{\mathrm{d}t} = c[m(t) - m_c(t)]$, *the we have*

$$m(t) = a[1 - \exp(-bt)]$$

and

$$m_c(t) = a\left[1 + \frac{c}{b - c}\exp(-bt) - \frac{b}{b - c}\exp(-ct)\right]. \tag{20}$$

Table 1 shows the relationship between SRGMs that can be derived by using various $d(t)$ and $\mu(t)$. Actually, from Table 1, we can see that many existing SRGMs based on NHPP can be re-examined and re-evaluated from the viewpoint of software correction process.

4. Numerical and data analysis

4.1. Descriptions of real data sets

In this section, we evaluate the performance of the proposed model by using two sets of software failure data. The first data set is the System T1 data of the Rome Air Development Center (RADC) (Musa, 1985; Musa et al., 1987) and shown in Table 2. The system T1 is used for a real-time

Table 2
DS1

Weeks	Execution time (CPU hour)	Cumulative execution time	Identified faults	Cumulative number of detected faults	Corrected faults	Cumulative number of corrected faults
1	0.00917	0.00917	2	2	1	1
2	0.010	0.01917	0	2	1	2
3	0.003	0.02217	0	2	0	2
4	0.023	0.04517	1	3	1	3
5	0.041	0.08617	1	4	1	4
6	0.004	0.09017	2	6	0	4
7	0.025	0.11517	1	7	1	5
8	0.302	0.41717	9	16	2	7
9	0.973	1.39017	13	29	6	13
10	0.020	1.41017	2	31	4	17
11	0.450	1.86017	11	42	1	18
12	0.250	2.11017	2	44	14	32
13	0.94	3.05017	11	55	5	37
14	1.34	4.39017	14	69	19	56
15	3.32	7.71017	18	87	19	75
16	3.56	11.27017	12	99	10	85
17	2.66	13.93017	12	111	12	97
18	3.77	17.70017	15	126	20	117
19	3.40	21.10017	6	132	12	129
20	2.40	23.50017	3	135	2	131
21	1.80	25.30017	1	136	5	136

command and control application. The size of the software is approximately 21,700 object instructions. It took 21 weeks, and nine programmers to complete the test. During the test phase, about 25.3 CPU hours were consumed, and 136 software faults were removed. The number of faults removed after 3.5 years of test is 188 (Kapur and Younes, 1995). Besides, Table 3 gives the second data set and it is from a study by Ohba (1984). The system is a PL/I database application software consisting of approximately 1,317,000 lines of code. During 19 weeks, 47.65 CPU hours were consumed, and 328 software faults were removed. The total cumulative number of detected faults after a long time of testing is 358.

4.2. Criteria for model comparison

The comparison criteria we use to judge models' performance are described as follows:

(1) The *accuracy of estimation* (AE) is defined as (Musa et al., 1987)

$$\left|\frac{M_{\mathrm{a}}-a}{M_{\mathrm{a}}}\right|, \tag{21}$$

where M_{a} is the actual cumulative number of detected faults during the testing process and after the test, and a is the estimated number of initial faults. Generally, M_{a} is obtained from software fault tracking after software testing. In addition to the better fit to the actual observed data, a good model should have the ability to predict the behavior of future failures well.

(2) The *mean of square faults* (MSF) is defined as (Lyu and Nikora, 1992)

$$\mathrm{MSF}=\frac{1}{n}\sum_{k=1}^{n}(m(t_k)-m_k)^2. \tag{22}$$

Table 3
DS2

Weeks	Cumulative execution time (CPU hours)	Cumulative number of detected faults	Weeks	Cumulative execution time (CPU hours)	Cumulative number of detected faults
1	2.45	15	11	26.23	233
2	4.9	44	12	27.67	255
3	6.86	66	13	30.93	276
4	7.84	103	14	34.77	298
5	9.52	105	15	38.61	304
6	12.89	110	16	40.91	311
7	17.1	146	17	42.67	320
8	20.47	175	18	44.66	325
9	21.43	179	19	47.65	328
10	23.35	206			

From the measure of MSF, we can obtain quantitative comparisons for long-term predictions between actual and predicted values. A lower MSF indicates better performance of fitting the real software data.

(3) The capability of the model to predict failure behavior from present and past failure behavior is called *predictive validity*, which can be represented by computing the *relative errors* (RE) for a data set (Musa et al., 1987; Huang, 2005):

$$\mathrm{RE} = \frac{m(t_Z) - Z}{Z}. \tag{23}$$

Suppose Z failures were observed by the end of test time t_Z, we use the failure data up to time t_i ($t_i \leqslant t_Z$) to estimate the parameters of $m(t)$. Moreover, a lower absolute value of RE indicates a better performance of fitting the real software data.

(4) The *mean of relative errors* (MRE) is defined as

$$\mathrm{MRE} = \frac{1}{l}\sum_{t_Z=t_1}^{t_l}\left|\frac{m(t_Z) - Z}{Z}\right|. \tag{24}$$

Note t_i is the time when the observed i failure is detected and l is the cumulative number of detected faults after the test. Besides, in order to check the predictive validity of the model, the procedure can be

Table 4
Comparison results for different models (DS1)

Model	AE (%)	MSF	|RE|	MRE	KS
$m_1(t)$	29.19	27.15	0.0384	0.3099	0.2246
$m_2(t)$	18.42	18.19	0.0138	0.3676	0.1654
$m_3(t)$	33.72	78.81	0.0894	0.5142	0.2104

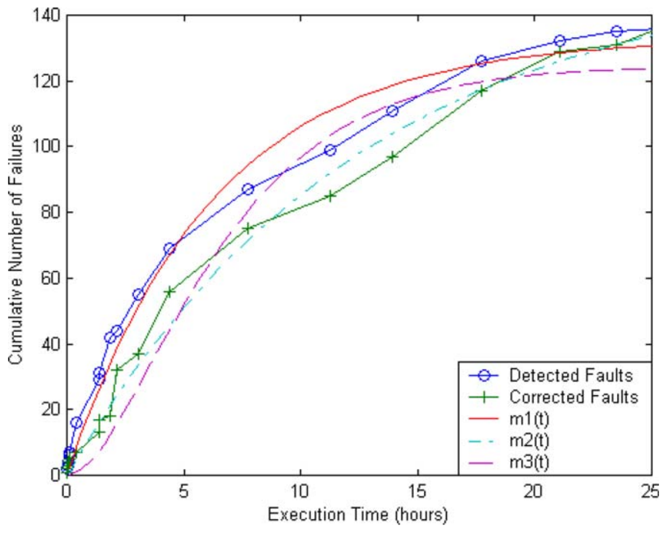


Fig. 1. Estimated MVFs vs. time (DS1).

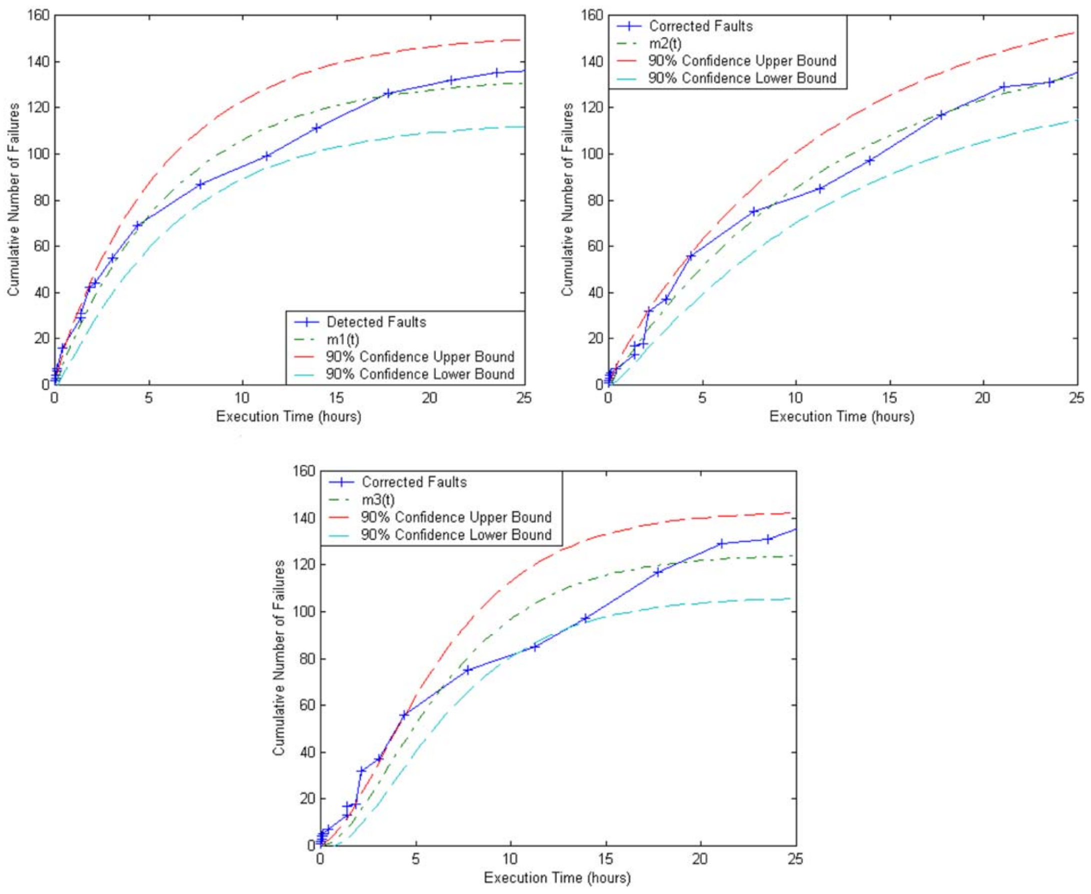
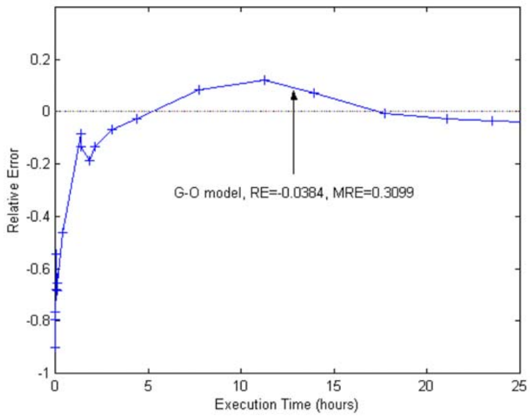
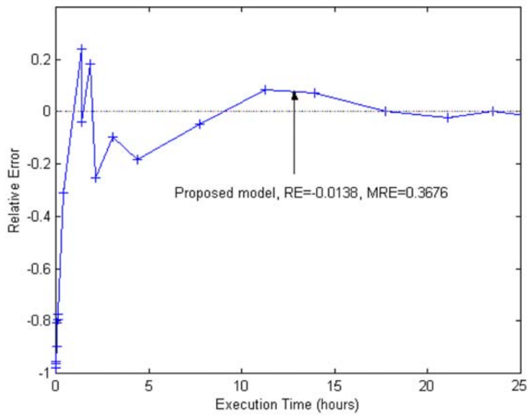


Fig. 2. The estimated MVFs of the correction process and the confidence bounds (DS1).





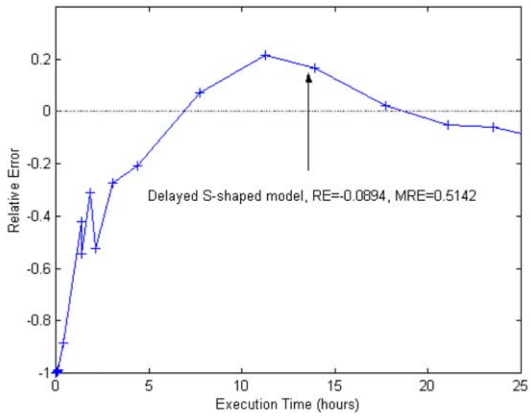


Fig. 3. RE curves vs. time (DS1).

executed for various values of t_i and the result can be shown by plotting the curve of the RE for different values of t_Z.

(5) The 100p% *upper and lower confidence limits* for $m(t)$ are defined as (Yin and Trivedi, 1999; Yamada et al., 2000)

$$\hat{m}(t) + \eta_p\sqrt{\hat{m}(t)} \quad \text{and} \quad \hat{m}(t) - \eta_p\sqrt{\hat{m}(t)}. \tag{25}$$

Suppose $\hat{m}(t)$ is the estimate for $m(t)$, then the 100p% upper and lower confidence limit for $m(t)$ can be bounded approximately as follows:

$$\hat{m}(t) + \eta_p\sqrt{\hat{m}(t)} \geqslant m(t) \geqslant \hat{m}(t) - \eta_p\sqrt{\hat{m}(t)}, \tag{26}$$

where η_p is the $100(1+p)/2$ percentage point of the standard normal distribution. For example, η_p is 1.645 when p is 0.9.

(6) The *Kolmogorov–Smirnov* (KS) statistic is defined as (Lyu, 1996)

$$\mathrm{KS} = \sup_x |F^*(x) - F(x)|. \tag{27}$$

where $F^*(x)$ is the normalized observed cumulative distribution at xth time point, $F(x)$ is the expected cumulative distribution at xth time point, based on the model, and $\sup_x$ represents the least upper bound of all pointwise differences $|F^*(x) - F(x)|$. In other words, the statistic compares the normalized cumulative distributions of the observed rates and the expected rates from the model at each point, takes the absolute difference, and chooses the maximum of them.

4.3. Performance analysis

In this section, we first estimate the parameters of all selected models and the corresponding mean value functions can be obtained. The maximum likelihood estimation (MLE) and least squares estimation (LSE) are used to estimate the parameters (Musa et al., 1987; Xie, 1991; Pham, 2000). Second, we will evaluate the performance of the proposed model and compare it with other selected SRGMs. In the following, we will discuss three cases to model the fault correction process. The first case is that the fault detection rate is a constant and fault correction rate is equal to zero, that is, $\lambda(t) = b$ and $\mu(t) = 0$. Obviously, from Eq. (13), we can see that $m(t)$ is the same as the Goel–Okumoto model. That is

$$m_{\text{case1}}(t) = m_1(t) = m(t) = a[1 - \exp(-bt)]. \tag{28}$$

The second case is that $\lambda(t) = b$ and $\mu(t) = c$. From Eqs. (13) and (14), the two equations can be solved under the initial condition $m(0) = 0$ and $m_c(0) = 0$. Thus we have

$$m_{\text{case2}}(t) = m_2(t) = m_c(t)$$
$$= a\left[1 + \frac{c}{b-c}\exp(-bt) - \frac{b}{b-c}\exp(-ct)\right]. \quad (29)$$

The third case is that $\lambda(t) = \mu(t) = b$. From Eqs. (13) and (14), we find that $m_c(t)$ is the same as the Yamada delayed S-shaped model. That is

$$m_{\text{case3}}(t) = m_3(t) = m_c(t) = a(1 - (1 + bt)\exp(-bt)). \quad (30)$$

4.3.1. Discussion of the fault correction process—DS1

Firstly, Table 4 gives the model's parameters and they are estimated by using MLE or LSE. Besides, the estimated values of RE, MRE, and KS are given in Table 4. Here we have to note that the lower the value of comparison criterion a model is, the better they fit the observed data. As seen from Table 4, it indicates that $m_2(t)$ provides the better fit and prediction for the first data set. That is, $m_2(t)$ has the smallest value of AE (18.42), MSE (18.19), |RE| (0.0138), and KS (0.1654). Fig. 1 shows the mean value functions of Eqs. (28)–(30), and the actual failure data set versus time. Fig. 2 graphically illustrates the comparisons between the observed data, the data estimated by Eqs. (28)–(30), and the 90% confidence bounds. In addition, we also compute the relative error in prediction for this data set at the end of testing. The results are plotted in Fig. 3. Fig. 4 provides the

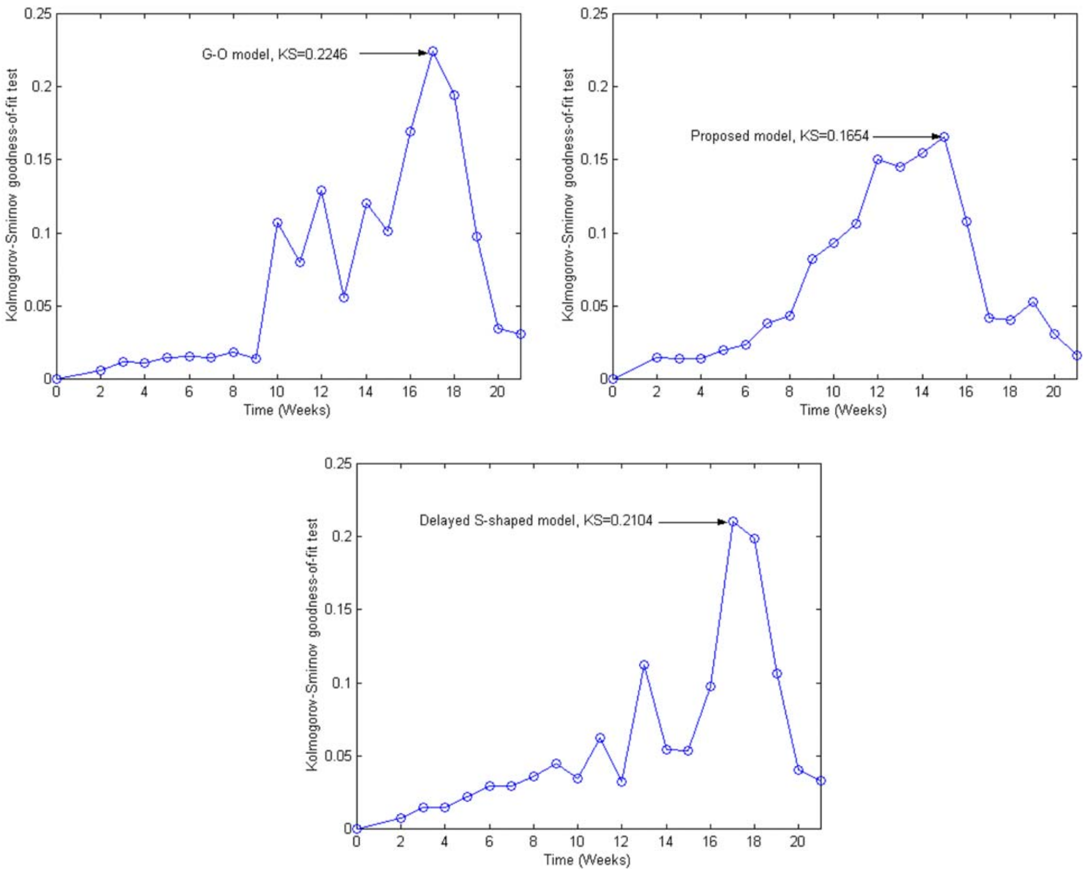


Fig. 4. KS curves vs. time (DS1).

Table 5
Comparison results for different models (DS2)

Model	AE (%)	MSF	|RE|	MRE	KS
$m_1(t)$	112.44	139.82	0.0625	0.0871	0.1155
$m_2(t)$	8.28	108.9	0.0044	0.0684	0.0642
$m_3(t)$	4.48	168.67	0.0135	0.1187	0.1027
Inflected S-shaped model	8.69	133.53	0.0274	0.0611	0.0737
Yamada exponential model	131.35	140.66	0.0475	0.0990	0.1049
Yamada Rayleigh model	28.23	268.42	0.0120	0.1495	0.1278
Yamada Weibull model	57.91	122.09	0.0404	0.0570	0.0871

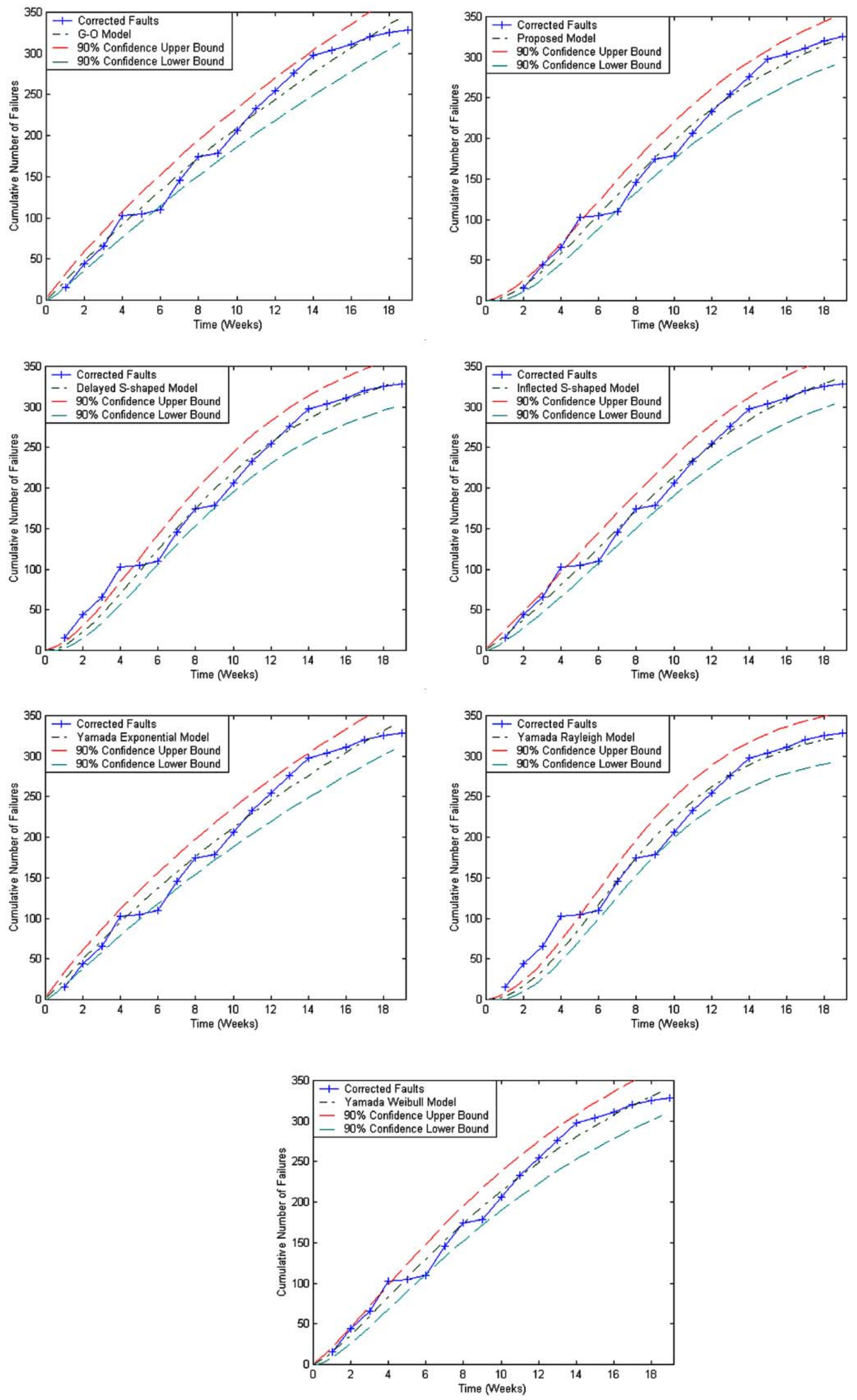


Fig. 5. The estimated MVFs of detection process and the 90% confidence bounds (DS2).

KS curves vs. time. Overall, the proposed model (i.e., $m_2(t)$) gives a better fit to the observed data and predicts the future behavior well.

4.3.2. Discussion of the fault detection process—DS2

Similarly, Table 5 gives the models' parameters and they are estimated by using MLE or LSE. Besides, Table 5 also

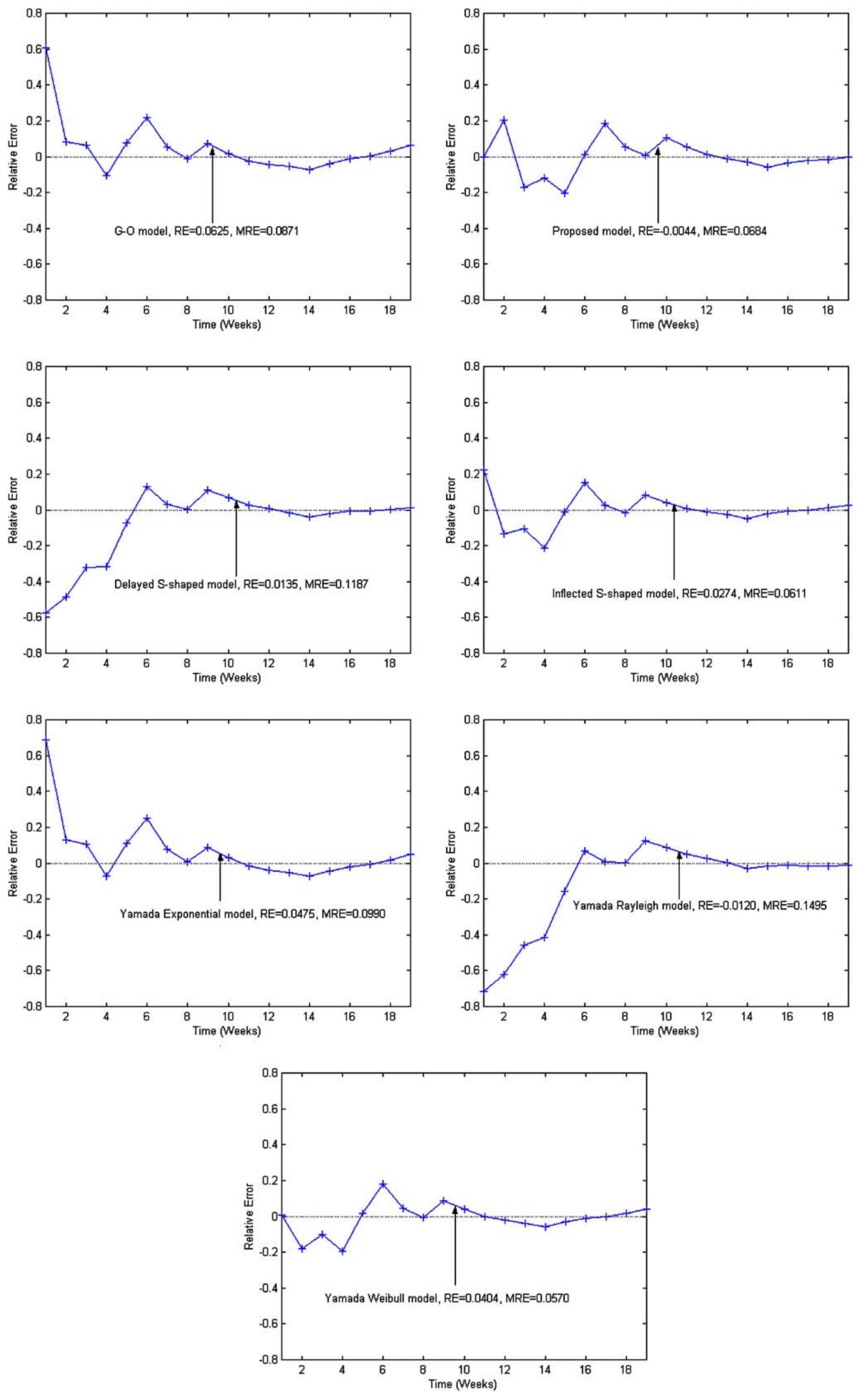


Fig. 6. RE curves vs. time (DS2).

gives the performance comparisons of different SRGMs. From Table 5, we can see that $m_2(t)$ provides the best fit and prediction for this data set. Although $m_3(t)$ (i.e., the Yamada delayed S-shaped model) gives the lowest AE,

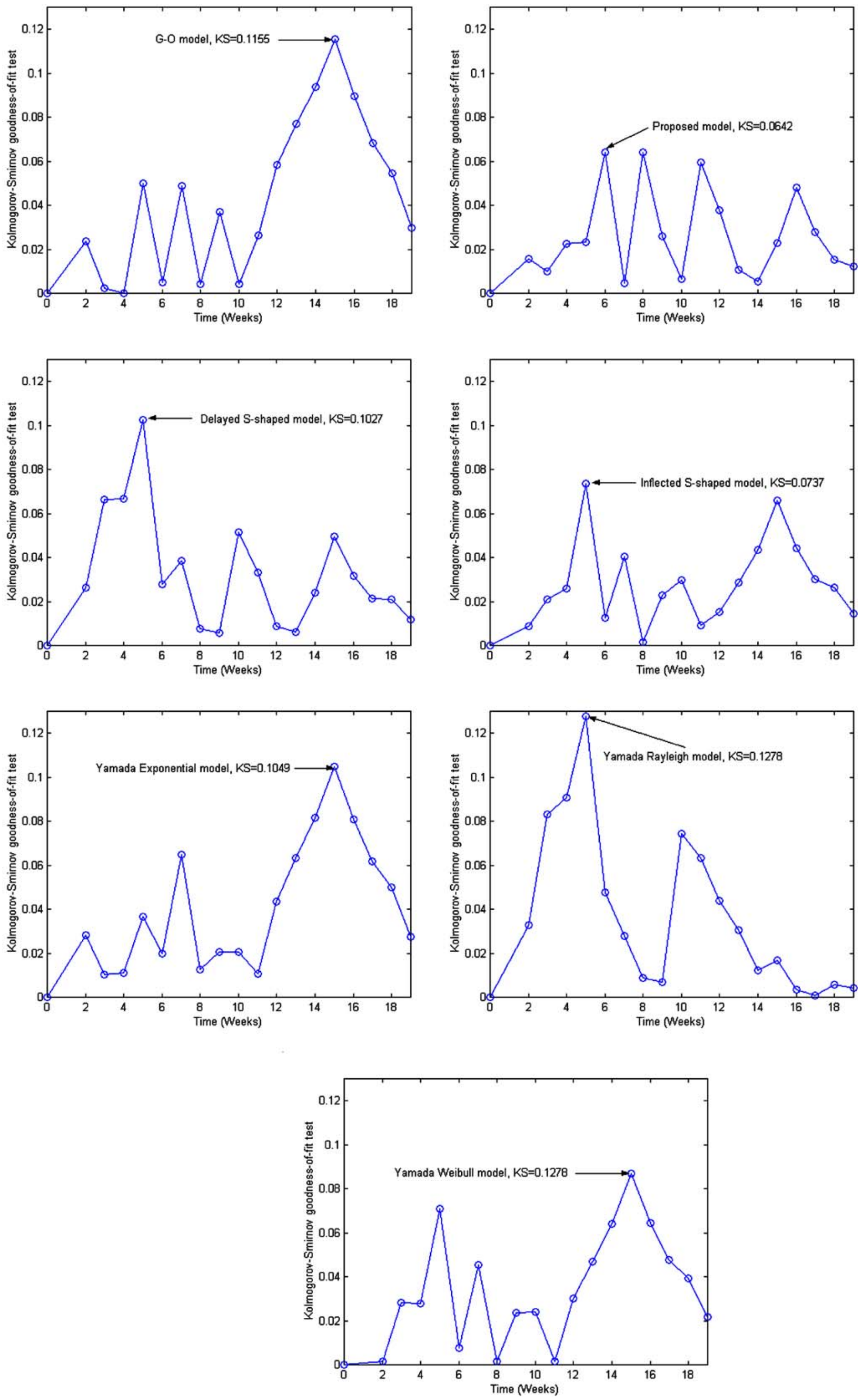


Fig. 7. KS curves vs. time (DS2).

$m_2(t)$ still provides the lower value of AE. It is noted that the estimated number of initial faults of $m_2(t)$ is very close to the actual reported number of initial faults (358). Fig. 5 depicts the estimated MVFs and the 90% confidence bounds. Finally, the curves of RE and KS versus time are plotted in Figs. 6 and 7, respectively. On the average, the proposed model gives a better fit to this data set.

5. Conclusions

Over the past 30 years, many SRGMs have been proposed by many researchers and some important metrics can be easily determined through SRGMs. However, from our studies, they assumed that detected faults are immediately corrected. In fact, this assumption may not be realistic in practice. In this paper, we first propose a general framework for modeling the fault detection and correction processes. We have showed that some existing SRGMs can be easily derived from the concept of the integration of the fault detection and correction processes. Two numerical examples are presented and experimental results show that the proposed framework considering both fault detection and correction processes for SRGM has a fairly accurate prediction capability.

Acknowledgements

This research was supported by the National Science Council, Taiwan, ROC, under Grant NSC 93-2213-E-267-001, and also substantially supported by a grant from the Ministry of Economic Affairs (MOEA) of Taiwan (Project No. 94-EC- 17-A-01-S1-038).

References

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

Gokhale, S.S., Lyu, M.R., Trivedi, K.S., 1998. Software reliability analysis incorporating fault detection and debugging activities. In: Proceedings of the 9th International Symposium on Software Reliability Engineering. November, pp. 64–75.

Huang, C.Y., 2005. Performance analysis of Software reliability growth models with testing-effort and change-point. J. Syst. Software 76, 181–194.

Kapur, P.K., Younes, S., 1995. Software reliability growth model with error dependency. Microelectron. Reliab. 35, 273–278.

Lo, J.H., Kuo, S.Y., Huang, C.Y., 2001. Reliability modeling incorporating error processes for internet-distributed software. In: The IEEE Region 10 International Conference on Electrical and Electronic Technology. August, pp. 1–7.

Lo, J.H., Kuo, S.Y., Lyu, M.R., Huang, C.Y., 2002. Optimal resource allocation and reliability analysis for component-based software applications. In: The 26th Annual International Computer Software and Applications Conference. August, pp. 7–12.

Lo, J.H., Huang, C.Y., Kuo, S.Y., Lyu, M.R., 2003. Sensitivity analysis of software reliability for component-based software applications. In: The 27th Annual International Computer Software and Applications Conference. November, pp. 500–505.

Lyu, M.R., 1996. Handbook of Software Reliability Engineering. McGraw-Hill, New York.

Lyu, M.R., Nikora, A., 1992. Applying software reliability models more effectively. IEEE Trans. Softw. 9, 43–52.

Musa, J.D., 1985. Software reliability data, report and database available from data and analysis center for software. In: Rome Air Development Center (RADC). Rome, NY, May, pp. 292–296.

Musa, J.D., 1998. Software Reliability Engineering: More Reliable Software, Faster Development and Testing. McGraw-Hill, New York.

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

Ohba, M., 1984. Software reliability analysis models. IBM J. Res. Develop. 28, 428–443.

Pham, H., 2000. Software Reliability. Springer-Verlag, Berlin.

Schneidewind, N.F., 1975. Analysis of error processes in computer software. Sigplan Notices 10, 337–346.

Schneidewind, N.F., 2003. Fault correction profiles. In: Proceedings of the 14th International Symposium on Software Reliability Engineering. November, pp. 60–67.

Xie, M., 1991. Software Reliability Modeling. World Scientific Publishing Company, Singapore.

Xie, M., 2000. Software reliability models-past, present and future. In: Limnios, N., Nikulin, M. (Eds.), Recent Advances in Reliability Theory: Methodology, Practice and Inference. Birkhauser, Boston, pp. 323–340.

Xie, M., Zhao, M., 1992. The Schneidewind software reliability model revisited. In: Proceedings of the 3rd International Symposium on Software Reliability Engineering. May, pp. 184–192.

Yamada, S., Ohba, M., Osaki, S., 1983. S-shaped reliability growth modeling for software error detection. IEEE Trans. Reliab. 32, 475–478.

Yamada, S., Hishitani, J., Osaki, S., 1993. Software-reliability growth with a Weibull test-effort: a model and application. IEEE Trans. Reliab. 42, 100–106.

Yamada, S., Tamura, Y., Kimura, M., 2000. A software reliability growth model for a distributed development environment. Electron. Commun. Jpn. Pt. 3: Fundam. Electron. Sci. 83 (12), 1–8.

Yin, L., Trivedi, K.S., 1999. Confidence interval estimation of NHPP-based Software reliability models. In: Proceedings of the 10th International Symposium on Software Reliability Engineering. November, pp. 6–11.

Jung-Hua Lo received the BS (1993) in Mathematics and the MS (1995) and the Ph.D. (2003) in Electrical Engineering from National Taiwan University. Since 1998, he has been with LanYang Institute of Technology, where he is currently an Assistant Professor and the Chairman of the Department of Information Management. His research interests are software engineering, software reliability and testing, etc.

Chin-Yu Huang is currently an Assistant Professor in the Department of Computer Science at National Tsing Hua University, Hsinchu, Taiwan. He received the MS (1994), and the Ph.D. (2000) in Electrical Engineering from National Taiwan University, Taipei. He was with the Bank of Taiwan from 1994 to 1999, and was a senior software engineer at Taiwan Semiconductor Manufacturing Company from 1999 to 2000. Before joining NTHU in 2003, he was a division chief of the Central Bank of China, Taipei. His research interests are software reliability engineering, software testing, software metrics, software testability, fault tree analysis, and system safety assessment, etc. He is a member of IEEE.