Estimation and Analysis of Some Generalized Multiple Change-Point Software Reliability Models

Chin-Yu Huang, *Member, IEEE*, and Michael R. Lyu, *Member, IEEE*

***Abstract*—Software typically undergoes debugging during both a testing phase before product release, and an operational phase after product release. But it is noted that the fault detection and removal processes during software development and operation are different. For example, the fault removal during operation occurs generally at a slower pace than development. In this paper, we derive a powerful, easily deployable technique for software reliability prediction and assessment in the testing and operational phases. We first review how several existing software reliability growth models (SRGM) based on non- homogeneous Poisson processes (NHPP) can be readily derived from a unified theory. With the unified theory, we further incorporate the concept of multiple change-points, i.e. points in time when the software environment changes, into software reliability modeling. Several models are proposed and discussed under both ideal and imperfect debugging conditions. We estimate the parameters of the proposed models by employing real software failure data, and give a fair comparison with some existing SRGM. Numerical results show that the proposed models can provide good software reliability prediction in the various stages of software development and operation. Our approach is flexible; we can model various environments ranging from exponential-type to S-shaped NHPP models.**

***Index Terms*—Change point, imperfect debugging, non-homogeneous Poisson process (NHPP), software reliability growth model (SRGM), software testing.**

ACRONYM[1]

ANSI	American National Standards Institute
CARATS	computer-aided reliability assessment tool for software
CP	change point
K-S	Kolmogorov-Smirnov
KD	Kolmogorov-Distance
LSE	least squares estimation
MLE	maximum likelihood estimation
MSE	mean square error
MTBF	mean time between failures
MVF	mean value function
NHPP	non-homogeneous Poisson process
RE	relative error
RMSE	root mean square error
RRMS	relative root mean square error
SRE	software reliability engineering
SRGM	software reliability growth model

NOTATIONS

$m(i)$	expected cumulative number of faults detected by i test runs
$m(t_i)$	mean value function, i.e., the expected number of software failures by time t_i
$F^*(x)$	normalized observed cumulative distribution at the x-th time point
$F(x)$	expected cumulative distribution at the x-th time point
a	expected number of initial faults
b	fault detection rate per fault in the steady state
$b(t)$	fault detection rate function
ψ	inflection factor
c	constant parameter
α	total amount of testing effort
β	scale parameter
γ	shape parameter
r	inflection rate
UCL_c	upper control limit
LCL_c	lower control limit
$\overline{C}$	Centerline

Manuscript received May 14, 2009; revised July 01, 2010; accepted September 08, 2010. Date of publication April 21, 2011; date of current version June 02, 2011. The work described in this paper was supported by the National Science Council, Taiwan, under Grants NSC 97-2221-E-007-052-MY3, NSC 98-2219-E-002-021, and NSC 99-2220-E-007-022 and by a Grant from the Research Grants Council of the Hong Kong Special Administrative Region, China, under Project CUHK4154/09E. Associate Editor: C. Smidts.

C.-Y. Huang is with the Department of Computer Science, National Tsinghua University, Hsinchu, Taiwan (e-mail: cyhuang@cs.nthu.edu.tw).

M. R. Lyu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TR.2011.2134350

[1]The singular and plural of an acronym are always spelled the same.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

m_i	cumulative number of detected faults in a given time interval $(0, t_i]$
θ	number of model parameters
$n(t)$	fault content function
β	fault introduction rate

I. Introduction

SOFTWARE is a key part of many critical applications. Modern software controls household appliances, financial payment systems, traffic light systems, and national defense systems, to name a few. The major problem facing the current computer industry is how to develop reliable software. Software is engineered in several distinct phases, including requirement specification, risk management, system analysis, system design, development, testing, and maintenance. In practice, at the end of the software testing phase, project managers always want an assessment of the software reliability (or quality) regarding the determination of whether and when the desired level of reliability has been reached. Software reliability measurement therefore plays a very important role in developing a robust, high-quality software products [1].

Reliability measurement is a set of mathematical techniques that can be used to estimate and predict the reliability behavior of software during its development and operation. According to the ANSI definition, *software reliability* is defined as *the probability of failure-free software operation for a specified period of time in a specified environment* [2], [3]. Measuring and computing software reliability can be used for planning and controlling all testing resources during software development. MM Moreover, software reliability relates directly to operation, rather than design, of the software. To assess software reliability, and assure software quality, one of the current methods is to apply SRGM. SRGM is the one particular aspect of SRE that has received the most attention. SRGM describe failures as a random process, which is characterized in either times of failures, or the number of failures at fixed time periods [4]. In general, SRGM can provide quantitative information about how to improve the reliability of software products, and greatly help software engineers to measure the defect levels, failure rates, MTBF, and reliability during the coding and testing phases. Furthermore, SRGM can help project managers to determine the testing resources and manpower needed to achieve desired reliability requirements [5], [6].

Our previous studies [7] have shown that several conventional SRGM can be unified under a general formulation. A unified theory is powerful for the study of general models without making strong assumptions. In this paper, we first review the unification of SRGM based on NHPP. Then we show how existing NHPP-based SRGM, such as the Goel–Okumoto model, the Yamada Delayed S-shaped model, the Generalized Goel NHPP model, and the Inflection S-shaped model, can be derived by applying the concept of three well-known means: the *weighted arithmetic mean*, the *weighted geometric mean*, and the *weighted harmonic mean*. Consequently, many existing SRGM based on NHPP are special cases of the unified NHPP model.

On the other hand, if a software system has already been developed, the developers usually want to predict its reliability in the operational phase. The operational reliability of software is the main concern for general users [8]–[11]. Sometimes developers can use the SRGM that perform accurately during testing to predict the reliability in the actual operation. Most SRGM estimate the reliability using the historical data collected during testing, and assume that the test environment can well represent the operation profile. However, this may not be correct because the testing environment may not represent the typical use of the system in field operation [1], [8]. The fault detection process in the operational phase is different from that in the testing phase. Thus we have to make an adjustment to the selected SRGM that was accurate in the testing phase. There are methods of modeling software operational reliability in the literature [8], [12]–[21].

Besides the fielded operation, due to the change of users' requirements or business needs, software systems usually have to be updated many times during their life cycle. In this case, traditional SRGM may be good for one revision period rather than the whole life cycle. Therefore, when SRGM are used in the various stages of software development and operation, the estimated parameters of selected SRGM should be adjusted to remove the prediction bias. In this paper, based on the unified theory in [7], we will show how to incorporate the concept of multiple CP into software reliability modeling. The models further derived from this approach can easily be used to describe the transition from the testing to the operational phase. In addition, the proposed models will also be assessed and discussed based on real data sets under both ideal, and imperfect debugging conditions.

The remainder of this paper is organized as follows. Section II reviews the unification of SRGM based on NHPP. Based on the unified theory, we will further study how to modify traditional SRGM by taking the concept of multiple CP into consideration. Section III gives some numerical examples with real software failure data, employing the proposed models. Section IV discusses the imperfect debugging problem attacked by the proposed models. Finally, Section V concludes this paper.

II. Software Reliability Modeling

A. Review of NHPP Models

First, we will give a brief review of the unification of SRGM based on NHPP [7].

1) Weighted Arithmetic, Weighted Geometric, and Weighted Harmonic Means: Let $x \geq 0$, and $y \geq 0$. We can define the *arithmetic mean* z of x and y as

$$z = \frac{1}{2}x + \frac{1}{2}y \tag{1}$$

More generally, we can define the *weighted arithmetic mean* z of x and y with weights w and $1 - w$ as

$$z = wx + (1-w)y, \qquad 0 < w < 1. \tag{2}$$

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

The *geometric mean* z of x and y is defined as

$$z = \sqrt{xy}. \tag{3}$$

That is,

$$\ln z = \frac{1}{2}\ln x + \frac{1}{2}\ln y. \tag{4}$$

Similarly, the *weighted geometric mean* z of x and y with weights w and $1-w$ is defined as

$$\ln z = w\ln x + (1-w)\ln y, \qquad 0 < w < 1. \tag{5}$$

Finally, the *harmonic mean* z of x and y is defined as

$$\frac{1}{z} = \frac{1}{2x} + \frac{1}{2y}, \tag{6}$$

and the *weighted harmonic mean* z of x and y with weights w and $1-w$ is defined as

$$\frac{1}{z} = w\frac{1}{x} + (1-w)\frac{1}{y}, \qquad 0 < w < 1. \tag{7}$$

Definition 1: Let g be a real-valued strictly monotone function. If x and y are two nonnegative real numbers, the *quasi-arithmetic mean* z of x and y with weights w and $1-w$ is defined as

$$z = g^{-1}\left(wg(x) + (1-w)g(y)\right), \qquad 0 < w < 1, \tag{8}$$

where g^{-1} is the inverse function of g.

2) General Discrete SRGM: In this section, we discuss a general discrete NHPP model. For the discrete Goel–Okumoto model, suppose that the expected number of failures detected per test run is proportional to the current fault content of a software system,

$$m(i+1) - m(i) = b(a - m(i)), \qquad a > 0, 0 < b < 1. \tag{9}$$

Taking $w = 1 - b$, we have

$$m(i+1) = wm(i) + (1-w)a, \qquad a > 0, 0 < w < 1. \tag{10}$$

Equation (10) indicates that $m(i+1)$ is equal to the weighted arithmetic mean of $m(i)$, and a with weights w, and $1-w$ respectively. That is, the discrete Goel–Okumoto model can also be derived based on the weighted arithmetic mean.

Because the *weighted arithmetic*, *weighted geometric*, and *weighted harmonic means* are all well-known means, we employ the other two means to derive other existing NHPP models. First, consider that $m(i+1)$ is equal to the weighted geometric mean of $m(i)$, and a with weights w, and $1-w$ respectively; then

$$\ln m(i+1) = w\ln m(i) + (1-w)\ln a, \quad 0 < w < 1, \text{ and } a > 0. \tag{11}$$

Next, let's consider the case that $m(i+1)$ is equal to the weighted harmonic mean of $m(i)$, and a with weights w, and $1-w$ respectively. In this case, we have

$$\frac{1}{m(i+1)} = w\frac{1}{m(i)} + (1-w)\frac{1}{a}, \qquad 0 < w < 1, a > 0. \tag{12}$$

More generally, let g be a real-valued, strictly monotone function; and $m(i+1)$ be equal to the quasi-arithmetic mean of $m(i)$, and a with weights w, and $1-w$ respectively. Then,

$$g\left(m(i+1)\right) = wg\left(m(i)\right) + (1-w)g(a), \qquad 0<w<1, a>0. \tag{13}$$

Solving (13) yields

$$\begin{aligned} g\left(m(i+1)\right) &= w^i g\left(m(0)\right) + (1 + w + w^2 + \ldots + w^{i-1}) \\ &\quad \times (1-w)g(a) \\ &= w^i g\left(m(0)\right) + (1 - w^i)g(a). \end{aligned} \tag{14}$$

Therefore,

$$m(i) = g^{-1}\left(w^i g\left(m(0)\right) + (1-w^i)g(a)\right). \tag{15}$$

3) General Continuous SRGM: We will further discuss a general continuous NHPP model in this section. We let $m(t+\Delta t)$ be equal to the quasi-arithmetic mean of $m(t)$, and a with weights $w(t,\Delta t)$, and $1 - w(t,\Delta t)$; then

$$g\left(m(t+\Delta t)\right) = w(t,\Delta t)g\left(m(t)\right) + \left(1 - w(t,\Delta t)\right)g(a), \quad 0 < w(t,\Delta t) < 1. \tag{16}$$

That is,

$$\frac{g\left(m(t+\Delta t)\right) - g\left(m(t)\right)}{\Delta t} = \frac{1 - w(t,\Delta t)}{\Delta t}\left(g(a) - g\left(m(t)\right)\right), \quad 0 < w(t,\Delta t) < 1. \tag{17}$$

Supposing $(1 - w(t,\Delta t))/\Delta t \to b(t)$ as $\Delta t \to 0$, we get the differential equation

$$\frac{\partial}{\partial t} g\left(m(t)\right) = b(t)\left(g(a) - g\left(m(t)\right)\right). \tag{18}$$

For $g(x) = x$ in (18) (i.e., considering the weighted arithmetic mean), then

$$\frac{\partial}{\partial t} g\left(m(t)\right) = b(t)\left(a - m(t)\right). \tag{19}$$

For example, if $b(t) = b$, then the Goel–Okumoto model can be derived from (19). The differential equations for $g(x) = \ln x$, and $g(x) = 1/x$ can also be derived from (18).

Theorem 1: Let

$$\frac{\partial}{\partial t} g\left(m(t)\right) = b(t)\left(g(a) - g\left(m(t)\right)\right),$$

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

TABLE I
SOME CLASSICAL SRGM CORRESPONDING TO $g(x)$, $b(t)$, AND k

MVFs	***g* (*x*)**	***b* (*t*)**	***k***
Goel-Okumoto (G-O) Model $m(t) = a(1 - \exp[-bt])$	x	b	1
Yamada delayed S-shaped (DSS) Model $m(t) = a(1-(1+bt)\exp[-bt])$	x	$\frac{b^2 t}{1+bt}$	1
Generalized Goel (gGO) NHPP Model $m(t) = a(1-k\exp[-bt^C])$	x	$bc\,t^{C-1}$	1
Inflection S-Shaped (ISS) Model $m(t) = a\left(\frac{1-\exp[-bt]}{1+\psi\exp[-bt]}\right)$	x	$\frac{b}{1+\psi\exp[-bt]}$	1
Yamada Weibull (YW) Model $m(t) = a(1-\exp[-bW(t)]$ and $W(t) = \alpha(1-\exp[-\beta t^\gamma])])$	x	$b\alpha\beta\gamma \times t^{r-1}\exp[-\beta \times t^\gamma]$	1

where g is a real-valued, strictly monotonic, and differentiable function. We have

$$m(t) = g^{-1}\left(g(a) + \{g(m(0)) - g(a)\}\exp[-B(t)]\right), \quad (20)$$

and $B(t) = \int_0^t b(u)du$.

Corollary 1: Based on the weighted arithmetic mean, take $g(x) = x$ in (20), and let $k = 1 - m(0)/a$; then the MVF, i.e., the expected mean number of faults detected in time $(0, t]$ is

$$m(t) = a\left(1 - k\exp[-B(t)]\right), a > 0, 0 < k \leq 1.$$

Below we briefly show that some classical SRGM based on NHPP, such as the Goel–Okumoto model, the Yamada delayed S-shaped model, the generalized Goel NHPP model, the Inflection S-shaped model, and the Yamada Weibull Model, can be directly derived from Corollary 1. Table I gives the SRGM corresponding to $g(x)$, $b(t)$, and k [7].

B. Software Reliability Models With Multiple CP

During the software development process, once failures are identified, related fault detection and isolation activities are activated to find the root causes. In practice, the fault detection process in the operational phase is typically different from that in the testing phase [22], but this fact is not distinctly incorporated in many software reliability modeling approaches. Ohtera and Yamada [16] considered that the fault detection rates per fault during the testing and operational phases are different. They assumed that the fault detection rate decreases during the testing phase, and then becomes stable during the operational phase. In reality, during the software development and operation, the fault detection rate may be neither a constant nor smooth, i.e., it may be changed at some moments in time called CP. In general, a CP is the time instant when a model's parameter experiences a discontinuity in time. That is, it is the time at which the parameter changes values. In recent years, a number of papers have addressed the change point problem in the field of software reliability modeling [23]–[29].

For example, Zhao [24] suggested that, before the software is released to field operation, a CP could occur when the testing strategy and resource allocation are changed. That is, the running environment may change at that moment. The availability of testing facilities and other random factors can be the causes of the CP. Besides, the testing effort may not be constant. Personnel may vary, and their time dedicated to job functions may change. In practice, software failure data sets are collected during testing, and this information is fed back to the software development teams for further improvements. Sometimes, if the company can afford the extra cost, and engineers are also eager to detect additional faults during the software development process, it is advisable to introduce new tools and techniques that are fundamentally different from the methods currently in use. The benefit of these methods is that engineers can design (or propose) several testing programs (or automated testing tools) to test software to satisfy the clients' technical requirements, schedule, and budget. That is, these approaches can provide a steady improvement in software testing and productivity. Therefore, the timing of introducing new tools or techniques can be treated as a CP [3].

In industry, users' requirements or business needs are frequently changed at some CP. Thus, software engineers have to update or revise the software many times during the life cycle of the software. In this case, traditional SRGM may be good for one revision period rather than the whole life cycle. Based on the fact that, when SRGM are used in different running environments or stages of software development, the estimated parameters of selected SRGM should be adjusted at the proper time to remove the prediction bias. I n the following discussions, based on the unified theory of NHPP models, we will show how to modify (and perhaps improve) the traditional SRGM by incorporating the concepts of multiple CP.

1) G-O Model With Multiple CP: Let us first consider the traditional Goel–Okumoto model. Its mathematical expression can be described as [4]

$$\frac{dm(t)}{dt} = b \times (a - m(t)), a > 0, 0 < b < 1. \quad (21)$$

Solving (21) using the boundary condition $m(0) = 0$, we have

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

Furthermore, we can describe the Goel–Okumoto model with a single CP

$$\frac{dm(t)}{dt} = b(t) \times (a - m(t)), \quad (22)$$

and

$$b(t) = \begin{cases} b_1(t) = b_1 & \text{if } 0 \leq t \leq \tau; \\ b_2(t) = b_2 & \text{if } t > \tau. \end{cases} \quad (23)$$

Solving the above two equations under the boundary condition $m(0) = 0$, we have

$$m(t) = \begin{cases} m_1(t) = a(1 - \exp[-b_1 t]) & ,0 \leq t \leq \tau \\ m_2(t) = a(1 - \exp[-(b_1\tau + b_2(t - \tau))]) & ,t > \tau. \end{cases} \quad (24)$$

In fact, (24) can also be derived from the unified theory, but we have to make some adjustments to Theorem 1 and Corollary 1 to accommodate multiple CP. First, from Corollary 1, we have $m_1(t) = a(1 - k_1\exp[-B_1(t)])$ when $0 \leq t \leq \tau$. If we take $g(x) = x$, $k_1 = 1 - m(0)/a$, and $B_1(t) = \int_0^\tau b_1 du$, we can

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

obtain the MVF $m_1(t) = a(1 - \exp[-b_1 t])$. When $t > \tau$, if we take $g(x) = x$, and $B_2(t) = \int_\tau^t b_2 du$, then from Corollary 1, we have

$$m_2(t) = a\left(1 - k_2 \exp\left[-b_2(t-\tau)\right]\right). \tag{25}$$

Note that if

$$k_2 = 1 - m_1(\tau)/a = \exp[-b_1\tau], \tag{26}$$

we can obtain $m_2(t) = a(1 - \exp[-(b_1\tau + b_2(t-\tau))])$. Furthermore, we can consider the Goel–Okumoto model with two CP. If the fault detection rate is given by

$$b(t) = \begin{cases} b_1(t) = b_1 & \text{if } 0 \le t \le \tau_1 \\ b_2(t) = b_2 & \text{if } \tau_1 < t \le \tau_2 \\ b_3(t) = b_3 & \text{if } \tau_2 < t \end{cases}, \tag{27}$$

the MVF can be obtained by following similar procedures. In this case, we have

$$m(t) = \begin{cases} m_1(t) = a\left(1 - \exp[-b_1 t]\right) & ,0 \le t \le \tau_1 \\ m_2(t) = a\left(1 - \exp\left[-(b_1\tau_1 + b_2(t-\tau_1))\right]\right) & ,\tau_1 < t \le \tau_2 \\ m_3(t) = a\left(1 - \exp\left[-(b_1\tau_1 + b_2(\tau_2 - \tau_1) + b_3(t-\tau_2))\right]\right) & ,\tau_2 < t \end{cases} \tag{28}$$

Finally, if we have n CP, i.e.

$$b(t) = \begin{cases} b_1(t) = b_1 & \text{if } 0 \le t \le \tau_1; \\ b_2(t) = b_2 & \text{if } \tau_1 < t \le \tau_2 \\ \vdots & \\ b_n(t) = b_n & \text{if } \tau_{n-1} < t. \end{cases}, \tag{29}$$

$$B_1(t) = \int_0^{\tau_1} b_1(u)du, \tag{30}$$

$$B_k(t) = \int_{\tau_{k-1}}^{\tau_k} b_k(u)du \quad (\text{for } 1 < k < n), \tag{31}$$

$$B_n(t) = \int_{\tau_{n-1}}^{t} b_n(u)du, \tag{32}$$

and

$$k_n = 1 - m_{n-1}(\tau_{n-1})/a = \exp\left[-\sum_{k=1}^{n-1} b_k \times (\tau_k - \tau_{k-1})\right], \tag{33}$$

then we get a generalized solution for describing the Goel–Okumoto model with multiple CP as

$$m_n(t) = a\left(1 - \exp\left[-\left(b_n \times (t - \tau_{n-1}) + \sum_{k=1}^{n-1} b_k \times (\tau_k - \tau_{k-1})\right)\right]\right), \tag{34}$$

where $\tau_0 = 0$.

2) Generalized Goel NHPP Model With Multiple CP: We can follow a similar procedure to that described in Section II-B-1. For the generalized Goel NHPP model with multiple CP, if

$$b_k(t) = b_k c t^{c-1}, \tag{35}$$

then we can get the MVF

$$m_n(t) = a\left(1 - \exp\left[-\left(b_n \times \left(t^C - \tau_{n-1}^C\right) + \sum_{k=1}^{n-1} b_k \times \left(\tau_k^C - \tau_{k-1}^C\right)\right)\right]\right). \tag{36}$$

3) ISS Model With Multiple CP: We follow a similar procedure to that described in Section II-B-1. For the Inflection S-shaped model with multiple CP, if

$$b_k(t) = \frac{b_k}{1 + \psi \exp[-b_k t]}, \tag{37}$$

we can get the MVF

$$m_n(t) = a\left(\frac{1 - \exp\left[-\left(b_n \times (t - \tau_{n-1}) + \sum_{k=1}^{n-1} b_k \times (\tau_k - \tau_{k-1})\right)\right]}{1 + \psi \exp\left[-\left(b_n \times (t - \tau_{n-1}) + \sum_{k=1}^{n-1} b_k \times (\tau_k - \tau_{k-1})\right)\right]}\right). \tag{38}$$

Here, ψ is defined as

$$\psi = (1-r)/r, \qquad 0 < r \le 1. \tag{39}$$

Note that when $r = 1$, that is all faults are detectable initially, then the exponential model is obtained, because $\psi = 0$. Only if $r < 0.5$ (or $\psi > 1$) does the reliability growth curve have an inflection [4].

4) Yamada Weibull Model With Multiple CP: We follow a similar procedure to that described in Section II-B-1. For the Yamada Weibull model with multiple CP, if

$$b_k(t) = b_k \alpha\beta\gamma \times t^{r-1} \exp[-\beta \times t^\gamma], \tag{40}$$

we can get the MVF

$$m_n(t) = a\left(1 - \exp\left[-\left(b_n \times (W(t) - W(\tau_{n-1})) + \sum_{k=1}^{n-1} b_k \times (W(\tau_k) - W(\tau_{k-1}))\right)\right]\right). \tag{41}$$

Note that the Weibull-type testing-effort function is given by [4], [7]

$$W(t) = \alpha\left(1 - \exp[-\beta t^\gamma]\right). \tag{42}$$

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

TABLE II
DS-1

WEEK	CNF*	WEEK	CNF	WEEK	CNF	WEEK	CNF	WEEK	CNF
1	7	18	174	35	319	52	387	69	454
2	8	19	183	36	323	53	387	70	456
3	36	20	196	37	324	54	387	71	456
4	45	21	200	38	338	55	388	72	456
5	60	22	214	39	342	56	393	73	457
6	74	23	223	40	345	57	398	74	458
7	82	24	246	41	350	58	400	75	459
8	98	25	257	42	352	59	407	76	459
9	106	26	277	43	356	60	413	77	459
10	115	27	283	44	367	61	414	78	460
11	120	28	286	45	373	62	419	79	460
12	134	29	292	46	373	63	420	80	460
13	139	30	297	47	378	64	423	81	461
14	142	31	301	48	381	65	429		
15	145	32	302	49	383	66	440		
16	153	33	310	50	384	67	443		
17	167	34	317	51	384	68	448		

*CNF: Cumulative Number of Failures

TABLE III
DS-2

WEEK	CNF	WEEK	CNF	WEEK	CNF	WEEK	CNF	WEEK	CNF
1	5	12	70	23	130	34	181	45	195
2	6	13	71	24	136	35	182	46	196
3	13	14	74	25	141	36	183	47	197
4	22	15	78	26	148	37	185	48	199
5	24	16	90	27	156	38	186	49	201
6	29	17	98	28	164	39	187	50	202
7	34	18	105	29	166	40	188	51	203
8	40	19	110	30	169	41	190		
9	46	20	117	31	170	42	191		
10	53	21	123	32	176	43	192		
11	63	22	128	33	180	44	193		

*CNF: Cumulative Number of Failures

5) Yamada DSS Model With Multiple CP: We follow a similar procedure to that described in Section II-B-1. For the Yamada delayed S-shaped model with multiple CP, if

$$b_k(t) = \frac{b_k^2 t}{1 + b_k t}, \tag{43}$$

we can get the MVF

$$m(t)=a\left(1-\left(1+\left(b_n\times(t-\tau_{n-1})+\sum_{i=1}^{n-1} b_i\times(\tau_i-\tau_{i-1})\right)\right)\right.$$
$$\left.\times\exp\left[-\left(b_n\times(t-\tau_{n-1})+\sum_{i=1}^{n-1} b_i\times(\tau_i-\tau_{i-1})\right)\right]\right). \tag{44}$$

III. Numerical Examples and Simulation Results

A. Data Description

To validate our approach, the first data set (DS-1) we employed was from a real software project [31]. The system was a Brazilian Electronic Switching system, TROPICO R-1500, for 1500 telephone subscribers. Its software size was about 300 kb written in assembly language. During an 81-week software execution period, 461 faults were removed. This data set has 81 corresponding data entries. Table II gives the complete data set. It is noted that entries 1 through 30 were obtained during the system validation phase, entries 31 through 42 were obtained during field trials, and entries 43 through 81 were obtained during system operation [2]. The second data set (DS-2) was obtained from a wireless network product in [32], and the software ran on an element within a wireless network switching center. Table III lists the data set. The third data set (DS-3) used in this paper was collected in the bug tracking system on the website of Xfce [33]. Xfce is a lightweight desktop environment for UNIX-like OS. Note that DS-3 was collected from 11/12/2003 to 04/07/2004, and is given in Table IV. In the following, we will use two existing methods, Laplace trend analysis, and c charts, to determine whether the software undergoes reliability growth or decrease, and to identify the location of CP, respectively.

Laplace test is generally used for identifying trends in grouped data or time-series. Among the analytical tests, Laplace test is the most commonly used, and has been discussed in detail [2], [31]. We will calculate the Laplace trend factor $U(t)$. For example, with reference to the data sets in Tables II–IV, the Laplace trend test results are shown in Figs. 1–3, respectively. As seen from Figs. 1–3, we observe that the values of $U(t)$ are almost always negative, which indicates a growth in reliability. In the case of reliability growth, our proposed models can

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
DS-3

WEEK	CNF	WEEK	CNF	WEEK	CNF	WEEK	CNF	WEEK	CNF
1	16	6	51	11	98	16	127	21	167
2	24	7	60	12	108	17	132		
3	26	8	67	13	114	18	135		
4	38	9	74	14	117	19	150		
5	45	10	89	15	124	20	164		

*CNF: Cumulative Number of Failures

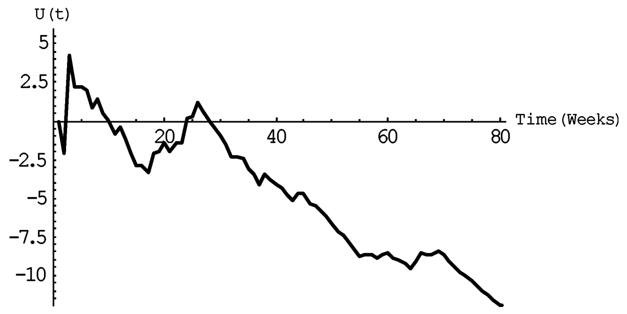


Fig. 1. Laplace trend test for DS-1.

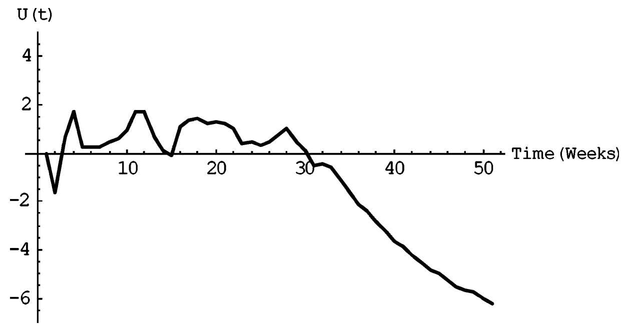


Fig. 2. Laplace trend test for DS-2.

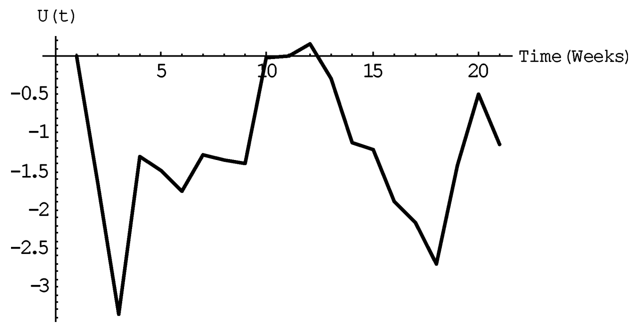


Fig. 3. Laplace trend test for DS-3.

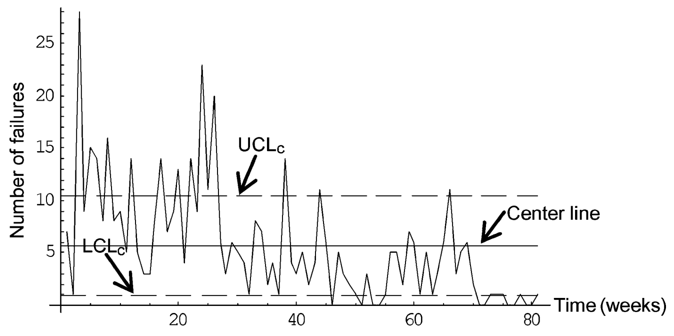


Fig. 4. C chart for DS-1.

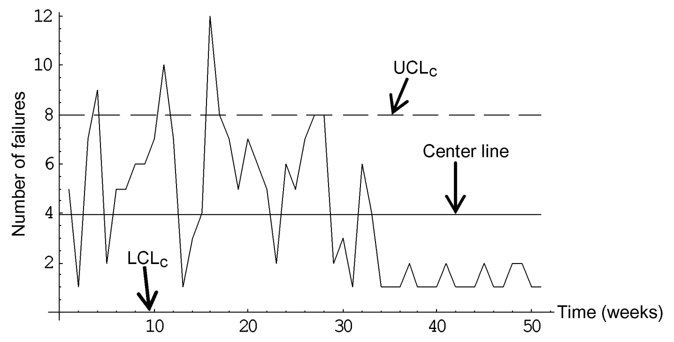


Fig. 5. C chart for DS-2.

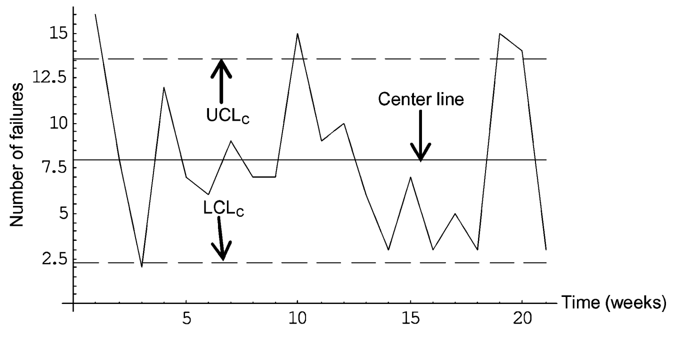


Fig. 6. C chart for DS-3.

be applied to predict the number of detected failures, and the failure intensity during the development phase.

On the other hand, the c chart is a commonly used tool for monitoring the software process. In the c chart, control limits are estimated. The parameters of the c chart are defined as [34]

$$UCL_c(LCL_c) = \overline{C} + (-)3\sqrt{\overline{C}}, \tag{45}$$

and

$$\overline{C} = \frac{\sum_{i=1}^{k} m(t_i)}{k}. \tag{46}$$

Figs. 4 –6 give the c chart diagrams for DS-1, DS-2, and DS-3, respectively.

The accuracy of projection is typically directly related to the sample size of the failure data. Without enough failure data, it

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

may not be possible to obtain unbiased or reduced biased estimates for some models. Thus, we assumed that if we would like to observe (or decide) the CP(s), the projection is required to be made at least after 50% of the end of test time. There are some existing rules (or tests) to detect unusual patterns and nonrandom behavior [34]. Here we assume that, if a single point falls outside the UCL_c or LCL_c, this will be a CP. On the other hand, it is also assumed that, if we observe at least eight successive values fall on the same side of the centerline, UCL_c or LCL_c, a CP may occur. In addition, a CP could be indicated by a trend of six or more values in a row steadily increasing or decreasing.

For DS-1, we have clearly known from [31] that entries 1 through 30 were obtained during the system validation phase, and entries 31 through 42 were obtained during field trials. Based on this reason, we have simply decided that the first CP will be located at the 31st week. Similarly, we assume the second CP to have occurred at the 43rd week. For DS-2, as seen from Fig. 5, after 50% of the end of test time, weeks 35-50 (more than eight successive values) are below the centerline and fall on the same side. Thus we can treat the 35th week as a CP. Finally, for DS-3, as shown in Fig. 6, we find that two point falls outside the UCL_c at the 10th and 19th weeks, after 50% of the end of test time. Besides, there are two obvious inflection points near at the 10th and 19th weeks. Consequently, it is decided that the 10th, and the 19th week are the first, and the second CP, respectively.

B. Performance Evaluation Criteria

A model can generally be analyzed according to its retrodictive capability (i.e., its ability to reproduce the observed behavior of the software), and predictive capability (i.e., its ability to predict future behavior of the software from the observed failure data) [1], [31]. Because the data sets listed in Tables II–IV are obtained as failure counts, we employ the following criteria to assess and compare the performance of all selected models.

1) The Goodness-of-Fit Criterion. The RMSE is expressed as [35]

$$RMSE = \sqrt{\frac{1}{k}\sum_{i=1}^{k}[m_i - m(t_i)]^2}. \tag{47}$$

A smaller RMSE indicates a smaller fitting error [5]. In addition, the RRMS can be calculated as [36]

$$\mathrm{RRMS} = \frac{\mathrm{RMSE}}{\frac{1}{\mathrm{k}}\sum_{\mathrm{i}=1}^{\mathrm{k}} \mathrm{m_i}}. \tag{48}$$

Conte *et al.* [37] suggested that $\mathrm{RMS} \leq 0.25$ is an acceptable performance criterion for a prediction model. The smaller value of RRMS suggests the better fit. On the other hand, the MSE is also used for comparison, and is defined as [35], [38]

$$\mathrm{MSE} = \frac{\sum_{i=1}^{k}[m_i - m(t_i)]^2}{k-\theta}. \tag{49}$$

2) After the proposed model is fitted to the actual observed data, the deviation between the observed and the fitted values can be evaluated by using the K-S test, or the chi-square (χ^2) test. The K-S test is considered to be the more informative of the two [2], [39]. The KD is defined by [2]

$$D_k = \mathrm{Sup}_x \left|F^*(x) - F(x)\right|, \tag{50}$$

where k is the sample size. It takes the absolute difference between the normalized cumulative distributions of the observed rates and the expected rates from the model at each point, and chooses the maximal value among these differences.

3) The *Predictive Validity* Criterion. This criterion was proposed by Musa *et al.* [39]. The capability of the model to predict failure behavior from the present and the past failure behavior is called *Predictive Validity*, which can be represented by computing the value of RE for a data set

$$\text{Relative Error(RE)} = \frac{m(t_q) - m_q}{m_q}. \tag{51}$$

Assuming we have observed m_q failures by the end of test time t_q, we employ the failure data up to time $t_e (t_e \leq t_q)$ to estimate the parameters of $m(t)$. Substituting the estimates of these parameters in the MVF yields the estimate of the number of failures $m(t_q)$ by t_q. The estimate is compared with the actual number m_q, and the procedure will be repeated for various values of t_e. We can generally check the predictive validity by plotting the RE for different values of t_q. Positive values of error indicate overestimation; negative values of error indicate underestimation. Musa [39] reported that, if a model is found to have the best predictive validity based on failure data, it may yield the best values of other reliability quantities.

C. Performance Analysis

In this section, we compare the proposed models to some existing SRGM. Due to the limitations of space, we only give detailed discussions of the results from two models: the G-O model with multiple CP, and the ISS model with multiple CP. On the other hand, the parameters of SRGM can typically be estimated by using the methods of LSE, and MLE [4], [30]. The method of least squares minimizes the sum of squares of the deviations between what we actually observe and what we expect. The maximum likelihood technique estimates parameters by solving a set of simultaneous equations. Because MLE generally tends to be biased [3], but LSE can produce unbiased results [30], we decide to use LSE to estimate the parameters of all selected models.

1) DS-1: As shown in Table II, entries 1–30 were obtained during the system validation phase, entries 31–42 were obtained during field trials, and entries 43–81 were obtained during system operation. Therefore, in the following, we will show the evaluation results of the models using 37% (i.e., the faults observed in (0, 30]), 52% (i.e., the faults observed in (0, 42]), and 100% (i.e., all faults observed in (0, 81]) of the historical failure data. Table V summarizes the estimated parameters, and the comparison results of the traditional G-O model using 37%, 52%, and 100% of the data. Furthermore, because this data set is collected from different phases, we can assume that the derivation of MVF would take CP into consideration. Here

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

TABLE V
PARAMETER ESTIMATION AND COMPARISON RESULT OF ALL SELECTED MODELS (DS-1)

Model	*a*	*b*	τ	RMSE	RRMS	MSE	KD	Remarks
G-O model (37% of data)	2383.960	4.471×10^{-3}	—	116.55	0.377	14554.6	0.239	
G-O model (52% of data)	727.566	1.652×10^{-2}	—	32.95	0.107	1140.35	0.127	
G-O model (100% of data)	546.082	2.421×10^{-2}	—	9.99	0.032	102.39	0.057	
G-O model with a single CP (52% of data)	669.845	$b_1=1.848\times10^{-2}$ $b_2=1.876\times10^{-2}$	$\tau_1=31$	27.2	0.088	818.31	0.112	
G-O model with two CP (100% of data)	531.548	$b_1=2.372\times10^{-2}$ $b_2=3.075\times10^{-2}$ $b_3=2.627\times10^{-2}$	$\tau_1=31$ $\tau_2=43$	9.35	0.03	94.44	0.051	
ISS model (37% of data)	405.679	6.936×10^{-2}	—	28.16	0.091	881.11	0.112	$\psi=1.883$
ISS model (52% of data)	556.212	3.355×10^{-2}	—	20.1	0.065	435.09	0.089	$\psi=0.646$
ISS model (100% of data)	499.502	3.735×10^{-2}	—	9.29	0.03	89.68	0.050	$\psi=0.581$
ISS with a single CP (52% of data)	424.529	$b_1=6.822\times10^{-2}$ $b_2=6.749\times10^{-2}$	$\tau_1=31$	19.63	0.064	437.50	0.098	$\psi=2.160$
ISS with two CPs (100% of data)	506.985	$b_1=2.949\times10^{-2}$ $b_2=4.395\times10^{-2}$ $b_3=3.397\times10^{-2}$	$\tau_1=31$ $\tau_2=43$	9.19	0.03	92.56	0.049	$\psi=0.355$
Yamada DSS model (37% of data)	343.89	1.049×10^{-1}	—	61.17	0.198	4009.14	0.214	
Yamada DSS model (52% of data)	382.06	9.341×10^{-2}	—	39.36	0.127	1627.28	0.160	
Yamada DSS model (100% of data)	472.637	6.901×10^{-2}	—	20.4	0.066	426.93	0.106	
Generalized Goel NHPP model (37% of data)	2071.16	5.08×10^{-3}	—	113.15	0.366	14224.33	0.235	$c=1.007$
Generalized Goel NHPP model (52% of data)	549.139	1.718×10^{-2}	—	15.69	0.051	265.31	0.076	$c=1.105$
Generalized Goel NHPP model (100% of data)	506.776	1.927×10^{-2}	—	9.16	0.03	87.09	0.049	$c=1.103$

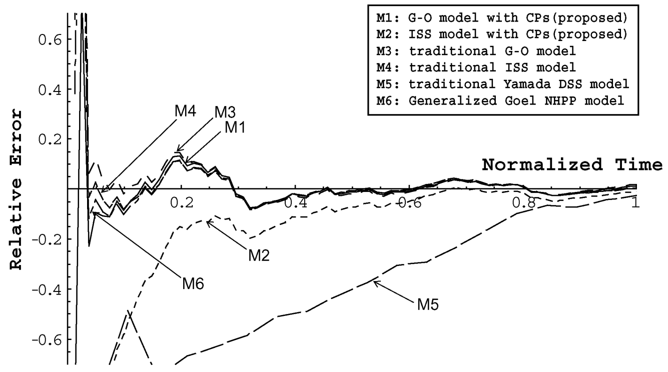


Fig. 7. Relative error curves for the selected models based on DS-1.

we can use (30) to estimate software reliability growth. Table V also shows the estimated parameters and the comparison results of all selected models using 37%, 52%, and 100% of the data. Fig. 7 shows the relative errors for the selected models against normalized time. We can see that, except for the traditional DSS model, after 50% of the end of test time, most models project the future behavior well for this data set, and the error curve is within ±5 percent. It is also noted that some of the dispersion in the projection in Fig. 7 may be due to different sizes (i.e., 37%, 52%, and 100%) of failure data. In fact, the accuracy of the project is direct related to the sample size of the failure data. Therefore, after this point, it could be more important to select other criteria.

From Table V, we can see that the RMSE, RRMS, and MSE of the G-O model with multiple CP using 52% and 100% of the data are less than those of the traditional G-O model using 52% and 100% of the data. Moreover, we also see that the G-O model with multiple CP achieves smaller values of the KD. On the whole, it is reasonable to conclude that the G-O model with multiple CP has a better goodness-of-fit than the traditional G-O model. It is also obvious that the performance improvement is achieved by introducing the concepts of multiple CP to reflect the possible changes of various fault detection rates and environments in the different phases instead of making the traditional assumption of a constant fault detection rate.

On the other hand, the estimated parameters for the traditional ISS model are also given in Table V. As with the previous model, we estimate the traditional ISS model using 37%, 52%, and 100% of the actual data. Table V summarizes the comparison results of the traditional ISS model using 37%, 52%, and 100% of the data. As before, we consider that the derivations of MVF needed to take CP into consideration, and engage (34) to estimate software reliability growth. Table V shows the estimated parameters for the ISS model with multiple CP using 37%, 52%, and 100% of the data. From Fig. 7, the ISS model with multiple CP seems to be biased to the underestimation side when projection is made before 40% of the end of test time. After that, the relative error would tend to approach zero. From Table V, we can see that the RMSE and RRMS of the ISS model with multiple CP using 52% and 100% of the data are still smaller than those of the traditional ISS model using 52% and 100% of the data. See from Table V that the ISS model with multiple CP using 52% and 100% doesn't provide the smaller MSE

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

TABLE VI
PARAMETER ESTIMATION AND COMPARISON RESULT OF ALL SELECTED MODELS (DS-2)

Model	a	b	τ	RMSE	RRMS	MSE	KD	Remarks
G-O model (66% of data)	5611.11	1.012×10^{-3}	—	29.53	0.229	924.69	0.252	—
G-O model (100% of data)	301.292	2.431×10^{-2}	—	8.74	0.068	79.56	0.113	—
G-O model with a single CP (100% of data)	669.845	b_1=3.173×10^{-2} b_2=3.383×10^{-2}	τ_1=35	6.79	0.053	75.74	0.087	—
ISS model (66% of data)	229.235	8.761×10^{-2}	—	7.62	0.059	63.62	0.076	ψ=3.694
ISS model (100% of data)	205.515	1.071×10^{-1}	—	2.58	0.02	7.08	0.037	ψ=4.681
ISS model with a single CP (100% of data)	204.779	b_1=1.121×10^{-1} b_2=1.104×10^{-1}	τ_1=35	2.53	0.02	7.15	0.036	ψ=5.548
Yamada DSS model (66% of data)	225.084	8.812×10^{-2}		5.62	0.044	33.47	0.055	—
Yamada DSS model (100% of data)	213.487	9.345×10^{-2}	—	3.96	0.031	16.36	0.051	—
Generalized Goel NHPP model (66% of data)	285.509	9.93×10^{-3}	—	14.28	0.111	223.14	0.142	c=1.319
Generalized Goel NHPP model (100% of data)	207.964	8.28×10^{-3}	—	3.32	0.026	11.72	0.045	c=1.542

compared to the traditional ISS model, but the differences are small. We still find that the ISS model with multiple CP achieves smaller values of the KD.

Finally, when using 52% (100%) of the data, from Table V, we can find that the RMSE of the G-O model with a single CP is about a 17.45% (28.24%) decrement compared with the value of RMSE computed by the traditional G-O model. This means that the G-O model with CP has more accurate early prediction capability of the defects compared with the traditional G-O model. Actually, this information is very useful for project managers to plan ahead, and to better schedule and manage the testing process. When using 100% of the data, from Table V, we have the inflection parameter $\psi = 0.581$ of the traditional ISS model. That is, the inflection rate (i.e., ratio of the number of detectable faults to the total number of faults in the program) is 0.633, indicating the growth curve is slightly S-shaped.

2) DS-2: Similarly, we show the evaluation results of the models using 66% (i.e., the faults observed in (0, 35]), and 100% (i.e., all faults observed in (0, 51]) of the historical failure data. Table VI summarizes the estimated parameters and the comparison results of the G-O model with or without single CP, and the ISS model with or without single CP using 66%, and 100% of the data. Table VI also shows the estimated parameters and the comparison results of the Yamada DSS model and Generalized Goel NHPP model using 66%, and 100% of the data. The relative errors for all selected models can be plotted as shown in Fig. 8. We can see that, except for the traditional DSS model, after 30 percent of the end of test time, most models project the future behavior well for this data set, and the error curve is within $\pm10\%$. Therefore, after this point, it could be more important to select other criteria. Fig. 8 shows that the G-O model with a single CP is superior to the traditional G-O model in projective validity. Also, the traditional G-O model, the traditional ISS model, the G-O model with a single CP, and the ISS model with a single CP tend to be biased to the underestimation side when projection is made before 30% of the end of test time. But the Yamada DSS model always tends to underestimate.

As we can see from Table VI, the RMSE, RRMS, and MSE of the G-O model with a single CP using 100% of the data are smaller than those of the traditional G-O model using 100% of the data. It is also clear from Table VI that the G-O model with a single CP has smaller values of the KD, and we can conclude that this model has a better goodness-of-fit. It is observed that incorporating the concept of CP(s) into the traditional G-O model improves the descriptive properties of the model, and the predictive properties as well.

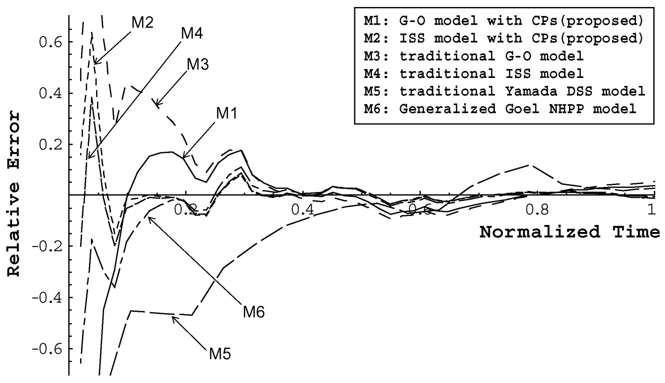


Fig. 8. Relative error curves for the selected models based on DS-2.

Table VI also summarizes the comparison results of the traditional ISS model using 66%, and 100% of the data. As shown in Table VI, both RMSE, and RRMS of the ISS model with a single CP are smaller than those of the traditional ISS model. Although the ISS model with multiple CP using 100% doesn't provide the least MSE compared to the traditional ISS model, but the difference is very small. However, we still see that the ISS model with a single CP obtains smaller values of the KD. Finally, it is also noted that that, when using 100% of the data, we have the inflection parameter $\psi = 4.681$ of the traditional ISS model. Therefore, the inflection rate is 0.176, and this indicates an S-shaped growth curve. This is equivalent to assuming that only a few faults are detectable at the beginning, and faults rapidly become detectable thereafter.

3) DS-3: As shown in Table VII, we list the evaluation results of all selected models using 43%, 86%, and 100% of the third data set. Table VII summarizes the estimated parameters of all selected models using 43%, 86%, and 100% of the data; and the performance of the selected models as measured by the four criteria. Fig. 9 plots the relative errors for all selected models.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

TABLE VII
PARAMETER ESTIMATION AND COMPARISON RESULT OF ALL SELECTED MODELS (DS-3)

Model	a	b	τ	RMSE	RRMS	MSE	KD	Remarks
G-O model (43% of data)	158.958	6.793×10^{-2}	—	20.76	0.226	553.934	0.159	
G-O model (86% of data)	339.316	2.959×10^{-2}	—	5.13	0.056	29.601	0.084	
G-O model (100% of data)	489.766	1.939×10^{-2}	—	4.32	0.047	20.625	0.074	
G-O model with a single CP (86% of data)	385.037	b_1=2.546×10^{-2} b_2=2.494×10^{-2}	τ_1=10	4.87	0.053	30.53	0.080	
G-O model with two CPs (100% of data)	758.559	b_1=1.029×10^{-2} b_2=1.328×10^{-2} b_3=1.14×10^{-2}	τ_1=10 τ_2=19	4.13	0.045	29.78	0.066	
ISS model (43% of data)	157.683	6.949×10^{-2}	—	20.83	0.227	650.65	0.160	ψ=0.016
ISS model (86% of data)	173.272	1.379×10^{-1}	—	6.92	0.075	57.501	0.117	ψ=1.865
ISS model (100% of data)	477.601	2.06×10^{-2}	—	4.32	0.047	21.782	0.073	ψ=0.037
ISS model with a single CP (86% of data)	203.717	b_1=1.008×10^{-1} b_2=9.744×10^{-2}	τ_1=10	5.73	0.062	45.47	0.099	ψ=1.321
ISS model with two CP (100% of data)	285.943	b_1=2.644×10^{-2} b_2=1.204×10^{-1} b_3=6.817×10^{-2}	τ_1=10 τ_2=19	4.1	0.045	25.21	0.073	ψ=1.539
Yamada DSS model (43% of data)	78.9474	4.093×10^{-1}	—	41.33	0.451	2196.27	0.441	
Yamada DSS model (86% of data)	152.703	2.009×10^{-1}	—	10.13	0.11	115.34	0.141	
Yamada DSS model (100% of data)	179.54	1.672×10^{-1}	—	7.91	0.086	69.078	0.119	
Generalized Goel NHPP model (43% of data)	183.236	6.475×10^{-2}	—	20.59	0.225	635.992	0.147	c=0.929
Generalized Goel NHPP model (86% of data)	253.685	3.534×10^{-2}	—	5.61	0.061	37.823	0.092	c=1.077
Generalized Goel NHPP model (100% of data)	612.331	1.638×10^{-2}	—	4.28	0.047	21.32	0.071	c=0.968

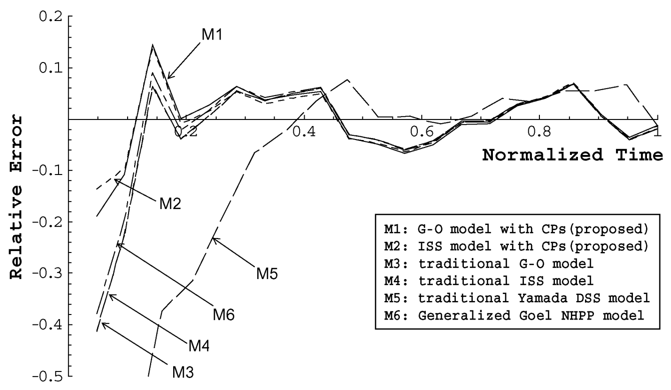


Fig. 9. Relative error curves for the selected models based on DS-3.

Note that, except the traditional DSS model, after 20% of the end of test time, other models project the future behavior well for DS-3, and the error curve is within $\pm 10\%$. We can find that the G-O model with multiple CP, and the ISS model with multiple CP, on the whole yield the best projection for this data set.

On the other hand, Table VII shows that the values of RMSE and RRMS of the G-O model with multiple CP using 86% and 100% of the data are smaller than those of the traditional G-O model using 86% and 100% of the data. Although the G-O model with multiple CP using 52% and 100% doesn't provide the least MSE compared to the traditional G-O model, but the differences are small. However, we still see that the G-O model with multiple CP has smaller KD.

Similarly, the estimated parameters for the ISS model with or without multiple CP are also given in Table VII. From Table VII, see that the values of RMSE and RRMS of the ISS model with multiple CP using 86% and 100% of the data are smaller than those of the traditional ISS model using 86% and 100% of the data. Note that the MSE of the ISS model with multiple CP using 86% of the data are smaller than that of the traditional ISS model using 86% of the data. Although the ISS model with multiple CP using 100% doesn't provide the smaller MSE compared to the traditional ISS model, but the difference is small. However, we still see that the ISS model with multiple CP results in smaller KD. Thus, it is obvious that the ISS model with multiple CP predicts more accurately than the traditional ISS model.

Finally, from Table VII, when 86% of the data are available, we can find that the MSE of the ISS model with a single CP is about 20.24% decrement compared with the value of MSE computed by the traditional ISS model. Similarly, when using 86% of the data, we also find that the RMSE of the ISS model with a single CP is about a 17.2% decrement compared with the value of the RMSE computed by the traditional ISS model. This shows that the ISS model with CP has more accurate capability of early fault prediction, and improves the descriptive properties of the traditional ISS model. It can also be noted that, when using 100% of the data, we have inflection parameter $\psi = 0.037$ of the traditional ISS model. That is, the inflection rate is 0.968, signifying that the growth curve is highly exponential. It is equivalent to assuming that all faults of a program

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

are detectable from the beginning of software testing activities, resulting in a substantial improvement in the prediction performance of the G-O model with multiple CP.

IV. Imperfect Debugging Problem

In general, different SRGM make different modeling assumptions, and therefore can be applied to different situations. Sometimes these assumptions help to reduce the complexity of modeling software reliability [1], [2], [4]. Most SRGM, in particular, assume that each time a failure occurs, the fault that caused the failure is immediately removed, and no new faults are introduced. However, we know that software debugging is a labor-intensive process of identifying the cause of software defective behavior and addressing that problem, and iI n reality, developers experience cases where they fix one bug but could create another new one.

Ohba and Chou [40], for example, reported that about 14 percent of the faults detected and removed during an observation period introduced new faults as a result of imperfect debugging. Goel and Okumoto [41] also showed that the imperfect debugging model provided a good fit to the software failure data from a real-time control system for a land-based radar system developed by the Raytheon Company. In another study, Yin *et al.* [42] reported that imperfect debugging should be taken into consideration when the software product is reaching a mature stage, where the number of remaining faults and the number of introduced faults are in the same order of magnitude [43]. Actually, the problem of imperfect debugging has been addressed by many papers [44]–[48]. In the following, we incorporate a relaxation of some common assumptions to make the SRGM more realistic and practical. Specifically, we consider SRGM with multiple CP, and imperfect debugging.

By modifying (16), if we let $m(t+\Delta t)$ be equal to the quasi-arithmetic mean of $m(t)$ and $n(t)$ with weights $w(t,\Delta t)$ and $1-w(t,\Delta t)$, then

$$\begin{aligned} g\left(m(t+\Delta t)\right) &= w(t,\Delta t)g\left(m(t)\right) \\ &\quad + \left(1-w(t,\Delta t)\right)g\left(n(t)\right), \qquad 0<w(t,\Delta t)<1. \end{aligned} \tag{52}$$

Note that $n(0)=a$. That is,

$$\begin{aligned} \frac{g\left(m(t+\Delta t)\right)-g\left(m(t)\right)}{\Delta t} &= \frac{1-w(t,\Delta t)}{\Delta t} \\ &\quad \times \left(g\left(n(t)-g\left(m(t)\right)\right)\right), \quad 0<w(t,\Delta t)<1. \end{aligned} \tag{53}$$

Supposing $(1-w(t,\Delta t))/\Delta t \to b(t)$ as $\Delta t \to 0$, we get the differential equation

$$\frac{\partial}{\partial t} g\left(m(t)\right) = b(t)\left\{g\left(n(t)\right)-g(m(t)\right\}. \tag{54}$$

For example, if $g(x)=x$, then (54) becomes

$$\frac{\partial}{\partial t} g\left(m(t)\right) = b(t)\left(n(t)-m(t)\right). \tag{55}$$

Theorem 2: Let $(\partial/\partial t)g(m(t)) = b(t)(g(n(t))-g(m(t))$, where g is a real-valued, strictly monotonic, and differentiable function. We have

$$m(t) = g^{-1}\left(g\left(n(t)\right)+\left\{g\left(m(0)\right)-g\left(n(t)\right)\right\}\exp\left[-B(t)\right]\right), \tag{56}$$

and $B(t)=\int_0^t b(u)du$.

In the following, we describe how to incorporate imperfect debugging in the modeling approach we have established.

A. G-O Model With Multiple CPS and Imperfect Debugging

Firstly, we can describe the G-O model with a single CP under imperfect debugging as

$$\frac{dm(t)}{dt} = b(t)\times\left(n(t)-m(t)\right), \tag{57}$$

and

$$b(t) = \begin{cases} b_1(t)=b_1 & \text{if } 0\le t\le\tau; \\ b_2(t)=b_2 & \text{if } t>\tau. \end{cases}$$

There are some fault content functions in [42]–[44], but further discussion of this subject is beyond the scope of this paper. Here we simply assume that

$$\frac{dn(t)}{dt} = \beta\frac{dm(t)}{dt}. \tag{58}$$

Therefore, (58) becomes

$$\frac{dm(t)}{dt} = b(t)\times\left[a+(\beta-1)m(t)\right]. \tag{59}$$

Solving (59), and assuming $m(0)=0$, we obtain the MVF as [See (60) at the bottom of the page]. In fact, (60) can also be derived based on the unified theory. When $0\le t\le\tau$, if we take $g(x)=x$, $k_1=1-m(0)/a$, and $B_1(t)=\int_0^\tau b_1du$, then from Theorem 2 and (58) with $n(0)=a$, we obtain the MVF

$$m_1(t) = \frac{a}{1-\beta}\left(1-\exp\left[-(1-\beta)b_1t\right]\right). \tag{61}$$

Furthermore, when $t>\tau$, if we take $g(x)=x$, and $B_2(t)=\int_\tau^t b_2du$, from Theorem 2, we have

$$m_2(t) = \frac{a}{1-\beta}\left(1-k_2\exp\left[-(1-\beta)b_2(t-\tau)\right]\right). \tag{62}$$

$$m(t) = \begin{cases} m_1(t)=\frac{a}{1-\beta}\left(1-\exp\left[-(1-\beta)b_1t\right]\right) & \text{,when } 0\le t\le\tau \\ m_2(t)=\frac{a}{1-\beta}\left(1-\exp\left[-(1-\beta)(b_2(t-\tau)+b_1\tau)\right]\right) & \text{,when } \tau<t \end{cases} \tag{60}$$

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

TABLE VIII
PARAMETER ESTIMATION OF SELECTED MODELS WITH MULTIPLE CPS CONSIDERING ID* (DS-1)

Model	a	b	ψ	τ	β	RMSE	RRMS	MSE
G-O model (37% of data)	2087.16	0.005	—	—	0.125	116.55	0.377	15093.66
G-O model with a single CP & ID (52% of data)	635.401	b_1=0.019 b_2=0.019	—	τ_1=31	0.051	27.21	0.088	840.432
G-O model with two CPs & ID (100% of data)	508.692	b_1=0.025 b_2=0.032 b_3=0.027	—	τ_1=31 τ_2=43	0.043	9.35	0.03	95.72
ISS model (37% of data)	530.412	0.043	1.255	—	0.025	26.21	0.085	792.65
ISS model with a single CP & ID (52% of data)	416.188	b_1=0.069 b_2=0.068	2.119	τ_1=31	0.022	19.33	0.063	436.04
ISS model with two CPs & ID (100% of data)	497.94	b_1=0.032 b_2=0.039 b_3=0.035	0.359	τ_1=31 τ_2=43	0.018	9.19	0.03	93.8

*ID: imperfect debugging

Note that

$$k_2 = 1-(1-\beta)m_1(\tau)/a = \exp\left[-(1-\beta)b_1\tau\right]. \tag{63}$$

Thus we have

$$m_2(t) = \frac{a}{1-\beta}\left(1-\exp\left[-(1-\beta)\left(b_1\tau + b_2(t-\tau)\right)\right]\right). \tag{64}$$

In this case, the fault content function is given by [See (65) at the bottom of the page]. Finally, if

$$\begin{aligned} k_n &= 1-(1-\beta)m_{n-1}(\tau_{n-1})/a \\ &= \exp\left[-(1-\beta)\sum_{k=1}^{n-1} b_k \times (\tau_k - \tau_{k-1})\right], \end{aligned} \tag{66}$$

we obtain a generalized solution for describing the G-O model with multiple CP under imperfect debugging:

$$m_n(t) = \frac{a}{1-\beta} \times\left(1-\exp\left[-(1-\beta)\left(b_n\times(t-\tau_{n-1})+\sum_{k=1}^{n-1} b_k\times(\tau_k-\tau_{k-1})\right)\right]\right), \tag{67}$$

and

$$n_n(t) = \frac{a}{1-\beta} \times\left(1-\beta\exp\left[-(1-\beta)\left[b_n(t-\tau_{n-1})+\sum_{k=1}^{n-1} b_k(\tau_k-\tau_{k-1})\right]\right]\right). \tag{68}$$

The mathematical expressions for other selected SRGM with multiple CP, and imperfect debugging, are given in the Appendix.

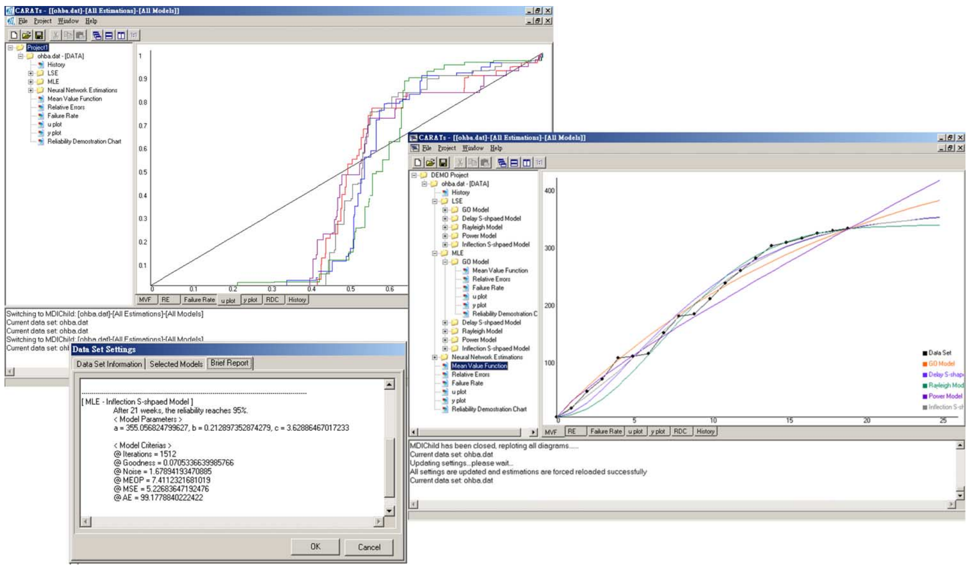
B. Experiment and Discussion: From Theory to Reality

Due to space limitations, here we only select the G-O model with multiple CP, and the ISS model with multiple CP to address the issue of imperfect debugging. Table VIII gives the estimated values of model parameters for DS-1. As shown in Table VIII, we observe that the fault removal process in the software development and testing environment may not be a perfect debugging process, because the estimated values of the fault introduction rate β are all close, but not equal to zero. For example, when using 37%, 52%, and 100% of the historical failure data, the estimated fault introduction rate of the G-O model with multiple CP is 12.5×10^{-2}, 5.1×10^{-2}, and 4.3×10^{-2}, respectively. So when we use 100% of the failure data, about five faults will be introduced per 100 removed faults. Similarly, when using 37%, 52%, and 100% of failure data, the estimated fault introduction rate of the ISS model with multiple CP is 2.5×10^{-2}, 2.2×10^{-2}, and 1.8×10^{-2}, respectively. On average, the result means that about two extra faults will be introduced if 100 faults are removed. Consequently, we can find that the introduction of new faults during the correction process is not a negligible issue for this data set, especially for the high severity faults if they are introduced accidentally.

In general, among various SRGM, there are two most important factors affecting reliability: the number of initial faults, and the fault detection rate [1]. The number of initial faults is the number of faults in the software at the beginning of the test, and is usually a representative measure of software reliability. But this number could be changed by an imperfect debugging phenomenon. Knowing the number of residual faults can also help us determine how much more testing resources are required, and whether the software is suitable for customers to use or not. On the other hand, the fault detection rate is engaged to measure the effectiveness of fault detection by test techniques and test cases. In this paper, we assume that, during the software

$$n(t) = \begin{cases} n_1(t) = \frac{a}{1-\beta}\left(1-\beta\exp\left[-(1-\beta)b_1t\right]\right) & \text{,when } 0 \le t \le \tau \\ n_2(t) = \frac{a}{1-\beta}\left(1-\beta\exp\left[-(1-\beta)\left[b_2(t-\tau)+b_1\tau\right]\right]\right) & \text{,when } \tau < t \end{cases} \tag{65}$$

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.



```
Switching to MDIChild: [ohba.dat]-[All Estimations]-[All Models]
Current data set: ohba.dat
Current data set: ohba.dat
Switching to MDIChild: [ohba.dat]-[All Estimations]-[All Models]
Current data set: ohl
```

```
[ MLE - Inflection S-shpaed Model ]
        After 21 weeks, the reliability reaches 95%.
        < Model Parameters >
        a = 355.056824799627, b = 0.212897352874279, c = 3.62886467017233

        < Model Criterias >
        @ Iterations = 1512
        @ Goodness = 0.0705336639985766
        @ Noise = 1.67894193470885
        @ MEOP = 7.4112321681019
        @ MSE = 5.22683647192476
        @ AE = 99.1778840222422
```

```
MDIChild has been closed, reploting all diagrams......
Current data set: ohba.dat
Updating settings...please wait...
All settings are updated and estimations are forced reloaded successfully
Current data set: ohba.dat
```

Fig. 10. Some user-interface views of CARATS.

development process, the fault detection rate may be neither a constant nor smooth, and may change at some CP. By adding some extra parameters in modeling software reliability growth, the estimation becomes more difficult as more numerical calculations are involved. But these additional calculations can be fully automated.

In the past, we have developed a versatile object-oriented software reliability assessment tool called CARATS [49], which provides quantitative estimation of software project management during testing. CARATS is originally developed for software failure data collection, software reliability analysis, parameter estimation, software cost estimation, etc. We use C++ to implement the tool. In this paper, our proposed models have also been incorporated into CARATS. Fig. 10 displays three screenshots of several working views of CARATS.

Finally, SRGM can produce reasonable results when enough failure data sets are used. When SRGM are employed for estimation of reliability growth, representative models need to be selected and applied sensibly. The objective of software reliability growth modeling is to predict the sequence of successive inter-failure execution times of a program, but the limitations of SRGM are inherent, and may not be eliminated by clever modeling [50]. It is thus generally accepted that, although SRGM sometimes give good results, there is no single model that can be selected *a priori* in providing accurate results under all circumstances [2], [36]. Nevertheless, software reliability researchers keep attacking the modeling problems from different angles, and they often come up with innovative ideas and approaches, such as the recalibration technique, the weighted combinational models, the simulation approach, the infinite server queueing approach, etc [1], [51]–[55]. In this paper, we take a new angle on the software reliability modeling process, and address multiple CP observed in the software development processes. From the numerical results in Sections IV and V, we see that the proposed new approaches can be considered as favorable software reliability models from both the theoretical and the practical viewpoints.

V. CONCLUSION

Diversity of software faults is one of main contributing factors of software unreliability. In practical, analyzing fault distribution and estimating failure intensity are very important tasks during software development because they provide very useful, valuable information for project managers to make decisions. Over the past three decades, many SRGM have been proposed for software reliability assessment, and the theory of SRGM typically involves the nature of stochastic process and statistical analysis. In this paper, we first review how several existing SRGM based on NHPP can be derived by adopting the concept of weighted arithmetic, weighted geometric, and weighted harmonic means. Based on a unified theory, we further propose several SRGM with multiple CP, and these models can be used for reliability estimation and prediction in the testing and operational phases. We also show that most existing SRGM based on NHPP can be improved by incorporating the concepts of multiple CP, which enables them to provide fairly good measurement capability for software reliability. Our goal is not to add one more model to the already-existing large number of SRGM, but to emphasize a new approach for the development of software reliability models. Moreover, our approach in describing the working status of various software operational environments is very flexible, as we can model various environments ranging from an exponential NHPP to an S-shaped growth curve. Based on the integrated theoretical foundation, the technologies and approaches presented in this paper offer a consistent, quantitative software reliability evaluation scheme in both the testing and operational phases.

Finally, by adding some extra parameters when modeling the fault detection process, the estimation becomes more tedious as more numerical calculations are involved. However, these additional calculations can be fully automated. When high reliability is required, as for some critical applications such as very large-scale commercial software, safety-critical flight software, or online banking service, the cost of the extra computation required to obtain more accuracy can be easily justified.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

Appendix
SRGM With Multiple CP, and Imperfect Debugging

Generalized Goel NHPP Model With Multiple CP, and Imperfect Debugging

We can follow a similar procedure to that described in Section IV-B. Therefore, for the generalized Goel NHPP model with multiple CP under imperfect debugging, we have the MVF

$$m_n(t) = \frac{a}{1-\beta} \times \left(1-\exp\left[-(1-\beta)\left(b_n\times\left(t^C-\tau_{n-1}^C\right)+\sum_{k=1}^{n-1} b_k\times\left(\tau_k^C-\tau_{k-1}^C\right)\right)\right]\right) \tag{69}$$

and

$$n_n(t) = \frac{a}{1-\beta} \times \left(1-\beta\exp\left[-(1-\beta)\left[b_n\times\left(t^C-\tau_{n-1}^C\right)+\sum_{k=1}^{n-1} b_k\times\left(\tau_k^C-\tau_{k-1}^C\right)\right]\right]\right), \tag{70}$$

where $\tau_0 = 0$, and $\tau_{n-1} < t$.

ISS Model With Multiple CP, and Imperfect Debugging

We follow a similar procedure to that described in Section IV-B; therefore, for the ISS model with multiple CP under imperfect debugging, we have

$$m_n(t) = \frac{a}{1-\beta}\left(1-\left(\frac{(\psi+\exp[-b_n\tau_{n-1}])\times\exp[-b_n t]}{1+\psi\exp[-b_n t]} \times \prod_{k=1}^{n-1}\frac{(\psi+\exp[-b_k\tau_{k-1}])\times\exp[-b_k\tau_k]}{1+\psi\exp[-b_k\tau_k]}\right)^{1-\beta}\right), \tag{71}$$

and

$$n_n(t) = \frac{a}{1-\beta}\left(1-\beta\left(\frac{(\psi+\exp[-b_n\tau_{n-1}])\times\exp[-b_n t]}{1+\psi\exp[-b_n t]} \times \prod_{k=1}^{n-1}\frac{(\psi+\exp[-b_k\tau_{k-1}])\times\exp[-b_k\tau_k]}{1+\psi\exp[-b_k\tau_k]}\right)^{1-\beta}\right). \tag{72}$$

Yamada Weibull Model With Multiple CP, and Imperfect Debugging

We follow a similar procedure to that described in Section IV-B; therefore, for the Yamada Weibull model with multiple CP under imperfect debugging, we have

$$m_n(t)=\frac{a}{1-\beta}\left(1-\exp\left[-(1-\beta)\left(b_n\times(W(t)-W(\tau_{n-1})) + \sum_{k=1}^{n-1} b_k\times(W(\tau_k)-W(\tau_{k-1}))\right)\right]\right), \tag{73}$$

and

$$n_n(t)=\frac{a}{1-\beta}\left(1-\beta\exp\left[-(1-\beta)\left(b_n\times(W(t)-W(\tau_{n-1})) + \sum_{k=1}^{n-1} b_k\times(W(\tau_k)-W(\tau_{k-1}))\right)\right]\right). \tag{74}$$

Yamada DSS Model With Multiple CP, and Imperfect Debugging

We follow a similar procedure to that described in Section IV-B; therefore, for the Yamada DSS model with multiple CP under imperfect debugging, we have the MVF [See (75) at the bottom of the page] and [See (76) at the bottom of the page].

$$m_n(t) = \frac{a}{1-\beta}\left(1-\left(\frac{(1+b_n t)\prod_{i=1}^{n-1}(1+b_i\tau_i)\exp\left[-\left(b_n\times(t-\tau_{n-1})+\sum_{i=1}^{n-1} b_i\times(\tau_i-\tau_{i-1})\right)\right]}{\prod_{i=1}^{n-1}(1+b_i\tau_{i-1})}\right)^{1-\beta}\right), \tag{75}$$

$$n_n(t) = \frac{a}{1-\beta}\left(1-\beta\left(\frac{(1+b_n t)\prod_{i=1}^{n-1}(1+b_i\tau_i)\exp\left[-\left(b_n\times(t-\tau_{n-1})+\sum_{i=1}^{n-1} b_i\times(\tau_i-\tau_{i-1})\right)\right]}{\prod_{i=1}^{n-1}(1+b_i\tau_{i-1})}\right)^{1-\beta}\right). \tag{76}$$

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

Acknowledgment

The authors would like to express their gratitude to the associate editor and anonymous reviewers for their insightful comments, encouragement, and suggestions.

References

[1] M. R. Lyu, "Software reliability engineering: A roadmap," in *Proceedings of the 29th International Conference on Software Engineering (ICSE 2007), Future of Software Engineering*, Minneapolis, MN, May 2007, pp. 153–170.

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

[3] J. D. Musa, *Software Reliability Engineering: More Reliable Software, Faster Development and Testing*, 2nd ed. Bloomington, IN: AuthorHouse, 2004.

[4] M. Xie, *Software Reliability Modeling*. New York: World Scientific Publishing Company, 1991.

[5] C. Y. Huang and C. T. Lin, "Analysis of software reliability modeling considering testing compression factor and failure-to-fault relationship," *IEEE Trans. on Computers*, vol. 59, no. 2, pp. 283–288, Feb. 2010.

[6] C. Y. Huang and M. R. Lyu, "Optimal testing resource allocation and sensitivity analysis in software development," *IEEE Trans. on Reliability*, vol. 54, no. 4, pp. 592–603, Dec. 2005.

[7] C. Y. Huang, M. R. Lyu, and S. Y. Kuo, "A unified scheme of some non-homogenous poisson process models for software reliability estimation," *IEEE Trans. on Software Engineering*, vol. 29, no. 3, pp. 261–269, March 2003.

[8] C. Y. Huang, S. Y. Kuo, M. R. Lyu, and J. H. Lo, "Quantitative software reliability modeling from testing to operation," in *Proceedings of the 11th IEEE International Symposium on Software Reliability Engineering (ISSRE 2000)*, San Jose, CA, Oct. 2000, pp. 72–82.

[9] K. Tokuno and S. Yamada, "Markovian availability measurement with two types of software failures during the operation phase," *International Journal of Reliability, Quality and Safety Engineering*, vol. 6, no. 1, pp. 43–56, March 1999.

[10] D. A. Christenson, "Using software reliability models to predict field failure rates in electronic switching systems," in *Proceedings of the 4th Annual National Joint Conference on Software Quality and Productivity*, Washington, DC, March 1988, pp. 158–163.

[11] P. K. Kapur, A. K. Bardhan, and N. L. Butani, "Modelling failure phenomenon of a commercial software in operational use," in *Proceedings of the 2001 International Conference on Quality, Reliability and Control*, Mumbai, 2001.

[12] T. Philip, P. N. Marinos, and K. S. Trivedi, A Multiphase Software Reliability Model: From Testing to Operational Phase Center for Advanced Computing and Communication, Duke University, , Technical Report (TR-96-01), January 1996.

[13] H. Okamura, T. Dohi, and S. Osaki, "A reliability assessment method for software products in operational phase—Proposal of an accelerated life testing model," *Electronics and Communications in Japan*, vol. 84, no. 8, pp. 25–33, 2001.

[14] B. Yang and M. Xie, "A study of operational and testing reliability in software reliability analysis," *Reliability Engineering and Systems Safety*, vol. 70, no. 2, pp. 323–329, 2000.

[15] Q. Kenney, "Estimating defects in commercial software during operational use," *IEEE Trans. on Reliability*, vol. 42, no. 1, pp. 107–115, March 1993.

[16] H. Ohtera and S. Yamada, "Optimum software-release time considering an error-detection phenomenon during operation," *IEEE Trans. on Reliability*, vol. 39, no. 5, pp. 596–599, Dec. 1990.

[17] S. Yamada, "Software Reliability Measurement during Operation Phase and its Application," *Journal of Computer and Software Engineering*, vol. 1, no. 4, pp. 389–402, 1993.

[18] S. Keene and C. Lane, "Reliability Growth of Fielded Software," in *Proceedings of the 1993 Annual Reliability and Maintainability Symposium (RAMS'93)*, Atlanta, GA, USA, Jan. 1993, pp. 360–365.

[19] D. R. Jeske, X. Zhang, and L. Pham, "Adjusting software failure rates that are estimated from test data," *IEEE Trans. on Reliability*, vol. 54, no. 1, pp. 107–114, March 2005.

[20] A. L. Goel, "Relating operational software reliability and workload: Results from an experimental study," in *Proceedings of the 1996 IEEE Annual Reliability and Maintainability Symposium (RAMS'96)*, Las Vegas, NV, January 1996, pp. 167–172.

[21] Y. Chen, "Modeling software operational reliability via input domain-based reliability growth model," in *Proceedings of the 28th IEEE International Symposium on Fault-Tolerant Computing (FTCS'98)*, Munich, Germany, Jun. 1998, pp. 314–323.

[22] R. Chillarege, R. K. Iyer, J. C. Laprie, and J. D. Musa, "Field failures and reliability in operation," in *Proceedings of the 4th IEEE International Symposium on Software Reliability Engineering (ISSRE'93)*, Denver, CO, Nov. 1993, pp. 122–126.

[23] Z. Wang and J. Wang, "Parameter estimation of some NHPP software reliability models with change-point," *Communications in Statistics: Simulation and Computation*, vol. 34, no. 1, pp. 121–134, Feb. 2005.

[24] M. Zhao, "Statistical reliability change-point estimation models," in *Handbook of Reliability Engineering*. New York: McGraw-Hill, 2003, pp. 157–163.

[25] C. T. Lin and C. Y. Huang, "Enhancing and measuring the predictive capabilities of the testing-effort dependent software reliability models," *Journal of Systems and Software*, vol. 81, no. 6, pp. 1025–1038, June 2008.

[26] C. Y. Huang, "Performance analysis of software reliability growth models with testing-effort and change-point," *Journal of Systems and Software*, vol. 76, no. 2, pp. 181–194, May 2005.

[27] X. Li, M. Xie, and S. H. Ng, "Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points," *Applied Mathematical Modelling*, vol. 34, no. 11, pp. 3560–3570, Nov. 2010.

[28] J. Zhao, H. W. Liu, G. Cui, and X. Z. Yang, "Software reliability growth model with change-point and environmental function," *Journal of Systems and Software*, vol. 79, no. 12, pp. 1578–1587, Dec. 2006.

[29] C. Y. Huang and C. T. Lin, "Reliability prediction and assessment of fielded software based on multiple change-point models," in *Proceedings of the 11th IEEE International Symposium on Pacific Rim Dependable Computing (PRDC'05)*, Changsha, Hunan, China, Dec. 2005, pp. 379–386.

[30] M. Xie, "Software reliability models—Past, present and future," in *Recent Advances in Reliability Theory: Methodology, Practice and Inference*. Birkhauser, Boston: , 2000, pp. 323–340.

[31] K. Kanoun, M. Martini, and J. Souza, "A method for software reliability analysis and prediction application to the TROPICO-R switching system," *IEEE Trans. on Software Engineering*, vol. 17, no. 4, pp. 334–344, April 1991.

[32] D. R. Jeske, X. Zhang, and L. Pham, "Adjusting software failure rates that are estimated from test data," *IEEE Trans. on Reliability*, vol. 54, no. 1, pp. 107–114, March 2005.

[33] Y. Tamura and S. Yamada, "Comparison of software reliability assessment methods for open source software," in *Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS 2005)*, Los Almitos, CA, July 2005, pp. 488–492.

[34] W. A. Florac and A. D. Carleton, *Measuring the Software Process: Statistical Process Control for Software Process Improvement (SEI Series in Software Engineering)*. New York: Addison-Wesley, 1999.

[35] W. K. Ehrlich and T. J. Emerson, "Modeling software failures and reliability growth during system testing," in *Proceedings of the 9th International Conference on Software Engineering (ICSE'87)*, Monterey, CA, March 30–April, 2, 1987, pp. 72–82.

[36] P. Rook, *Software Reliability Handbook*. New York: Elsevier Science Inc., 1990.

[37] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models*. Redwood City, CA: Benjamin-Cummings Publishing Co., Inc., 1986.

[38] N. R. Draper and H. Smith, *Applied Regression Analysis*, 3rd ed. New York: Wiley, 1998.

[39] J. D. Musa, A. Iannino, and K. Okumoto, *Software Reliability, Measurement, Prediction and Application*. New York: McGraw Hill, 1987.

[40] M. Ohba and X. Chou, "Does imperfect debugging affect software reliability growth?," in *Proceedings of the 11th International Conference on Software Engineering (ICSE'89)*, Pittsburgh, PA, May 1989, pp. 237–244.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.

[41] A. L. Goel and K. Okumoto, "An analysis of recurrent software errors in a real-time control system," in *Proceedings of the 1978 Annual Conference (ACM'78)*, Washington, DC, Dec. 1978, pp. 496–501.

[42] M. L. Yin, L. E. James, S. Keene, R. R. Arellano, and J. Peterson, "An adaptive software reliability prediction approach," in *Proceedings of the 23rd Annual Software Engineering Workshop*, Greenbelt, MD, Dec. 1998, NASA/Goddard Space Flight Center.

[43] M. C. J. van Pul, "Statistical Analysis of Software Reliability Models," PhD. Dissertation, Department of Mathematics, Utrecht University, , Netherlands, 1993.

[44] S. Yamada, K. Tokuno, and S. Osaki, "Software reliability measurement in imperfect debugging environment and its application," *Reliability Engineering & System Safety*, vol. 40, no. 2, pp. 139–147, 1993.

[45] H. Pham, L. Nordmann, and X. Zhang, "A general imperfect software debugging model with S-shaped fault detection rate," *IEEE Trans. on Reliability*, vol. 48, no. 2, pp. 169–175, June 1999.

[46] X. Zhang, X. Teng, and H. Pham, "Considering fault removal efficiency in software reliability assessment," *IEEE Trans. on Systems, Man, and Cybernetics—Part A: Systems and Humans*, vol. 33, no. 1, pp. 2241–2252, Jan. 2003, 114-120.

[47] M. Xie and B. Yang, "A study of the effect of imperfect debugging on software development cost," *IEEE Trans. on Software Engineering*, vol. 29, no. 5, pp. 471–473, May 2003.

[48] P. E. Amman, S. S. Brilliant, and J. Knight, "The effect of imperfect error detection on reliability assessment via life testing," *IEEE Trans. on Software Engineering*, vol. 20, no. 2, pp. 142–148, Feb. 1994.

[49] C. C. Chen, C. T. Lin, H. H. Huang, S. W. Huang, and C. Y. Huang, "CARATS: A computer-aided reliability assessment tool for software based on object-oriented design," in *CD-ROM Proceedings of 2006 IEEE Region 10 Conference (TENCON 2006)*, Hong Kong, China, Nov. 2006.

[50] B. Littlewood, "Dependability assessment of software-based systems: State of the art," in *Proceedings of the 27th International Conference on Software Engineering (ICSE 2005)*, St. Louis, MO, May 2005, pp. 6–7.

[51] S. Brocklehurst, P. Y. Chan, B. Littlewood, and J. Snell, "Recalibrating software reliability models," *IEEE Trans. on Software Engineering*, vol. 16, no. 4, pp. 458–470, April 1990.

[52] C. J. Hsu and C. Y. Huang, "Reliability analysis using weighted combinational models for web-based software," in *Proceedings of the 18th International World Wide Web Conference (WWW 2009)*, Madrid, Spain, April 2009, pp. 1131–1132.

[53] C. T. Lin and C. Y. Huang, "Staffing level analysis of software debugging through rate-based simulation approaches," *IEEE Trans on Reliability*, vol. 58, no. 4, pp. 711–724, Dec. 2009.

[54] S. S. Gokhale and M. R. Lyu, "A simulation approach to structure-based software reliability analysis," *IEEE Trans. on Software Engineering*, vol. 31, no. 8, pp. 643–656, Aug. 2005.

[55] C. Y. Huang and W. C. Huang, "Software reliability analysis and measurement using finite and infinite server queueing models," *IEEE Trans. on Reliability*, vol. 57, no. 1, pp. 192–203, March 2008.

Chin-Yu Huang (M'05) received the M.S. (1994), and the Ph.D. (2000) in electrical engineering from National Taiwan University, Taipei. He is currently an Associate Professor in the Department of Computer Science at National Tsing Hua University, Hsinchu, Taiwan. 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. He received the Ta-You Wu Memorial Award from the National Science Council of Taiwan in 2008. He also received an Honorable Mention Best Paper Award in IEEM'2010. His research interests are software reliability engineering, software testing, software metrics, software testability, fault tree analysis, and system safety assessment.

Michael R. Lyu (M'10) received the B.S. (1981) in electrical engineering from National Taiwan University, the M.S. (1985) in computer engineering from University of California, Santa Barbara, and the Ph.D. (1988) in computer science from University of California, Los Angeles. He is a Professor in the Computer Science and Engineering Department of the Chinese University of Hong Kong. He worked at the Jet Propulsion Laboratory, Bellcore, and Bell Labs; and taught at the University of Iowa. His research interests include software reliability engineering, software fault tolerance, distributed systems, image and video processing, multimedia technologies, and mobile networks. He has published over 330 papers in these areas. He has participated in more than 30 industrial projects, and helped to develop many commercial systems and software tools. Professor Lyu is frequently invited as a keynote or tutorial speaker to conferences and workshops in U.S., Europe, and Asia. He initiated the International Symposium on Software Reliability Engineering (ISSRE), and was Program Chair for ISSRE'1996, Program Co-Chair for WWW10, SRDS'2005 and ICEBE'2007, and General Chair for ISSRE'2001 and PRDC'2005. He also received Best Paper Awards in ISSRE'98 and in ISSRE'2003, and SigSoft Distinguished Paper Award in ICSE'2010. He is the editor-in-chief for two book volumes: Software Fault Tolerance (Wiley, 1995), and the Handbook of Software Reliability Engineering (IEEE and McGraw-Hill, 1996). He has been an Associate Editor of IEEE Trans. Reliability, IEEE Trans. Knowledge and Data Engineering, Journal of Information Science and Engineering, and Wiley Software Testing, Verification & Reliability Journal. Professor Lyu is an IEEE Fellow, and an AAAS Fellow, for his contributions to software reliability engineering and software fault tolerance.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 28,2020 at 07:37:51 UTC from IEEE Xplore. Restrictions apply.