doi:10.3772/j.issn.1006-6748.2015.01.008

A software reliability growth model for component-based software incorporating debugging delay and imperfect debugging①

Zhang Ce (张 策)②*** , Cui Gang* , Meng Fanchao** , Liu Hongwei* , Bian Yali**
(* School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, P. R. China)
(** School of Computer Science and Technology, Harbin Institute of Technology at Weihai, Weihai 264209, P. R. China)

Abstract

In view of the problems and the weaknesses of component-based software (CBS) reliability modeling and analysis, and a lack of consideration for real debugging circumstance of integration testing, a CBS reliability process analysis model is proposed incorporating debugging time delay, imperfect debugging and limited debugging resources. CBS integration testing is formulated as a multi-queue multichannel and finite server queuing model (MMFSQM) to illustrate fault detection process (FDP) and fault correction process (FCP). A unified FCP is sketched, given debugging delay, the diversities of faults processing and the limitations of debugging resources. Furthermore, the impacts of imperfect debugging on fault detection and correction are explicitly elaborated, and the expressions of the cumulative number of fault detected and corrected are illustrated. Finally, the results of numerical experiments verify the effectiveness and rationality of the proposed model. By comparison, the proposed model is superior to the other models. The proposed model is closer to real CBS testing process and facilitates software engineer's quantitatively analyzing, measuring and predicting CBS reliability.

Key words: software reliability, component-based software (CBS), debugging delay, imperfect debugging, queuing theory

0 Introduction

Software reliability plays a very important role in the software life cycle, especially in the development phase. With the improvement of scale and complexity of software, the development of object-oriented techniques, component-based software (CBS) has been widely used, and this research on CBS reliability has already become the focus of study in recent years.

CBS testing process can be divided into unit testing, integration testing and system testing. Most of research efforts available concentrate on the integration testing stage. For example, for architecture-based software reliability, Gokhale[1] represented a unified framework model based on state, and CBS reliability prediction was carried out using an analytical method. In Ref. [2], the reliability of individual component was evaluated firstly, and then the whole system reliability was predicted. Likewise, Ref. [3] assessed the whole CBS reliability by calculating the execution path reliability based on the reliability of component and system structure. The CBS reliability was analyzed dynamically through calculating and evaluating execution path reliability of three standard structures in program in Ref. [4]. These research efforts focus on directly calculating CBS reliability, but give little consideration to reliability improvement analysis.

In real software testing, the faults detected go through a series of processes: detecting, isolating, analysing, allocating, correcting and verifying, and there exist delays[5]. For example, Ref. [6] carried out an analysis of delay between fault detection process (FDP) and fault correction process (FCP), and concluded that mean value function (MVF) of FCP can be obtained by failure density function $\lambda_d(t)$ of FDP and time delay Δ. Ref. [7] got MVF of FCP by introducing a delay impact factor $\varphi(t)$. Considering the delay in fault correcting, the number of corrected lags behind the number of detected in $[0,t]$, and often there are

① Supported by the National High Technology Research and Development Program of China (No. 2008AA01A201), the National Natural Science Foundation of China (No. 60503015), the National Key R&D Program of China (No. 2013BA17F02) and the Shandong Province Science and Technology Program of China (No. 2011GGX10108, 2010GGX10104).

② To whom correspondence should be addressed. E-mail: zhangce@hitwh.edu.cn

Received on Dec. 4, 2013

circumstances that the faults detected are waiting to be corrected. So, applying queuing theory into software testing becomes a natural choice. For instance, Kapur[8] employed an infinite server queue to build a relatively unified software reliability analysis model, conducted the reliability analysis of different types of fault correction process and achieved good effect finally. Huang[9] finished the software reliability analysis using finite and infinite server queue, and considered the change point problem[10]. Lin[11] used a single queue multichannel model to make an analysis of perfect and imperfect debugging based on simulation, and incorporated debugging delay and the limitations of debugging resources. Hou[12] adopted a hybrid queue model to conduct CBS reliability analysis considering the incompleteness and unlimited debuggers, but the shortages were the absences of considering debugging delay and introducing new faults.

Thus it can be seen that, in these software reliability studies available including CBS based on analytical model or simulation, some studies have involved explicit correction efforts and delay between FDP and FCP, but the study of real debugging process is still not deep and thorough, and there are still some insufficiencies in the following aspects:

(1) There are too many assumptions in analyses available: ignore the sub-processes of debugging and time delay, debugging is perfect, and no new faults are introduced[5-8];

(2) The severity and the differences of faults detected in testing are ignored;

(3) The research efforts[6-11] above are not suited to reliability analysis of CBS integration testing;

(4) Furthermore, the debuggers are regarded as unlimited.

In response to the problems and deficiencies of extant research, this study is mainly concentrated on the CBS reliability analysis when incorporating debugging delay, limited debugging resources and imperfect debugging in the integration testing stage.

1 Finite server queuing model MMFSQM considering debugging delay and imperfect debugging

Let CBS S consists of N components, that is $S = \{C_i, 1 \leqslant i \leqslant N\}$. CBS integration testing of S can be described as a multi-queue multichannel and finite server queuing model (MMFSQM), as shown in Fig. 1.

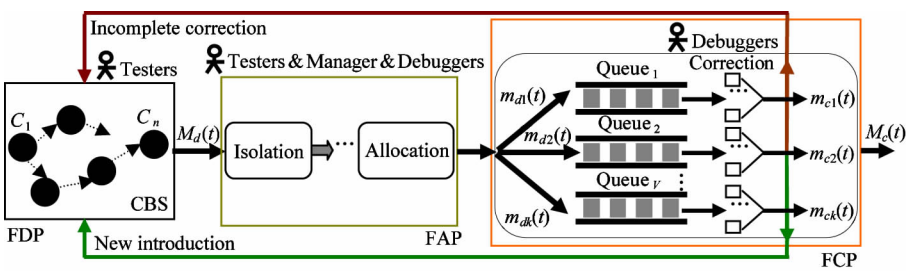

Fig. 1 Multi-queue multichannel and finite server queuing model considering debugging delay and imperfect debugging

Fig. 1 is a gray-box test model, which executes the integration testing based on an operation profile of CBS. In order to describe accurately the testing process, testing is divided into FDP and debugging process. Thereinto, FDP denotes detecting faults by testers and debugging process can be furthermore partitioned into isolating, analyzing (determining the root), allocating and correcting. As shown in Fig. 1, debugging process includes fault analysis process (FAP) and fault correction process (FCP), and there exist debugging delays. FCP consists of different types of fault correction queues and debugging resources to accomplish fault correcting. The faults of component C_i enter different fault correction queue FCQ_j according to the fault allocation strategy (FAS) and are allocated debugging resources in the light of current queues state. Here, debugging resources mainly refer to as the debuggers of queues.

Compared with perfect debugging, there will exist incomplete debugging and introduction of new faults in the debugging process. So, there will be two feedback lines between FCP and FDP, and both cases might be called imperfect debugging in this work.

Testing begins at the component level and different reliability enhancement techniques are appropriate according to complexity, function and important place of different components. Obviously, the faults from different components will go through different sub-processes of FAP and be allocated to different correction queues of FCP, due to complexity and severity. In or-

der to reflect the diversity of debugging, the faults detected from components could be divided into two groups: $C = F_C \cup F_G$.

(1) The faults from critical component C_i are classified into serious fault set F_C;

(2) The faults from the other component C_j are classified into simple fault set F_G.

Hereon, the faults outputted by FDP are allocated to different correction queues by FAS. Accordingly, we set V fault correction queues, the maximum capacity of each queue to be $m_i (1 \leqslant i \leqslant V)$ and the queues can be classified into two kinds of fault correction queues:

(1) Serious fault correction queue FCQ_{FC}: serious fault F_C enters FCQ_{FC} according to first come first server (FCFS);

(2) Simple fault correction queue FCQ_{FG}: simple fault F_G enters FCQ_{FG} according to FCFS.

Compared to available queue models with unlimited debuggers[8,9,12], each fault correction queue of FCP consists of limited debuggers.

The subsequent analysis is based on the following assumptions[8-15]:

(1) Let the failure process satisfy non-homogeneous poisson process (NHPP) distribution, that is (easy to be proven) time interval of failure is random variable with the same distribution of exponential type;

(2) In the process of CBS testing, detecting, isolating, determining root and correcting are independent;

(3) Debugging process is imperfect, that is, there exist incomplete debugging and introducing new faults;

(4) The failures among the components are independent.

By the assumptions and analysis above, the CBS integration testing process consists of an $M/M/V/m_i$ queue system with multi types of service windows.

2 CBS integration testing process analysis considering debugging delay and imperfect debugging

2.1 The unified CBS integration testing process analysis considering debugging delay

It is assumed that random variable X is the moment of detecting fault in the integration testing and the fault w occurs at $x(0 \leqslant x \leqslant T)$. Let random variable Z be the time of fault isolating and determining the root with probability distribution function $Z(t)$ and probability density function $z(t)$ and random variable Y be the time of fault correction by FCQ_i with probability distribution function $F(t)$ and probability density function $f(t)$.

Due to limited debuggers of MMFSQM, fault w will enter into the waiting queue when it cannot be allocated debugging resources timely. Thus, considering the actual working process of MMFSQM, there will be following cases where fault w is corrected in $[x, T]$:.

Fault w is removed thoroughly with probability p and is unfinish corrected (including w in waiting queue and in correcting) with probability r. Furthermore, incorporating imperfect debugging, incomplete correction and introduction of new faults are denoted by probability q and $h(h << p$, in most cases). Obviously, $p + r + q + h = 1$ will be obtained.

Let random variable K be the number of debuggers and k be the number of available debuggers at t. Due to limited debuggers, the probability of the debuggers being idle at t is $k(t)$.

Hereon, it is supposed that a complete fault correction process needs to go through detection, isolation, determining root cause and correction sub-processes. So, let $N_{jd}(t)$, $N_{ji}(t)$, $N_{js}(t)$ and $N_{jr}(t)$ be four stochastic counting processes representing fault detecting, isolation, determining root and correction respectively. Obviously, it can be drawn:

$$\begin{aligned} P\{N_{jr}(t) = m\} &= \sum_{n=0}^{a_j(t)} P\{N_{jr}(t) = m \mid N_{jd}(t) = n\} \times P\{N_{jd}(t) = n\} \\ &= \sum_{n=0}^{a_j(t)} P\{N_{jr}(t) = m \mid N_{jd}(t) = n\} \times \frac{[m_{jd}(t)]^n e^{-m_{jd}(t)}}{n!} \\ &= \sum_{n=0}^{a_j(t)} C_n^m [p_j(t)]^m [1 - p_j(t)]^{n-m} \times \frac{[m_{jd}(t)]^n e^{-m_{jd}(t)}}{n!} \\ &= \left\{ \frac{[m_{jd}(t) p_j(t)]^m e^{-[m_{jd}(t) p_j(t)]}}{m!} \right\} \times \\ &\quad \left\{ \sum_{n=0}^{a_j(t)} \frac{[m_{jd}(t) - m_{jd}(t) p_j(t)]^{n-m} e^{-[m_{jd}(t) - m_{jd}(t) p_j(t)]}}{(n-m)!} \right\} \end{aligned} \tag{1}$$

where $a_j(t)$ is the number of total faults of component C_j in $[0,t]$, $m_{jd}(t)$ is the cumulative number of faults detected from C_j and $p_j(t)$ is the probability of complete fault correction in correcting queue. It was clear that when considering imperfect debugging $a_j(t)$ is the increasing function with testing time t, and when $a_j(t)$ is large especially $a_j(t) \to \infty$, it can be got:

$$P\{N_{jr}(t) = m\} \approx \frac{[m_{jd}(t) p_j(t)]^m e^{-[m_{jd}(t) p_j(t)]}}{m!} \tag{2}$$

Based on the nature of Poisson distribution process, it can be got:

$$m_{jr}(t) = m_{jd}(t) p_j(t) \tag{3}$$

2.2 CBS integration testing analysis considering two types of fault debugging delay

In the integration testing of CBS, the severity and complexity of different components are considerably different. Due to central key business functionality, the faults coming from complex and critical component (for example, fault-tolerant component) will be corrected going through multiple sub-processes. By contrast, the faults from simple and non-critical component can be corrected going through only relatively small number of sub-processes. Of course, different levels of imperfect debugging phenomena may exist due to some uncertain influence factors during the debugging process above.

Considering fault severity, limited debuggers and imperfect debugging in MMFSQM, the performance of CBS integration testing process analysis includes two types of faults. These two kinds of faults refer to serious fault F_c and simple fault F_s respectively.

2.2.1 The correction process of serious fault considering debugging delay

For serious fault $F_c \in F_C$, the time spent from detection to correction is the longest. Hereon, it is assumed that F_c will go through detecting, isolating and correcting sub-processes. Based on the assumptions above, the following can be got:

$$\begin{aligned} &P\{Y \leqslant t-y \cap X = x \cap Z = y \cap K < k_1\} \\ &\quad = P\{Y \leqslant t-y \cap X = x \cap Z = y\} \times P\{K < k_1\} \\ &\quad = P\{Y \leqslant t-y \mid X = x\} \times P\{X = x\} \times P\{Z = y\} \\ &\qquad \times P\{K < k_1\} \\ &\quad = F(t-y) \times P\{X = x\} \times P\{Z = y\} \times P\{K < k_1\} \end{aligned} \tag{4}$$

At some point, the probability that fault has been corrected or being corrected is $p_1 + r_1 = 1 - q_1 - h_1$. Based on the analysis above, the following can be obtained:

$$\begin{aligned} &p_1 + r_1 = 1 - q_1 - h_1 \\ &= \int_0^t \int_x^t P\{Y \leqslant t-y \cap X = x \cap Z = y \cap K < k_1\} \,\mathrm{d}y\mathrm{d}x \\ &= \int_0^t \int_x^t F(t-y) \times P\{X = x\} \times z(y) \times P\{K < k_1\} \,\mathrm{d}y\mathrm{d}x \\ &= P\{K < k_1\} \int_0^t \int_x^t F(t-y) \times z(y) \times P\{X = x\} \,\mathrm{d}y\mathrm{d}x \end{aligned} \tag{5}$$

Let fault F_c enter FCQ_{FC} and the probability of complete correction in FCQ_{FC} is ρ_{c_1}. So,

$$\begin{aligned} p_{j1}(t) &= \rho_{c1} \times (p_1 + r_1) \\ &= \rho_{c1} P\{K < k_1\} \int_0^t \int_x^t F(t-y) \times z(y) \\ &\quad \times P\{X = x\} \,\mathrm{d}y\mathrm{d}x \end{aligned} \tag{6}$$

Here, let $P\{K < k_1\} = k_1(t)$ satisfy the following relation[15]:

$$P\{K < k_1\} = k_1(t) = \sum_{l=0}^{k_1-1} \frac{\beta_1^l e^{-\beta_1}}{l!} \tag{7}$$

where l denotes the number of faults being corrected and β_1 is a positive constant. For component C_j, at t failure probability is

$$P\{X = x\} = \frac{m_{jd}'(\kappa_j x)}{m_{jd}(\kappa_j t)} \tag{8}$$

where κ_j is the execution time proportion of C_j in the integration testing of CBS S:

$$\kappa_j = \frac{\omega_j \sum_{i=1}^{n} p_{ji} t_{ji}}{\sum_{i=1}^{n} \omega_i \sum_{k=1}^{n} p_{ik} t_{ik}} \tag{9}$$

Among these parameters, p_{ij} is transition probability from C_i to C_j, $\omega = [\omega_i]$ represents execution probability of C_i in steady state and t_{ij} denotes mean execution time of C_i when the transition (C_i, C_j) occurs.

Substitute Eqs(6) ~ (9) into Eq. (3), the following equation can be derived as

$$\begin{aligned} m_{jr}(\kappa_j t) &= m_{jd}(\kappa_j t)\rho_{c1} \times (p_1 + r_1) \\ &= m_{jd}(\kappa_j t)\rho_{c1} P\{K < k_1\} \int_0^t \int_x^t F(t-y) z(y) \\ &\quad P\{X = x\} \,\mathrm{d}y\mathrm{d}x \\ &= m_{jd}(\kappa_j t)\rho_{c1} \left(\sum_{l=0}^{k_1-1} \frac{\beta_1^l e^{-\beta_1}}{l!} \right) \int_0^t \int_x^t F(t-y) z(y) \\ &\quad \left(\frac{m_{jd}'(\kappa_j x)}{m_{jd}(\kappa_j t)} \right) \mathrm{d}y\mathrm{d}x \\ &= \rho_{c1} \left(\sum_{l=0}^{k_1-1} \frac{\beta_1^l e^{-\beta_1}}{l!} \right) \int_0^t \int_x^t F(t-y) z(y) \\ &\quad m_{jd}'(\kappa_j x) \,\mathrm{d}y\mathrm{d}x \end{aligned} \tag{10}$$

2.2.2 The correction process of simple fault considering debugging delay

Hereon, it is assumed that simple $F_g \in F_G$ will go through detecting and correcting two sub-processes. Taking simple fault F_g from C_s, correction process analysis is done.

Also, like serious fault above, the following can be got:

$$P\{N_{sr}(t) = m\} = \frac{[m_{sd}(t) p_{s2}(t)]^m e^{-[m_{sd}(t) p_{s2}(t)]}}{m!} \tag{11}$$

$$m_{sr}(t) = m_{sd}(t) p_{s2}(t) \tag{12}$$

where $p_{s2}(t)$ is the probability that fault is removed completely in correcting queue.

$$\begin{aligned} p_2 + r_2 &= 1 - q_2 - h_2 \\ &= \int_0^t P\{Y \leqslant t-x \cap X = x \cap K < k_2\} \,\mathrm{d}x \\ &= \int_0^t F(t-x) \times P\{X = x\} \times P\{K < k_2\} \,\mathrm{d}x \end{aligned}$$

$$= P\{K < k_2\} \times \int_0^t F(t-x) \times P\{X = x\}\,\mathrm{d}x \tag{13}$$

Let fault F_g enter FCQ_{FG} and complete correction probability in FCQ_{FG} is ρ_{s_2}. So

$$\begin{aligned} p_{s2}(t) &= \rho_{s2} \times (p_2 + r_2) \\ &= \rho_{s2} \times P\{K < k_2\} \times \int_0^t F(t-x) \times P\{X = x\}\,\mathrm{d}x \end{aligned} \tag{14}$$

Substitute Eq. (14) into Eq. (12), with simplification, Eq. (15) can be obtained:

$$\begin{aligned} m_{sr}(\kappa_s t) &= m_{sd}(\kappa_s t)\rho_{s2} \times (p_2 + r_2) \\ &= m_{sd}(\kappa_s t)\rho_{s2} P\{K < k_2\} \int_0^t F(t-x) P\{X = x\}\,\mathrm{d}x \\ &= m_{sd}(\kappa_s t)\rho_{s2}\left(\sum_{l=0}^{k_2-1} \frac{\beta_2^l e^{-\beta_2}}{l!}\right)\int_0^t F(t-x)\left(\frac{m_{sd}'(\kappa_s x)}{m_{sd}(\kappa_s t)}\right)\mathrm{d}x \\ &= \rho_{s2}\left(\sum_{l=0}^{k-1} \frac{\beta_2^l e^{-\beta_2}}{l!}\right)\int_0^t F(t-x) m_{sd}'(\kappa_s x)\,\mathrm{d}x \end{aligned} \tag{15}$$

Obviously, when considering more sub-processes of debugging, $m_{jr}(\kappa_j t)$ and $m_{sr}(\kappa_s t)$ will become complicated.

2.2.3 Fault detection and correction of CBS integration testing considering debugging delay

Exponential distribution is usually taken to be cumulative distribution function of service time (namely fault correction time). So let fault correction time spent in queues of MMFSQM obey parameter μ exponent distribution. Thus $F(y) = 1 - e^{-\mu y}$ is got with probability density function $f(y) = \mathrm{d}F(y)/\mathrm{d}y = \mu e^{-\mu y}$. Substitute $f(y)$ into Eq. (10) and Eq. (15), with simplification, one can be obtained:

$$\begin{cases} m_{jr}(\kappa_j t) = \rho_{c1}\left(\sum_{l=0}^{k_1-1} \frac{\beta_1^l e^{-\beta_1}}{l!}\right)\int_0^t\int_x^t (1 - e^{-\mu(t-y)}) z(y) \\ \qquad\qquad \times m_{jd}'(\kappa_j x)\,\mathrm{d}y\mathrm{d}x,\ w \in F_c \\ m_{sr}(\kappa_s t) = \rho_{s2}\left(\sum_{l=0}^{k_2-1} \frac{\beta_2^l e^{-\beta_2}}{l!}\right)\int_0^t \mu e^{-\mu(t-x)} m_{sd}(\kappa_s x)\,\mathrm{d}x \\ \qquad\qquad ,w \in F_s \end{cases} \tag{16}$$

The cumulative number of faults detected and corrected of CBS S in $[0,t]$ can be derived as

$$\begin{cases} M_d(t) = \sum_{v=1}^{V} \sum_{C_i \in FRQ_v} m_{di}(\kappa_i t) \\ M_c(t) = \sum_{v=1}^{V} \sum_{C_i \in FRQ_v} m_{ci}(t) = \\ \rho_{c1}\left(\sum_{l=0}^{k_1-1} \frac{\beta_1^l e^{-\beta_1}}{l!}\right)\sum_{C_s \in F_c}\int_0^t\int_x^t (1 - e^{-\mu(t-y)}) z(y) m_{dj}'(\kappa_j x) \\ \mathrm{d}y\mathrm{d}x + \rho_{s2}\left(\sum_{l=0}^{k_2-1} \frac{\beta_2^l e^{-\beta_2}}{l!}\right)\sum_{C_s \in F_s}\int_0^t \mu e^{-\mu(t-x)} m_{sd}(\kappa_s x)\,\mathrm{d}x \end{cases} \tag{17}$$

2.3 Fault detection and correction of CBS integration testing considering debugging delay and imperfect debugging

Here, the reliability modeling of component C_i is mainly based on G-O model[13] and imperfect debugging. In subsequent analysis, MMFSQM is formulated based on the following assumptions:

(1) Fault detection rate is proportional to the number of faults undetected, and $b_i(t)$ is the proportion function;

(2) Fault correction is incomplete with probability function $p_i(t)$;

(3) New faults can be introduced during correction, fault introduction rate is proportional to the number of faults corrected and the probability function is $r_i(t)$ ($r_i(t) << p_i(t)$).

Based on the assumptions above, differential equations can be derived as

$$\begin{cases} \dfrac{\mathrm{d}m_i(t)}{\mathrm{d}t} = b_i(t)[a_i(t) - c_i(t)] \\ \dfrac{\mathrm{d}c_i(t)}{\mathrm{d}t} = p_i(t)\dfrac{\mathrm{d}m_i(t)}{\mathrm{d}t} \\ \dfrac{\mathrm{d}a_i(t)}{\mathrm{d}t} = r_i(t)\dfrac{\mathrm{d}c_i(t)}{\mathrm{d}t} \end{cases} \tag{18}$$

Solving the differential equations above with the boundary conditions of: $m_i(0) = 0$, $a_i(0) = a_i$, $c(0) = 0$ yields:

$$m_i(t) = \int_0^t a_i b_i(v) \times \left\{\left[1 - \int_0^v [1 - r_i(u)] p_i(u) b_i(u) e^{-\int_0^u [1-r_i(\tau)] p_i(\tau) b_i(\tau) d\tau}\mathrm{d}u\right]\mathrm{d}v\right\} \tag{19}$$

$$\lambda_i(t) = \frac{\mathrm{d}m_i(t)}{\mathrm{d}t} = a_i b_i(t) \times \left[1 + \int_0^t [r_i(u) - 1] p_i(u) b_i(u) e^{-\int_0^u [1-r_i(\tau)] p_i(\tau) b_i(\tau) d\tau}\mathrm{d}u\right] \tag{20}$$

For simplicity and tractability, let $b_i(t) = b$; $p_i(t) = p$; $r_i(t) = r$, so Eq. (19) can be simplified as

$$m(t) = \frac{a}{(1-r)p}(1 - e^{-(1-r)pbt}) \tag{21}$$

Substitute Eq. (21) into Eq. (17), the cumulative number of detected and corrected of CBS S in the integration testing can be derived as

$$\begin{cases} M_d(t) = \sum_{v=1}^{V} \sum_{C_i \in FRQ_v} m_{di}(\kappa_i t) = \\ \qquad \sum_{v=1}^{V} \sum_{C_i \in FRQ_v} \frac{a}{(1-r)p}(1 - e^{-(1-r)pb\kappa_i t}) \\ M_c(t) = \sum_{v=1}^{V} \sum_{C_i \in FRQ_v} m_{ci}(t) = \\ ab\rho_{c1}\left(\sum_{l=0}^{k_1-1} \frac{\beta_1^l e^{-\beta_1}}{l!}\right) \sum_{C_j \in F_c} \int_0^t \int_x^t z(y) e^{-(1-r)pb\kappa_j x}(1 - e^{-\mu(t-y)}) \\ \mathrm{d}y\mathrm{d}x + \left(\frac{\mu a e^{-\mu t}\rho_{s2}}{(1-r)p}\right)\left(\sum_{l=0}^{k_2-1} \frac{\beta_2^l e^{-\beta_2}}{l!}\right) \times \sum_{C_s \in F_s} \left[\frac{1}{\mu}(e^{\mu t} - 1) \right. \\ \qquad \left. - \frac{1}{(\mu - (1-r)pb\kappa_s)}(e^{[\mu-(1-r)pb\kappa_s]t} - 1)\right] \end{cases} \tag{22}$$

In this study, incorporating debugging delay and imperfect debugging into integration testing of CBS, we refer to the proposed model as debugging delay and imperfect debugging-CBS reliability growth modeling (DDID-CBSRGM).

3 Numerical illustration

3.1 Numerical example

In recent years, performing software reliability analysis and generating failure data by simulation[9-11,16] (based on discrete event simulation) have been greatly used. Without CBS failure data set available, we verify the potential of proposed model: DDID-CBSRGM through a CBS reported in Ref. [17] as an illustration example that has been extensively used[12,16]. The layout of the specified CBS application is depicted in Fig. 2 and transition probabilities among the components are shown in Table 1.

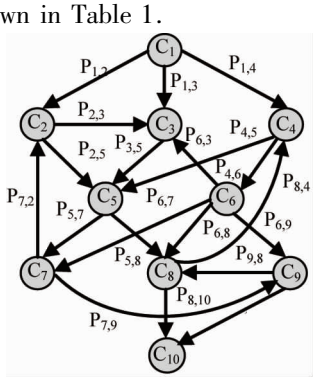

Fig. 2 Architecture of a CBS

Table 1 Transition probabilities among the components

$P_{1,2}=0.60$	$P_{1,3}=0.20$	$P_{1,4}=0.20$	$P_{2,3}=0.70$	$P_{2,5}=0.30$
$P_{3,5}=1.00$	$P_{4,5}=0.40$	$P_{4,6}=0.60$	$P_{5,7}=0.40$	$P_{5,8}=0.60$
$P_{6,3}=0.30$	$P_{6,7}=0.30$	$P_{6,8}=0.10$	$P_{6,9}=0.30$	$P_{7,2}=0.50$
$P_{7,9}=0.50$	$P_{8,4}=0.25$	$P_{8,10}=0.75$	$P_{9,8}=0.10$	$P_{9,10}=0.90$

3.2 Parameter settings

In the previous work, CBS simulation procedure has been developed considering imperfect debugging, and performed fault detection and correction analysis. On that basis, let $b_i = 0.0085$ and $a_i = 40(1 \leqslant i \leqslant N)$. In integration testing, correction rate is related to the skillful degree of debuggers, testing environment and other many factors and cannot be accurately estimated. Without loss of generality, let correction rate $\mu_i = 0.04$. The extent of imperfect debugging is determined by fault introduction rate r_i, let $r_i = 0.15$. Here, simulation procedure is exploited to get failure data set of CBS integration testing. Due to page-limitations, it is not in this paper (contact the author for more information on it). In identifying critical component, critical component is considered to be the one that has more interactive exchange with the other components[2]. So, execution probability ω_i of the components in steady state is calculated, and the maximum value of ω_1 and ω_5 of C_1 and C_5 is discovered in 10 components in Fig. 2. Moreover, Ref. [16] determines that C_1 and C_5 are critical components by means of simulation. Thus, the faults from C_1 and C_5 enter serious fault correction queue: FCQ_1 and FCQ_2, and the faults from the other components enter simple fault correction queue: FCQ_3.

3.3 Evaluation Criteria

To evaluate the models, *MSE*, *Variance*, *RMS-PE* and *R-square* are used to measure the potential performances of the models.

$$MSE = \sum_{i=1}^{n} \frac{(y_i - m(t_i))^2}{n} \tag{23}$$

$$R - square = \frac{\sum_{i=1}^{n} [m(t_i) - \bar{y}]^2}{\sum_{i=1}^{n} [y_i - \bar{y}]^2}, \ \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{24}$$

$$Variance = \sqrt{\frac{\sum_{i=1}^{n} (y_i - m(t_i) - Bias)^2}{n-1}} \tag{25}$$

$$Bias = \frac{\sum_{i=1}^{n} [m(t_i) - y_i]}{n};$$

$$RMS\text{-}PE = \sqrt{Bias^2 + Variance^2} \tag{26}$$

thereinto, $m(t_i)$ represents the estimated value of faults by time t_i, y_i denotes the cumulative number of faults detected and n is the sample size of the real failure data set. Obviously, smaller values of *MSE*, *Vari-*

ance, *RMS-PE* and the closer to 1 of *R-square* indicate a better model than the others.

3.4 Performance validation and analyses

First, the operation processes of FCQ_1, FCQ_2 and FCQ_3 are carried out in the case of different debuggers utilizing CBS simulation procedure and the corresponding residual faults profiles in waiting queue are obtained, as shown in Fig. 3. As can be seen from Fig. 3, the cumulative number of faults waiting to be corrected continues to fall in FCQ_1, FCQ_2 and FCQ_3, as the number of debuggers increases. Thereinto, the faults in waiting queue of FCQ_1 and FCQ_2 are corrected completely when the debugger is 9 persons (i. e. $Debugger_{FCQ1} = Debugger_{FCQ2} = 9$) at the testing time of 4000 and 5000 respectively. This indicates that allocating 9 persons to FCQ_1 and FCQ_2 can meet the requirement for real testing. By contrast, due to the large number of initial faults in FCQ_3, the number of debugger required to be allocated is larger than that of FCQ_1 and FCQ_2 accordingly. It is clearly observed that there are no faults in waiting queue of FCQ_3 under the condition of $Debugger_{FCQ3} = 29$ and $t = 5000$, from the diagram in Fig. 3(c).

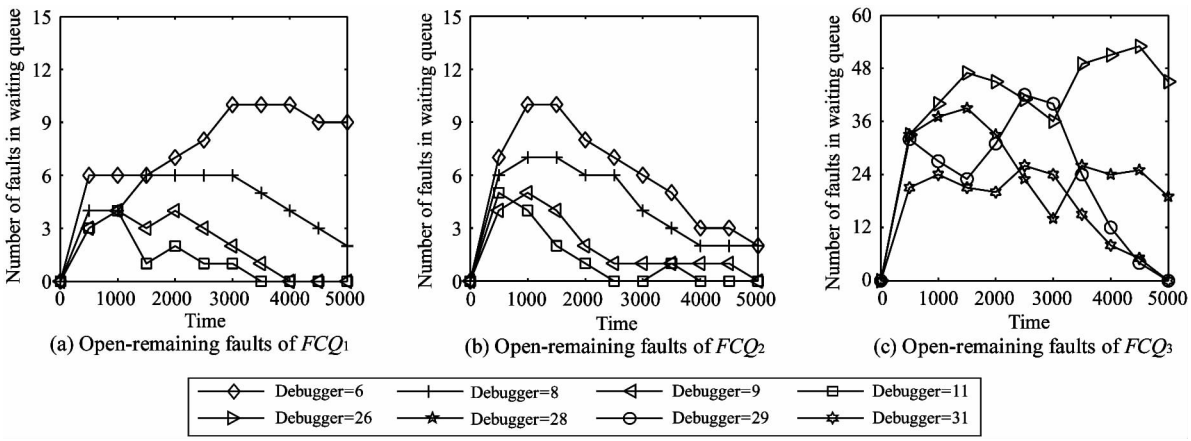

Fig. 3 Open-remaining fault profile

Fig. 4 illustrates the fault profiles including detection and correction of FCQ_1, FCQ_2 and FCQ_3 in case of the numbers of debugger allocated to three queues above. It can be seen fault correction profile of CBS S lags behind evidently the detection profile. It indicates that there exists debugging delay in fault correction process and sub-process of correction can not be ignored. Moreover, as Fig. 4 shows, the number of faults corrected is the same with that of faults detected when $t = 4000$, 5000 and 5000 for FCQ_1, FCQ_2 and FCQ_3, which indicates that all the faults detected have been corrected completely.

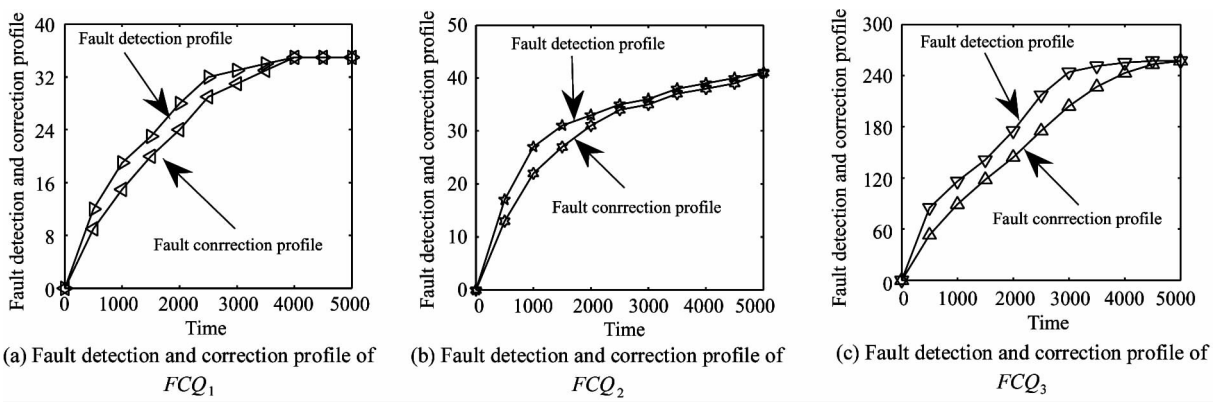

Fig. 4 Fault profile of FCQ_1, FCQ_2 and FCQ_3

Next, to validate the effectiveness of DDID-CB-SRGM, it is compared with CBS reliability models available and imperfect debugging models. These models encompass Hou model[12], the models obtained by

applying three imperfect debugging models proposed respectively by Kapur[8], Pham[18], Ohba[19] to CBS S in Fig. 2, and the model obtained by DDID-CBSRGM without considering debugging delay and imperfect debugging. Hereon, these models can be referred as Hou model, Pham model, Ohba model, Kapur model and CBSRGM respectively. Fig. 5 illustrates graphically the comparisons between the observed failure data, and the data estimated by the six models above. From Fig. 5 (f), the estimated results of DDID-CBSRGM are very close to real fault correction profile especially before t = 2500. By contrast, the other five models engender large errors in different degrees.

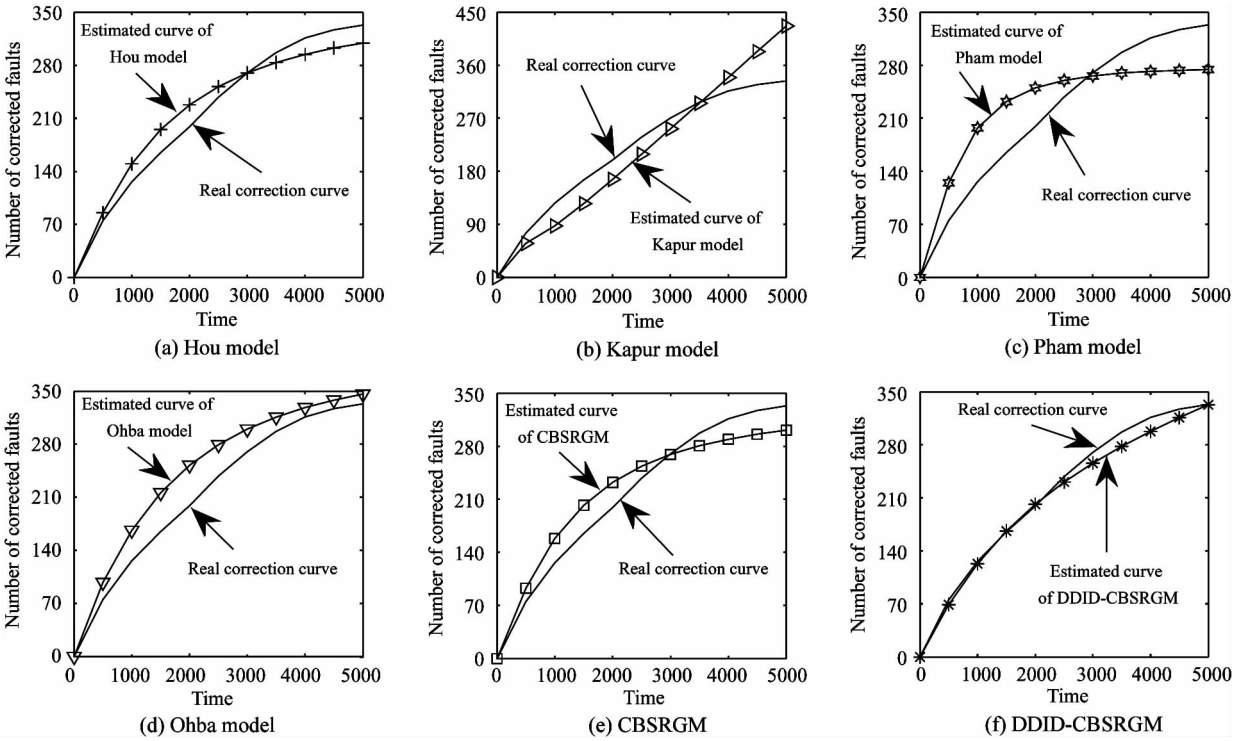

Fig. 5 Observed and estimated cumulative number of corrected faults versus time

In order to makes a further analysis and comparison on the performances, comparison criterion results of performance of six models have been calculated, as shown in Table 2.

Table 2 Comparison results of different models

Model	*MSE*	*R-square*	*Variance*	*RMS-PE*
Hou model	411.637	0.78372	21.3287	21.4689
Kapur model	1607.3233	1.5932	41.2555	41.256
Pham model	2235.153	0.54397	50.2623	50.7598
Ohba model	1022.9015	1.0589	59.0444	65.1367
CBSRGM	655.5762	0.72168	26.808	26.9517
DDID-CBSRGM	104.5748	0.92642	16.4357	17.8973

It is observed that the values *MSE*, *Variance* and *RMS-PE* are the lowest (104.5748, 16.4357 and 17.8973, respectively) among the models considered. Besides, we also see that the value of *R-square* is also the nearest to 1 (0.92642), which demonstrates that DDID-CBSRGM achieves maximum performance in all these models. The reason for this is that, the proposed model incorporates debugging delay and imperfect debugging into reliability modeling, so DDID-CBSRGM has a more accurate description of real debugging process. Moreover, in the other five models, Hou model and CBSRGM have a better performance than the other three models. The probable explanation is that fault detection functions of Hou model and CBSRGM can describe accurately the failure situation than the other three models. In Kapur model, Pham model and Ohba model, the performance of Pham model is inferior to the other two models. This is because Pham model ignores debugging delay, and fault detection rate function $b(t)$ of imperfect debugging model proposed is too subjective and can not describe real testing environment.

Altogether, from the Fig. 5 and Table 2, it can be concluded that the proposed analytical model built by DDID-CBSRGM has a more accurate description of the fault detection and correction process of CBS integration testing, due to considering real testing process.

Especially, the proposed model explicitly incorporates debugging delay of different serious faults and imperfect debugging phenomena, making DDID-CBSRGM perform the better performance than the others. Furthermore, as a practical matter, software engineer can select and set appropriate parameters to conduct experiment based on historical data, and obtain quantitatively insightful information tightly related to software testing and reliability evaluation, and determine the best decision reference on software development and testing resources allocation.

4 Conclusions

The major contribution of this paper is that we establish multi-queue multichannel and finite server queuing model with limited debuggers, and explicitly elaborates the impacts of debugging delay and imperfect debugging on CBS integration testing. The practical numerical example justifies that the proposed DDID-CBSRGM illustrates more accurately the integration testing process, explores a more efficient approach for research on CBS reliability growth model and the results reveal that the proposed model has better performance as compared to the other models. In particular, CBS integration testing is very complex, stochastic process involving various factors and sub processes, so incorporating more real situations into CBS reliability model is a main research direction. The research trends mainly encompass fault tolerant component reliability configuration, test coverage, incorporating testing effort (TE), the differences in transition probability and operational profile, and changeable structure into CBS reliability modeling and analysis, etc.

References

[1] Gokhale S S, Trivedi K S. Analytical models for architecture-based software reliability prediction: a unification framework. *IEEE Trans on Reliability*, 2006, 55(4): 578-590

[2] Palviainen M, Evesti A, Ovaska E. The reliability estimation, prediction and measuring of component-based software. *The Journal of Systems and Software*, 2011, 84: 1054-1070

[3] Lo J H, Huang C Y, Chen I Y, et al. Reliability assessment and sensitivity analysis of software reliability growth modeling based on software module structure. *The Journal of Systems and Software*, 2005, 76: 3-13

[4] Hsu C J, Huang C Y. An adaptive reliability analysis using path testing for complex component-based software systems. *IEEE Trans on Reliability* 2011, 60(1): 158-170

[5] Yuan F Q, Kumar U. A general imperfect repair model considering time-dependent repair effectiveness. *IEEE Trans on Reliability*, 2012, 61(1): 95-100

[6] Wu Y P, Hu Q P, Xie M, et. al. Modeling and analysis of software fault detection and correction process by considering time dependency. *IEEE Trans on Reliability*, 2007, 56(4): 629-642

[7] Huang C Y, Lin C T. Software reliability analysis by considering fault dependency and debugging time lag. *IEEE Trans on reliability*, 2006, 55(3): 436-450

[8] Kapur P K, Anand S, Inoue S, et al. A unified approach for developing software reliability growth model using infinite server queuing model. *International Journal of Reliability, Quality and Safety Engineering*, 2010, 17(5): 401-424

[9] Huang C Y, Huang W C. Software reliability analysis and measurement using finite and infinite server queueing models. *IEEE Trans on Reliability*, 2008, 57(1): 192-203

[10] Huang C Y, Hung T Y. Software reliability analysis and assessment using queueing models with multiple change-points. *Computers and Mathematics with Application*, 2010, 60: 2015-2030

[11] Lin C T. Analyzing the effect of imperfect debugging on software fault detection and correction processes via a simulation framework. *Mathematical and Computer Modelling*, 2011, 54: 3046-3064

[12] Hou C Y, Cui G, Liu H W, et al. A hybrid queueing model with imperfect debugging for component software reliability analysis. *Intelligent Automation and Soft Computing*, 2011, 17(5): 559-570

[13] Goel A L, Okumoto K. Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Trans on Reliability*, 1979, R-28(3): 206-211

[14] Xie M, Yang B. A study of the effect of imperfect debugging on software development cost. *IEEE Trans on Software Engineering*, 2003, 29(5): 471-473

[15] Huang W C, Huang C Y, Sue C C. Software reliability prediction and assessment using both finite and infinite server queueing approaches. In: Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing, 2006: 194-201

[16] Gokhale S S, Lyu M R. A simulation approach to structure-based software reliability analysis. *IEEE Trans on Software Engineering*, 2005, 31(8): 643-656

[17] Cheung R C. A user-oriented software reliability model. *IEEE Trans on Software Engineering*, 1980, SE-6(2): 118-125

[18] Pham H, Lars N, Zhang X M. A general imperfect-software-debugging model with s-shaped fault-detection rate. *IEEE Trans on Reliability*, 1999, 48(2): 169-175

[19] Ohba M, Chou X. Does imperfect debugging affect software reliability growth? In: Proceedings of the 11th International Conference on Software Engineering, Pittsburgh, USA, 1989: 237-244

Zhang Ce, born in 1978. He received Bachelor and Master degrees of computer science and technology from Harbin Institute of Technology (HIT) and Northeast University (NEU), China in 2002 and 2005, respectively. He has been a Ph. D. candidate of HIT major in computer system structure since 2010. His research interests include software reliability modeling and Trusted Computing (TC).