A Component-Based Software Reliability Growth Model Considering Differences in Components

Qi-Shun Xie, Jun Bai, Ce Zhang, Fan-Chao Meng
School of Computer
Harbin Institute of Technology at Weihai
Weihai, China
e-mail: xqswinner@163.com, baijun69@sina.com,
zhangce@hitwh.edu.cn, mengfanchao74@163.com

Gang Cui, Hong-Wei Liu
School of Computer
Harbin Institute of Technology
Harbin, China
e-mail: cg@ftcl.hit.edu.cn, lhw@ftcl.hit.edu.cn

***Abstract*—In the light of the problems in reliability modeling and analyzing for CBS (Component-Based Software), occurring time-invariant, less attention to debugging process, absence of considering differences in the components, a CBS SRGM (Software Reliability Growth Model) is presented. In terms of characters of different components, components are divided into critical and non-critical component sets. An imperfect debugging model considering fault correction probability and introducing new fault is used to describe critical component, and G-O model is applied to non-critical component. The formulation of the cumulative number of faults detected in CBS is derived. The proposed CBS SRGM can appropriately describe real testing process. Numerical examples verify the effectiveness of the proposed model and it outperforms other models.**

Keywords-software reliability; Component-Based Software; software testing; SRGM

I. INTRODUCTION

Software reliability is the key attribute in software life cycle, especially in development and operation stages. Generally speaking, large and complex software are made of multiple modules or components. Many efforts have been devoted to CBS(Component-Based Software) reliability recently [1-6]. From the view of architecture, reliability analysis of CBS integration testing includes four main types: state-based model [3], path-based model [7], additive model [8] and simulation method [9]. CBS integration testing process can be divided into unit testing, integration testing and system testing.

In reliability prediction for architecture-based software, S. S. Gokhale [1] first proposed a unified framework which formulated reliability analysis from three aspects: application architecture model, failure model of component and solving methods of reliability prediction, and conducted reliability prediction using analytical method based on state-based model. Related research can also see literatures [10]. The dynamic analysis has been carried out for CBS reliability in literature [11] using three standard program structure including sequence, loop and branch to conduct execution path reliability calculation and estimation. In addition, literature [12] proposed CBS reliability model and algorithm based on the specified scenario could identify the critical component and analyzed the impact of single component on system reliability. Obviously, CBS reliability depends on the reliability of every component and application structure. Testing begins with component level, and different component should employ diverse reliability enhancement technology in terms of complexity, function and importance. In conclusion accurately describing reliability growth of each component with testing time is the key to solving CBS reliability growth model.

However, existing researches still have certain insufficient in the following aspects. (1) difficulty in describing the growth of reliability of CBS, occurring so-call "time-invariant", (2) ignoring some real problems, for example, imperfect debugging, (3) Absence of considering the differences in the components.

A CBS SRGM incorporating the differences in the components is presented in this paper. Components are classified into critical and non-critical components set in term of characteristics, and are modeled by different SRGMs. Based on NHPP, the special imperfect debugging and G-O model are applied to different components. As a result, the solving process of the cumulative number of detected faults in CBS is illustrated.

II. IMPERFECT DEBUGGING

There are multiple situations in imperfect debugging [13-16]. Faults may not be removed completely and may be introduced into software after debugging by debuggers. To describe real software testing in detail, component C_i is modeled based on G-O model [17] and considering imperfect debugging. The analyses in model proposed are based on the following assumptions [13-15]:

(1) Let $N(t)$ denote the cumulative number of failure of C_i in time interval $(0,t)$ and $N(t)$ can be considered a NHPP (Non-Homogeneous Poisson Process) with mean value function $m(t)$, that is

$$\Pr(N(t)=k)=\frac{(m(t))^k}{k!}e^{-m(t)},\ k=0,1,2... \quad (1)$$

where $m(t)=E[N(t)]$.

(2) Fault detection rate in t is proportional to the faults remaining in software and the proportion is b.

(3) Faults may be removed incompletely and the successful rate is p.

(4) The new faults may be introduced during debugging and introduction probability β ($p >> \beta$) is proportional to the number of the faults detected in t.

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

$$\begin{cases} \dfrac{dm(t)}{dt} = b\left[a(t) - pm(t)\right] \\ \dfrac{da(t)}{dt} = \beta \dfrac{dm(t)}{dt} \end{cases} . \quad (2)$$

where $a(t)$ denotes the total number of faults. Solving the above differential equations with the boundary condition of $m(0) = 0, a(0) = a$ yields

$$m(t) = \frac{a}{p-\beta}\left(1 - e^{-(p-\beta)bt}\right) . \quad (3)$$

Thus, current failure rate can also be get:

$$\lambda(t) = m'(t) = ab\left(e^{-(p-\beta)bt}\right) . \quad (4)$$

Substitute (3) into (2), we can obtain $a(t)$:

$$a(t) = \frac{a}{p-\beta}\left(p - \beta e^{-(p-\beta)bt}\right) . \quad (5)$$

It is obvious that $m(t)$ and $a(t)$ are both increasing functions with time t, which indicates the increasing number of detected faults and the total faults with time due to considering imperfect debugging (incomplete correction and introducing the new faults).

III. A CBS RELIABILITY GROWTH MODEL CONSIDERING DIFFERENCES IN COMPONENTS

The critical component can be viewed as the complex component in CBS which has a more complicated interactions with the other components [6]. According to the testing information of component, the growth model of C_i can be modeled. Let CBS S consist of N components, $C=\{C_i, 1 \le i \le N\}$, and the critical components set is $CC=\{C_j | C_j$ is critical component$\}$,then $C=CC \cup NCC$. Hereon, the reliability of every component is described by the different SRGM. Obviously, due to the different complexity and importance of component in CBS, the reliability growth will not be the same. The different characteristics of components should be incorporated into the software reliability modeling. So, the detected faults of S during integration test can be classified into two groups in terms of the differences:

(1) The faults from critical component Ci are treated as complicated faults Fc;

(2) The faults from the other components are set as simple faults F_g.

Accordingly, we set two kinds of fault correcting queues:

(1) Complex faults correcting queues $Queue_{CC}$: F_c come into $Queue_{CC}$ in the light of FCFS (First come first serve).

(2) Simple faults correcting queues QueueNCC: Fg come into QueueNCC in the light of FCFS.

The failure mode of C_i is modeled based on failure intensity function $\lambda_i(t)$ with t. The fault detection process of F_c and F_g are analyzed in the following.

The fault detection process of critical component C_i is formulated as (2), then the cumulative number of faults detected in $[0,t]$ is:

$$m_i(t) = \frac{a_i}{p_i - \beta_i}\left(1 - e^{-(p_i-\beta_i)b_i t}\right) . \quad (6)$$

So, the cumulative number of faults in $Queue_{CC}$ by t can be obtained:

$$M_{CC}(t) = \sum_{C_i \in CC} m_i(t) . \quad (7)$$

For non-critical component, the classified G-O model[17]is used to describe the reliability growth, that is:

$$m_j(t) = a_j(1 - e^{-b_j t}) . \quad (8)$$

So, the cumulative number of faults in $Queue_{NCC}$ by t can be obtained:

$$M_{NCC}(t) = \sum_{C_j \in NCC} m_j(t) . \quad (9)$$

Obviously, in the interval of [0, t], the cumulative number of the detected faults is:

$$M(t) = \sum_{C_i \in CC} m_i(t) + \sum_{C_j \in NCC} m_j(t) . \quad (10)$$

Furthermore, given that the last failure occurred at T, the software reliability in $(T,T+x)$ is:

$$R_{CBS}(x \mid T) = e^{-\left\{\left[\sum_{i=1}^{n} m_i(T+x_i) - \sum_{i=1}^{n} m_i(T)\right]\right\}} . \quad (11)$$

$$m_i(t) = \begin{cases} \dfrac{a_i}{p_i - \beta_i}\left(1 - e^{-(p_i-\beta_i)b_i t}\right) & , C_i \in CC \\ a_i(1 - e^{-b_i t}) & , C_i \in NCC \end{cases} . \quad (12)$$

Due to considering the differences in components, the proposed model is referred to as CBSDC-RGM (Component-Based Software with Differences in Components-RGM).

IV. NUMERICAL EXAMPLE

Employing simulation method in conducting reliability analysis and generating failure data has been applied in a large quantity recently [9,15,18,19]. Due to the absence of real CBS failure data, the effectiveness of the proposed model is verified by a specified CBS reported in literature [20] which has been extensively used to illustrate CBS reliability model in recent years [9]. The structure of the CBS is presented in Fig. 1 and transition probabilities among the components are listed in Table 1.

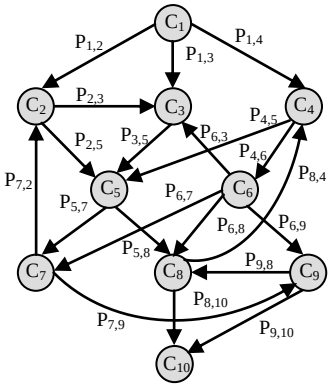


Figure 1. Architecture of an CBS

TABLE I. 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

In our previous work, based on the CBS simulation procedure reported in [9], we have developed the CBS integration testing simulation procedure considering imperfect debugging. On the basis, let a_i=40.14, b_i=0.0086 ($1 \le i \le N$). The level of imperfect debugging covers fault correction probability and introduction rate, without loss of generality, let p_i=0.95, $\beta_i = 0.15$ ($1 \le i \le N$). In this example, CBS failure data is obtained using the above simulation procedure. Literature [9] determined that C_1 and C_5 are critical components based on simulation. We have calculated component execution probability in steady state ω_i, namely,

$$\omega = [0.1297, 0.1177, 0.1181, 0.0543, 0.1751, 0.0326, 0.0798, 0.1133, 0.0497, 0.1297] .$$

We can see that ω_1 and ω_5 are the maximum in value. In addition, the same results can be obtained using $I_R \equiv \partial R / R_i = R(R_i = 1) - R(R_i = 0)$ [21]. So, we let the faults from C_1 and C_5 come into $Queue_{CC}$, and the other faults come into $Queue_{NCC}$.

A. Criteria For Comparisons

In order to verity the performance of the models, *MSE* (Mean Square Error), *R-square* and *Variance* are used to measure goodness of fitting.

$$MSE = \sum_{i=1}^{k} \frac{\left(m(t_i) - y_i\right)^2}{k} . \quad (13)$$

$$R - square = \frac{\sum_{i=1}^{k} \left[m(t_i) - \bar{y} \right]^2}{\sum_{i=1}^{k} \left[y_i - \bar{y} \right]^2}, \quad \bar{y} = \frac{1}{k} \sum_{i=1}^{k} y_i . \quad (14)$$

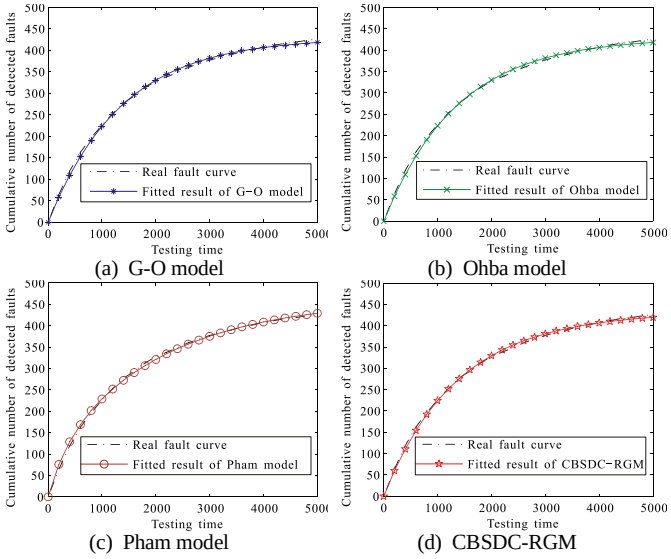
$$Variance = \sqrt{\frac{\sum_{i=1}^{k} \left(y_i - m(t_i) - Bias \right)^2}{k-1}} . \quad (15)$$

$$Bias = \frac{\sum_{i=1}^{k} \left[m(t_i) - y_i \right]}{k} . \quad (16)$$

where, y_i denotes the cumulative number of detected faults by t_i, $m(t_i)$ is the estimated value, and k is the sample size of the failure data.

B. Performance Validation

Fig. 2 plots the comparisons between the fault detection profile of real failure data and the data estimated by CBSDC-RGM. In the meanwhile, to verify the validity of the proposed model and the importance of differences in the components to reliability improvement, we compare Pham+ model [13] (Pham model is used to the critical component considering imperfect model) Ohba+ model [14] (the same to that of Pham+) , G-O+ model (the same to that of Pham+) with CBSDC-RGM.



(a) G-O model (b) Ohba model

(c) Pham model (d) CBSDC-RGM

Figure 2. Fault detection profile of the models

In order to further differentiate the fitting capacity of the models, criteria for comparisons are calculated and listed in Table 2.

As can be seen from Table 2, *MSE* and *Variance* of CBSDC-RGM are minimums in all the models, and *Variance* is also most close to 1. So, it indicates that CBSDC-RGM can be considered as the optimal one, and can describe the testing process well than the others. The reason for this is that the proposed model adequately considers the different characteristics of the components and imperfect debugging can describe the imperfect phenomena covering incomplete debugging and new faults introduction compared with the other models.

TABLE II. PERFORMANCE COMPARISON OF THE MODELS

Model	*MSE*	*R-square*	*Variance*
Pham+	18.8562	0.9716	4.5707
Ohba+	24.3110	1.0300	5.1725
G-O+	24.3110	1.0300	5.1725
CBSDC-RGM	16.8149	1.0243	4.3025

V. CONCLUSIONS AND THE FURTHER RESEARCH

A SRGM for CBS is presented considering the differences in components and imperfect debugging. The proposed model is closer to real software testing, and can help software engineer to analyze, measure and predict CBS reliability. In fact, CBS testing process is very complex, and there exists some influencing factors. Given the randomness and complexity in software testing, the further research is concentrated on incremental integration testing and CBS reliability process analysis considering fault-tolerant configuration, etc.

ACKNOWLEDGMENT

This work was supported in part by the National Nature Science Foundation of China (No. 60503015), the National High Technology Research and Development Program of China (No. 2008AA01A201), the Shandong province Science and Technology Program, China (No. 2011GGX10108, 2010GGX10104).

The National Nature Science Foundation of China (No. 60503015);

The Shandong province Science and Technology Program, China (No. 2011GGX10108, 2010GGX10104)

REFERENCES

[1] S. S. Gokhale and K. S. Trivedi, "Analytical Models for Architecture-Based Software Reliability Prediction: A Unification Framework," IEEE Trans on Reliability, vol. 55(4), 2006, pp. 578-590, doi: 10.1109/TR.2006.884587.

[2] S. S. Gokhale, W. E Wong, J.R. Hogan, and K. S. Trivedi, "An analytical approach to architecture-based software performance and reliability prediction," Performance Evaluation, vol. 58, 2004, pp. 391-412, doi: 10.1016/j.peva.2004.04.003.

[3] S. S. Gokhale, "Architecture-Based Software Reliability Analysis: Overview and Limitation," IEEE Trans on Dependable and Secure Computing, vol. 4(1), 2007, pp. 32-40, doi: 10.1109/TDSC.2007.4.

[4] W. L.Wang, T. L. Hemminger, and M. H. Tang, "A Moving Average Non-Homogeneous Poisson Process Reliability Growth Model to Account for Software with Repair and System Structures," IEEE Trans on Reliability, vol. 56(3), Sept. 2007, pp. 411-421, doi: 10.1109/TR.2007.903119.

[5] J. H. Lo, C. Y.Huang, I. Y. Chen, S. Y. Kuo, and M. R. Lyu, "Reliability assessment and sensitivity analysis of software reliability growth modeling based on software module structure," The Journal of Systems and Software, vol. 76, 2005, pp. 3-13, doi: 10.1016/j.jss.2004.06.025.

[6] W. L. Wang, T. L. Hemminger and M. H. Tang, "A Moving Average Non-Homogeneous Poisson Process Reliability Growth Model to Account for Software with Repair and System Structures," IEEE Trans. Reliability, vol. 56(3), Sept. 2007, pp. 411-421, doi: 10.1109/TR.2007.903119.

[7] S. S. Gokhale and K. S. Trivedi , "Dependency characterization in path-based approaches to architecture-based software reliability prediction," Proc. IEEE Workshop on Application-Specific Software Engineering Technology, Richardson, TX , USA, 1998, pp. 86-89, doi: 10.1109/ASSET.1998.688239.

[8] M. Xie and C. Wohlin, "An additive reliability model for the analysis of modular software failure data," Proc. Proceedings of the Sixth International Symposium on Software Reliability Engineering, Toulouse, France, 1995, pp. 188-194, doi: 10.1109/ISSRE.1995.497657.

[9] S. S. Gokhale and M. R. T. Lyu, "A simulation approach to structure-based software reliability analysis," IEEE Trans on Software Engineering, vol. 31(8), 2005, pp. 643-656, doi: 10.1109/TSE.2005.86.

[10] M. Palviainen, A. Evesti and E. Ovaska, "The reliability estimation, prediction and measuring of component-based software," The Journal of Systems and Software, vol. 84, 2011, pp. 1054-1070, doi: 10.1016/j.jss.2011.01.048.

[11] C. J. Hsu and C. Y. Huang, "An Adaptive Reliability Analysis Using Path Testing for Complex Component-Based Software Systems," IEEE Trans on Reliability, vol. 60(1), 2011, pp. 158-170, doi: 10.1109/TR.2011.2104490.

[12] S. Yacoub, B. Cukic and H. H. Ammar, "A Scenario-Based Reliability Analysis Approach for Component-Based Software," IEEE Trans on Reliability, vol. 53(4), 2004, pp. 465-480, doi: 10.1109/TR.2004.838034.

[13] H. Pham, L. Nordmann and Z. Zhang, "A general imperfect-software-debugging model with S-shaped fault-detection rate," IEEE Trans on Reliability, vol. 48(2), 1999, pp. 169-175, doi: 10.1109/24.784276.

[14] M. Ohba and X. M. Chou, "Does Imperfect Debugging Affect Software Reliability Growth?" Proc. Proceedings of the 11th

International Conference on Software Engineering, Pittsburgh, USA, 1989, pp. 237-244, doi: 10.1109/ICSE.1989.714425.

[15] C. T. Lin, "Analyzing the effect of imperfect debugging on software fault detection and correction processes via a simulation framework," Mathematical and Computer Modelling, vol. 54, 2011, pp. 3046-3064, doi: 10.1016/j.mcm.2011.07.033.

[16] Y. Fuqing, U. Kumar, "A General Imperfect Repair Model Considering Time-Dependent Repair Effectiveness," IEEE Trans on Reliability, vol. 61(1), 2012, pp. 95-100, doi: 10.1109/TR.2011.2182222.

[17] A. L. Goel, K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," IEEE Trans on Reliability, vol. R-28(3), 1979, pp. 206- 211, doi: 10.1109/TR.1979.5220566.

[18] C. T. Lin, C. Y.Huang, "Staffing Level and Cost Analyses for Software Debugging Activities Through Rate-Based Simulation Approaches," IEEE Trans on Reliability, vol. 58(4), 2009, pp. 711-724, doi: 10.1109/TR.2009.2019669.

[19] S. S. Gokhale, M. R. Lyu and K. S. Trivedi, "Incorporating Fault Debugging Activity Into Software Reliability Models: A simulation approach," IEEE Trans on Reliability, vol. 55(2), 2006, pp. 281-292, doi: 10.1109/TR.2006.874911.

[20] R.C. Cheung, "A user-oriented software reliability model," IEEE Trans on Software Engineering, vol. SE-6(2), 1980, pp. 118-125, doi: 10.1109/TSE.1980.234477.

[21] F. Belli, P. Jedrzejowicz, "Fault-tolerant programs and their reliability," IEEE Trans on Reliability, vol. 39(2), 1990, pp. 1844-192, doi: 10.1109/24.55880.