ATLANTIS PRESS
Advances in Computer Science Research (ACSR), volume 73
7th International Conference on Education, Management, Information and Computer Science (ICEMC 2017)

A Test Suite Generation Method for Component Interaction Testing

Liangming Li
Zibo Vocational Institute, Zibo, China
lilm@zbvc.edu.cn

Keywords: Interaction testing; Test case; Component composition; Unit testing

Abstract. Composition technology is the key factor of component based software development. Assembling component models to derive interaction test cases is a common way of component interaction testing; however, one of the consequences is the difficulty to test different component models sufficiently and the other is state explosion problem. We propose a method to generate interaction testing suite based on the composition of unit test case. The proposed method is a generally-used way for interaction testing and could meet the demand of appointed test criteria.

Introduction

Component-Based Software Development (CBSD) is the process of building soft-ware system by assembling existing components. Although unit testing can validate the correctness of local module [1], interaction errors may be still existed during component composition in practice [2], so component interaction should test thoroughly and completely during integration [3].

The current researchers usually perform compositional testing by means of model combination, such as Beydeda and Gruhn [4] that use the component state machine (CSM) to describe the behavior of individual component, and then construct the component-based software flow graph (CBSFG) for generation of test cases according to the CSMS; Wu et al. [5] first utilizes the component interaction graph (CIG) to describe the interactions and the dependence among components, and then generate the test case based on the CIG.

However, this research focus on the analysis of the integrate system function and locating the interactions of model combination. The consequences are the problems of state space explosion when the models contain considerable states and the high requirement for tester to understand different models.

Concepts and Terminology

Some concepts to be used are introduced in this section and the definition is simplified to facilitate the study.

Definition 1 a component is a 3-tuple $M = \langle IP_M, IR_M, P_M \rangle$

IP_M is the interface set of M to provide services;

IR_M is the interface set of M to require services;

P_M is the protocol of M to describe the behavior.

Definition 2 The test suite of component M (T_M) is the set of test cases that satisfy a user's testing requirements, which is provided by the component developer or generated by the component user according to the interface instruction and protocol. We use $t = ((i_1,...,i_m),(o_1,...,o_n))$ to describe a test case in which i is input information and o is output information.

Definition 3 Two component P and Q are composable if:

$IP_P \bigcap IP_Q = \varnothing$, $IR_P \bigcap IR_Q = \varnothing$, $IP_P \bigcap IR_Q \neq \varnothing$ or $IP_Q \bigcap IR_P \neq \varnothing$. We let $Shared(P,Q) = (IP_P \bigcap IR_Q) \bigcup (IP_Q \bigcap IR_P)$ and $P \bullet Q$ is the symbol of two component composition.

The deadlock and illegal state are assumed will not occur when component composition which mean the assembly ability has validated at the design stage. The following definition of test criteria is mainly from reference [5].

Subdomain is defined as containing all the inputs that can cause a particular code to be executed in the subdomain policy based on the program structure. There is a program P contain the component M and a test adequacy criterion C based on subdomain, $\delta D_C(D_P)$ is the induced multi-set of P according C, $\delta D_C(D_M)$ is the induced multi-set of M according C, $M-traverse(D_P)$ is the input set of P which result in M is called, $M-bypass(D_P)$ is the input set of P which is irrelevant to M.

Definition 4 Test suite T_P is $C-adequate$ for P if T_P contain each test sample of $\delta D_C(D_P)$; T_M is $C-adequate$ for M if T_M contain each test sample of $\delta D_C(D_M)$.

The Test Suite Generation Method for Component Interaction. The universal model from component composition is introduced, then the correctness and effectiveness of the proposed method are theoretically analyzed, after that the implementation process of the method is illustrated.

The General Model Of Component Composition. System development is replaced by the idea of assembling existing components in the process of CBSD. The general model of component composition is shown in Fig. 1 and the composition of several components could be performed by pair-wise composition

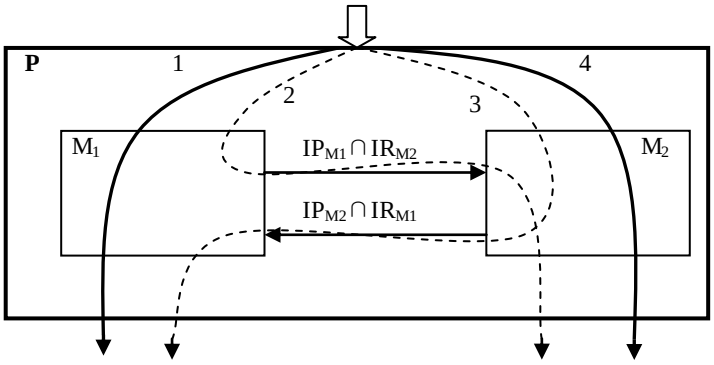

Figure 1. The model of component composition

The program P is constructed by composition of component M_1 and M_2 in Fig. 1 in which the line which is tagged with number 1 and 2 represent the input set of $M_1-traverse(D_P)$ and the dash 2 represent the related input subdomain of $IP_{M1} \cap IR_{M2}$; Similarly the line 3 and 4 represent the input set of $M_2-traverse(D_P)$ and dash 3 represent the related input subdomain of $IP_{M2} \cap IR_{M1}$.

From Fig. 1, One part of the system functions is implemented by M_1 and M_2 individually and the other functions are cooperative performed by the interaction of M_1 and M_2. If M_1 and M_2 had passed the unit testing and gotten the test suit T_{M1} and T_{M2} then line 1 and 4 in figure 1 can be tested by mapping the related part of T_{M1} and T_{M2} to input of P. The dash line 2 represent the input of M_2 that is provided by M_1, which can be tested by composition of the related test cases in $IP_{M1} \cap IR_{M2}$ from T_{M1} and T_{M2}, similarly the line 3 can be tested by combination of related test cases in $IP_{M2} \cap IR_{M1}$.

The Method of Test Suite Generation for interaction Testing. The test case set T_M can be divided into relevant and irrelevant part for interface I_M of component M.

Definition 5 $T_M-relevant(I_M)$: the relevant part of T_M with component I_M

Definition 6 $T_M-irrelevant(I_M)$: $T_M - (T_M-relevant(I_M))$

Definition 7 let $t_1=((i_1,...,i_m),(o_1,...,o_n))$, $t_2=((i'_1,...,i'_{m'}),(o'_1,...,o'_{n'}))$ the combination of t_1 and t_2 is $t_1 \bullet t_2$

$t_1 \bullet t_2=(((i_1,...,i_m) \cup (i'_1,...,i'_m)-X),((o_1,...,o_n) \cup (o'_1,...,o'_n)-X))$, $\forall x \in X, x \in (o_1,...,o_n) \cap (i'_1,...,i'_{m'})$, in which $X \neq \varnothing$ and $t_1 \bullet t_2$ represent the input of t_2 is the output of t_1.

After the analysis, $T_{M1}-relevant(IP_{M1}\cap IR_{M2})$ is the test suite of M_1 that could call the function of M_2 and $T_{M2}-relevant(IP_{M1}\cap IR_{M2})$ is the test suite of M_2 that could call the function of M_1. The combination of $T_{M1}-relevant(IP_{M1}\cap IR_{M2})$ and $T_{M2}-relevant(IP_{M1}\cap IR_{M2})$ could test the activity of $IP_{M2}\cap IR_{M1}$. Therefor we get the interaction testing method by combination of test suite, which is IT-on-CUTC (Interaction Testing based on Combination of Unit Test Case, IT-on-CUTC), $T_1 \# T_2$ is used to describe the combination of test suite T_1 and T_2.

The steps of IT-on-CUTC are as follows:

Premise: component M_1 and M_2, test criterion C and test suite T_{M1} and T_{M2}

Step1: according IP_{M1}, IR_{M1} and IP_{M2}, IR_{M2} to determine if M_1 and M_2 is composable and the operation will stop if M_1 and M_2 is not composable;

Step2: compute the $Shared(M_1, M_2)$, namely compute the $IP_{M1}\cap IR_{M2}$ 与 $IP_{M2}\cap IR_{M1}$;

Step3: determine the $T_{M1}-relevant(IP_{M1}\cap IR_{M2})$ and $T_{M1}-relevant(IP_{M2}\cap IR_{M1})$ from T_{M1}, determine the $T_{M2}-relevant(IP_{M1}\cap IR_{M2})$ and $T_{M2}-relevant(IP_{M2}\cap IR_{M1})$ from T_{M2};

Step4: compute $T_{M1}-relevant(IP_{M1}\cap IR_{M2})$ # $T_{M2}-relevant(IP_{M1}\cap IR_{M2})$ and $T_{M2}-relevant(IP_{M2}\cap IR_{M1})$ # $T_{M1}-relevant(IP_{M2}\cap IR_{M1})$ then get the union $T_{M1\bullet M2}$;

Step5: analyze $T_{M1\bullet M2}$ and remove the meaningless test cases; run $T_{M1\bullet M2}$ and finish the interaction testing of M_1 and M_2.

Theorem 1. The test suit $T_{M1\bullet M2}$ generated from IT-on-CUTC is $C-adequate$ for component interaction

Proof: the composition of M_1 and M_2 do not generate new interface and no new partition is made for the input domain. Then $|IP_{M1\bullet M2}|+|IR_{M1\bullet M2}|< |IP_{M1}|+|IR_{M1}| + |IP_{M2}|+|IR_{M2}|$ and $\delta D_C(D_{M1\bullet M2}) \subseteq (\delta D_C(D_{M1})\cup\delta D_C(D_{M2}))$. So the testing of $M_1 \bullet M_2$ could implement in the original input domain according to the testing criteria C. On the other hand T_{M1} and T_{M2} is $C-adequate$ for M_1 and M_2 and $T_{M1\bullet M2}$ is generated by the related test cases to $Shared(M_1, M_2)$ from T_{M1} and T_{M2} which could cover all the interaction path of M_1 and M_2 based on C. Therefor the test suit $T_{M1\bullet M2}$ generated from IT-on-CUTC is $C-adequate$ for interaction of $M_1 \bullet M_2$.

Literature References

Weyuker E J [6] define 8 axioms to formalize the basic properties of the adequacy of test cases and the anti- decomposition and the anti-combination axiom illustrate the test adequacy between the overall procedure and the components. Frankl F G, and Weyuker E J [7] further analyze the different test criteria based on subdomain division. Rosenblum D S [5] define the testing criteria C-adequate-for-P of unit testing and C-adequate-for-M of integration testing. LIU Ling, MIAO Huai-kou [8] define an axioms system to describe the relations among the logic coverage criteria. Some interaction testing are based on component model, Hierons R M [9] present the method using CIS(Constrained identification sequence) as state recognition sequence to test the interaction among FSM. The approach of integration testing is proposed in [10] based on UML, first, According to the model, the interaction activities of different components are merged and then the test cases are derived from the composed model. Gotzhein and Khendek [11] propose a compositional test method, which test the concurrent component composition without constructing the global state machines.

From the above survey, we find that the studies usually concern how to establish testing criteria and discuss the relation between them, which do not concentrate on describing the method of generating interaction test suite. In this paper we study the method of obtaining interaction test suite from unit test cases and prove the method could meet the test criteria.

Summary

Interaction testing is the important part of softer testing and which is the essential way to validate component composition. The study on component interaction testing is usually based on model combination, but if the components are described by different models, the completion of interaction testing will be very difficult and inefficient when a lot works of model transformation are needed. We propose a novel interaction testing approach, called IT-on-CUTC, which derives test cases from interaction path. The method performs interaction testing with low computational complexity and more generality; therefore the development cost is reduced.

In the future, we will further optimize the IT-on-CUTC algorithm for generating interaction path more efficiently, and conduct more experiments at different scale to verify its validity and practicability.

References

[1] E.J. Weyuker, Testing Component-Based Software: A Cautionary Tale: IEEE Software, 1998, 15(5): 54-59.
[2] S. Eldh, S. Punnekkat, H. Hansson, et al, Component Testing Is Not Enough -A Study of Software Faults: Proc. 19th IFIP International Conference on Testing of Communicating Systems TESTCOM/FATES, Springer LNCS-4581, Tallinn, Estonia 2007, 74–89.
[3] S. Beydeda S, V. Gruhn. An integrated testing technique for component-based software: Proc. of the ACS/IEEE Int'l Conf on Computer Systems and Applications. Beirut, IEEE Computer Society Press, 2001. 328-334.
[4] Y. Wu, D. Pan, M. H. Chen. Techniques for testing component-based software: Proc. of the 7th IEEE Int'l Conf on Engineering of Complex Computer Systems(ICECCS'01). Skovde, IEEE Computer Society Press, 2001. 222-232.
[5] D. S. Rosenblum. Adequate Testing of Component-Based Software: Technical Report, UCI-ICS-97-34, University of California, Irvine. August, 1997.
[6] E. J. Weyuker. Axiomatizing software test data adequacy: IEEE Tran. on Software Engineering, 1986, 12(12):1128-1138.
[7] F. G. Frankl, E. J. Weyuker. A formal analysis of the fault- detecting ability of testing methods: IEEE Transactions on Software Engineering, 1993, 19(3): 202- 213.
[8] L. Ling, L. Huai-kou. Axiomatic Assessment of Logic Coverage Software Testing Criteria: Journal of Software, 2004 ,15 (9) :1301-1310.
[9] R. M. Hierons. Checking states and transitions of a set of communicating finite state machines: Microprocessors and Microsystems, 2001, 24(9): 443-452.
[10] J. Hartmann, C. Imoberdorf, M. Meisinger. UML-based integration testing: Proc. of 2000 ACM SIGSOFT international symposium on Software Testing and Analysis, Portland, Oregon, United States, 2000. 60-70.
[11] R. Gotzhein, F. Khendek. Compositional Testing of Communication Systems: Proc. of IFIP Testcom 2006, Berlin: Springer LNCS 3964, 2006. 227–244.