A study on the methods of software testing based on the design models

Tangtang Xie
College of Computer and Information Science
Southwest China University
Chongqing, China
xtt@swu.edu.cn

Jun Li
Southwest China University Press
Southwest China University
Chongqing, China

Hanrong Chen
College of Computer and Information Science
Southwest China University
Chongqing, China
chrong@swu.edu.cn

Hailing Xiong
Southwest China University Press
Southwest China University
Chongqing, China

***Abstract*—With the expanding range of computer applications, computer applications is used more and more in key areas, so it requires higher quality and reliability for software. As the most important techniques for software quality and reliability, software testing becomes more and more important in software development. Though software engineering technology development continuously put forward new requirements for software test technique, software test model and test case generation have always been the core of the software testing, and it is important to choose the better test models and test methods to improve the efficiency of test cases. Considering that most of software errors can be attributed to the difference between design and implementation, this paper presents a software testing method based on the design model by analyzing the design model and the implement model extracted from design process and converting to formal test models, and finally compare the design model and the implement model to find out the difference between requirement and implement.**

***Index Terms*—design model, test model, software testing, test case**

I. INTRODUCTION

With the development and popularization of computer technology, the software has been widely used in all the spheres of society, and becomes the core technology support for the enterprises, industrial control, and communications industry, and even plays an important role in the Military departments, which makes people increase demand for the higher and higher reliability of computer systems. How to improve software quality is one of the key issues to address for software engineering. Software testing is the most fundamental and important tool to ensure accuracy of software and improve reliability of software, and mainstream technology used in the industrial.

Software test techniques and methods are affected by design models, development process, programming languages and other software development technologies and methodologies; therefore there are not common test methods that are applicable to all the software and test requirements. In the past 20 years, the software engineering research has achieved many important progresses that are many new applications, development models and development methodologies, so the research on software testing will also be continuously changing [1].

II. THE SIGNIFICANCE OF THE STUDY

Most of the software errors can be attributed to the difference between the design and the implementation in the development. Traditional test methods have shortcomings especially in the test efficiency, thus people are trying to find a way to use existing resources for automatic software test methods. This paper proposes to identify the difference between the design and the implement by comparing the design models and the implementation models, and finally aims to develop the software test tool for automatic testing.

III. THE IDEAS AND THE METHODS OF SOFTWARE TESTING BASED ON THE DESIGN MODEL

The design model in this paper is referred to the UML design model that is widely used in industry now. This paper uses the formal method to analyze UML design models to get the formal test models, at the same time transforms the implement models from the code, then converts the implement models to the formal test models, and eventually compares the two formal models to identify the difference to generate test cases. We can design the automatic test software by this method to improve the test efficiency.

A. The formal Analysis of the design models

The software models abstract and describe the software in the software development in order to help people to understand, refine and develop the software. In the software development process, there are different software models in the various development phases. Along with the development phase going on, software models become more detailed until the code is generated. Software testing can also use the design models

which are generated in the design process, but in reality different designers can get different design models. So the design models should be formally described in order to standardization and reasoning comparison, which is a basis for all subsequent work. The example for the formal description of collaboration diagrams is as follows:

The collaboration diagram emphasizes the objects organization structure between the sending messages of objects and the receiving messages of objects. The collaboration diagram can be used to model a specific scene of the use case according to the role and their relationship of the interaction. It describes the specific behavior of the objects in the static structure and the dynamic interaction that is a set of messages.

Definition: the UML collaboration diagram CD can be expressed as a pair: $CD = (OS,\ M)$.

$OS = \{OS_1, OS_2, \cdots, OS_m\}$ is the nonempty finite of object set in CD.

$M = \{m_1, m_2, m_3, \cdots, m_n\}$ is the finite of messages set described in CD. Each a message is defined as

$M = (guard, sequence\_expression, m\_name, parameter\_list, return\_value, receive, send, type)$, m_name is the message name, $guard$ is the guard condition of the sending message, $sequence\_expression$ is the order of entry list, $parameter\_list$ and $return\_value$, are respectively message of the parameters and return values of the message, $receive$ is the sending object of the message, $send$ is the receiving object of the message. The type of M is defined as: $type \in \{syncall, asyncall, send, return\}$, $syncall$ represents the synchronous call of the method, $asyncall$ represents the asynchronous call of the method, $send$ represents the sending of signal ; $return$ represents the return of synchronous call.

B. The extraction and the analysis of the implementation models

The implement model can't be directly gotten, which need to reverse the code to generate. Although there are many researches on generation the implementation model from reverse engineering that have not reached the industrial production requirements [2-4]. The implementation models converted from the code should be formally described in order to standardize. This formal description methods of the implement models refer to formal description method methods of the design models.

C. The choice of the test models

The test model is a very important to the testing method based on design models. Design models and implementation models are compared by test models. When the different design models are used as the test models, the construction method of design models, the extraction method of implementation models, the comparison method of test models, and the test cases generation method will be different. And the appropriate test model language will benefit the entire test process to improve the efficiency and reliability. The test model described by the natural language can not be reasoned and compared; therefore the description language of the test model must be the formal language.

The choice of the abstraction level of the test model is essential. If the abstraction level of the test model is too high, the information source of code will be reduced a lot in the extraction process of the implementation model so that it is likely to lose the difference of the implementation model and the design model. On the other hand, if the level abstraction of the test model is too low, the extraction process of the implementation model is relatively simple to achieve and will retain more information. However, testers need to construct the test model with the formal language by themselves from the existing information. Even if testers had to go through more complex and larger amount of work to complete the construction process, they should introduce the error which is the understanding of them and nothing to do with the code in the construction process. The implementation model that is changed in the construction process does not truly reflect the code, so the comparison result of the design model and the implementation model is not meaningful. Therefore, the choice of the test model should be based on detailed analysis of the testers in the actual situation.

D. Generation test cases by the model comparison

The model comparison must be described and deduced based on unambiguous language, so it must firstly formalize the design models and the implementation models according to the test models, and then will compare the design models and the implementation models to get the difference between the two models.

Definition: Two equivalent class models $M = \{< P_1, f_1 >, < P_2, f_2 >, ..., < P_n, f_n >\}$ and $M' = \{< P_1', f_1' >, < P_2', f_2' >, ..., < P_n', f_n' >\}$ .If M and M' equivalent, denoted $M \equiv M'$,that is $\forall i \in P\left(i \in P_i \wedge i \in P_j' \wedge f_i(i) = f_j'(i)\right)$. Therefore the conditions that M and M' are not equivalent can be defined as $\exists f_i \left(\wedge_j f_i' f_i \neq f_j'\right) \vee \exists f_j' \left(\forall f_i f_i \neq f_j'\right) \vee \exists f_i f_j \left(f_i \neq f_j \wedge P_i \cap P_j' \neq \varnothing\right)$.

Specific algorithm is as follows:

Input: the design model M, the implementation model M'

Output: the different set of M and M' model

{ Int flag

For each$< f_1, P_1 > \in M$

{flag=0

For (each$< f', P' > \in M'$)

If ($f_1 = f'$)

{calculate $P_N = \left(P_1 - P'\right) \cap \left(P' - P_1\right)$

If P_N is not empty

{ flag=1

add P_N into N

}

}

Else

{Calculate $P_N = P_1 \cap P'$

If P_N is not empty

{ flag=1

add P_N into N

}

}

If (flag=0)

{add P_N into N

Remove $< f_1, P_1 >$ from M

Remove $< f', P' >$ from M'

Loop

Loop

If M' is not empty

{ for (each $< f', P' > \in M'$)

add P' into N

}

}

}

IV. SUMMARY

This paper presents the guidelines and the specific implementation method on the software testing based on the design model. There are still some key problems for further research, such as the specific selection rules of the test model, the transformation of formal models and so on. The following picture shows that the test cases are automatically extracted from the activity diagram of beverage vending machines by the Modeltest tools that designed by the author.

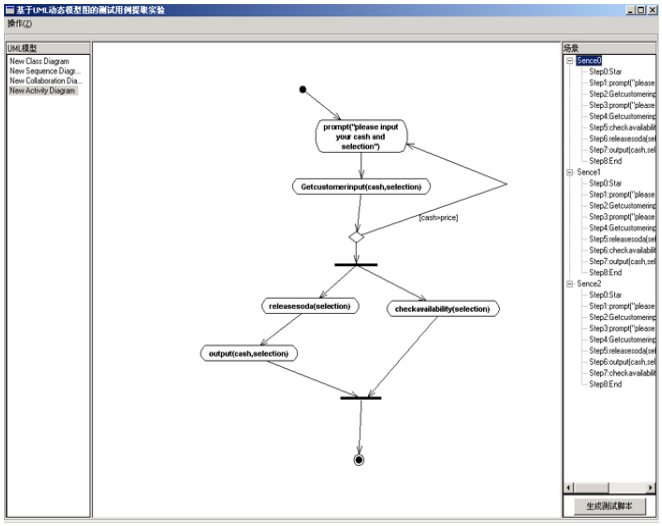

Figure 1 test scenarios from the activity diagram of beverage vending machines

This paper describes the current research status of software testing based on design model, research directions and the work of author to give some references in this field.

ACKNOWLEDGMENT

This work has been supported by the Fundamental Research Funds for the Central Universities (project No. XDJK2010C031,project No. XDJK2010C032).) ,Southwestern University Education Reform Project (project No. 2010JY027) and the Doctoral Foundation Project of the Southwest China University ((project No. SWUB2008073).

REFERENCES

[1] A.Polini A.Bertolino.A Framework for Component Deployment Testing.In:A.J.a.F.Titsworth Ed.Proc.of ICSE'03.Portland,Oregon USA.Los Alamitos,CA:IEEE Computer Society Press,Press,2003,221~231.

[2] Fu Jianjing. Code Protection Based on Feature of JAVAC and JVM. Computer Engineering.2010(1).

[3] Lawrence P, Sergey B, Rajeev M, et al. The PageRank Citation Ranking: Bringing Order to the Web[Z]. Stanford, CA, USA:Stanford Digital Libraries, 1998.

[4] Liao Husheng. Data Specialization and Program Specialization Based on Control Flow Graph. Chinese Journal of Computers.2001(9).

[5] T.Garwin,M.P.Crozat,B.L.Merrell,et al.,"Metrics-Based Test and Evaluation of Group Detection Software for Counter-Terrorism,"in 2006 IEEE Aerospace Conference,2006.

[6] P.Runeson.A Survey of Unit Testing Practices.IEEE Software,2006,23(4):22-29.

[7] Xie Xiaodong; Lu Yansheng; Mao Chengyin. Software Testing Method Based on Model Comparison. Journal of Southwest Jiaotong University.2008(2)

[8] Meudec C.Automatic Generation of Software TestCases From Formal Specifications [D]. Belfast,UK:The Queen's University of Belfast,1998.