2011 International Conference on Mechatronic Science, Electric Engineering and Computer
August 19-22, 2011, Jilin, China

An Application of Ontology to Test Case Reuse

Shaojie Guo
1. School of Computer Engineering and Science
Shanghai University
2. Shanghai Key Laboratory of Computer Software
Evaluating & Testing
Shanghai, China, 200072
e-mail: shaojie215@163.com

Weiqin Tong
School of Computer Engineering and Science
Shanghai University
Shanghai, China, 200072
e-mail: wqtong@shu.edu.cn

Juan Zhang
School of Computer Engineering and Science
Shanghai University
Shanghai, China, 200072
e-mail: zhangj@ssc.stn.sh.cn

Zongheng Liu
School of Computer Engineering and Science
Shanghai University
Shanghai, China, 200072
e-mail: liuzongheng@shu.edu.cn

***Abstract*—Test case is one of the most important part of software testing. Reusing available test cases is an effective means to reduce the cost and improve the efficiency of software testing. In order to efficiently reuse test cases, a unified and standard format to describe test cases is needed. Therefore, this paper presents a reusing method based on ontology. The test case ontology is created by Protégé Ontology Editor and described by OWL (Web Ontology Language). At last, an example to illustrate the application of this method is to be discussed.**

Keywords-test case; reuse; ontology; OWL;

I. INTRODUCTION

Software testing is a critical element of software quality assurance and the cost of testing accounts for a large part of the cost of software development and maintenance[1].In the field of software engineering , there is a well-known rule of thumb: in a typical programming project approximately 50 percent of the elapsed time and more than 50 percent of the total cost were expended in testing the program or system being developed[2].As an example, in Microsoft Inc 50% employees are test engineers and the programmer of Microsoft spend 50% of their time on the testing [3].

As is well-known in software engineering, software reuse has been thought as a key strategy for reducing development costs and improving quality[4]. With the wide application of object-oriented method and component-based development method in recent years , software reuse has been very popular. However, the reuse research of software test case is just beginning. The most important consideration in program testing is the design and creation of effective test cases. The excellent test cases will be stored in the library as a valuable and reusable resource for the future applications. In the test cases reuse process, it is a key issue that the effective test cases organization and management, especially the depiction and analysis of test cases .

Much research has been devoted to the reuse of the test case. Mayrhauser[5] proposed a new test data generation method, Domained Based Testing(DBT), Govind Kulkarni[6] discussed the reusability of test case for web application. Yongbo Wang[7] proposed a approach of test case generation based on ontology. Luxiao L[8] developed a test case library and discussed the model of testing case managing. To support effective test reuse, Z. L. Shao[9] proposed a software test design model based on the analysis of reusable test assets and their relationships. Most of the previous research focus on how to manage or generate test case, however, few study is chiefly concerned with how to describe test case.

To describe precisely and accurately test case, this paper points out an ontology-based method which as a basis for the sharing and reuse of knowledge has been widely used in information science.

II. BRIEF REVIEW OF ONTOLOGY

A. The definition of ontology

Research on ontology is becoming increasingly widespread in the computer science community[10]. The word "ontology" derives form philosophy in which it refers to the subject of existence. In the 1980s, it began to be used in the Artificial Intelligence community to create computational models that enable certain kinds of automated reasoning. In 1993, Tom Gruber proposed the most famous definition of ontology: a formal specification of a conceptualization[11]. This definition is the most referenced in the literature. In essence, an ontology describes the concepts in the domain and also the relationships that hold between those concepts . Most ontologies which share many structural similarities describe individuals (instances), classes (concepts), attributes, and relations. Ontology is formalized using a 5-tuple[12]:

<Classes, Relations, Functions, Axioms, Insurances >

- **Classes:** sets, collections, concepts, classes in programming, types of objects, or kinds of things.

978-1-61284-722-1/11/$26.00 ©2011 IEEE

- **Relations:** ways in which classes and individuals can be related to one another. They are formally defined as any subset of a product of *n* sets, that is: R:C1 x C2 x … x C*n*. The relations widely used include: part-of, kind-of, instance-of, and attribute-of.

- **Function terms:** complex structures formed from certain relations that can be used in place of an individual term in a statement. Formally, functions are defined as: F: C1 x C2 x ... x C*n*-1→C*n*.

- **Axioms:** model sentences that are always true.

- **Instances:** represent elements.

B. Principles and methodologies for building ontologies

In the absence of a standard ontology construction method, lots of principles, design criteria and phase in the ontology building process are presented by many development groups. The most influential among these principles is the 5 rules proposed by Gruber in 1995： *clarity and objective*, *coherence*, *extendibility*, *minimal encoding bias*, *minimal ontological commitment*[13].

Due to each development team usually follows its own principles and methodology in the ontology development process, there does not yet exist a common methodology that everybody agrees on. The most commonly construction methods include: TOVE methodology, Methontology method, Skeletal methodology, KACTUS engineering methodology, and so on. This paper adopts Skeletal Methodology[14] which proposes the comprehensive steps for developing ontologies :

(1) Identify the purpose and scope of the ontology being built;

(2) Build the ontology, inclusion of capturing knowledge, coding knowledge and integrating the knowledge;

(3) Evaluate the ontology

(4) Documentation

(5) Guidelines for each phase

C. Language and tools for ontologies modeling

Once the main components of ontologies have been represented, the ontology can be implemented in a various languages. Note that different ontology languages provide different facilities. In order to avoiding ambiguous understanding between description languages, it's need to use a common standard to express ontology. The most recent development in standard ontology languages is OWL(Web Ontology Language) from the World Wide Web Consortium(W3C). This paper uses the Protégé-OWL to build OWL ontologies which makes it possible to describe concepts but it also provides new facilities.

III. ONTOLOGY-BASED TEST CASE SPECIFICATION

As the core assets, a test case in software engineering is a set of conditions or variables under which a test engineer could determine whether an application or software system is working correctly or not. Based on the analysis of a large number of test cases, the properties of a test case consist of id, name, precondition, input, operation, except result, author, date of creation and so on. In order to clearly exhibit the process of reusing test cases, the properties above is simplified as a 6-tuple:{Id, Name, Precondition, Input, Operation, Expectation}. The 6-tuple is shown in the Table 1.

TABLE I. THE PROPERTIES OF A TEST CASE

Name of property	Type	Description
Id	int	Unique identifier of a test case
Name	string	The name of a test case
Precondition	string	The condition to meet before test activities
Input	sting	Input data
Operation	string	Describe the process and method of testing
Expectation	string	The expectation result

According to these concepts, a minimum test case ontology can be built, which only contains a class named "*test_case*". Of course, every ontology is slowly evolving, and software test case ontology is no exception. The ontology is described using OWL, which can be easily exchange and share between different test engineers. The core code of property of concept *test_case* is defined as follows.

```
<owl:Class rdf:ID="test_case"/>
  <owl:DatatypeProperty rdf:ID="ID">
    <rdfs:range rdf:resource="http://www.w3.
        org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#test_case"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Name">
    <rdfs:range rdf:resource="http://www.w3.
          org/2001/XMLSchema#string"/>
   <rdfs:domain rdf:resource="#test_case"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Precondition">
        <rdfs:range rdf:resource="http://www.w3.
            org/2001/XMLSchema#string"/>
        <rdfs:domain rdf:resource="#test_case"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Input">
       <rdfs:rangerdf:resource="http://www.w3.
           org/2001/XMLSchema#string"/>
       <rdfs:domain rdf:resource="#test_case"/>
  </owl:DatatypeProperty>
                        ……
```

IV. THE METHOD AND PROCESS OF TEST CASE REUSE

During software testing, the first step is that determine the test target, namely the set of test requirements. According to these requirements, query the satisfied test case from library and reuse. In order to reuse test cases effectively, we need to comply with certain methods and strategies, as following:

1. According to the function point of the software being tested, test engineer defines the type of test cases that is needed.

2. According to the defined test cases, the test cases to meet the requirements will be found in the test case library.

3. If can be found, the reusable test cases will be taken out from the library. Then, the program ends.

4. Otherwise, design the test cases using OWL, and verify its correctness. If correct, the new test case will be added to the library for future reuse. Then, the program ends.

The flow chart is shown in Figure1.

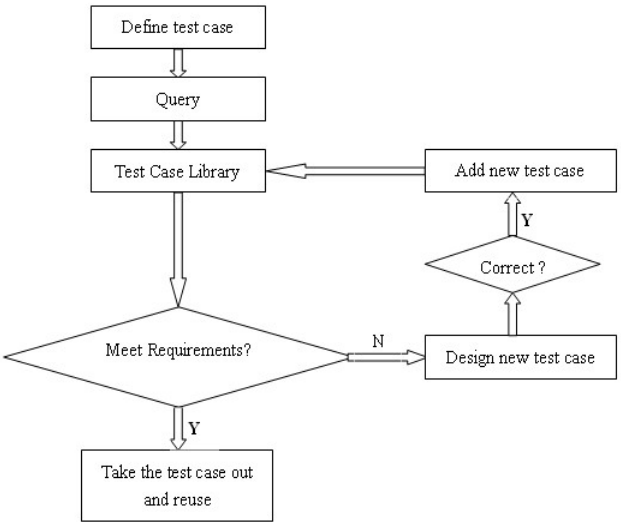

Figure 1. The flow chart of reusing test case

V. CASE STUDY

To more clearly explain the reusing process, a Simple Automated Teller Machine System(SATMS) is taken as an example in this section.

A. Analysis of SATMS

A simplified withdrawal or deposit transactions in the SATMS only consists of three steps: insert credit card, enter password, withdrawal or deposit. A typical withdrawal transaction is briefly described as following.

1. Costumers insert the credit card;

2. If the credit card is invalid, ATM will retire it. Otherwise, promote user to enter password;

3. Costumers enters the password. If the entered password is error , ATM will retire it. Otherwise, prompt user to choose the type of transaction;

4. Costumers choose the Withdrawal;

5. The ATM prompts costumers enter the withdrawal amount that must be an integer multiple of 50. Costumers enter the amount, and then click "OK";

7. The ATM checks whether the balance is sufficient. If inadequate, the ATM prompts costumers to re-enter the withdrawal amount; Otherwise, the ATM will spit out the cash;

8. After the withdrawal transaction, the ATM retire credit card if no other transaction is selected.

From the above description, we can see that the implementation of the withdrawal transaction is based on inserting the valid credit card and entering the correct password. So test engineer could reuse the both test cases of inserting credit card and entering password when designs a test case for the withdrawal function.

B. Design test cases

Based on the above analysis, some test cases for "inserting credit card" and "entering password" are designed as follows.

1.Test cases for inserting credit card:

Id: 001 **Name:** insert a valid card
Precondition: the SATMS can work
Input: a valid credit card
Operation: inserting the credit card
Exception: Please enter your password

Id: 002 **Name:** insert a invalid card
Precondition: the SATMS can work
Input: a invalid credit card
Operation: inserting the credit card
Exception: Error! Please exit!

2. Test cases for entering password:

Id: 003 **Name:** verify the error password
Precondition: inserted card is valid
Input: correct password
Operation: enter the password
Exception: Please choose the type of transaction

Id: 004 **Name:** verify the right password
Precondition: inserted card is valid
Input: error password
Operation: enter the password
Exception: Error! Please re-enter!

A complete withdrawal operation includes inserting credit card, entering password and entering amount. Given that the test cases in step 2、3 have been in test cases library. The test cases for first two step can reuse the test cases in table 2、3. And then design new test cases for "entering amount " shown as follows.

3.Test cases for entering amount:

Id: 005 **Name:** adequate balance
Precondition: costumers choose "withdrawal "
Input: the amount is integer multiple of 50
Operation: entering the amount
Exception: The SATMS spit out cash

Id: 006 **Name:** inadequate balance
Precondition: costumers choose "withdrawal "
Input: the amount is greater than balance
Operation: entering the amount
Exception: Error! Balance is inadequate! Please re-enter !

C. Describe test cases individual using by OWL

Based on the above designed test case, six test case individuals is created by Protégé. They can be described using by OWL as follows.

```
<?xml version="1.0"?>
<rdf:RDF
<test_case rdf:ID="testcase_individual_1">
 <ID rdf:datatype="http://www.w3.org/2001/
XMLSchema#int">001</ID>
  <Name xml:lang="en">insert a valid card</Name>
  <Input xml:lang="en">a valid credit card</Input>
 <Operation xml:lang="en">inserting the credit card
      </Operation>
<Precondition xml:lang="en">the SATMS can work
      </Precondition>
</test_case>
<test_case rdf:ID="testcase_individual_2">
   <ID rdf:datatype="http://www.w3.org/2001/
   XMLSchema#int">……</ID>
  <Name xml:lang="en">……</Name>
  <Input xml:lang="en">……</Input>
  <Operation xml:lang="en">……</Operation>
  <Precondition xml:lang="en">……</Precondition>
</test_case>
      ……
      ……
</rdf:RDF>
```

This simply example briefly describes the process of test case designing and reusing based on ontology. First, abstract the common properties of test cases to form a unified structure of test cases, and then build the test case ontology. According to the specific test environment and requirements, design test project and test cases. Every actual test case is an instance of the test case ontology. At last, the reusable test cases already exists in library are reused to for new test cases. Of course, in the actual testing process, the process is much more complex.

VI. SUMMARY

By analyzing a large number of software test cases, this paper proposes a ontology-based method of test case reuse and summarizes some common characteristics of the reusable test cases. After that, the process of building ontology and reusing test cases are discussed. Our work is focused on building a test case ontology which gives a specification of all concepts in test case, and reusing test case. And our method promotes the efficiency of reusing. Currently we are concentrating on the management and query of test case with the purpose of better efficiency of test case reuse.

VII. ACKNOWLEDGMENT

This work is supported by National Torch Foundation of China (Grant No.2009GH510068) and the Foundation of Shanghai Committee of Science and Technology (Grant No. 10DZ2291800).

REFERENCES

[1] B. Beizer. Software testing techniques. Van Nostrand Reinhold Company, 1990.

[2] Glenford J. Myers, The Art of Software Testing, 2nd ed., John Wiley and Sons, Hoboken, N.J., 2004

[3] B Gates. Trustworthy computing. Microsoft internal memo (15 January), see http://*www.wired.com/news/business/0*, 1367(49826):00, 2002.

[4] Frakes W. Systematic Software Reuse: A Paradigm Shift. In Proceedings of Third International Conference on Software Reuse: Advances in Software Reuse. Los Alamitos, California: IEEE Computer Society Press, 1994

[5] Anneliese von Mayrhauser, Richard T. Mraz, Jeff Walls, and Pete Ocken. Domain based testing: Increasing test case reuse. In ICCS ’94: Proceedings of the 1994 IEEE International Conference on Computer Design: VLSI in Computer & Processors, pages 484–491, Washington, DC, USA, 1994. IEEE Computer Society. ISBN 0-8186-6565-3.

[6] Kulkarni. Reusable test cases how can it facilitate web testing. In 3rd Annual International Software Testing 2001 Conference in India, 2001.

[7] Yongbo Wang , Ontology-Based Test case Generation for testing web service. Autonomous Decentralized Systems, 2007. ISADS '07. Eighth International Symposium.

[8] H. G. Luxiao Li, Gewei Chenxin Li. Designing a test case library system of supporting sharing and reusing(in Chinese).Journal of computer science, 33(5):290-291, 2006.

[9] Z. L. Shao, X. Y. Bai, and C. C. Zhao. Research and implementation of a reuse-oriented test design model. Journal of mini-micro systems(In Chinese), 27(11):2150–2155, 2006.

[10] Nicola Guarino**,** Formal Ontology and Information Systems**.** Proceedings of FOIS’98, Trento, Italy, 6-8 June 1998. Amsterdam, IOS Press, pp. 3-15.

[11] T. R. Gruber. A translation approach to portable ontology specifications. Knowledge Acquisition, 5:199–220, 1993

[12] T. R. Gruber. A translation approach to portable ontology specifications. Knowledge Acquisition, 5:199–220, 1993.

[13] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies, 43:907–928, 1995.

[14] M. Uschold and M. Gruninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.