A Test Case Selection from Using Use Case Description Changes

Monthawan Raengkla, Taratip Suwannasart

***Abstract*— The software industry is in a strong market competition. The software development method to support the present market with rapid requirement changes should reduce the time to market with customer needs. Developers are able to reduce the implementation time but to have adequate quality of software spends much development time. In software quality assurance, the solution to reduce time spend is using automation testing. Many test case selection techniques such as risk base by code, object, or model are commonly used in the industry. This paper offers another test case selection technique by detecting changes of use case descriptions.**

***Index Terms*— use case description, test case, change impact**

I. Introduction

THE software industry has many competitors due to merging of new technologies and new requirements from young generation user. New technologies drive the software industry to produce many products to serve existing users and pursue new users to invest their money using the software. The software development methodology also transforms to serve the rapid market changes as well. The methodology being commonly used in the organization and the company requires to reduce time in software production such as the requirement gathering phase, the requirement analysis phase, the software implementation phase, and the software testing phase. The two keys of the software production are to deliver user's requirements and to deliver good quality software. Due to the software can be changed, many tools have been developing to support the effect of the risk software management, the change management, and the risk-based change analysis. The most accuracy change impact analysis is detected the change in source code level but it spends times to understand the code change and to apply the change to the source code. The change detection in the early phase improves much time reduction in the development life cycle. At the beginning of development life cycle, the requirement gathering phase is the appropriate time to emphasize the change and the change impact in many record forms such as system requirements documents, use case diagrams, XML documents, and etc. The software requirement changes impact the software quality mechanism of the enhancement and existing features quality assurance. A new requirement implementation requires new test suite to ensure software quality for the enhancement and the existing test suite will be re-used to examine any failure in other existing features. The empirical regression test selection techniques [1] were very well-known in the past to perform a testing by all test cases because it is the strongest way to ensure software quality but it is the most cost consumption. Another technique to select some test cases to perform testing was introduced to reduce time consumption but it is unable to fulfill the software quality because some cases might be ignored even it was a main functionality. So the change impact analysis is able to identify the software impact and to select test cases more accuracy point of changes. Another approach to detect the change impact is in the requirement document such as use case by use case map specification [2] and case spec [3] which the relationship between requirement and test case has been created to figure out the change impact and manually look through test cases to identify the valid cases but the programming knowledge of use case symbol transformation is required. This paper proposes the solution to detect change impact on requirement phase by using use case description to reduce the symbol transformation and identify the test cases received the change impact to reduce the time consumption to perform testing.

The reminder of this paper is organized as follows: Section 2 provides the background of Use Case Description and change detection work in the past. Section 3, 4, and 5 describe Use Case Diagram, Use Case Description, and XML Language respectively. Section 6 provides the test case selection framework. Conclusion is presented in Section 7.

II. Background

Previous work [8] proposes an approach for generating test cases from use case based on a decision table. The use cases are reduced complexity and rearranged to be one-column table style based on use case description by Cockburn [6]. The input of use case description and the result of test case generation are in XML format but it is unable to detect the requirement changed. The framework for change detection and re-used testing in requirement level [9] doesn't require any coding knowledge but the change detected by Use Case Map (UCM) notation in formal concept analysis does require coding knowledge. The most effective changes detected in requirement level by the slicing

Manuscript received December 26, 2012.

M. R. is with the Software Engineering Laboratory Center of Excellence in Software Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand (e-mail: monthawanr@gmail.com).

T. S. is with the Software Engineering Laboratory Center of Excellence in Software Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand (e-mail: Taratip.S@chula.ac.th).

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

analysis and dependency analysis in use case map specification level [2]. However the changes of input and output in use case description are excluded. Case Spec [3] is a tool for software life cycle management. It is able to support many input formats and also trace failure from e-mail notification automatically. Whereas the mapping requirement with test case and identifying change area in related test case have to do manually.

III. Use Case Diagram

A use case diagram [4] portrays actors, use cases, and the relationship among them.

A. Actor

An actor is a thing having role to play such as a person, group of person, or a system. The actor name might be job description or relationship. Figure 1 is a stick figure.

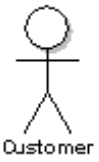

Fig. 1. An Actor

B. Use Case

A use case describes the interactions and activities of system and actors. The use case connects a primary actor and describes various scenario related to that actor. Figure 2 is an ellipse meant to a use case.

Fig. 2. A Use Case

C. Relations

The relations are being in-use for two types between use cases

Includes

A base use case calls another use case. The base use case describes higher level than the included use case. In figure 3, the arrow starts from base use case to included use case.

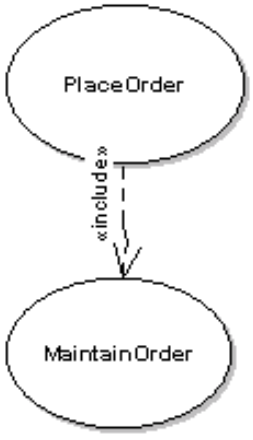

Fig. 3. An Include Relation

Extend

The base use case embeds an extension use case inside. The extension use case states the internal point from the base use case and outgrew scenario then pushes back the actor at the interrupt point. In figure 4, the arrow starts from the extension use case to the base use case. It reverts direction of the include relation.

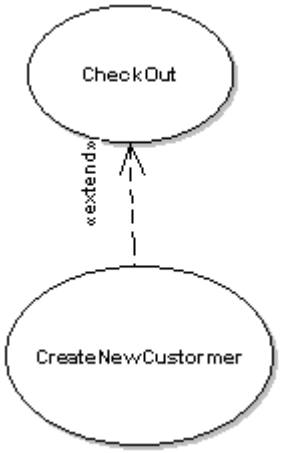

Fig. 4. An Extend Use Case

IV. Use Case Description

A use case description [5] is the detail of use case which is a part from the use case diagram. By the definition from UML standard states various formats of use case description as plain text, activity diagram, state machine, and etc. Cockburn [6] proposes a use case description format as "One-Column Table" which is simple to understand. Table 1 is a use case description in One-column table format.

TABLE I Use Case Description in One-column table format

USE CASE #	<the name is the goal as a short active verb phrase>	
Context of use	<a longer statement of the context of use if needed>	
Scope	<what system is being considered black box under design>	
Level	<one of summary, primary task, subfunction>	
Primary Actor	<a role name for the primary actor, or a description>	
Stakeholder and interests	Stakeholder	Interests
	<stakeholder name>	<put here the interest of the stakeholder>
	<stakeholder name>	<put here the interest of the stakeholder>
Preconditions	<what we expect is already the state of the world>	
Minimal Guarantees	<the interests as protected on any exit>	
Success Guarantees	<the interests as satisfied on a successful ending>	
Trigger	<the action upon the system that starts the use case>	
Description	Step	Action
	1	<put here the steps of the scenario from trigger to goal delivery and any cleanup after>
	2	<…>
	3	
Extensions	Step	Branching Action
	1a	<condition causing branching>:
		1a1 <action or name of sub use case>
		1a2 <…>

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

V. XML Language

XML language [7] is a language to be used in the communication between different systems, especially data part. The well-known formedness constraint should be as follows:

A. Start-tags And End-tags

An element should consist of start-tag, end-tag and content (as optional if it is empty-element). The start- and end-tag define the element's type. The start-tag should appear for every element with an unique name and the name should not contain a < or any external entities. The end-tag should contain the same name as start-tag. The text between start- and end-tag is an element's content but the element could have start-tag immediately followed by end-tag for empty element. For example,

<name>John Doe</name>

B. Element Type Declarations

The element type must be declared only once. The element content should be contained in the properly order start- and end-tag. For example;

<customer>
<name>John Doe
<id></name>123
</id>
</customer>

C. Type Name

A type name in the same start- and end-tag should have the same case. The XML interprets the name different for case-sensitive between, for example, <customer> and <Customer>

D. Attribute Form

An attribute could be added into start-tag after element type. The attribute should be declared with its value inside quote symbol. For example; <patient name="John Doe"></patient>

E. Text Declaration

The document text must have XML declaration at the beginning of document. It contains the version information or encoding declarations. The grammar should be start with a ?. For example;

<?xml version="1.0"?>
<customer>
<name>John Doe</name>
<id>123</id>
</customer>

VI. Test Case Selection Framework

This paper proposes a framework to classify valid and invalid test cases for re-using test cases in software development to reduce time in the development life cycle. Figure 5 describes the test case selection framework.

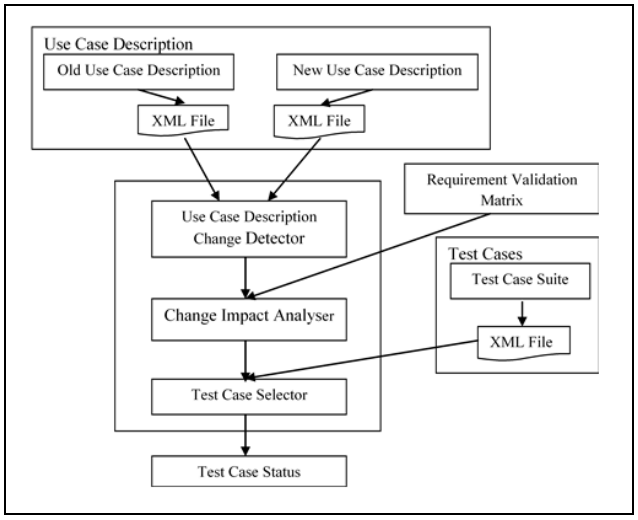

Fig. 5. Conceptual of Use Case Description Change Detection Tool

The component and functionality are described as follows;

A. Input Data

The input data contains 3 sections.

Use case description

A use case description comprises a use case ID, a use case name, an actor, an input, an output, a procedure, a success scenario, and an alternative scenario. The use case description should;

1) Normalize the included by identify the interrupt point with {UC+<use case ID>} or extend relation by indentify the interrupt point with {ExToUC+<use case ID>}.

2) Transform one-column table into XML document in well-formedness.

Test cases

A test case comprises a test case ID, a use case ID, a test case name, a prerequisite, an input, a procedure, and an output.

Requirement Validation Matrix

A requirement validation matrix is mapping between use cases and test cases.

B. Use Case Description Change Detector

The scope of use case description change impact consists of 3 parts:

Use case description input

The change of use case description input is detected the different of number, type, and size in the input items.

Use case description output

The change of use case description output is detected the different number, type, and size in the output items.

Use case description procedure

The change of use case description procedure is detected the different number, sequence, condition, and loop in the procedures.

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

C. Test Case Impact Analyser

The test case impact determines the change of related use case descriptions and the test cases mapping by the requirement validation matrix.

D. Test Case Selector

The conditions to identify valid and invalid test case are listed as follows;

Use case description input change

The input of use case description related to the input of test case is going to be invalid if the change affects the test case.

Use case description output change

The output of use case description related to the output of test case is going to be invalid if the change of output affects the test case.

Use case description procedure change

The procedure in use case description related to procedure in test case is going to be invalid if the change of procedure affects the test case.

E. Outcome

The various change from use case description impacts the test case validation. The result is the test case identification status to keep valid test cases being re-used in the development and to mark invalid test cases to being further proceed.

VII. Conclusion

This paper proposes the use case description change detection framework and test case validation identification methodology for test case re-using of software change implementation.

In section 6, we propose a test case selection framework. Our framework can automatically detect the change in use case descriptions and identify the test case impact. As a result, the test cases status differentiates test cases which are ready to be re-used or require an update test case before re-use them.

Furthermore, we have applied our framework to develop a tool which will be our future work.

References

[1] Graves Todd L., Harrold Mary Jean, Kim Jung-Min, Porter Adam, Rothermel Gregg, “Empirical study of regression test selection techniques”, (1998) Proceedings - International Conference on Software Engineering, pp. 188-197.

[2] Hassine, J., Rilling, J., & Hewitt, J. (2005), “Change Impact Analysis for Requirement Evolution using Use Case Maps”, Information Systems.

[3] GODA SOFTWARE. "CASE Spec 10.0", September 2011, Available from: http://www.analysttool.com/index.htm

[4] Alan D.,Barbara W.,David T., “Systems Analysis and Design with UML: An Object-Oriented Approach”, John Wiley & Sons, Inc., Third Edition

[5] Coleman, D., “A Use Case Template: Draft for discussion”, Fusion Newsletter, April 1998, September 2011, Available from: http://www.hpl.hp.com/fusion/md_newsletters.html.

[6] Cockburn. A., “Writing Effective Use cases”, United States of America: Addison-Wesley. 2000.

[7] W3C Recommendation, “Extensible Markup Language (XML) 1.0 (Fifth Edition)”, Tim Bray, Textuality and Netscape <tbray@textuality.com>, Jean Paoli, Microsoft <jeanpa@microsoft.com>, C. M. Sperberg-McQueen, W3C <cmsmcq@w3.org>, Eve Maler, Sun Microsystems, Inc. <eve.maler@east.sun.com>, Fran?ois Yergeau, 26 November 2008, September 2011, Available from: http://www.w3.org/TR/REC-xml/

[8] Setapong Leeraharattanarak, “An Approach For Automatically Generating Test Cases From Use Cases” , A Thesis for the Degree of Master of Science in Computer Science Department of Computer Engineering Faculty of Engineering Chulalongkorn University, 2004.

[9] Shiri, M., Hassine, J., & Rilling, J. (2007), “A Requirement Level Modification Analysis Support Framework”, Third International IEEE Workshop on Software Evolvability 2007, 67-74. Ieee.G. O. Young, “Synthetic structure of industrial plastics (Book style with paper title and editor),” in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)