Test Case Impact Analysis from Use Case Description Changes

Tawan Sakkarinkul, Taratip Suwannasart

***Abstract*— Software testing is an important activity in software development in terms of quality control. Software testing requires test cases for testing the system. Requirements changes can occur during the development phase, as a result, use case descriptions are changed. Many test cases are unusable because of use case description changes. It is not easy to specify whether they are usable or unusable. This paper proposes an approach for analyzing impact on test cases when use case description are changed.**

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

I. Introduction

Software testing is an important activity in the software development life cycle. Test cases are key factors in software testing. Test cases are created from work products such as use cases and source codes. The construction of use case occurs in requirements gathering phase. Therefore, at the beginning of development life cycle, requirements gathering phase is the appropriate time to emphasize the change. Changes can be occurred to any artifacts such as system requirements documents, use case diagrams, and XML documents [1]. When requirements changes during software development impact existing work products, test cases must be updated to consistently maintain. It has commonly been reported that impacted test cases associate with time consumption and costly.

This paper attempts to propose a framework to identify impacted test cases on requirements changes through use case description. Our approach is divided into three steps: requirements validation matrix generating, use case description changes analysis, and impacted test case analysis. The overall structure of the study takes the form of six sections, including this introductory section. Section II begins by laying out the theoretical dimensions of the research. Section III is concerned with the background of use case description, and requirements validation matrix. Section IV defines use case description changes impact test cases. Section V explains the framework with an example. Finally, the conclusion gives a brief summary, and areas for further research are identified.

Manuscript received December 28, 2014.

T. Sakkarinkul and T. Suwannasart are with the Software Engineering Laboratory Center of Excellence in Software Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand (e-mail: tawan.s@student.chula.ac.th, taratip.s@chula.ac.th).

II. Related Work

There are relatively few historical studies in the area of test case impact analysis. S. Phetmanee [2] developed a tool for impact analysis of test cases based on changes of a web application. The tool generates new test cases based on the black-box testing techniques by comparing two types of document: HTML document file, and XML Schema file. The HTML document file describes inputs and their forms. Further, the XML Schema file is a description of grammatical rules of XML document which supports boundary value definitions and can be used for well-formed input validation. Consequently, new test cases are generated by either equivalence class or boundary value analysis.

Similarly, J. Jainae [3] proposed a framework for test case impact analysis of database schema changes using use cases. The framework generates new test cases based on black-box testing techniques. There are four steps: analyzing database schema file, finding and repairing affected use cases, analyzing affected test cases, and generating new test cases. In addition, existing test cases generated from the use case description are taken into the analysis. As a result, affected test cases are replaced by new generated test cases which are either valid or invalid test case.

Whereas the previous studies use the black-box testing techniques to generate test cases, M. Raengkla [1] demonstrated a use case description change detection tool. She focuses on three parts: use case description input, output, and procedure. The changes of use case description input and output are detected by the difference of their numbers, types, and sizes. Furthermore, the changes of use case description procedure are detected by the difference of its number, sequence, condition, and loop in the procedures. The inputs of the tool consist of an old use case description, a new use case description, test cases, and a requirements validation matrix. Difficulties arise, however, when the requirements validation matrix does not exist. This paper is based on her approach. Although, we argue that requirements validation matrix is not necessary for an input because it can be generated by other inputs.

III. Background

A. Use Case Description

A use case description is a detail of a use case in the use case diagram. Cockburn [4] proposed a use case description format by “One-Column Table”, which is able to understand simply, shown in Table I.

ISBN: 978-988-19253-2-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I. USE CASE DESCRIPTION IN ONE-COLUMN TABLE

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	<…>
Extensions	Step	Branching Action
	1a	<condition causing branching>:
		1a1 <action or name of sub use case>
		1a2 <…>
Technology and Data Variations	1	<list of variations>

Likewise, S. Leeraharattanarak [5] adapted Cockburn's format to generating test cases. Table II compares the two formats.

TABLE II. USE CASE DESCRIPTION COMPARING BETWEEN COCKBURN AND S. LEERAHARATTANARAK

Use Case Description	Cockburn	S. Leeraharattanarak
Use Case No.	-	Include
Use Case Name	Include	Include
Context of use	Include	Include
Scope	Include	-
Level	Include	-
Primary Actor	Include	Include
Stakeholder and interests	Include	-
Preconditions	Include	Include
Input	-	Include
Minimal Guarantees	Include	Include
Success Guarantees	Include	Include
Trigger	Include	Include
Description	Include	Include
Extensions	Include	Include
Technology and Data Variations	Include	-
Abstract Use Case	-	Include

B. Requirements Validation Matrix

Requirements validation matrix [1], [6] shows the relationship between use cases and test cases. It is used for forward and backward traceability, therefore, it is able to indicate test cases that are not impacted by use case description changes. Table III gives an example of requirements validation matrix.

TABLE III. AN EXAMPLE OF REQUIREMENTS VALIDATION MATRIX.

		Use Case Description	
		UC1	***UC2***
Test Case	***TC1-01***	X	
	TC1-02	X	
	TC2-01		X
	TC2-01		X

IV. USE CASE DESCRIPTION CHANGES IMPACT TEST CASES

This paper uses requirements validation matrix to identify test cases which can be categorized into the following types of testing

- valid input testing
- invalid input testing
- success scenario testing for pass
- success scenario testing for fail
- alternative scenario testing

The differences of use case description changes have an effect on test cases. This paper focuses on five forms of changes.

1) Use Case Name

Use case name changes affect only the test case name. The result of the test case is still valid.

2) Use Case Input

There are four main use case input changes. Firstly, the number of input changes affects the amount of the test case inputs. Secondly, if the input names are changed, the test case input names will be affected. Another input change is that its types can affect the test case input types and values. Although the test cases are valid input testing, and input type changes do not decrease any possible valid input values, the input values are not affected. Finally, the input constraint changes have consequences for the test case precondition and the input values. However, the test cases that are valid input testing, and the input constraint changes do not reduce any possible valid input values. Then, the precondition and input values are not affected.

3) Use Case Output

A use case output is an additional element of use case description. It describes a proper output of the system. Thus, the use case output is related to the expected outputs of test cases. There are three main use case output changes. Firstly, the number of output changes affects the amount of test case expected outputs. Secondly, output name changes affect test case expected output name. Finally, output type changes also have influences on test case expected output type and values.

4) Success Scenario

Success Scenario condition changes affect test case precondition and input values. If the test cases are success scenario testing for pass, and condition changes do not reduce possible input values for pass the condition, precondition and input values will not be affected.

ISBN: 978-988-19253-2-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

5) Alternative Scenario

Alternative Scenario condition changes can have an impact on the test case precondition and the input values. Their consequences are similar to the success scenario.

V. Test Case Impact Analysis Framework

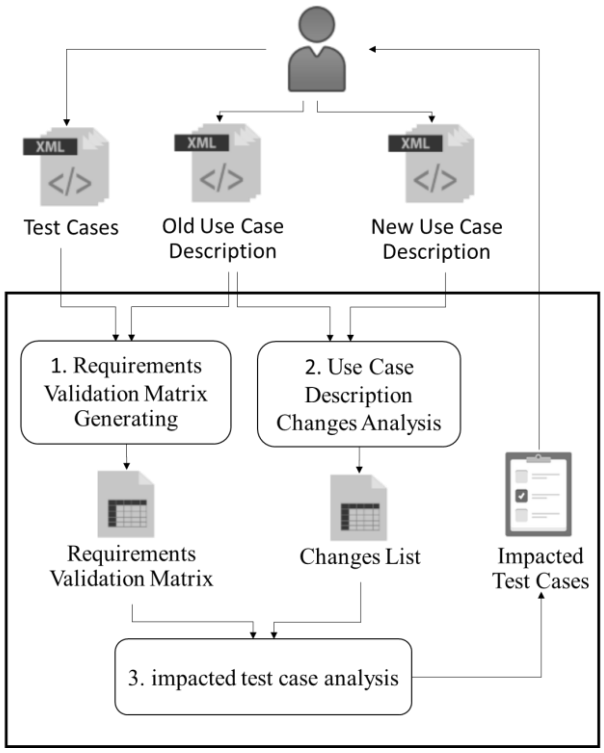

Fig. 1. The test case impact analysis framework

According to use case description changes, figure 1 shows the three-step test case impact analysis framework: requirements validation matrix generating, use case description changes analysis, and impacted test cases.

A. Inputs

1) Use Case Descriptions

A use case description consists of a use case ID, a use case name, inputs, outputs, a success scenario, and alternative scenarios. The use case description can be in many forms such as database, spreadsheet and text file, then it should be transformed into XML document in well-formedness.

2) Test Cases

A test case consists of a test case ID, a test case name, a precondition, inputs, and expected outputs. Likewise, it has to be transformed into XML document in well-formedness.

B. Requirements validation matrix generating

The framework analyzes use case descriptions in order to map with test cases. This paper focuses on three components of use case description which are name, input and output. The use case name and the test case name should be similar while the use case inputs must be the same as the test case inputs. Moreover, the use case outputs must be the same as the test case outputs. Normally, test cases are generated from a use case description of which name is similar to the test cases. It can be implied that the naming concept is applied.

This study determines the similarity of use case name and test case name by using the similarity percentage which is calculated by a number of words contained in both use case name and test case name divided by the total number of words in the use case. The similarity percentage is greater than or equal to the acceptable level, and inputs and outputs of the use case and the test case are same, then they have a relationship.

C. Use case description changes analysis

In order to analyze each pair of the use case descriptions, the framework indicates the component changes and their details. The scope of the analysis is focused on four components of use case descriptions including a name, inputs, outputs, and a success scenario. The input type and the input constraints change analysis require additional information to determine whether those changes affect the previous valid values of the inputs or not. Success scenario condition change analysis consists of two types. The first type is constraint reduction determined by adding "OR" or removing "AND" condition. Another is constraint change. The result of the use case description changes analysis is a changes list.

D. Impacted test case analysis

In order to analyze impacted test cases, there are two steps. Firstly, the possible impacted test cases that related to use case description changes are identified by requirements validation matrix. Secondly, the components of impacted test cases are indicated by the changes list and the test case types, including valid input testing, invalid input testing, and success scenario testing for pass. Moreover, additional information is required to describe the type of testing.

E. Framework demonstration

The framework can be illustrated briefly by the following case. There are two use case descriptions with two versions and four test cases as inputs. Table IV and V show the two old use case descriptions. Table VI and VII explain the two new use case descriptions. Table VIII, IX, X and XI are four test cases.

TABLE IV. Use Case Description UC_VA_01 Version 1

ID	UC_VA_01		
Name	add a volunteer's activity		
Input	**Name**	**Type**	**Constraints**
	volunteerCode	String	required = true
	activityCode	String	required = true
	workingDate	Date	required = true
	hours	Integer	required = true min = 1 max = 24
Output	**Name**	**Type**	
	activityRecordId	Long	
Success Scenario	isActiveVolunteer(volunteerCode) isActiveActivity(activityCode) !isFutureDate(workingDate)		

ISBN: 978-988-19253-2-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE V. USE CASE DESCRIPTION UC_VD_01 VERSION 1

ID	UC_VD_01		
Name	add a volunteer's donation		
Input	**Name**	**Type**	**Constraints**
	volunteerCode	String	required = true
	donationCode	String	required = true
	donationDate	Date	required = true
	amount	Double	required = true scale = 2 min = 100.00 max = 100000.00
Output	**Name**	**Type**	
	donationRecordId	Long	
Success Scenario	isActiveVolunteer(volunteerCode) isActiveDonation(donationCode) !isFutureDate(donationDate)		

TABLE VI. USE CASE DESCRIPTION UC_VA_01 VERSION 2

ID	UC_VA_01		
Name	add volunteer activity		
Input	**Name**	**Type**	**Constraints**
	volunteerCode	String	required = true
	activityCode	String	required = true
	activityDate	Date	required = true
	startHour	Integer	required = true min = 0 max = 23
	endHour	Integer	required = true min = 1 max = 24
Output	**Name**	**Type**	
	activityRecordId	Long	
Success Scenario	isActiveVolunteer(volunteerCode) isActiveActivity(activityCode) !isFutureDate(workingDate) endHour > startHour		

TABLE VII. USE CASE DESCRIPTION UC_VD_01 VERSION 2

ID	UC_VD_01		
Name	add volunteer donation		
Input	**Name**	**Type**	**Constraints**
	volunteerCode	String	required = true
	donationCode	String	required = true
	donationDate	Date	required = false
	Amount	Double	required = true scale = 2 min = 100.00 max = 100000.00
Output	**Name**	**Type**	
	donationRecordId	Long	
Success Scenario	isActiveVolunteer(volunteerCode) isActiveDonation(donationCode)		

TABLE VIII. TEST CASE TC_VA_01_01

ID	TC_VA_01_01		
Name	add volunteer activity : valid		
Pre-condition	isFuture(workingDate) = false		
Input	**Name**	**Type**	**Value**
	volunteerCode	String	V01
	activityCode	String	A01
	workingDate	Date	2014-11-15
	hours	Integer	8
Output	**Name**	**Type**	**Value**
	activityRecordId	Long	*

TABLE IX. TEST CASE TC_VA_01_02

ID	TC_VA_01_02		
Name	add volunteer activity : invalid hour		
Pre-condition	isFuture(workingDate) = false		
Input	**Name**	**Type**	**Value**
	volunteerCode	String	V01
	activityCode	String	A01
	workingDate	Date	2014-11-15
	hours	Integer	25
Output	**Name**	**Type**	**Value**
	activityRecordId	Long	null

TABLE X. TEST CASE TC_VD_01_01

ID	TC_VD_01_01		
Name	new volunteer donation : valid		
Pre-condition	isFuture(donationDate) = false		
Input	**Name**	**Type**	**Value**
	volunteerCode	String	V01
	donationCode	String	D01
	donationDate	Date	2014-11-15
	amount	Double	10000
Output	**Name**	**Type**	**Value**
	donationRecordId	Long	*

TABLE XI. TEST CASE TC_VD_01_02

ID	TC_VD_01_02		
Name	new volunteer donation : invalid date		
Pre-condition	isFuture(donationDate) = true		
Input	**Name**	**Type**	**Value**
	volunteerCode	String	V01
	donationCode	String	D01
	donationDate	Date	2016-11-15
	amount	Double	10000
Output	**Name**	**Type**	**Value**
	donationRecordId	Long	Null

The first step is that the framework generates a requirements validation matrix shown in table XII. This example sets the acceptable level of similarity percentage at 50%. The use case UC_VA_01 is related to the test case TC_VA_01_01 and TC_VA_01_02 with 100% similarity because every words of the use case name is appeared in those test case names. It means that similarity percentage is greater than the acceptable level, as a result, they have a relationship. The use case UC_VD_01 is related to the test case TC_VD_01_01 and TC_VD_01_02 with 66.67% similarity because there are two out of three words of use case name are appeared in those test case names. In the same way, they also have a relationship.

TABLE XII. REQUIREMENTS VALIDATION MATRIX

		Use case	
		UC_VA_01	***UC_VD_01***
Test case	***TC_VA_01***	X (100%)	
	TC_VA_02	X (100%)	
	TC_VD_01		X (66.67%)
	TC_VD_02		X (66.67%)

The second step is that the framework compares between two versions of the use case descriptions and displays the changes list of the use case descriptions in table XIII.

ISBN: 978-988-19253-2-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE XIII. CHANGE LIST OF THE USE CASE DESCRIPTIONS

Use Case Description		
Component	**Change Type**	**Detail**
UC_VA_01		
Input	Rename	from "workingDate" to "activityDate"
	Remove	Hours
	Add	startHours
	Add	endHours
Success Scenario	Increase Constraints	endHour > startHour
UC_VD_01		
Input	Decrease Constraints	from "require = true" to "require = false"
Success Scenario	Decrease Constraints	!isFutureDate(donationDate)

Finally, the framework displays a list of the impacted test case in table XIV and XV. Table XIV shows the status of the test cases, and table XV shows the details of the impacted test cases.

TABLE XIV. REQUIREMENTS VALIDATION MATRIX WITH TEST CASE STATUS

		Use case	
		UC_VA_01	***UC_VD_01***
Test case	***TC_VA_01***	X (100%) modify and retest	
	TC_VA_02	X (100%) modify and retest	
	TC_VD_01		X (66.67%) retest
	TC_VD_02		X (66.67%) modify and retest

TABLE XV. DETAIL OF IMPACTED TEST CASES

Test case		
Component	**Recommendation**	**Remark**
TC_VA_01-1		
Input	change "workingDate" into "activityDate"	UC_VA_01 - input name change
	remove "hours"	UC_VA_01 - input decrease
	add "startHours"	UC_VA_01 - input increase
	add "endHours"	UC_VA_01 - input increase
	edit value of "startHours" and "endHours"	UC_VA_01 - success scenario constraint change
TC_VA_02-2		
Precondition	edit message	UC_VD_02 - success scenario constraint decrease

VI. CONCLUSION AND FUTURE WORK

This paper aims to propose an approach for test case impact analysis from use case description changes. This study has shown that the test case impact can be identified automatically by the framework. In addition, the framework can be used for generating a requirements validation matrix. Finally, the framework helps to reduce an effort to find impacted test cases.

For future research, it would be interesting to implement a prototype using the proposed framework. It can be tested by use case description changes focused on the four components including a name, inputs, outputs, and a success scenario. Then, the results of prototype are evaluated by comparing the time reduction and resource consumption with a traditional method. In addition, the information retrieval techniques can be used in order to map use cases and test cases by their names in the requirements validation matrix generating. The further studies could be improved the knowledge of the test case impact analysis.

REFERENCES

[1] M. Raengkla and T. Suwannasart, "A Test Case Selection from Using Use Case Description Changes," presented at the International MultiConference of Engineers and Computer Scientists, Hong Kong, 2013.
[2] S. Phetmanee and T. Suwannasart, "A Tool for Impact Analysis of Test Cases Based on Changes of a Web Application," presented at the the International MultiConference of Engineers and Computer Scientists, Hong Kong, 2014.
[3] J. Jainae and T. Suwannasart, "A Tool for Test Case Impact Analysis of Database Schema Changes Using Use Cases," presented at the Information Science and Applications (ICISA), 2014 International Conference on, 2014.
[4] A. Cockburn, Writing Effective Use Cases: Addison-Wesley Longman Publishing Co., Inc., 2000.
[5] S. Leeraharattanarak, "An Approach for Automatically Generating Test Cases from Use Cases," Master of Science in Compurter Science, Computer Engineering, Chulalongkorn University, 2004.
[6] "IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries," IEEE Std 610, pp. 204, 1991.

ISBN: 978-988-19253-2-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)