Impact Analysis to Database Schema and Test Cases from Inputs of Functional Requirements Changes

Apirak Kampeera, Taratip Suwannasart

***Abstract*—Changes always occur in software development. Software consists of functions which are defined in term of capabilities, inputs, and outputs. A function which is verified by test cases is always associated with attributes in a database. If inputs of functional requirements are changed, they will effect to the database schema and test cases directly. Therefore, the impact to database schema and test cases are needed to clarify before software testing. Thus, this paper proposes an approach to analyze an impact to database schema and test cases from inputs of functional requirements which are changed. After that database schema is updated to cooperate with the functions properly. Finally, test cases related to the functions are verified to check if they need to be updated.**

***Index Terms*— functional requirements, database, test case, impact analysis**

I. Introduction

Changes always occur in software development. They might occur not only in a development phase but also in a software design phase. The impact may effect to functions and their inputs as well as outputs. The function is always associated with attributes of a database schema for handling data. If inputs of a function are changed, they may affect the function execution as well as the database. Thus, it is necessary to analyze an impact to the database, and then update the database so that the function and the database can work together properly.

Software Testing is an important process in the software development, because the objective of software testing is verify if the accuracy of the system meets the user needs and to find errors in a function by using test cases as an equipment to verify the function's operation. Black box test cases are constructed based on functional requirements. When inputs of a function are changed, test cases are affected directly. Therefore, it is necessary to analyze an impact to the test cases.

This paper proposes an approach to analyze the impact to database schema and test cases when inputs of functional requirements are changed. We analyze the changes to the database schema, and then create a SQL statement to update the database schema in order to reflect the changes. The test cases related to the changes are updated as well.

Manuscript received December 10, 2015.

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

The reminder of this paper is organized as follows. Section 2 provides related work while section 3 explains background related to this paper. Section 4 presents an approach for impact analysis to database schema and test cases from inputs of functional requirements changes. Finally, a conclusion and future work are described in section 5.

II. Related Work

There are several related studies that inspired this research. J. Jainae and T. Suwannasart[1] proposed a tool that provides a test case analysis from schema changes. Use case description is a source used to create new test cases for replacing original test cases. This research gets a set of SQL queries, analyzes the database schema changes, compares a consistency of attributes in the database schema, and then finds and compares the use case description. The tool selects the affected use case description to compare data in the database. It shows type of SQL commands such as DROP, ADD, and UPDATE. The next step is to analyze the test case by comparing between test data and the use case description conditions. The result shows affected test cases and unaffected test cases. At last, generates new test cases.

Another approach proposed by M. Raengkla and T. Suwannasart[2] presents a method for selecting test cases to verify use case description by importing use case description, test case, and requirement validation matrix. The method analyzes use case description changes to focus on an input, an output and a use case description steps. After that, affected test cases are discovered by using a requirements validation matrix. Then the test cases are verified by comparing input, output, and steps to use case description.

S. Phetmanee and T. Suwannasart[3] presented a methodology of the document file comparison. Values from two versions of HTML files and XML Schema files are compared to find the differences. The result obtained from this tool consists of several changed values. This research suggests 7 patterns of the change of a web application as follows.

1. The change of variable name.
2. The change of data type.
3. The change of variable value.
4. The change of variable tag.
5. The change of order.
6. The change of link.
7. The change of variable number.

III. Background

A. *Requirement Traceability Matrix[4]*

Requirements Traceability Matrix (RTM) is a document that links functional requirements throughout the validation process, to ensure that all requirements defined for a system are tested. The traceability matrix is a tool for the validation team, to ensure that requirements are not lost during the validation process, and for auditors, to review the validation documentation. An example of requirements traceability matrix is shown in table I.

Table I Requirements Traceability Matrix

Requirement ID	Requirement description	TC 001 (user login)	TC 002 (add user)	TC 003 (delete user)
SR-1.1	User should be able to login using LDAP user	x		
SR-1.2	User should be able to see error message when fail to login	x		
SR-1.3	Admin user should be able to add user on admin page		x	
SR-1.4	Admin user can not add duplicate user on admin page		x	

IV. Approach for Impact Analysis to Database Schema and Test Cases

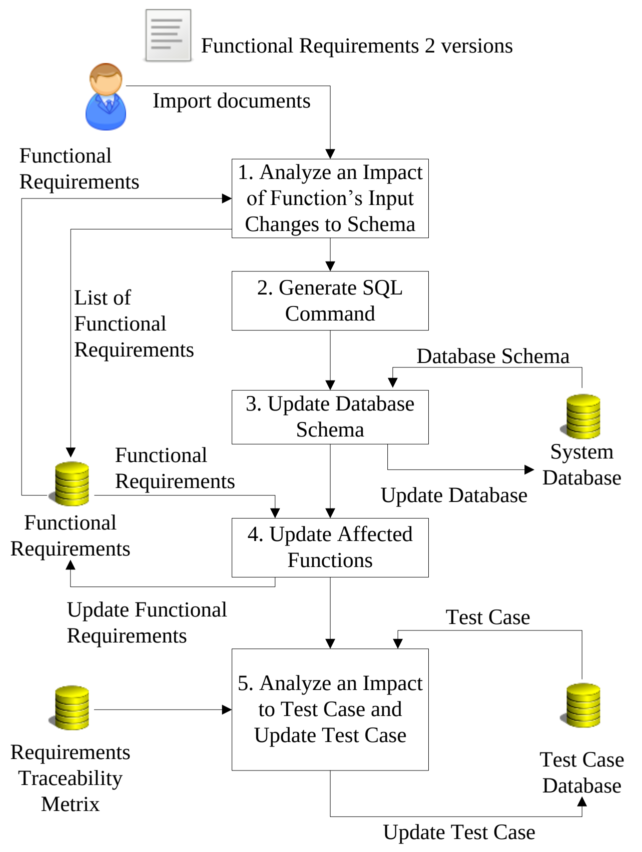
Functional Requirements 2 versions
Import documents
Functional Requirements
1. Analyze an Impact of Function's Input Changes to Schema
List of Functional Requirements
2. Generate SQL Command
Database Schema
3. Update Database Schema
System Database
Functional Requirements
Update Database
Functional Requirements
4. Update Affected Functions
Update Functional Requirements
Test Case
5. Analyze an Impact to Test Case and Update Test Case
Requirements Traceability Metrix
Test Case Database
Update Test Case

Fig. 3 Framework for Impact Analysis to Database schema and Test Cases

In order to analysis impact for database schema and test cases, fig.3 depicts a concept to analyze an impact. The concept consists of 5 processes. First, to find which inputs are changed. Next, changes of the inputs are compared to the database. Then, SQL command to update in the schema is generated to reflect the changes. Then, test cases related to the function are analyzed.

1) *Analyze Impact of Function's Input Changes to Database Schema*

At the beginning, user imports 2 versions of functional requirements which have different inputs. A functional requirement consists of a function number, a function name, an objective, a function version and input specifications which include an input name, data type, length, and constraints. A functional requirement is illustrated in table II.

TABLE II An Example of Functional Requirements

Function No.	FC_01			
Function Name	Add Customer			
Objective	User able to add new customer profile			
Function Version	1.0			
Input Specifications				
Input Name	Type	Length	Constraints	Table Name
CUSTOMER_ ID	int			CUSTOMER
CUSTOMER_ NAME	varchar	50		CUSTOMER
CUSTOMER _CREDIT_ LIMIT	decimal	10,2	min=20.00 &&max=5 00.00	CUSTOMER
Output Specifications				
Output Name	Type	Length	Validation	

Input changes are only compared between 2 versions of functional requirements that exclude the database. The result shows five types of changes which are explained as follows

A. Add inputs
B. Delete inputs
C. Update names of inputs
D. Update types of inputs
E. Update length of inputs
F. Update constraints

Table III shows the result of comparisons between two versions of the functional requirements that there are 3 changes. First, a CUSTOMER_ID in CUSTOMER table is changed to CUSTOMER_NUMBER. Second, a CUSTOMER_BIRTHDAY is added into a CUSTOMER table. Finally, a CUSTOMER_CREDIT_LIMIT in CUSTOMER table is removed.

TABLE III An Example of List of Input Changes

No	Table	The Original Input Name	Changes	
			Type	Description
1	CUSTOMER	CUSTOMER_ ID	Alter	Edit field name from Customer_id to Customer_number
2	CUSTOMER	CUSTOMER_ BIRTHDAY	Add	Add field Customer_birthday as CUSTOMER
3	CUSTOMER	CUSTOMER _CREDIT_ LIMIT	Delete	Delele Field

After comparison of inputs' changes, the next step is analyzing the changes of input to a database schema by using SQL statement to check field properties and relations in the database, and creating a condition for generating SQL command to update database.

There are 3 operations for analyzing impact to a database schema which are divided by the changes of input as shown in Fig. 4 – 6.

1.1) Analysis updating database by adding input

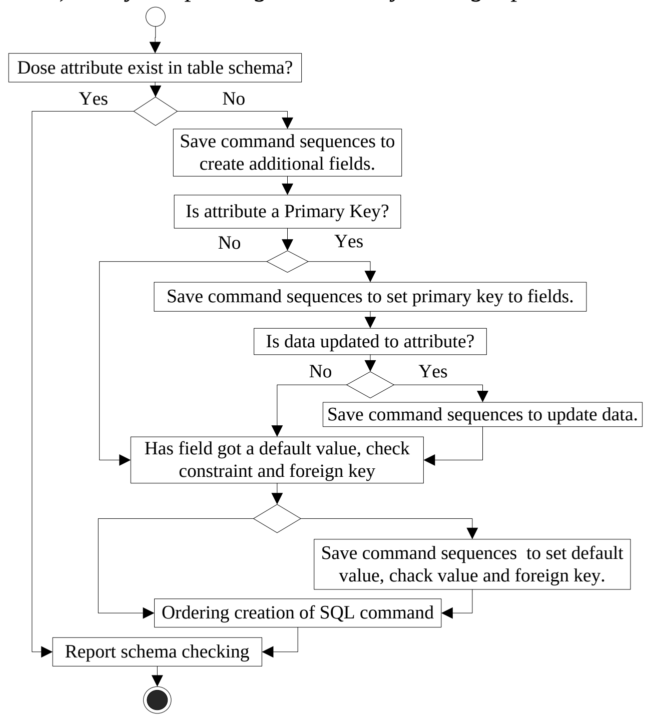
Dose attribute exist in table schema?
Yes
No
Save command sequences to create additional fields.
Is attribute a Primary Key?
No
Yes
Save command sequences to set primary key to fields.
Is data updated to attribute?
No
Yes
Save command sequences to update data.
Has field got a default value, check constraint and foreign key
Save command sequences to set default value, chack value and foreign key.
Ordering creation of SQL command
Report schema checking

Fig. 4 Activity for analysis for updating database in case of adding inputs of function

1.2) Analysis update database by deleting input

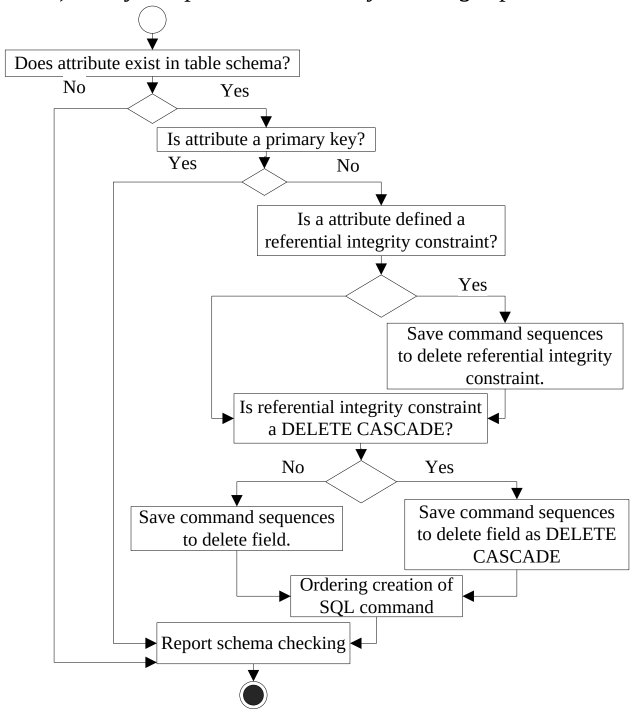
Does attribute exist in table schema?
No
Yes
Is attribute a primary key?
Yes
No
Is a attribute defined a referential integrity constraint?
Yes
Save command sequences to delete referential integrity constraint.
Is referential integrity constraint a DELETE CASCADE?
No
Yes
Save command sequences to delete field.
Save command sequences to delete field as DELETE CASCADE
Ordering creation of SQL command
Report schema checking

Fig. 5 Activity for analysis a database update in case of deleting inputs of function.

1.3) Analysis update database by updating input

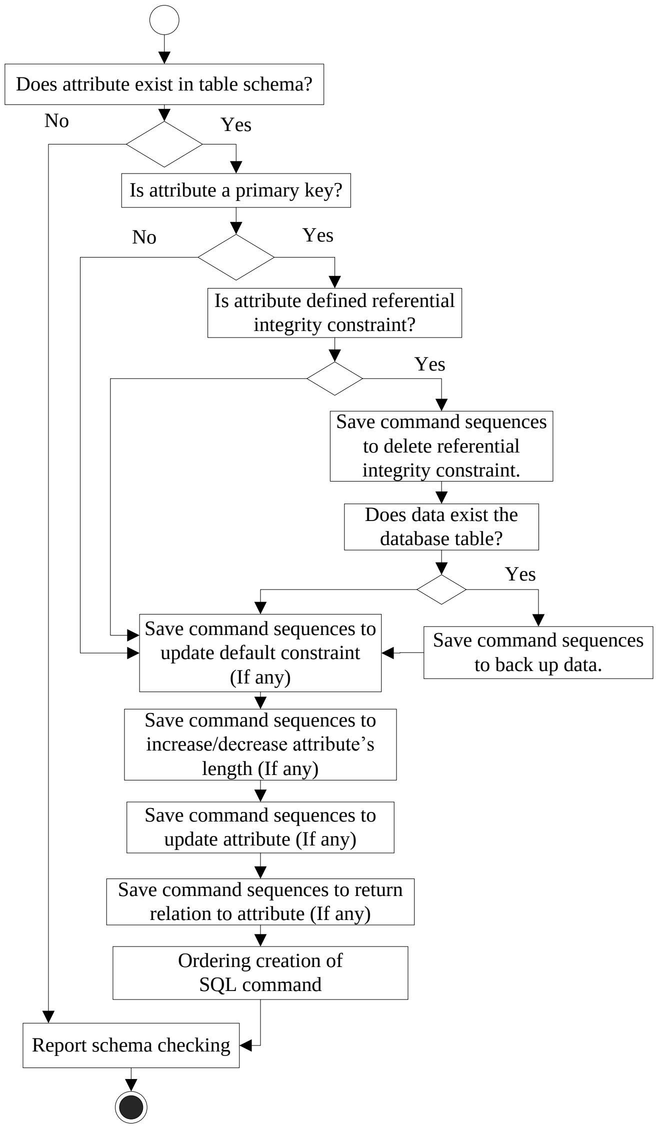
Does attribute exist in table schema?
No
Yes
Is attribute a primary key?
No
Yes
Is attribute defined referential integrity constraint?
Yes
Save command sequences to delete referential integrity constraint.
Does data exist the database table?
Yes
Save command sequences to update default constraint (If any)
Save command sequences to back up data.
Save command sequences to increase/decrease attribute's length (If any)
Save command sequences to update attribute (If any)
Save command sequences to return relation to attribute (If any)
Ordering creation of SQL command
Report schema checking

Fig. 6 Activity for analysis a database update in case of updating inputs of function.

Fig.4 shows an activity to analyze the database in case of adding inputs of a function. Firstly, the activity checks an existing of attributes related to the new inputs in a database table. If the attributes do not appear in the database table, an expression to add the attributes is created. Secondly, the activity checks if the new inputs are a primary key. If they are a primary key, an expression to assign a primary key constraint is created. Then the activity checks an updating data to attributes. If attribute need to be updated, an expression to update data is assigned for updating data to the attributes. Finally, the activity checks the database constraints of inputs such as a DEFAULT constraint, a CHECK constraint, and a FOREIGN KEY constraint. If any constraints exist, an expression to add those constraints is created.

Fig.5 shows an activity to analyze the database schema in case of deleting inputs of a function. Firstly, the activity checks an existing of attributes in a database table. Then the activity checks if the attributes are a primary key. If they are not a primary key, the activity checks a referential integrity constraint of the attributes. If there are any referential integrity constraints, an expression to delete those referential integrity constraints is created. Then the activity checks if a referential integrity constraint is a DELETE CASCADE. In case of a referential integrity constraint is a DELETE CASCADE, an expression to delete as DELETE CASCADE is created. In case of schema is not a DELETE CASCADE, an expression to delete attributes is created.

Fig.6 shows an activity to analyze database in case of updating inputs of a function. Firstly, the activity checks an existing of attributes related to the inputs in a database table. Then the activity checks if the attributes are a primary key. If they are a primary key, the activity checks a referential integrity constraint of the attributes. If there are any referential integrity constraints, an expression to delete those referential integrity constraints is created. Then the activity checks data in the database table. If there are data in the database, an expression to back up the data is created. If there are DEFAULT constraints, increment or decrement a length of attribute, updating attribute, and returning the relations to the attribute, an expression to response those action is created.

Table IV Sequences of Actions to Generate SQL Command

Ordering No	Action	Table Field and Properties
1	Check Exist	CUSTOMER.CUSTOMER_ID
2	Drop constraint	CUSTOMER_CUSTID_PK
3	Alter field name	CUSTOMER.CUSTOMER_NUMBER
4	Create constraint	CUSTOMER_CUSTID_PK

Table IV shows an example output for analysis an impact of function’s input changes. The output is a sequential for generating SQL command, Table IV includes of 4 steps. First, CUSTOMER_ID is checked an existing in a CUSTOMER table. Second, drop the field’s constraint. Third, alter CUSTOMER table by adding field CUSTOMER_NUMBER. Finally, a constraint is assigned to the attribute.

2) *Generate SQL Command*

The result from analysis an impact of function’s input changes to schema is a list of creation SQL command. According to table IV CUSTOMER_ID is changed to CUSTOMER_NUMBER, which can be generated SQL command shown in Fig.7.

```
/*---Check Exist---*/
If exists(select * from INFORMATION_SCHEMA.COLUMNS
where TABLE_NAME='CUSTOMER' and
COLUMN_NAME='CUSTOMER_ID')

BEGIN

/*---Drop Constraint---*/
ALTER TABLE CUSTOMER
DROP FOREIGN KEY CUSTOMER_PRIMARY_KEY

/*Alter Field*/
SP_RENAME CUSTOMER_ID, CUSTOMER_CODE


/*--Create Constraint--*/
ALTER TABLE CUSTOMER
ADD CONSTRAINT CUSTOMER_PRIMARY_KEY PRIMARY KEY
(CUSTOMER_NUMBER)

END
```

Fig. 7 Example of SQL command which is generated from table IV

3) *Update Database Schema*

In order to update the database schema, there are two steps. First, the original database schema is backed up for restoring database when error occurs in updating the database. Second, the database schema is updated by executing SQL command that is generated from the previous step.

4) *Update an Affected Function*

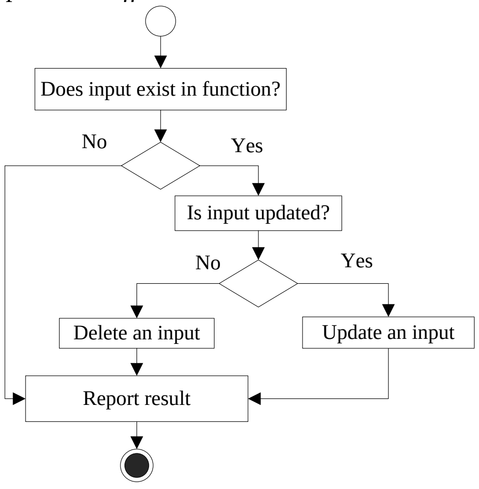

Fig. 8 Activity for updating an affected function

The inputs of a function do not exist in a single function. Hence, finding of affected functions is performed for updating the functions.

According to update the affected functions, fig.8 shows the first step is checking existing inputs of a function. If inputs are in the function, they will be checked types of inputs’ change. If the change is an updating, the inputs of function will be updated. If the change is a deleting, the inputs of function will be deleted. After that, the function will be set a new version.

5) *Analyze an impact to test cases and update test cases*

This step is a comparison of the relationship between a function and test cases. Affected test cases are determined by using a requirement traceability matrix as shown table V. Table V is a matrix which shows a relation between function number FC_01 to test cases number TC01.

Then, the test cases are compared by types of inputs’ change which consist of 3 types as follows.

1) Analysis changes of adding inputs, inputs are added into test case and test data are assigned to the inputs.

2) Analysis change of deleting inputs, the inputs are checked if the inputs exist in a test case, they will be removed from the test case.

3) Analysis changes of editing inputs, the original inputs are deleted. New inputs are added into a test case and test data are assigned to the new inputs.

Table V A Requirement Traceability Matrix

		Test Case Number					
		TC01	TC02	TC03	TC04	TC05	TC06
Function Number	FC_01	x					
	FC_02		x		x		
	FC_03			x		x	x

Table VI An Original Test Case

Test case no.	TC01	
Test case name	Test Case Customer Profile - New	
Test case objective	To test for adding a new customer	
Test case version	1.0	
Function no.	FC_01	
Function version	1.0	
Test data	Input Name	Value
	CUSTOMER_ID	000001
	CUSTOMER_NAME	John
	CUSTOMER_CREDIT_ LIMIT	50.00
Expect result	Able to add a new customer	

Table VI shows an original test case, which consists of a test case no, a test case name, a test case objective, a test case version, a function no., a function version, test data, and an expected result.

If the inputs of a function have been changed, the test cases will be updated. Test case will be updated test data according to type of change in table III as follows.

A. Adding Input

In case of adding inputs, a test case is added a test data. In row 2 of table III shows input CUSTOMER_BIRTHDAY is added into table named CUSTOMER, therefore the test case is add the input named CUSTOMER_BIRTHDAY as well.

B. Deleting Input

In case of deleting input, in row 3 of table III shows field named CUSTOMER_CREDIT_LIMIT is deleted, therefore an input named CUSTOMER_CREDIT_LIMIT is deleted from a test case.

C. Updating Input

In case of updating inputs, the existed input is deleted, and then a new input is added into a test case. In row 1 of table III shows an input CUSTOMER_ID which is become CUSTOMER_NUMBER. Therefore, an input CUSTOMER_ID is deleted from a test case. Then an input CUSTOMER_NUMBER is added into a test case.

Table VII A New Test Case

Test case no.	TC01	
Test case name	Test Case Customer Profile - New	
Test case objective	To test for adding a new customer	
Test case version	2.0	
Function no.	FC_01	
Function version	2.0	
Test data	Input Name	Value
	CUSTOMER_NUMBER	000001
	CUSTOMER_NAME	John
	CUSTOMER_BIRTHDAY	01/01/2000
Expect result	Able to add a new customer	

After the test cases are updated, they will be generated into a new version as shown in test case version. Table VII shows a test case is updated, which contains three inputs – CUSTOMER_NUMBER, CUSTOMER_NAME, and CUSTOMER_BIRTHDAY.

V. Conclusion And Future Work

Changes is always happen every moment in software development, therefore an impact analysis from changes is important. This paper presents an approach to analyze an impact to database schema and test cases from inputs of functional requirement changes. The approach shows comparison between 2 versions of inputs, checks an impact to database, records an action sequentially, generates SQL command, and then updates functional requirements. Moreover, test cases must be identified by requirement traceability matrix. Finally, test cases are updated.

The future research will be applied the presented approach to develop an actual software tool.

References

[1] J. Jainae and T. Suwannasart, "A tool for test case impact analysis of database schema changes using use cases," in *ICISA 2014 - 2014 5th International Conference on Information Science and Applications*, 2014.

[2] M. Raengkla and T. Suwannasart, "A test case selection from using use case description changes," in *Lecture Notes in Engineering and Computer Science*, 2013, pp. 507-510.

[3] S. Phetmanee and T. Suwannasart, "A tool for impact analysis of test cases based on changes of a web application," in *Lecture Notes in Engineering and Computer Science*, 2014, pp. 497-500.

[4] O. S. Inc. *Requirements Traceability Matrix*. Available: http://www.ofnisystems.com/services/validation/traceability-matrix/