A Tool for Test Case Impact Analysis From User Interface Changes in Android Mobile Application

Chatchawan Prongsang and Taratip Suwannasart

***Abstract*— Mobile applications have influence in our daily life. There are a lot of different mobile application development platforms and Android gains the highest market share. Hence, Android application development has grown up, and it actually has an effect on testing process. The problem of testing process is that some test cases cannot be used when a program is changed. This causes testers to spend more time, energy, and resources for testing the changes before testing the others. Consequently, test case impact analysis from user interface changes for applications are important. We have proposed a tool to analyze an impact on test cases when an user interface of Android applications is changed by comparing two versions of user interface files to analyze the changes, highlighting and correcting which test cases can be used and finally generating new test cases if necessary.**

***Index Terms*— test cases, impact analysis, user interface changes, mobile application, android**

I. Introduction

Mobile applications have influence in our daily life and there are a lot of mobile application platforms such as Android, iOS, Windows Phone, and BlackBerry. However, in year 2011-2014, Android gains the most market share with an over 50% share in two years and more than 75% over the next two years [1].

Previous researches about test case generation for several applications [2]-[4] presented test case creation from the user interface regardless to user interface changing which impact for old test cases. Although, the researchers studied on change impact analysis in several possibilities [5]-[7] but they have not covered user interface for Android mobile application, and they have not focused on change impact analysis of personal computer application or web application.

This paper presents a tool to examine the impact on test case from user interface changes for Android mobile applications by comparing two versions of XML user interface files, deleting the test cases when those test cases can be no longer used, and correcting test cases that can be used. In addition, new test cases will be created to replace the old ineffective test cases. In our research, we use boundary value analysis which is a black-box testing technique to assign input values for each test case.

Manuscript received December 22, 2015; revised January 15, 2016.

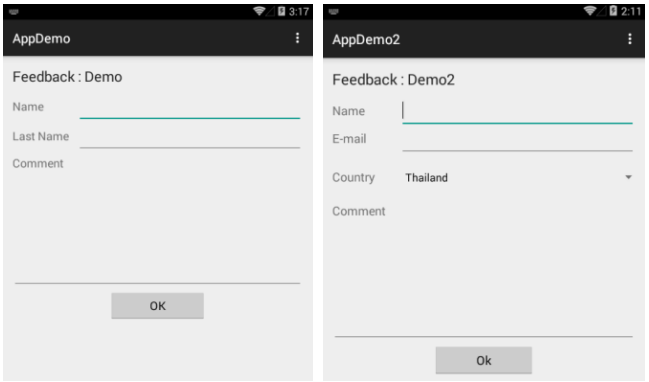
C. Prongsang and T. Suwannasart are with the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand e-mail:chatchawan.p@student.chula.ac.th, Taratip.S@chula.ac.th

II. Background

A. A User Interface of Android Mobile Application

A user interface of Android mobile application [8,9] comprises of 2 object tags which are ViewGroup and View. ViewGroup is user interface layout which stores widgets into group such as LinearLayout, RelativeLayout, and ScrollView. View is user interface widget which is a graphic format interacting with users such as Button, and EditText.

This research focuses on 9 widgets of View including Button, EditText, CheckBox, RadioButton, ToggleButton, Spinner, DataPicker, TimePicker, and NumberPicker. Fig. 1 shows an example of graphical user interfaces where (a) is the old version and (b) is the new version. The differences of two versions of GUI are as follow: <EditText> LastName is removed, <EditText> E-mail is inserted, <Spinner> Country is inserted, and label of <Button> OK is updated.



(a) Old Version (b) New Version

Fig. 1. An example of two versions of Graphical User Interface

An user interface is stored in XML document files which consists of user interface file and resource file. Fig. 2 shows the new version of XML user interface file and Fig. 3 shows the new version of XML resource file. A user interface file contains information about widgets, and a resource file contains information about boundary values and list values of widgets.

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

```
20    <LinearLayout
21        android:layout_width="match_parent"
22        android:layout_height="wrap_content"
23        android:layout_marginTop="15dp" >
24
25        <TextView
26            android:id="@+id/textView2"
27            android:layout_width="100dp"
28            android:layout_height="wrap_content"
29            android:text="Name"
30            android:textAppearance="?android:attr/textAppearanceMedium" />
31
32        <EditText
33            android:id="@+id/editText1"
34            android:layout_width="wrap_content"
35            android:layout_height="wrap_content"
36            android:layout_weight="1"
37            android:ems="10"
38            android:inputType="textPersonName" >
39            <requestFocus />
40        </EditText>
41    </LinearLayout>
```

Fig. 2. A XML user interface file

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <resources>
3
4       <string name="app_name">AppDemo</string>
5       <string name="hello_world">Hello world!</string>
6       <string name="action_settings">Settings</string>
7
8       <string-array name="spinner_array">
9           <item >Thailand</item>
10          <item >Japan</item>
11          <item >China</item>
12      </string-array>
13
14      <item name="editText1_min" type="integer" format="string">6</item>
15      <item name="editText1_max" type="integer" format="string">25</item>
16
17  </resources>
```

Fig. 3. An example of a XML resource file

B. *Test Cases*

In this research, test cases are generated by using Black-box testing technique. An example of test case is depicted in Table I.

TABLE I
AN EXAMPLE OF TEST CASE

Application		AppDemo		
Screen		activity_main		
TestCase ID		001	**Version**	1.0
Data				
Order	**Widget ID**	**Text Value**	**Input Data**	
1	editText1	Name	Mr.Hello	
2	editText2	Last Name	Wolrd	
3	editText3	Comment	Good Luck!	
4	Button	OK	Click	
Output		Show Toast		

C. *Impact Analysis*

Since a user interface has been changed, the change has an effect on XML document files and test cases. Thus, the affected test cases must be corrected and we concern with 4 main issues as follows:

1. How do changes affect an user interface file and a resource file?
2. How do we locate the affected positions on the user interface file and the resource file?
3. How do changes affect test cases?
4. Can we fix and update the test cases to be ready for use?

III. DEVELOPMENT OF THE PROPOSED TOOL

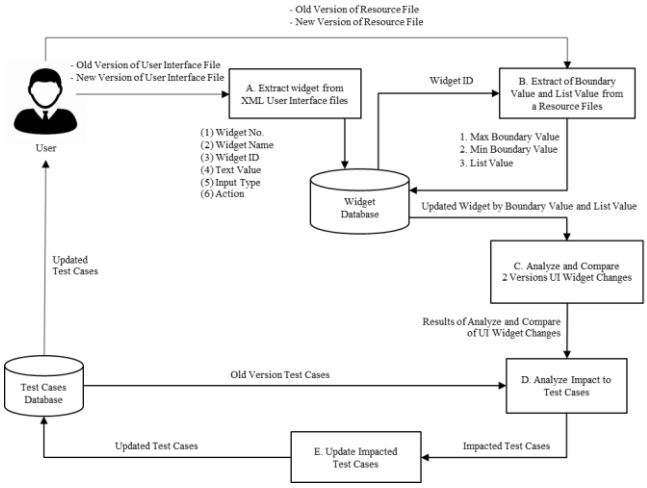


Fig. 4. The proposed tool structure

Fig. 4 presents the proposed tools structure for test cases impact analysis from user interface changes of an Android mobile application. There are five main steps in our approach starting from a user imports the old and the new version of user interface files in XML format then the tool extracts widgets from XML document files and keeps them into a database. Next, values from the resource file will be extracted.

The values include maximum, minimum, and list values for updating the widget attributes in the database. Afterwards, the tool will analyze and compare between 2 versions of the user interface, analyze the impact to test cases, and update the impacted test cases. Finally, the tool will provide updated test cases for testing the new version of user interface. The tool operations are described in the steps as follows.

A. *Widget Extraction from XML User Interface files*

With the imported user interface files of the old and the new version of user interface, the tool extracts attributes of both versions which include attributes below.

(1) Widget No: an order of widget
(2) Widget Name: a name of widget
(3) Widget ID: an id of widget
(4) Text Value: a text value of widget
(5) Input Type: an input data type of widget
(6) Action: an action value of widget

The extracted attributes are stored in the widget database. TABLE II shows an example of extracted attributes from <EditText> tag in line 32-40 of Fig. 2

TABLE II
AN EXAMPLE OF EXTRACTED WIDGET ATTRIBUTES

Attributed Extraction	**Valuable Extraction**
Widget No.	1
Widget Name	EditText
Widget ID	editText1
Text Value	-
Input Type	textPersonName
Action	-

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

B. Extraction of Boundary Value and List Values from Resource Files

The tool extracts properties from both resource files which include max boundary value, min boundary value, and list values. These properties are updated in the widget database by the matched Widget ID. The boundary values are extracted in 4 attributes which include

(1) Name: a name of boundary value as Widget ID
(2) Format: a data type
(3) Min Value: a minimum boundary value
(4) Max Value: a maximum boundary value

For example, table III shows extracted boundary values from <item name> tag in line 14–15 of Fig. 3. It presents editText1 which is string data type, max boundary value is 25, and min boundary value is 6.

TABLE III
AN EXAMPLE OF EXTRACTED BOUNDARY VALUE ATTRIBUTES FROM RESOURCE FILES

Attributed Extraction	Valuable Extraction
Name	editText1
Format	String
Min Value	6
Max Value	25

Table IV shows extracted list values whether there is <Spinner> tag in the user interface file. They are extracted from <string-array name> tag in line 8–12 of Fig. 3.

TABLE IV
AN EXAMPLE OF EXTRACTED LIST VALUE ATTRIBUTES FROM RESOURCE FILES

Attributed Extraction	Valuable Extraction
Name	spinner1
List	Thailand, Japan, China

After extracting resource files, the extracted attributes are updated in the widget database by the matched Widget ID field in the widget database to complete widget information.

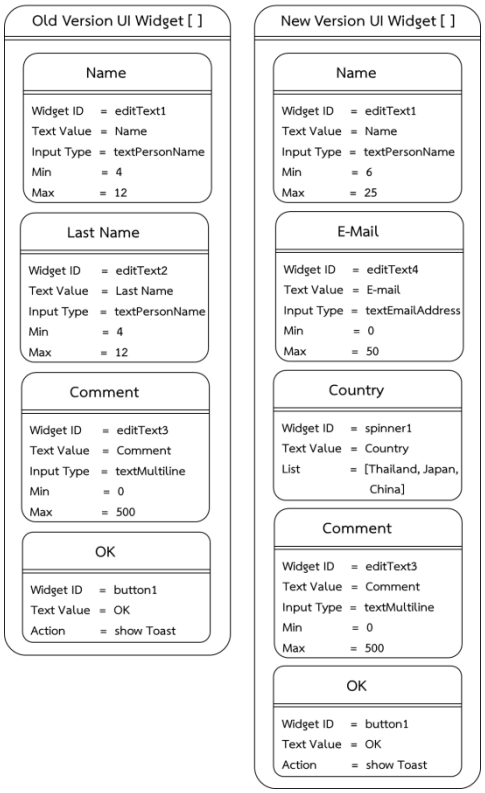
TABLE V
AN EXAMPLE OF COMPLETE WIDGET

Attribute	Attribute Value
Widget No.	1
Widget Name	EditText
Widget ID	editText1
Text Value	Name
Input Type	textPersonName
Action	-
Min Value	6
Max Value	25

C. User Interface Analysis and Comparison

The tool selected the complete widget information from the widget database to create widget list of the old and the new version of user interface as shown in Fig. 5 where (a) is widget list of the old version in Fig.1 (a) and (b) is widget list of the new version in Fig.1 (b)

After both widget lists have been created, the tool will analyze and compare change of both widget lists. Tool supports the changes with following cases (1) adding a new widget, (2) removing an existing widget, (3) changing widget order, and (4) editing widget data. The result of this step is shown in table VI



(a) Old version (b) New version

Fig. 5. A structure of two Versions UI Widget List

TABLE VI
A CONCLUSION OF CHANGES AND TEST CASES IMPACT ANALYSIS

Order	Widget ID	Text Value	Changes Description	Impacted Changes
1	editText1	Name	Updated minimum boundary value from 4 to 6 and updated maximum boundary value from 6 to 25 in new version	Updated test case
2	editText2	Last Name	Deleted widget in new version	Deleted and updated test case
3	editText3	Comment	Changed Widget Order from 2 to 4	Updated test case
4	button1	OK	Changed Widget Order from 4 to 5	Updated test case
5	editText4	E-Mail	Created and inserted new widget in new version in order 2	Created New test case, update test case
6	spinner1	Country	Created and inserted new widget in new version in order 3	Updated test case

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

D. Analyze Impact to Test Cases

The objective of this step is to analyze the affected test cases from the result from step C which has an effect on 4 types of changes described below

1) In case of adding a new widget, the test case is updated by adding test case order of the old test case but there is an exception for this widget that includes EditText, NumberPicker, CheckBox, and RadioButton. Because of adding EditText and Number Picker, the test tool must add a new test case based on their boundary value. By Adding of CheckBox and RadioButton, the tool must add a new test case to follow numbers of Checkbox and RadioButton.
2) In case of removing an existing widget, the test case is updated by removing test case order of the old test case but there is an exception for this widget that includes EditText, NumberPicker, CheckBox, and RadioButton. Because of removing EditText and Number Picker, the test tool must remove a test case based on their boundary value. By removing CheckBox and RadioButton, the tool must remove a test case to follow numbers of Checkbox and RadioButton.
3) In case of changing widget order, the tool will rearrange the Widget ID in each test case.
4) In case of editing widget data, the tool will analyze the changes as follows
   a. Editing boundary value, if the test data does not go along with a new boundary value, the test case will be updated by following a new boundary value. However, the test case will not be changed if the existing test data goes along with a new boundary value.
   b. Editing list values, if there is a change in list values, the tool will randomly generate list values for a new test case.
   c. Editing text value and action, a test case is updated by using a new text value and action value.
   d. Editing input type, if the input type of test case is updated, it needs to follow a new input type.

E. Update Impacted Test Cases

The results from the previous step are used to update test cases. A test case in table I is updated a new test case as shown in table VII according to the changes in table VI.

TABLE VII
An Example of Impacted Test Case

Application		AppDemo2		
Screen		activity_main		
TestCase ID		001	**Version**	2.0
Data				
Order	**Widget ID**	**Text Value**	**Input Data**	
1	editText1	Name	Mr.Hello	
2	editText4	E-Mail	Hello@mail.com	
3	spinner1	Country	Thailand	
4	editText3	Comment	Good Luck!	
5	Button	OK	Click	
Output		Show Toast		

IV. Conclusion

In this paper, we present an idea to create a tool for analyzing the effects to test cases when an Android user interface is changed.

This tool can analyze the changes by comparing the old and the new version of Android user interface, analyze impact to test cases and automatically update impacted test cases. Finally, the benefit of this tool is reduction of time and effort to rectify test cases which affect the changes.

References

[1] International Data Corporation. Smartphone OS Market Share, Q42014. Available from: http://www.idc.com/
[2] Surasak Phetmanee and Taratip Suwannasart. “A Tool for Impact Analysis of Test Cases based on Changes of a Web Application,” in, Proc. IMECS, 2014.
[3] R. Selvam and V. Karthikeyani. “Mobile Software Testing - Automated Test Case Design Strategies,” in, Proc. IJCSE, 2011, vol. 3, pp.1450-1461.
[4] P. Pocatilu and F. Alecu. “An UI Layout Files Analyzer for Test Data Generation,“ in Proc. IE, 2014, vol. 18, pp.53-61.
[5] Bohner S. A. “Software Change Impacts-An Evolving Perspective.” Software Maintenance, Proceedings International Conference on; 2002:263-72.
[6] Sprenkle S., Pollock L., Esquivel H., Hazelwood B., Ecott S., “Automated Oracle Comparators for Testing Web Application” International Symposium on Software Reliability Engineering; 2007:117-26.
[7] Yadav D., Sharma A. K., Gupta J. P., “Change Detection in Web Pages” Information Technology, (ICIT 2007) 10th International Conference on;2007:265-70.
[8] Android Open Source Project. Input Controls | Android Developers. Available from: http://developer.android.com/guide/topics/ui/controls.html
[9] Android Open Source Project. UI Overview | Android Developers. Available from: http://developer.android.com/guide/topics/ui/overview.html

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)