An Image Comparing-based

GUI Software Testing Automation System

Hyunjun Jung, Sukhoon Lee, Doo-Kwon Baik
Department of Computer and Radio Communications Engineering, Korea University, Seoul, Republic Of Korea
junghj85@gmail.com, leha82@korea.ac.kr, baikdk@korea.ac.kr

Abstract - This paper proposes an automated GUI (Graphical User Interface) testing method to support regression testing when a company requests new functionality additions or program changes during the development phase. In this paper, we propose an automated GUI testing method based on two processes. An event-driven testing method can test the target program directly. A capture and replay testing method can repeat recordings of the tester's actions. GUI verification is image comparing-based. To demonstrate the advantages of our proposed method, we implemented a prototype system based on C#.

Keywords: Capture & Replay test, Event-driven test, Image comparing-based test, Test automation, Test automation tool

1 Introduction

Quality assurance (QA) is very important in large scale software projects but manual testing cannot guarantee the quality of large scale programs. A company may request the addition of a new functionality or software changes during the development phase. Software changes can contain new types of errors. Thus, software changes demand repeated testing to ensure there are no errors. Software is needed for regression testing to meet the specified requirements [1]. The regression testing method uses previously executed test scenarios to check for the presence of errors. Weak version management can lead to problems with the loss of the previous bug fixes. Previously, fixing the software version solved the problem. However, temporary fixes of software often lead to the same problem when software is redesigned by refactoring. Thus, this method is not a fundamental solution. To solve the problem, software is locked up with the software version while a test scenario is created to find the bug. The test scenario should also be repeated when the software changes. QA testing proceeds by manual or automated software testing. However, this repeats a lot of regression testing when using automated testing tools, which can be expensive in terms of cost and efficiency [2]. However, quality management with automated test tools can facilitate the early detection of defects in modified software. If errors are detected before complete software development, this also has the advantage of reducing time and costs. Automated testing method include code-driven and GUI testing. Code-driven testing methods use a class, module, or library interface and they return results that confirm whether a variety of input arguments is satisfied. The GUI testing method uses a mouse click or mouse input to generate user interface events, such as changes in a program, to ensure that the program functions correctly while observing the results. In this study, we proposed event-driven testing and capture and replay testing for automated GUI testing. The GUI also includes image comparing-based validation, such as graphs and charts. The paper is organized as follows. Section 2 provides a brief review of previous automated GUI testing tools. Section 3 describes the automated GUI testing method. Section 4 presents an implementation of a prototype system. We conclude the paper and consider future work in Section 5.

2 Relate Work

Table 1 GUI automated testing tools

<table>
<thead>
<tr>
<th>Testing Tool</th>
<th>Feature</th>
<th>input</th>
<th>Report Function</th>
</tr>
</thead>
</table>
| abbot        | • Measured via a test script GUI state  
               • An interface for controlling the replay  
               • Event-based testing | Java Application | Coverag e Report |
| Guitar       | • Provide a test case generator plug-in  
               • Event flow measurement is useful | Java Application | Unsuppo rted |
| Pounder      | • Records test scripts and provides an interface for measuring the results | Java Application | Unsuppo rted |
| Selenium IDE| • Records the actions of the tester using HTML script | Web UI | Unsuppo rted |
Many studies have been conducted to support automated testing [3-11]. For example, Abbot [12] assists Java UI testing where test scripts are used to measure the state of the GUI, while event-based replay provides a control interface. GUITAR [13] is a Java and Microsoft Windows application that provides a GUI testing framework with a test case generator and event flow measurement. Pounder [14] records a test script that can be measured while the results are provided via an interface. Selenium IDE [15] records the behavior of a test script as HTML generated in the tests. Table 1 shows examples of automated GUI testing tools.

3 GUI testing automation methods

The automated GUI testing procedure is shown in Figure 1. The tester selects the appropriate method during the Method Selection phase. Special control testing is difficult, image-based buttons and user creation control are provided. In this case, the tester has selected the capture and replay method. The event-driven method sends the event directly to the target program, depending on whether scenario testing is performed by sending an event. In addition, a scenario can be created using the event to perform tests rapidly and without error. The exact value of the unit tested is checked before verifying whether an event-driven method is required.

Method selection during this stage of the target application development requires a unit test or verification of the exact value when it is necessary to initiate the event. Figure 2 shows the use of the event-driven method. The event-driven method can be used to send events directly to the target program to perform precision tests of the control value of the property involved in the event, by comparing the verification accurately. This would be available at the code level with other methods for scripting an event in a direct target program. An event-driven method is required to perform tests using the event name and to create a test script for the operation. Test scripts are written in scripting languages, while parsing the script will create the event. The event script is a list of events that are passed to the actual event. All events are dispatched to the target program, before ending the test.

During the capture and replay method selection step, as shown in Figure 3, the tester selects how the test is performed throughout the process. The test script is written by hand or previously written scripts are reused where available. This method is based on the coordinates of the target program features when performing a test of added or changed events. In this case, you the previous scenario can be reused. Test scenarios keyboard strokes and mouse movements can be used to automatically generate scripts by recording. Replaying the generated script reproduces the movement of the testers, which alters the program or input data so the test can be performed.

The capture and replay method does not communicate directly with the event in the target program. Targeted programs are recognized based on their coordinates. A comparison of the output image should verify the correct value. A comparison of images is shown in Figure 4 using the verification procedure. The tester extracts and generates a script to control the capture program, which is targeted directly in the middle of the verification process for features that require an output value from the GUI and the image.
The necessary features are implemented for each verification result before comparing the differences in the output between the images on the screen and the previous screen output, which creates an image that is recorded.

The image shown in Figure 5 is generated by comparing the images. Figure (a) is the previous screen output results on the GUI. Figure (b) shows a GUI screen with the results for the test output. Replaying the created image facilitates testing of the changed GUI and a change in the target program will be detected as an error. Figure (c) shows a comparison of the image before and after performing the image testing. The result of performing the functions on the replay screen during testing are compared with the extracted features to determine whether they perform correctly.

4 An Image Comparing-based Automated Software Testing System

4.1 Conceptual Model

Figure 6 shows the concept model of our event-driven method. The tester extracts information to perform a test scenario by parsing the XML-based test. The event generator produces an event script based on this information. A saved script file can be reused if necessary for future testing of the same units as those found in the target program. An event testing engine script is generated for events and passed to the target program to perform the test. Verifying the results of the control target program for events yields the value of the property.

The event-driven method can be verified directly to yield the property values of controls, which is advantageous for verifying the exact value. The XML tester must specify the expected results to verify the values directly. The test result is deduced from the tests performed by the function generator, including successful and failed test results.

Figure 7 shows a conceptual model of the capture and replay method. The capture engine generates test scripts according to the tester while entering the time for recording mouse and keyboard inputs automatically. The tester input is activated when the target program is generated. The capture engine uses Global Hooker global event analysis of the OS to generate the script. This function is necessary for the extracted image in the target program verification system to activate the image verification engine accelerator to deliver the results. Replayed functions during image performance are passed through a procedure to test the capacity to compare the output of the identified portions and to verify the test results that are based on it.
4.2 Test Case

Figure 8 shows an example implementation of a target program when applying the automated test tool. The path is typed in the target program to enumerate the files by selecting the check box, which can be deleted. Menus, list boxes, and edit boxes are common features in the GUI.

Events in the target program are tested using the event-driven Method, with the tester shown in Figure 9 to transfer the event to XML-based scripts. The root node of an XML-based script in the <test> center node of each test is divided into a function <step> to perform any function that indicates whether the value was passed as an argument.

Figure 10 shows the main screenshot of the event-driven method. In this stage, the tester has loaded XML-based scripting of the test step to select the requirements. The step chosen by the tester moves to the automated GUI test tool, which is passed through a targeted program of events. The tester includes a previously created XML-based script scenario for mouse and keyboard inputs. The automated GUI test tool passes sequentially through each step of the scenario at the rate set by the tester.

The capture and replay method is shown in Figure 11 where the number and speed of the repeat can be set. The capture start button is used to record the movements of the tester. When the capture the tester is finished, the Stop button is pressed to create a script and test the tool automatically when it is ready to Replay. Capture and replay is performed using a script, which can be reused.

```xml
<xml version="1.0" encoding="utf-8"> 
<test>
    <step>
        <comment>View->Trace Data</comment>
        <type>click</type>
        <content>Trace Data</content>
        <eventName>listViewToolStripMenuItem_Click</eventName>
    </step>
    <step>
        <comment>TreeView</comment>
        <type>text</type>
        <content>tree selection</content>
        <eventName>treeBox1</eventName>
    </step>
    <step>
        <comment>DataType Selection</comment>
        <type>user Define</type>
        <content>comboBox1</content>
        <eventName>comboBox1</eventName>
    </step>
    <step>
        <comment>Select Button</comment>
        <type>click</type>
        <content>button</content>
        <eventName>button2_Click</eventName>
    </step>
    .......
</test>
```

Figure 9 Example of a written script
5 Conclusions

This paper proposed an Automated GUI Testing Method to support regression testing. The GUI provides image comparing-based validation using graphs and charts. The automated test method uses two processes. The event-driven method generates an event that tests the target software. The capture and replay method records the action, which needs to be repeated. We developed the test method, which was defined for this prototype system. Using the proposed method, automated testing is possible without knowing the source code. Further comparisons of the proposed automated test method and other commercial tools are required. The test method also requires simpler scripting to improve the efficiency of the input method.

6 Acknowledgement

This study was supported by Second Brain Korea 21 Project and by the National IT Industry Promotion Agency (NIPA) under the program of Software Engineering Technologies Development. The corresponding author is Doo-Kwon Baik.

7 References

Selenium IDE. Web application testing tool, http://seleniumhq.org/projects/ide/