GUI Based Test Case Generation- A Review

Mrs. Aarti Chugh
Assistant Professor
Amity University, Haryana
achugh@ggn.amity.edu

Abstract:- Our society is becoming more and more dependent on software systems. The most widespread activities to increase the confidence in the correctness of software is testing. Today’s software system usually features GUI. GUI has become an important and accepted means of interacting with today’s software. The testing needs to be performed in a way that it meets its written specifications and to detect if application is working functionally correct. In this paper we describe different techniques used for test case generation for GUI.

General Terms
GUI testing, test case, manual testing, automated testing

INTRODUCTION

With growing complexity of web application, identifying web interfaces that can be used for testing such application has become increasingly challenging and today’s software system usually feature GUI. Testing process is a method of executing a program on set of test cases and comparing result with expected result.

Test cases derived from software artifacts such as specification and design. Testers go through the requirements document to understand specifications. After this, they start preparing test cases using test case templates. Test case templates should have details like test steps, data, expected result, actual result, pass/fail and comments.

GUI testing having difficulties like GUI test automation is difficult and this test automation is technology dependent and GUI test maintenance is hard and costly. GUI testing having advantages also like it is feasible, easy to conduct for non expert people, great test coverage and increased test speed, test efficiency, software quality.

1. RELATED WORK

There are different GUI testing techniques that had been purposed. Some of them are as follows:

1.1 Testing using finite state machine
Shehday et al [7] proposed a technique for automation of a limited portion of GUI testing. This GUI is first transformed in variable finite state machine model. Then convert this model to Finite State Machine, and then test cases are generated from FSM. This technique is feasible and provides coverage of all paths and all transitions.

1.2 Using a goal driven approach
Memon et al [8][1] purposed a technique which is based on plan generation. Its objective is to achieve predefined goals for a specified GUI. The goal act as input and as output, generate sequences of actions, which reach these goals. These sequences of actions are termed as test cases for GUI. It is not capable of handling huge number of states and it does not cater all events.

1.3 Model based testing
Dalal et al [9] purposed a technique that used to generate test cases from requirements and used data model to generate test cases. In this first generate data model as an intermediate representation from requirements and constraints of given GUI and from data model test cases generated. It is semi automated because tester has to verify
data model manually. It is most applicable to a system for which data model is sufficient to capture the systems behavior.

1.4 Generating test cases for GUI responsibilities using complete interaction sequence
While et al.[10] purposed a technique in which first of all a given GUI is transformed into different tasks and their complete interaction sequences (CIS). Then from these, for every CIS a reduced finite state machine model is created. Then test cases are generated by traversing the created reduced FSM model.

1.5 Finite state testing and analysis of GUI
Belli et al.[11] extended work of While with certain improvements like for effectiveness of generated test case, introduction of test coverage criteria, suggestion to cover all possible combination of node with edges for the complete coverage of nodes and edges. It detects more faults but still not considered as a cost effective because tool support is still required.

1.6 Automation of GUI testing using model driven approach
Vieira et al.[12] purposed a technique which aim is to improve effectiveness of testing and developing cost effective model based technique for GUI testing.

1.7 Dynamic path testing
Korel[13] purposed a technique which generate test cases by executing the program with different possible test case value. The test cases are prioritized according to the shorter path by applying random testing method.

1.8 Multiple test suite prioritation
Rotherma[14] purposed a technique which is used to prioritize the test cases from multiple test suites. The entire program is divided into number of test suites and these test suites contains multiple test cases. The test cases are prioritized according to weight and rank that are used for testing program.

1.9 Using GUI run time state as feedback
Xun Yuan and Atif M Menon[3], used GUI run time state as feedback for test case generation and feedback is obtained from the execution of a seed test suite on an Application Under Test. This feedback is used to generate additional test cases and test interaction between GUI events in multiple ways. This feedback mechanism is not automated.

1.10 Using covering array technique
Xun Yuan et al. [4], proposed a new automated technique for test case generation using covering arrays for GUI testing. Usually 2-way covering are used for testing. Because as number of events in a sequence increases, the size of test suite grows large, preventing from using sequences longer than 3 or 4. But certain defects are not detected using this coverage strength. Using this technique long test sequences are generated and it is systematically sampled at particular coverage strength. This technique having disadvantages are event partition and identifying constraints are done manually.

1.11 Dynamic adaptive automated test generation
Xun Yuan et al. [5], suggested algorithm to generate test suites with fewer infeasible test cases and higher event interaction coverage. Due to dynamic state based nature of GUIs, it is necessary and important to generate test cases based on feedback from the execution of tests. The disadvantages are event contexts are not incorporated and need coverage and test adequacy criteria to check how these impacts fault detection.

2. RESEARCH METHODOLOGY:
Basically the GUI testing methodology is described in the block diagram in Figure 1.

Figure 2: GUI Testing methodology

3. COMPARISION OF TECHNIQUES:
The comparison of GUI testing techniques are described in the table 1.
The GUI Testing techniques are evaluated on the basis of these parameters:

### Table 1: Comparison of GUI Testing Techniques

<table>
<thead>
<tr>
<th>Types of techniques</th>
<th>Input representation</th>
<th>Intermedi ate representation</th>
<th>Coverage criteria</th>
<th>Automation</th>
<th>Tool used</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFSM</td>
<td>Input, output, variable states</td>
<td>FSM model, VFSM model</td>
<td>All paths, all transitions</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>By goal driven approach</td>
<td>Source file of GUI</td>
<td>Hierarchical model</td>
<td>All transitions</td>
<td>Semi</td>
<td>PATHS</td>
</tr>
<tr>
<td>Model based testing in practice</td>
<td>Requireme nts &amp; constraints of GUI</td>
<td>Data model</td>
<td>Pairwise</td>
<td>Semi</td>
<td>AETG</td>
</tr>
<tr>
<td>Using complete interaction sequence</td>
<td>Identification of CIS</td>
<td>Reduced FSM</td>
<td>All paths, all transitions</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Finite state testing &amp; analyisi s</td>
<td>Identification of CIS &amp;FCIS</td>
<td>FSM model</td>
<td>All paths, all transitions</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>On the test case definitio n for GUI testing</td>
<td>Identification of single actions &amp;Output</td>
<td>Mealy machine</td>
<td>All paths, event transition</td>
<td>Semi</td>
<td>Win Runner</td>
</tr>
<tr>
<td>Using model driven approach</td>
<td>GUI into UML model</td>
<td>UML</td>
<td>Data coverage, graph coverage</td>
<td>Semi</td>
<td>UML</td>
</tr>
<tr>
<td>Cost effective model based techniqu e</td>
<td>Auto extraction of GUI model</td>
<td>GUI</td>
<td>All transitions</td>
<td>Full</td>
<td>No</td>
</tr>
<tr>
<td>Event flow model</td>
<td>Source file of GUI</td>
<td>Event flow model</td>
<td>All transitions, event interaction</td>
<td>Semi</td>
<td>GUITAR</td>
</tr>
<tr>
<td>Hierarchical predicate transiti on nets</td>
<td>Given GUI into six tuples</td>
<td>GUI model</td>
<td>All transitions</td>
<td>-</td>
<td>No</td>
</tr>
</tbody>
</table>

### A. Input representation of GUI under test

When a GUI is given for testing, each technique requires some information according to the mechanism of that technique. This parameter describes the type of information required by a technique as a first step [15].

### B. Intermediate representation

The information acquired from the GUI under test may be transformed into intermediate representation to make them clear and easy to generate test cases. This parameter helps to determine the test case generation mechanism for a technique [15].

### C. Coverage criteria

This parameter describes the type of coverage criteria adopted by a technique. Since in GUI testing we have to evaluate events, states and their relationships because we know that every action is associated with some events and every event generates a new state [15].

### D. Automation

This parameter describes whether the said technique is automatable or not. Automation is a key factor for any GUI test case generation technique[15].

### E. Tool support

This parameter describes whether the said technique has any tool support or not. It could be possible that a technique might be automatable but no tool support is available for each technique. Values are “yes” and “no”[15].

### 4. CONCLUSION

As GUI is the most popular human-computer interface in today’s software system. As GUI testing is the process of testing a product’s graphical user interface to ensure it meets its written specification. This is normally done through the use of a variety of test cases.

In this paper, we have studied different GUI test case generation techniques. All techniques have their own merits and demerits. Among all of these techniques Model-Driven Approach technique is most convenient technique due to the reason that it is the faster than other techniques, it is cost effective and less error-prone and Cost Effective Model based technique is also good because it is fully automated technique. Techniques like GUI Run Time State as Feedback and Covering Array Technique are not so good in these techniques most of the work is done manually.
REFERENCES


[10] L.While and H.Aimezen(2005),"Generating test cases for GUI responsibilities using complete interaction sequences,” in Proc. of the 21st IEEE Int. Conf. on Software Maintenance(ICSM)


