IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 4 Issue 11, November 2017
ISSN (Online) 2348 – 7968 | Impact Factor (2016) – 5.264
www.ijiset.com

Event Coverage Criteria in GUI Interaction Testing

Kanchan Gautam , Om Prakash Pal

Department of Computer Science and Engineering, Bharat Institute of Technology, Dr. Abdul Kalam Technical University, Meerut, India

Department of Computer Science and Engineering ,Bharat Institute of Technology, Dr. Abdul Kalam Technical University, Meerut, India

Abstract- The importance of GUI tests lies is that they are performed from view of end user of product. Therefore goal of GUI testing should be enhanced fault detection rate, path coverage . In this paper, we develop a family of coverage criteria for GUI testing grounded in combinatorial interaction testing. The motivation of using combinatorial techniques is that they enable us to "context" into the criteria in terms of event combinations, sequence length, and by including all possible positions for each event. Our criteria range in both efficiency and effectiveness.

Index terms- GUI Testing, modal based testing ,combinational interaction testing.

INTRODUCTION

Graphical User Interfaces have become very important and significant in the software engineering. GUIs are the interacting points between users and programs.GUI testing is not a single testing rather it is an activity in which we test GUI from different which includes test coverage, test case generation, test oracle .The testing has been done in the such a way that it ensures tested GUI is error-free, we should select such test cases that are capable to capture the errors if they exist. Each technique has it limitations.

One method of modeling a GUI for testing creates a representation of events within windows called an event-flow-graph (EFG).It encodes all possible execution paths in a program, an EFG represents all the possible sequences of the events that can be executed on the GUI. Coverage criteria based on the EFG have provided coverage of a GUI's event space for functional correctness . However, these criteria cover sequences of events bounded by a specific length, which are the essentially sub-paths through an EFG. Since there are a large number of events that do not interact with the GUI application, such as those responsible for opening and closing windows, a refinement of the EFG was developed called an event-interaction graph (EIG). In an EIG events.are interact with the application, called system interaction events .We refer to the testing only these events as GUI interaction testing.

GUI TESTING

Model-based techniques have been used to automate aspects of GUI testing .The GUI testing has focused on the graph models to minimize manual work. The most successful graph models that have been used

for GUI test-case generation include EFGs and EIGs . The nodes in these graphs represent GUI events; edges represent different types of relationships between pairs of events.An EFG models all of possible event sequences that may be executed on a GUI. It is a directed graph that contains nodes and edges that represent a relationship between the events. An edge from node nx to node ny means that the event represented by ny performed immediately after the event represented by nx. This relationship is called follows.

The EFG is represented by two sets: (1) a set of nodes N representing events in the GUI and (2) a set E of ordered pairs(ex, ey) representing the directed edges in the EFG. EIG nodes, on the other hand, do not represent events to open or close menus, or open windows. The result is a more compact, and hence more efficient. Figure 1 presents a GUI that consists of the four events, Cut,Copy, Paste, and Edit. Figure 1(b) shows the GUI's EFG; the four nodes represent the four events; the edges represent the follows relationships. For example, in this EFG, the event Copy follows Edit, represented by a directed edge from the node labeled edit to Copy.

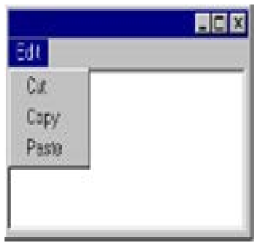

Fig1 (a) GUI

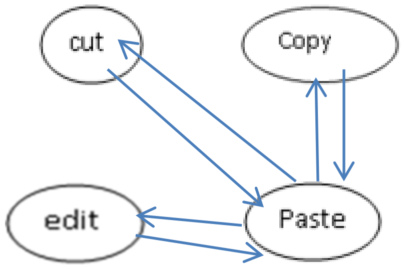

Fig1(b) EFG of GUI

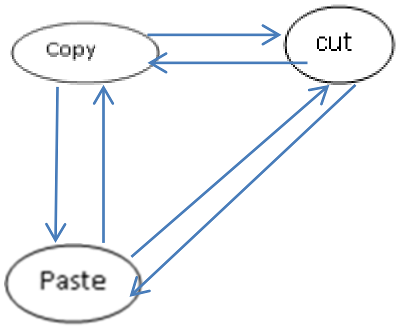

Fig1(c) EIG of GUI

Figure 1(c) shows the corresponding EIG. Note that the EIG does not contain the Edit event. EIG was to (1) delete Edit because it is a menu-open event. A technique to generate test cases, eachcorresponding to an EIG edge has been developed; these test cases are called smoke tests. Two examples of such length two smoke test cases for our example of Figure 1(c) are< Copy,Cut> and <Cut,Paste>.There are a total of nine such tests – one for each EIG edge, because EIG nodes do not represent events to open/close menues or open windows.

Combinatorial Interaction Testing

The combinatorial interaction testing is a covering array. A covering array (written as CA(N; t, k, v)) is an N×k array on v symbols with the property that of every N × t subarray contains all ordered subsets of size t of the v symbols at least once . In the other words, any subset of t-columns of this array will contain all t-combinations of the symbols. We use this definition of a covering array to the define the GUI event sequences.Covering arrays have been used to test input parameters of programs as well as to thetest system configurations. Other uses of covering array sampling, such as testing software product and databases .

Covering Array: (9, 2, 4, 3)

Events Name: (cut, copy, paste)

cut	cut	Cut	Cut
cut	paste	Paste	Copy
cut	copy	Copy	Paste
copy	cut	Paste	Paste
copy	copy	Cut	Copy
copy	paste	Copy	Cut
paste	cut	Copy	Copy
paste	paste	Cut	Paste
paste	copy	Paste	Cut

Fig2 (a). covering array

Cut	Cut
Cut	Copy
Cut	Paste
Copy	Cut
Copy	Copy
Copy	Paste
paste	Cut
paste	Copy
paste	Paste

Fig2 (b). Smoke Tests t=2

cut	cut	Cut
cut	copy	Paste
cut	paste	Copy
copy	cut	Paste
copy	copy	Cut
copy	paste	Copy
paste	cut	Copy
paste	copy	Cut
paste	paste	Paste

Fig2(c) Smoke Tests t=3

EVENT COVERAGE TEST ADEQUACY

This represents event coverage adequacy criteria created to capture the interaction coverage.We begin by defining event-tuples and event positions in a sequence. Assume we have a set of events E, and each event can occupy any location p in the sequence S of length k. In first definition does not assume a specific position within a sequence of the events, but rather defines a combination of events that occur in order somewhere within the sequence.

Definition: An event-t-tuple (ei, ej, . . . , et) is an ordered tuple of size t of events from E. A set of events E gives rise to |E|t event-t-tuples, i.e., all possible permutations of events.

Definition: A test suite is t-cover adequate if it executes all possible event-t-tuples (all possible permutations of events) in the form of an event-consecutive-t-sequence at least once.

Definition: An event-non-consecutive is an event-t-sequence <(ei,p1), (ej,p2),…….(en, pt)>,where p1<p2<….<pt, such that at the least one interval (p2-p1),…..(pt - pt-1) is greater than 1.

Definition: A test suite is t+-cover adequate if it executes all possible event-t-tuples in the form of an event-nonconsecutive-t-sequence at least once. Adequacy is zero when t==k.

Definition: A test suite is t*-cover adequate if it is both t-cover adequate and t+- cover adequate for a common set of events E.

PROCEDURE

1. The first step to creates the system interaction event set (SIES) to be used as the basis for the covering array.

2. The second step generates the test cases

3. In third step takes the output of the covering arrays and converts them into executable tests.

4 Our tests, detect faults and capture coverage.

5. The last step analyzes our results for the various coverage criteria .

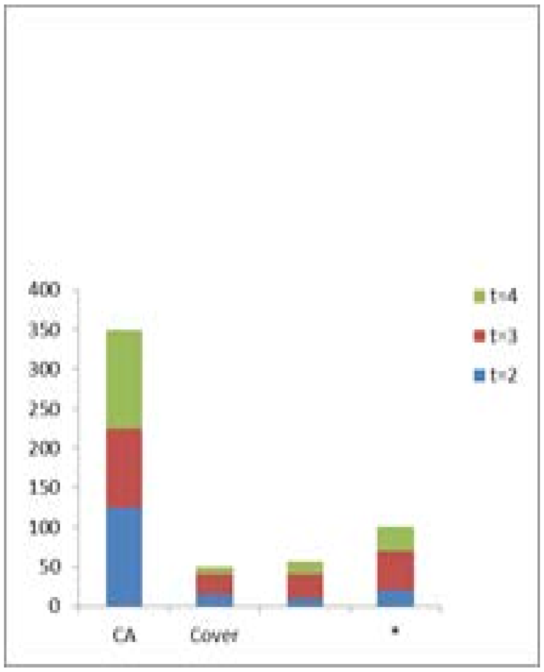

Fig3 Analysis of Fault detected by CA, Cover, t +, t*.

CONCLUSION AND FUTURE WORK

It present a test adequacy criteria for GUI testing. The criteria are unique in that they allow for “context” in GUI test cases in terms of event combination strength, sequence length, and all possible positions for each event. We abstracted our event space into a new model, a system interaction event set, that does not have strict ordering constraints, so that the we could leverage ideas from CIT for sampling.

We are currently examining techniques to generate test cases that can systematically improve coverage, as specified by our new criteria; in this we are used subsets of larger test suites rather than directly generated the test cases. The covering array based test case generation approach provided a good starting point, but unexecutable parts of test cases suggest that our approach needs to be augmented.

In covering array the no. of rows were fixed to length 9. In the future, we will vary this length and study the impact of the test-case length on faults and coverage. In future also increase the no. of events (v) of GUI and also increase the value of t i.e. event–t-tuples , study the impact of this on fault and coverage.

REFERENCES

[1] Imran Ali Qureshi , Aamer Nadeem,” GUI Testing Techniques: A Survey,” International Journal of Future Computer and Communication, Vol. 2, No. 2, April 2013

[2] Atif M.Memon, Sree Sampath,” Developing a single model and test priorization strategies for event driven software” ,IEEE Transactions on Software Engineering, Vol.X., 2010.

[3] Lee White and Husain Almezan,” Generating Test Cases for GUI Responsibilities using Complete Interaction Sequences”.

[4] Atif Memon,” GUI Testing: Pitfalls & Process, software technologies”,2002.

[5] Atif Memon, Martha E.Pollack,” Using a goal driven approach to generate test cases for GUIs.”

[6] Mario Brcic and Damir Kalpic,” Combinatorial testing in software projects”.

[7]Cemal Yilmaz ,Adam Porter,” Moving Forward with Combinatorial Interaction Testing”.