GUI Testing Techniques: A Survey

Imran Ali Qureshi and Aamer Nadeem

***Abstract*—This paper presents a survey of existing GUI test case generation techniques; we have thoroughly reviewed the existing literature, identified *Analysis* parameters and provided a comparative *Analysis* of all the existing techniques. We have classified the GUI test case generation techniques on the basis of Fault Models which is a novel work in the context of GUI testing.**

Our Analysis shows a clear picture about the usefulness of each technique. Classification of the techniques according to the fault model expected to help determine which technique is most suitable to capture which type of faults.

***Index Terms*—GUI, testing, techniques, classification, fault model.**

I. Introduction

Graphical User Interfaces (GUIs) have become very important and significant in the software engineering discipline. GUIs are the interacting points between users and programs. Sophisticated and complex systems often possess sophisticated and complex GUIs and to ensure the correctness of these, GUI testing is necessary.

GUI testing is not a single way testing rather it is a set of activities, which, as a whole confirm us that GUI testing, has been done successfully. In other words, it is an activity in which we test GUI from different perspectives, which includes test coverage, test case generation, test oracle and regression testing. Among these, the test case generation is the focal point. For this reason to ensure that testing has been done in such a way that it ensures tested GUI is error-free, we should select such test cases that are enough capable to capture the errors if they exist. For this purpose, different test case generation techniques have been proposed in the literature. However, each technique has its limitations.

Different researchers have proposed techniques keeping in view the different aspects and features of a GUI. Eventual objective of every technique is to generate GUI test cases which are capable to detect maximum faults.

Harris has classified the different faults into fault models [1]. We have classified the available GUI test case generation techniques according to the fault models which is a novel work in this area.

II. Evaluation Criteria

Since various approaches have been proposed for GUI test case generation and each has its own advantages and disadvantages so it was necessary to evaluate the techniques with such parameters which not only highlight the weaknesses and strengths of the techniques but also give us a picture about the effectiveness of the approach. For this purpose, following parameters were identified.

All the surveyed techniques were evaluated on the basis of these parameters:

A. Input Representation of GUI Under Test

Submit your manuscript electronically for review.

This parameter describes how the GUI under test is represented as an input. 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.

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 precisely explains, what attributes are required as an input to generate GUI test cases. This parameter helps to determine the test case generation mechanism for a technique.

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. In this parameter, we specify the coverage criteria.

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. A technique is fully automated if from start to end process of the test case generation there is no human involvement. Semi-automated means some human involvement is required for test case generation. A value of "No" for this parameter means that the technique is not automatable at all.

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 such technique. Values are "Yes" and "No".

F. Case Study

This parameter describes whether technique has been practically evaluated or not. Possible values are "Yes" and "No".

G. Fault Model

Fault model is an abstraction of types of faults. This parameter describes the fault model in which a technique is placed. Following fault model have been used in GUI perspective: SMFM (State Machine Fault Model), CDFM

Manuscript received August 6, 2012; revised September 20, 2012.

The authors are with Center for Software Dependability, Mohammad Ali Jinnah University, (MAJU) and Islamabad, Pakistan (e-mail: i4imran@mail.com, anadeem@jinnah.edu.pk).

DOI: 10.7763/IJFCC.2013.V2.139

(Control Flow Fault Model), TFM (Textual Fault Model) and IFM (Interface Fault Model) [1].

H. Fault Injection

This parameter describes whether fault injection/ seeding has been used in a technique or not. Its values are "Yes" and "No".

III. Test Case Generation Techniques for GUIs

We have conducted a thorough survey to explore and find out existing test case generation techniques. In the following paragraphs, we explained different test case generation techniques for GUIs.

A. A Method to Automate User Interface Testing Using Variable Finite State Machines (Shehday et. al., 1997)

Shehday *et al.* proposed a technique for the automation of a limited portion of GUI testing [2]. In this technique a given GUI is first transformed in Variable Finite State Machine (VFSM) Model. The objective to used VFSM is to reduce the complexity of modeling GUIs. Then this model is converted into an equivalent FSM, and then test cases are generated from FSM. Implementation showed that technique was feasible because it revealed such errors which were not identified through the conventional debugging process. In current scenario this technique could be consider as an early work.

Analysis this technique provides coverage for all-paths and all-transitions, it is an automatable but no tool support is provided. Case study showed that no fault injection is used. This technique is *classified in SMFM and CDFM fault models [1]*.

B. Using a Goal-Driven Approach to Generate Test Cases for GUIs (Memon et. al., 1999)

Memon *et al.* proposed a technique which is primarily based on Plan Generation [3],[4],[5],[6]. Basic objective of this technique is to achieve the predefined goals for a specific GUI. These goals act as input in this technique and as output, generate sequences of actions, which reach these goals. These sequences of actions are termed as test cases for GUIs [3],[4]. This technique is said to be test designer oriented rather than user oriented. Author argues that a test designer has a good perspective about GUI goals and events as compared to a tester (user) who simply follows a sequence of events to get the desired goal.

Analysis According to the presented technique, source file of GUI under test is given as input, which is transformed into hierarchical model, that is an intermediate representation; then an initial state, a goal state, a set of operators, and a set of objects are given as input, and a planner returns a set of steps to achieve the goal state. On the basis of this, a test case generation system by the name of Planning Assisted Tester for grapHical user interface Systems (PATHS) is proposed [5]. It is semi-automated technique because in second phase test designer has to give certain input values. PATHS creates hierarchical model as seen in the case study. This technique is not capable of handling huge number of states and also it does not cater all events [7],[8],[9] whereas it provides transition coverage. Author has not used any fault injection mechanism. This technique is classified in SMFM fault model [1].

C. Model-Based Testing in Practice (Dalal et. al., 1999)

Dalal *et al.* proposed technique is used to generate test cases from requirements [10]. The generated test case suite includes inputs, expected outputs and necessary infrastructure to execute the tests automatically. Basically this approach uses data model to generate test cases. A model is a specification of the inputs, so it can be developed at early stage from the requirements. This techniques primarily depends on three key factors; notation used for data model, test-generation algorithm, and tools that generate supporting infrastructure for the tests (including expected outputs). First two are portable.

Analysis this technique first of all generates a data model as an intermediate representation from requirements and constraints of given GUI using ATEGSpec notation and then from data model test cases are generated. This technique provides Pairwise-interaction coverage, event-interaction coverage. It is semi-automated because a tester has to verify data model manually. After verification ATEG Software generates the test cases. The most difficult aspect of this approach is, from requirement it can not be predict what could be the constraints. This technique is most applicable to a system for which data model is sufficient to capture the system's behavior [10]. The small case study proposed for GUIs showed that no fault injection is used. This technique revealed such faults which were not detected through conventional testing. This technique is classified in CDFM and IFM fault models [1].

D. Generating Test Cases for GUI Responsibilities Using Complete Interaction Sequences (White et. al., 2000)

White *et al.* proposed a technique, in which first of all a given GUI is transformed into different tasks and their complete interaction sequences (CIS) [11]. Then from these, for every CIS, a reduced finite state machine (FSM) Model is created. Then test cases are generated by traversing the created reduced FSM Model.

Analysis this technique mainly relies on the tester, because it requires substantial amount of work from tester. Case study showed that this technique has detected faults but no fault injection is used. To minimize number of test cases, author suggested that components of CIS should be reduced. This approach provides coverage of all-paths and event-interaction. No tool support is provided. This technique is classified in SMFM and CDFM fault models [1].

E. Finite State Testing and Analysis of Graphical User Interfaces (Belli, 2001)

Belli [12] has extended the work of While *et al.* [11] with certain improvements, like for effectiveness of generated test cases, introduction of test coverage criteria, suggestion to cover all possible combinations of node with edges for the complete coverage of nodes and edges.

Analysis Technique provided coverage for all-IPs (Interaction Pairs), all-FIPs (Faulty interaction pairs) of CIS and FCIS respectively along with all-paths and event-interaction. Case study showed that technique has detected more faults after improvement, but still it can not be considered as a cost effective because tool support is still required. No fault injection is used. This technique is classified in SMFM and CDFM fault models [1].

TABLE I: SHOWING COMPARISON OF GUI TEST CASE GENERATION TECHNIQUES

		Evaluation Parameters							
		Input representation of GUI under Test	*Intermediate representation*	*Coverage Criteria*	*Automation*	*I. TOOL SUPPORT*	*Case Study*	*Fault Model*	*Fault Injection*
1.	R. Shehady *et. al* (1997)	Input, Output, Variables,States	VFSM Model, FSM Model which is used by Wp Method	All-paths, All-transitions	Yes	No	Yes	SMFM CDFM	No
2.	A.M.Memon *et. al.* (99, 00, 01)	Source file of GUI	Hierarchical Model Scenario (Initial State, Goal State, Operators, Objects)	All-transitions	Semi	PATHS	Yes	SMFM	No
3.	S. R. Dalal *et. al.* (1999)	Req. & constraints of GUI	Data Model (Specification of inputs for AETGSpec)	Pairwise-interaction event-interaction	Semi	AETG	Yes	CDFM IFM	No
4.	L. White and H. Almezen (2000)	Identification of CIS	Reduced FSM (Trace all distinct paths)	All-paths, event-interaction	No	No	Yes	SMFM CDFM	No
5.	Belli (2001)	Identification of CIS & FCIS	FSM Model (Trace all valid & invalid paths)	All-paths, event-interaction All-IPs, All-FIPs	No	No	Yes	SMFM CDFM	No
6.	Kai-Yuan Cai *et. al.* (2005)	Identification of single actions and their outputs	Mealy machine/ FSM Model(Convert into quintuple form) 1.States, 2.Actions,3.Outputs,4.State transition function, 5. Output function	All-path, All-transitions; Event-interaction	Semi	Win Runner	Yes	SMFM CDFM IFM	Yes
7.	M. Vieira *et. al.* (2006)	GUI into UML Model (category-partition method)	UML Model(Use case, Class and Activity diagrams) Class (category data) and activity diagrams	Graph Coverage, Data Coverage	Semi	TDE/ UML	Yes	CDFM	No
8.	Qing Xie (2006)	Auto-extraction of GUI Model	GUI Model (extract all the widgets, properties and their values)	All- transitions, Event-interaction	Full	No	No	SMFM IFM	Yes
9.	A. M. Memon *et. al.* (2007)	Source file of GUI	Event-flow Model (extract all the widgets, properties and their values)	All-transitions, Event-interaction Statement, branch	Semi	GUITAR	Yes	TFM CDFM SMFM	Yes
10.	H. Reza *et. al.* (2007)	Given GUI into six tuple HPrTNs	GUI Model using HPrTNS(Create set of paths & transition execution seq)	all-transitions all-threads, all-states	-	No	No	SMFM CDFM	No
11.	I. Alsmadi and K. Magel(06,07)	Source file of GUI	XML Tree Model (XML file containing all GUI controls and their selected properties)	All-path, All-transitions, branch;	Full	GUI Auto	Yes	SMFM CDFM	No
12.	X. Yuan, A. M. Memon (2007)	Auto extraction of EIG Model	EIG Model	All- transitions, Event-interaction	Full	No	Yes	SMFM IFM	No

F. On the Test Case Definition for GUI Testing (Cai et al., 2005)

Cai *et al.* proposed an approach that defines GUI test cases as a sequence of primitive GUI actions and treats GUI test suites as an inner hierarchy of formal languages [13]. In this technique the test case generation and execution process is sequential i.e. one by one, and treats GUI Test cases as a sequence of primitive GUI actions and GUI test suites as hierarchical language. In order to investigate how test cases should be defined for GUI testing, Mealy machines are used here.

Analysis this technique covers all-paths, all-transition and event-interaction coverage criterion. This technique provides a systematic way for GUI test case generation. It is a semi-automated supported by WinRunner tool. Case study showed that this technique is more effective in fault detection as compare to random. Fault injection is also used. This technique is classified in SMFM, CDFM and IFM fault models [1].

G. Automation of GUI Testing using a Model-driven Approach (Vieira et al., 2006)

Vieira *et al.* have proposed an approach that follows category-partition method to generate UML Model [14]. Authors said that this approach is the only approach that generates test cases on activity diagram control flow basis. Major aim of this technique is to improve the effectiveness of testing. Authors said that effectiveness is directly related to the completeness, consistency and accuracy of supplied information and tester experience.

Analysis this technique first transformed the given GUI into a UML Model (use case, class and activity diagrams), which is an intermediate representation. This technique provided Graph Coverage which includes Round Trip criterions, Happy Path criterion, All-Paths criterion and All-Activities criterion and Data Coverage which includes Sampling, Group coverage, Expression, Choice coverage and Exhaustive coverage. It is a semi-automated. In this technique authors have provided their own tool, Test Development Environment using UML (TDE/UML) for test case generation. No fault injection is used. This technique is classified in CDFM fault model [1].

H. Developing Cost-Effective Model-Based Techniques for GUI Testing (Xie, 2006)

Xie proposed a technique whose primary goal is to provides the best combination of fault detection effectiveness and cost [15]. Author proposed a framework, which consists of a model of the GUI, and modules that employ this model to perform various GUI testing tasks. The GUI model is obtained using a GUI analyzer that automatically reverse engineers the GUI and extracts all the widgets, properties and their values. Given the GUI model as input, a test case generator automatically generates focused test cases. The generated test cases are provided to a test oracle generator, which derives the expected output of the GUI for each test case.

Analysis According to the author, work done in-house has proof that this technique is feasible, scalable and cost effective, but neither case study nor any tool support is provided by the author. Author also claims that the technique, if automated, is capable of detecting a large number of faults. This technique provides event-interaction coverage and all-states coverage. Fault injection can also be used. This technique is classified in SMFM and IFM fault models [1].

I. An Event-flow Model for GUI-Based Applications for Testing (Memon, 2004, 2007)

Memon proposed a technique that consolidates different

models into one scalable event-flow model and outlined algorithms to semi-automatically reverse engineer the model from an implementation [16],[17]. This technique forms model by defining event-space exploration strategies (ESES), which addressed the problem of reusability of test cases, effectiveness of test cases for detecting GUI faults, scalability of test cases in the form of reduction of test case suites.

Analysis this technique transforms the given GUI into event-flow model by using GUI Ripper, which is a component of GUITAR tool. This technique is a semi-automated author has provided a tool, i.e., GUITAR. Case study showed that this technique has detected faults. Event-flow model can be edit to minimize the generation of test cases. This technique provides all-transition and event-interaction coverage. Fault injection is used in the technique. This technique is classified in TFM, SMFM and CDFM fault models [1].

J. A Model-Based Approach for Testing GUI Using Hierarchical Predicate Transition Nets (Reza et al., 2007)

Reza *et al.* proposed a model based testing method to test the structural representation of GUIs specified in high class of Petri nets known as Hierarchical Predicate Transitions Nets (HPrTNs) [18]. Main reason for using HPrTNs to test GUI is, HPrTNs recognize and treat both Events (desirable and undesirable behaviors) and states (desirable and undesirable conditions) equally. Authors have also proposed a new coverage criteria for GUI testing but they have not proved the feasibility of their approach neither they have done any implementation.

Analysis this technique provides all-states, all-transitions, all-threads coverage. Authors have not given any tool support neither they have stated that their technique is automated if so then upto what extend, no case study is provided. No fault injection is used. This technique is classified in SMFM and CDFM fault models [1].

K. Generating Test Cases from The GUI Model (Alsmadi et. al., 2006,2007)

Alsmadi *et al.* proposed a technique that first transforms the GUI from implementation to a XML tree model, Test cases are then generated from XML model which makes it easier to automate generating test cases and executing them [5], [6]. The proposed technique has four algorithms that do not need any user involvement and generates test cases from different perspective, which is an edge on other techniques. Through this approach automation of first few test cases is expensive, but cheaper for generating large test cases. GUI state reduction is the key contribution of this research. Tool that is developed by the authors have a component GUI modeler whose purpose is to transform the GUI into model that becomes easier for testing using an automated tool.

Analysis this technique is fully automated and provides coverage of all-paths, all-transitions and branch. Authors have provided the case study and claimed that their technique is cost-effective, scalable and effective in fault detection. They have developed their own tool, i.e., GUI Auto, which generates the test cases independently and allow the user in a later stage to define pre- and post-conditions for the verification process. No fault injection is used. This technique is classified in SMFM and CDFM fault models [1].

L. Using GUI Run-Time State as Feedback to Generate Test Cases (Yuan et. al., 2007)

Yuan *et al.* in this technique have introduced a mechanism how to generate test cases from the feedback of the executing application under test (AUT) [19]. This technique is used for the automated GUI testing. In this technique first of all given GUI is automatically converted into an Event-Interaction Graph (EIG) Model, then test cases are generated from this EIG Model. This technique is also useful for testing GUI events-interaction with possible multi-ways interaction.

Analysis It is a fully automated technique but authors have not specifically mentioned the tool name. This technique provides all-states and event-interaction coverage. Case study showed that feedback technique was helpful in detecting various types of faults. This technique needs to be reviewed to make it cost-effective and more effective in fault detection. No fault injection is used. This technique is classified in SMFM and IFM fault models [1].

IV. Evaluation of Test Case Generation Techniques for GUIs

We have analyzed the existing techniques with the help of eight identified evaluation parameters and showed in Table 1. We have thoroughly investigated how each technique generates test cases for GUI, which type of coverage it provides, we have also explored either the process of the said technique is manual or automated, and if automated then any tool support is provided by the author, whether the technique is practically evaluated, in which type of fault model this technique is classified, and fault injection is used by this technique or not. Our analysis showed a clear picture about the functionality of each technique. Classification of surveyed techniques according to the fault model is an important attempt which, help us to determine which technique is most suitable to capture which type of faults.

V. Conclusion

We have presented a survey of existing GUI test case generation techniques; we have thoroughly examined the existing GUI test case generation techniques and given a brief analysis for each technique. For a comprehensive and detailed analysis, we identified various analysis parameters and with the help of these parameters, we compared all the techniques.

We have classified the GUI test case generation techniques on the basis of fault models.

Our analysis shows a clear picture about usefulness of each technique. Classification of surveyed techniques according to the fault model is an important attempt which helps in determining the suitability of a technique to capture certain types of faults.

References

[1] I. G. Harris, "Fault Models and Test Generation for Hardware-Software Covalidation," *IEEE Design and Test*, vol. 20, no. 4, pp. 40-47, January 2003.

[2] R. Shehady and D. Siewiorek, "A Method to Automate User Interface Testing Using Variable Finite State Machines," in *Proc. of the 27th International Symposium on Fault Tolerant Computing (FTCS-27)*, pp 80-88, June 24-27, 1997, *Seattle, WA, USA.*

[3] A. M. Memon, M. E. Pollack, and M. L. Soffa, "Using a Goal-Driven Approach to Generate Test Cases for GUIs," in *Proceedings of the 21st International Conference on Software Engineering (ICSE'99),* pp. 257-266, May 16-22, 1999, Los Angeles, USA.

[4] A. M. Memon, M. E. Pollack, and M. L. Soffa, "Plan Generation for GUI Testing," in *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems(AIPS'00)*, pp. 226-235, April 14-17, 2000, Breckenridge, CO, USA.

[5] A. M. Memon, M. E. Pollack, and M. L. Soffa, "Hierarchical GUI Test case generation using automated planning," *IEEE Trans. on Soft. Eng. (TSE)*, vol. 27, no. 2, pp. 144-155, February 2001.

[6] A. M. Memon, M. E. Pollack, and M. L. Soffa. "A Planning-based Approach to GUI Testing," in *Proceedings of the 13th International Software / Internet Quality Week (QW'00)*, May 30-June 2, 2000, San Francisco, California, USA.

[7] I. Alsmadi and K. Magel, "Generating Test Cases from The GUI Model," *World Scientific and Engineering Academy and Society* (WSEAS), 2006.

[8] I. Alsmadi and K. Magel, "An Object Oriented Framework for User Interface Test Automation," in *Proceedings of Midwest Instruction and Computing Symposium (MICS'07)*, April 20-21, 2007, *Alerus Center, Grand Forks, North Dakota, USA.*

[9] M. U. Hayat and N. Qadeer, "Intra Component GUI Test Case Generation Technique," in *Proc. of the Int. Conf. on Information and Emerging Tech. (ICIET'07), July 6-7, 2007, Karachi, Pakistan.*

[10] S. R. Dalal, A. Jain, N. Karuanith *et al.,* "Model-Based Testing in Practice" in *Proc. of the 21st Int. Conf. on Software Engineering (ICSE'99),* pp. 285-294, May 16-22, 1999, Los Angeles, USA.

[11] L. White and H. Almezen, "Generating test cases for GUI responsibilities using complete interaction sequences," in *Proc. of the 21st IEEE Int. Conf. on Software Maintenance (ICSM'05)*, pp. 473-482, September 25-30, 2005, Budapest, Hungary.

[12] F. Belli, "Finite State Testing and Analysis of Graphical User Interfaces," in *Proc. of the 12th Int. Symposium on Soft. Reliability Eng. (ISSRE'01)*, pp. 34-43, November 27-30, 2001, Hong Kong.

[13] 10. K. Cai, L. Zhao, H. Hu, and C. Jiang "On the Test Case Definition for GUI Testing," in *Proc. of the 5th Int. Conf. on Quality Soft.(*QSIC'05), pp. 11-18, Sept.19-20,2005, Melbourne, Australia.

[14] M. Vieira, J. Leduc, B. Hasling, R. Subramanyan, and J. Kazmeie, "Automation of GUI Testing using a Model-driven Approach," in *Proceedings of the 2006 Int. Workshop on Automation of Software Test (AST'06)*, pp. 9-14, May 23, 2006, Shanghai, China.

[15] Q. Xie, "Developing Cost-Effective Model-Based Techniques for GUI Testing," in *Proc. of the 28th Int. Conf. on Software Eng. (ICSE'06),* pp. 997-1000, May 20-28, 2006, Shanghai, China.

[16] A. M. Memon, "An Event-flow Model for GUI-Based Applications for Testing," *Software Testing, Verification and Reliability*, vol. 17, no. 3, pp. 137-157, September 2007.

[17] A. M. Memon, "Developing Testing Techniques for Event-driven Pervasive Computing Applications," *OOPSLA'04 Workshop on Building Software for Pervasive Computing (BSPC'04)*, October 24, 2004, Vancouver, BC, Canada.

[18] H. Reza, S. Endapally, and E. Grant, "A Model-Based Approach for Testing GUI Using Hierarchical Predicate Transition Nets," in *Proc. of the 4th Int. Conf. on Inf. Tech. New Generations (ITNG'07)*, pp. 366-370, April 2-4, 2007, Las Vegas, USA.

[19] X. Yuan and A. M. Memon, "Using GUI Run-Time State as Feedback to Generate Test Cases," in *Proceedings of the 29th International Conference on Software Engineering (ICSE'07),* pp. 396-405, May 20-26, 2007, Minneapolis, MN, USA.