COMPARATIVE STUDY ON CORRECTNESS AND TIME TAKEN OF TEST CASE GENERATION USING EFG AND BXT TECHNIQUES FOR GUI APPLICATION

MOSTAFA NSER BRKA

A dissertation submitted in partial
fulfillment of the requirements for the award of the
Degree of Master of Computer Science (Software Engineering)

Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia

SEPTEMBER 2015

Specially dedicated to…

This project is dedicated to my parents who have supported me all the way since the beginning of my studies and who have been a great source of motivation and inspiration. I also dedicate this work and give special thanks to my closest friends May God bless us always.

ACKNOWLEDGEMENT

First and foremost I thank God for the strength and courage that had made this humble effort a preality.

I would like to express my deepest gratitude to my dissertation projectsupervisor, Dr. Noraini Bt. Ibrahim for her invaluable advices and guidancethroughout this project. Her profound knowledge, ideas and support keeps onmotivating me to give my all for this project.

I wish to thank all my friends, staffs and those who directly or indirectly guiding and helping me in this project. The knowledge and support that they shared with me will always be remembered.

Lastly, and most importantly I wish to dedicate my appreciation to my beloved father, mother , brothers and my fiancee for always being there for me all these years. I thank them for their unconditional love, encouragement and support of Universiti Tun Hussein Onn Malaysia (UTHM) is also gratefully acknowledged.

ABSTRACT

The previous decade witnessed the increased popularity and rapid development of graphical user interface (GUI). GUI is a type of computer– human interface thatallows users to communicate with their computers. This interface has significantly contributed to the popularity of recently developed software applications. Given the importance of working under an error-free GUI, the correctness of such GUI must be tested at all times. GUI testing involves checking the screens for controls, such as menus, buttons, icons and all types of bars, including tool bars, menu bars, dialog boxes and software windows. Event Flow Graph (EFG) and Behavior ExplorerTesting (BXT) are two of the techniques used to generate test cases on GUIcomponents. The correctness and time required for these two techniques to generate GUI test cases are compared in this project by using the techniques in Paint and Present applications. EFG and BXT obtained correctness rates of 9.61% and 36.81% respectively for the paint application and 5.18% and 39.33% respectively for the Present application. Therefore, BXT exhibits more correctness than EFG. In term of elapsed time, EFG and BXT spent 0.16 millisecond (ms) and 0.18 ms in the paint application respectively and 0.14 millisecond (ms) and 0.17 millisecond (ms) in the Present application respectively. Therefore, EFG is slightly faster than BXT in generating test cases. Overall, BXT is better than EFG in generating GUI test cases.

ABSTRAK

Dekad yang lepas menyaksikan peningkatan dalam popularity dan pembangunan yang pesat untuk antara muka pengguna grafik (GUI). GUI adalah antara muka manusia komputer yang membolehkan pengguna berkomunikasi dengan komputer mereka. Antara muka ini mempunyai sumbangan besar kepada popularity aplikasi perisian yang dibangunkan. Memandangkan pentingnya GUI bebas dari kesalahan, maka ketepatan kes GUI perlu sentiasa diuji. Pengujian GUI melibatkan pemeriksaan skrin untuk kawalan seperti menu, butang, ikon dan semua jenis bar, termasuk bar alat, bar menu, kotak dialog dan tetingkap perisian. Event Flow Graph (EFG) dan Behavior Explorer Testing (BXT) adalah antara teknik yang digunakan untuk menjana kes-kes ujian untuk komponen GUI. Dalam projek ini, ketepatan dan masa yang diperlukan untuk kedua-dua teknik menjana kes-kes ujian untuk komponen GUI dibandingkan dengan menggunakan dua kajian kes iaitu aplikasi mewarna dan aplikasi persembahan. Dari segi ketepatan, EFG dan BXT menunjukkan kadar ketepatan 9.61% dan 36.81% masing-masing untuk aplikasi mewarna. Sebaliknya, EFG dan BXT masing-masing menunjukkan kadar ketepatan 5.18% dan 39.33% dalam aplikasi persembahan. Oleh itu, teknik BXT menghasilkan kes ujian yang lebih tepat berbanding EFG. Dari segi masa, EFG dan BXT menggunakan 0.16 dan 0.18 millisaat untuk aplikasi mewarna dan 0.14 dan 0.17 millisaat untuk aplikasi persembahan. Oleh itu, EFG adalah sedikit lebih cepat daripada BXT dalam menjana kes-kes ujian. Namun begitu, secara keseluruhan, teknik BXT lebih baik daripada teknik EFG dalam menjana kes-kes ujian.

CONTENTS

.

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AUT	–	Application Under Testing
BXT	–	Behavior Explore Testing
CPU	–	Center Processor Unit
CLIs	–	Command-Line Interfaces
DOS	–	Disk Operating System
DOM	–	Document Object Model
ESIG	–	Event Semantic Interaction Graphs
EFG	–	Event Flow Graphs
EIG	–	Event Interaction Graphs
GUI	–	Graphical User Interface
IDE	–	Integrated Development Environment
JDT	–	Java Development Tools
MRE	–	Magnitude of Relative Error
Ms	–	Millisecond
SDLC	–	Software Development Lifecycle
STL	–	Software Testing Lifecycle
SFG	–	State Flow Graph
STS	–	Seed Test Suite

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background of Study

The Palo Alto Research Center of the Xerox Corporation designed the first graphical user interface (GUI) in the 1970s. However, this GUI was not as popular as the GUIs launched by Apple Macintosh in the 1980s. The substantial power consumption of the central processing unit, high-quality graphicsand high cost of GUI slowed down its growth. Since 1997, Xerox Star (Aris, 2007) has developed and expanded the use of GUIs.

A GUI is sometimes referred to as “gee-you-eye” or “gooey” in computing. It facilitates user interaction with electronic devices by using graphical icons and visual indicators, such as secondary notation or visual cues, as opposed to text based interfaces, keywords, text linksand text navigation designs (Martinez, 2011). GUIs sharply reduce the use of command line interfaces (CLIs)that use the keyboard to type commands (Melchior *et al.,* 2009). The disk operating system (DOS) is an example of the a typical user–computer interface that used a keyboard’s typed commands before GUIs were designed (Nadira & Sani, 2009). The intermediate step in user interfaces between GUI and CLIs was the non-graphical, menu-based interface, where a user interacts with a device using a mouse instead of typing keyboard commands. GUIs are among the crucial parts of a software (Xie *et al*., 2006).

The design of human–computer interaction, which is the application of programming in software technology, depends on the visual arrangement and temporal behavior of a GUI. The GUI enhances the efficiency and ease of the

underlying logical design of a stored program, a design discipline known as usability. In a user-centered design system, the usage of visual language ensures the efficiency and usability of tasks. Thus, a good user interface depends on the design instead of the system architecture (Shneiderman & Ben, 2003).

In software engineering, testing all the components of GUIs is vital to ensure that the GUI meets written specifications. Using a variety of test cases ensures that the GUI design specifications are fulfilled. Nowadays, software testing is an important stage in software projects (Nah *et al.*, 2001). It is one of the most expensive and time-consuming phases and usually stops after available resources are used or even in the middle of the development process because of the duration of this phase. However, manual testing allows a programmer to pledge each test, interrelate with the test, as well as interpret, investigate and report the collected results.

Software testing is automated with a tester-free mechanism. GUI is used to design various software applications because it interacts directly with users. Thus, the accuracy of the applications can easily meet the quality specification set by users. Other tools, such as capture replay, can be laborious and error-prone in designing software applications. However, GUI is particularly automated for easy management. Hence, substantial research focused on different methodologies to test GUIs, particularly the techniques for automated GUI test case generation (Padmawar & Sarwate, 2014).

Event flow graphs (EFGs) and behavior explorer technique (BXT) are two of the most employed techniques to generate test cases for GUI. All possible event flows within a particular system are provided by EFGs. These graphs are structural in nature, which is a limitation of the subpaths within an event that can increase exponentially. This step is theoretically feasible but is limited to practical applications. The second method called BXT determines the effect of the first event on the subsequent event and is useful in selecting two-way interaction. Hence, this research applies these techniques to test GUIs in the two case studies.

1.2 Problem Statement

GUI is designed to improve the interaction between a user and an electronic system. It involves the use of a mouse to select menu options; decisions are made by clicking screen buttons and programs are launched by clicking icons on screen. GUI must always be error-free and normal users must not experience any difficulty when using this interface. Numerous software applications depend on GUIs to coordinate the interactions of users with their systems. However, testing the correctness of a GUI is difficult because of the numerous possible interactions in the interface, such as the command button, text box, combo box and radio button. The sequence of GUI events can also lead to different states, increase the cost and length of software development and consume more time (Giuseppe & Di, 2012). A test case includes the input and expected output, pass/fail criteria and environment where the test will be conducted. Input refers to the data required to generate a test case. Software products can be tested in two ways. First, tests are performed on each function of a product to determine whether a software is fully operational. Second, the internal mechanisms of a product are tested to determine whether these functions actually occur. Many techniques have been used to generate test cases for GUI applications, including Event Interaction Graph (EIG) (Memon *et al*., 2005), Event Semantic Interaction Graph (ESIG) (Yuan & Memon 2010), EFG (Memon *et al*., 2012) and BXT (Bertolini *et al*., 2009). These techniques improve, the correctness and consumed time of test cases. Based on the software engineering perspective, the correctness refers to the adherence of a GUI to specifications that determine how users can interact with software and how the software must behave when used correctly. If the software behaves incorrectly, then users may spend a considerable amount of time to complete their tasks or may even fail to complete such tasks. Thus, the current research will compare the correctness and time taken of the two techniques, namely, EFG and BXT, in generating test cases.

1.3 Project Objectives

The objectives of this research are as follows:

1. To define steps for generating test cases by using Event Flow Graphs (EFG) and Behavior Explore Testing (BXT) techniques.
2. To apply techniques in (1) to the paint application and Present application.
3. To compare the correctness and time taken of test case generated between both techniques in (1) for the case studies in (2) .

1.4 Scope of Project

This research uses two case studies and two techniques to investigate the problem of correctness and time taken in GUI applications. The study compares the result of the two techniques for the two case studies to determine the better technique between the two chosen techniques. The two techniques are EFG and BXT. The case studies of the current research are the paint and Present applications.

1.5 Dissertation Outline

The dissertation includes six chapters. Chapter 1 is an overview of the research and provides the main objectives of the project. The chapter also includes the problem statement and scope of the work covered by this project. Chapter 2 provides the literature review of EFG and BXT, as well as a brief explanation of the general information about generating test cases for GUI applications and some definitions used in this project. Chapter 3 discusses the methodology and tools to achieve all the objectives of this project. Chapter 4 explains the implementation and detailed steps employed in this work. Chapter 5 discusses this project. Finally, Chapter 6 explains the achieved objectives, conclusion of the project and future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter provides an overview of previous research on GUI testing. Research on techniques for generating test cases are also Present ed in this chapter. An introduction of a framework for tools that employ these techniques and some topics related to this study are explained to identify the appropriate approach for investigating the objectives of the project.

2.2 Graphical User Interface

GUIs have three main bases: windows applications, web applications and mobile applications. At Present , software GUIs are one of the most commonly used components. A typical GUI provides degrees of freedom and various facilities to an end-user. A test designer handles particular design challenges, such as enormous input interaction space of the GUI, deals with development and examines the test cases (Huang & Lu, 2012).

Currently, almost 70% of software systems are developed using GUIs. GUIs are primarily used to faciliate an end-user to promote the features of GUIs that provide ease and natural interaction between a system and its users.

GUI is hierarchical in producing the deterministic graphical output when a graphical front-end software system accepts an input user-generated and system-generated sequence of events from a fixed set of events. This interface is composed

of graphical objects where every object has a fixed set of properties. In some cases, GUI properties have separate values that contain a set of constituted GUI states during execution (Memon, 2007).

GUIs have become nearly omniPresent by interacting with software systems. Figure 2.1 descibes the front-end underlying code for a GUI, where an end-user interacts with the software using the GUI.

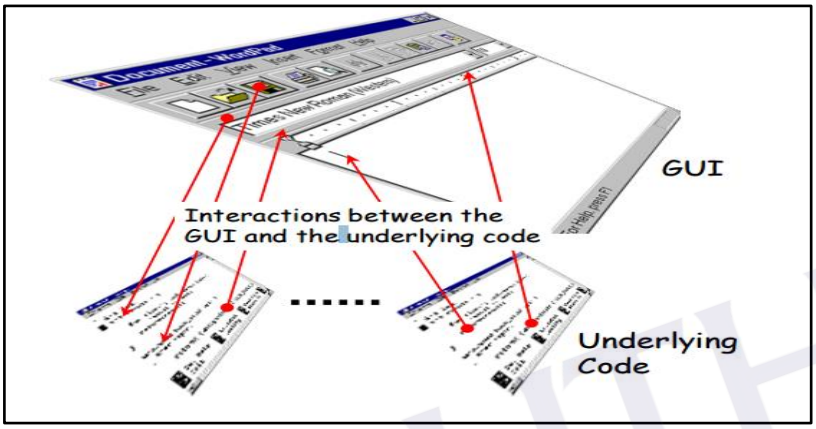

Figure 2.1: GUI is the front-end to the underlying code (Memon, 2001)

2.3 Software Testing

Software testing is an important activity in software engineering. It can help people obtain accurate findings for software quality and diagnose errors in software execution (Ref *et al*., 2011). Software testing helps achieve the required results by evaluating an attribute or capability of a program or system (Choudhary & Kumar, 2011).

In software testing, analysis is performed to ensure that the quality of the tested product or service meets the specific requirement of stakeholders. Software testing can also provide an objective and independent view of a software to allow a business to escalate and determine the risks of software implementation. In a testing technique, a user can execute a program or application to find software bugs, errors, or other defects but is not limited to these. Software testing can validate and verify a software program application or product (Karnavel & Santhosh, 2013).

2.4 GUI Testing

GUI is crucial in perfecting a software product. Thus, GUI testing ensures software reliability. It is usually performed using a test script that interacts with a GUI through a sequence of actions. Moreover, generating event sequences is always challenging. Thus, the testing phase is divided into GUI testing, logical testing, unit testing and integration testing. These various testing approaches provide effectiveness, efficiency, correctness and accuracy in developing a quality product (Chen *et al*., 2008).

Testing a GUI is an important and difficult concern in developing quality software. In GUI testing, one of the most challenging considerations is the significantly large or infinite input domain of a non-trivial GUI application. Defining the region of convergence is necessary to help testers select the test cases from the input domain of GUI applications (Zhao & Cai, 2010). GUI testing provides information about the functionality of software for it to meet the design requirements in the GUI and develop the standard required product before being launched in the market.

GUI testing helps ensure that the requirements specified for a particular GUI are met. These specifications contain the navigation path or sequences that will be performed by a normal user and other sequences that a user can freely obtain through the GUI. These sequences can generate faults or failures in the software system. Thus, testing these sequences is crucial (Isabella & Emi, 2012).

2.5 Performance Parameters for Testing

Developers should achieve accuracy, correctness and performance-related issues in developing a program that can rePresent an algorithm. To obtain appropriate design specifications of a software system, developers must confirm the desired quality. This step is achieved using a process called software verification. This process involves verification activities during specification, design and implementation. Software testing is another process used to assess the functionality and correctness of software. The process analyzes the execution of a program (Gregory, 2008).

REFERENCES

Aris, K. A.(2007). *Development of motor control using graphical user interface*. University Malaysia Pahang, Ph.D. Thesis.

Alferez, M., Santos, J., Moreira, A., Garcia, A., Kulesza, U., Araujo, J., & Amaral, V. (2010). *Software Language Engineering*. Berlin Heidelberg : Springer.

Boghdady, P. N., Badr, N., Hashem, M., & Tolba, M. F. (2011). Test Case Generation and Test Data Extraction Techniques. *International Journal of Electrical and Computer Sciences, 11(03),* pp. 87-94.

Bunin, G., & Schneider, A. (2009). *U.S. Patent No. 7,523,425*. Washington, DC: U.S. Patent and Trademark Office.

Bertolini, C., Peres, G., d'Amorim, M., & Mota, A. (2009). An empirical evaluation of automated black box testing techniques for crashing guis. In*Software Testing Verification and Validation, 2009. ICST'09. International Conference on IEEE* . pp. 21-30.

Bae, G., Rothermel, G., & Bae, D. H. (2012). On the relative strengths of model-based and dynamic event extraction-based gui testing techniques: An empirical study. In *Software Reliability Engineering (ISSRE), 2012 IEEE 23rd International Symposium on (pp. 181-190). IEEE.*

Chen, W. K., Shen, Z. W., & Chang, C. M. (2008). GUI test script organization with component abstraction. *In Second International Conference on Secure System Integration and Reliability Improvement, 2008, IEEE.* pp.128-134.

Choudhary, D., & Kumar, V. Software testing. *Journal of Computational Simulation and Modeling*. 2011. 1(1): 01-09.

DesRivieres, J., & Wiegand, J. (2004). Eclipse: A platform for integrating development tools. *IBM Systems Journal*, 2004. *43 (2)*: 371-383.

Dexter, M. (2007). Eclipse And Java For Total Beginners Companion Tutorial Document. *Mark Dexter. Licensed under the Educational Community License.*

Eldh, S. (2011). *On Test Design*. Doctoral dissertation, Vasteras*:* Malardalen University. Ph.D. Thesis.

Gandhi, G. M. D., & Pillai, A. S. (2014). Challenges in GUI Test Automation. *International Journal of Computer Theory and Engineering*, *6*(2),pp 192.

Gautam and Sharma (2014). An Approach to Generate the Test Cases for GUI Testing, IJISET - *International Journal of Innovative Science, Engineering and Technology*, Vol. 1 Issue 6, August 2014.

Di Lucca, G. A., & Fasolino, A. R. (2006). Testing Web-based applications: The state of the art and future trends. *Information and Software Technology*, *48*(12), pp.1172-1186.

Huang, Y., & Lu, L. (2012). Apply ant colony to event-flow model for graphical user interface test case generation. *IET software, 6(1),* pp.50-60.

Fernando, S. G. S., & Perera, S. N. (2014). Empirical Analysis of Data Mining Techniques for Social Network Websites. *Compusoft, 3*(2), pp.582**.**

Isabella, A., & Retna, E. (2012). Study Paper on Test Case generation for GUI Based Testing. *arXiv preprint arXiv:1202.4527*,3(1).pp 139-147

Kanchan, G., & Sharma, P.M. (2014). An Approach to Generate the Test Cases for GUI Testing, *International Journal of Innovative Science, Engineering and Technology, 1 (6).*

Karnavel, K., & Santhoshkumar, J. (2013). Automated software testing for application maintenance by using bee colony optimization algorithms (BCO). *In IEEE International Conference on Information Communication and Embedded Systems (ICICES)* 2013,pp.327-330.

Kaur, M., Sharma, N., & Kaur, R. K. (2010). Xml Schema Based Approach for Testing of Software Components. *International Journal of Computer Applications, 6(11),PP, 7-11*.

Latiu, G. I., Cret, O., & Vacariu, L (2013).Graphical user interface testing using evolutionary algorithms. *In IEEE 8th Iberian Conference on Information Systems and Technologies,* 2013.pp.1-6.

Liu, S. (2001*). Generating test cases from software documentation*; McMaster University. Ph.D. Thesis

Martinez, W. L. (2011). Graphical user interfaces. *WIREs Comp Stat, 3,* pp.119–133.

Melchior, J., Grolaux, D., Vanderdonckt, J., & Van Roy, P. (2009). A toolkit for peer-to-peer distributed user interfaces: concepts, implementationand applications. *In Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pp.69-78.

Memon, A. M. . (2001). *A comprehensive framework for testing graphical user interfaces.*University of Pittsburgh;. Ph.D. Thesis

Memon, A. M. (2004). Developing testing techniques for event-driven pervasive computing applications. *In Proceedings of The OOPSLA 2004 workshop on Building Software for Pervasive Computing*, (BSPC 2004).pp.1-3.

Memon, A. M. (2007). An event-flow model of GUI-based applications for testing. *Software Testing, Verification and Reliability.. 17(3),*pp. 137-157.

Memon, A. M., Pollack, M. E., & Soffa, M. L. (2001). Hierarchical GUI test case generation using automated planning. *IEEE Transactions on Software Engineering*.. 27(2),pp. 144-155.

Mesbah, A., Bozdag, E., & Van Deursen, A. (2008). Crawling Ajax by inferring user interface state changes. *In Web Engineering,. ICWE'08. Eighth International Conference on.* IEEE. *pp. 122-134.*

Nadira, I., & Sani, A. (2009). *Voice Recognition Technology To Control Electronic/Electrical Appliances*. UTeM, Melaka,Malaysia. Ph.D. Thesis

Nah, F. H., Lau, J. L. S., & Kuang, J. (2001). Critical factors for successful implementation of enterprise systems. *Business process management journal,. 7(3):* 285-296.

Navarro, P. L. M., Ruiz, D. S., & Pérez, G. M. (2010). A proposal for automatic testing of GUIs based on annotated use cases. *Advances in Software Engineering*.pp 8.

Padmawar, N. V., & Sarwate, D. A. (2014). Impact of Test Automation and Test Case Design Techniques-Challenges. *International Journal of Innovative Research in Computer Science & Technology.. 2(4): 40-42*

Rauf, A., & Alanazi, M. N. (2014). Using artificial intelligence to automatically test GUI. *In IEEE 9th International Conference on Computer Science and Education.*. pp. 3-5.

Singh, S. K. (2011). An event-based framework for object-oriented analysis computation of metrics and identification of test scenarios. *University: Jaypee Institute of Information Technology*.

Silva Simao, A. L., Moura, A. V. & Bonifácio, A. (2008). A generalized model-based test generation method. In *Software Engineering and Formal Methods, 2008. SEFM'08. Sixth IEEE International Conference on (pp. 139-148). IEEE.*

Tai, K. C. (1980). Program testing complexity and test criteria. *IEEE Trans. Software Engineering.*, *6*(6), 531-538

Tron F, Stensrud, E, Barbara K, & Ingunn M (2002). An empirical validation of the relationship between the magnitude of relative error and project size. *In Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on (pp. 3-12).*

Xie., Qing., & Memon A.M. (2006). Model-based testing of community-driven open- source GUI applications. *Software Maintenance, 22nd IEEE International Conference on IEEE.* 2006.pp :145-154

Xie, Q., & Memon, A. M. (2006,). Studying the characteristics of a" Good" GUI test suite. In *Software Reliability Engineering, 2006. ISSRE'06. 17th International Symposium on (pp. 159-168). IEEE.*

Wasif Afzal. (2007). Metrics in Software Test Planning and Test Design Processes. *School of Engineering Blekinge Institute of Technology*

Yu, H., Lan, Y., & Ren, H. (2011). The Research about an Automated Software Testing System RunTool. *In IEEE 3rd International Workshop on Intelligent Systems and Applications*, 2011. pp.1-4.

Yuan, X., & Memon, A. M. (2010). Generating event sequence-based test cases using GUI runtime state feedback. Software Engineering, *IEEE Transactions*. *36(1),*pp. 81-95.

Zacharias, B. (2012).Test Case Generation and Reusing Test Cases for GUI Designed with HTML. *Journal of Software*.7(10): 2269-2277.

Zhao, L., & Cai, K. Y.(2010). Event handler-based coverage for GUI testing. *In 10 th IEEE International Conference on Quality Software*..pp. 326-331.