UNIVERSITI PUTRA MALAYSIA

COMPARISON OF SOFTWARE TESTING TOOLS ON GUI FEATURES

FATIMAH GHAZALI

FSKTM 2005 11

COMPARISON OF SOFTWARE TESTING TOOLS ON GUI FEATURES

By

FATIMAH GHAZALI

A project Paper submitted to the Schools of Graduate Studies, Universiti Putra Malaysia in Fulfillment of the Requirement for the Degree of Master of Science

April 2005

Abstract of thesis presented to the Senate of University Putra Malaysia in fulfillment of the requirement for degree of Master of Science

COMPARISON OF SOFTWARE TESTING TOOLS ON GUI FEATURES

By

FATIMAH BINTI GHAZALI

APRIL 2005

Chairwoman: Puan Norhayati Mohd Ali

Faculty: Faculty of Computer Science and Information Technology

Abstract

Software testing is an important approach to software quality assurance. It is a kind of software development activity throughout all the life cycle of software development, whose purpose is to find the potential errors or bugs of software as many as possible. Hence, software testing tools were developed to assists software engineering to gauge the quality of software automating the mechanical aspects of the software testing tasks. Software testing tools vary in their underlying approach, quality and ease-of-use among the other features.

This project paper compares and measures three selected software testing tools that available in current market. These testing tools were downloading from Internet. Comparison of three selected testing tools based on graphical user interface (GUI) features and usability. The GUI features are proposed by Stephen Morris. The software testing tools also measured on usability aspect from user’s perspective to determine the efficiency, affect, helpfulness, control and learnability of software testing tools. The result of the study was to propose general features of GUI for software testing tools that should be considered in software testing tools.

Abstrak tesis yang dikemukan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk Ijazah Master Sains

COMPARISON OF SOFTWARE TESTING TOOLS ON GUI FEATURES

By

FATIMAH BINTI GHAZALI

APRIL 2005

Pengerusi: Puan Norhayati Mohd Ali

Fakulti : Fakulti Sains Komputer dan Teknologi Maklumat

Pengujian perisian adalah satu pendekatan yang penting kepada jaminan perisian yang berkualiti. Ia adalah satu aktiviti pembangunan perisian melalui keseluruhan kitar hayat pembangunan perisian, yang mana tujuannya adalah untuk mencari ralat dan pepijat sebanyak mungkin di dalam sesuatu perisian. Maka dengan itu, alat bantuan pengujian perisian (*software testing tools*) dibangunkan untuk membantu kejuruteraan perisian untuk mengukur kualiti perisian mengautomasikan aspek makanikal tugas pengujian perisian. Alat bantuan pengujian perisian mempunyai pendekatan yang berbeza-beza, kualiti dan senang guna di kalangan ciri-ciri yang lain.

Projek ini membanding dan mengukur tiga alat bantu pengujian perisian yang terdapat dalam pasaran semasa. Alat bantu pengujian ini dimuaturun daripada Internet. Perbandingan ketiga-tiga alat bantu pengujian dibuat berdasarkan kepada ciri-ciri antaramuka pengguna bergrafik dan kebolehgunaan. Ciri-ciri antaramuka pengguna bergrafik dicadangkan oleh Stephen Morris. Alat bantu pengujian perisian juga diukur berdasarkan kepada kebolehan perisian dari perspektif pengguna bagi menentukan keberkesanan, pengaruh, bantuan, kawalan dan penggunaan alat bantu pengujian perisian. Hasil daripada kajian telah mencadangkan ciri-ciri umum antaramuka pengguna bergrafik untuk alat bantu pengujian perisian yang diperlukan dipertimbangkan dalam alat bantu pengujian perisian.

ACKNOWLEDGEMENT

Alhamdulillah, all praises to Allah s.w.t the only true God that rules the earth, and everything beyond it. Nothing can be accomplished without His consent. He is the all merciful, all giving. Never once has He left me in moment of despair. He is the all powerful, all knowing, the creator, which bestowed me with the wisdom to take a pace in this odyssey of knowledge. I am forever grateful to His unconditional care.

I would like to express my greatest appreciate to my supervisor, Puan Norhayati Mohd Ali for her guidance, analytical advice and support that led to the success of this project. I would like to express my thankful to FSKTM students who participate as respondent. I’m also thankful to my family and all of my friends, for their support and encourage in completing this study. Lastly, I also would like to express my pleased to everybody who involve direct or indirectly in completing this study.

APPROVAL

This project paper is submitted to the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia and was accepted as fulfillment of the requirement for the degree of Master of Science

………………………………….

Puan Norhayati Bt Mohd Ali

Project Supervisor

Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

Date :

DECLARATION

I hereby declare that this project report is based on my original work except for quotations and citations which are duly acknowledge. I also declare that it has not been previously or concurrently submitted for any other degree at UPM or other institutions.

…...…………………

…

FATIMAH

GHAZALI

Date :

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	**Words**
GUI	Graphical User Interface
SUMI	Software Usability Measurement Inventory
RSM	Resource Standard Metrics
OO-SQA	Object-oriented-Software Quality Assurance
IDE	Integrated Development Environment
HTML	Hypertext Markup language

CHAPTER ONE

INTORDUCTION

1.1 General Background for study

Software testing is a critical component of software development. Its goal is to uncover and correct errors found in software. Since software testing becomes an important branch of software engineering, there have been abundant researches on software testing. Software testing is an important approach to software quality assurance, is a kind of software development activity throughout all the life cycle of software development, whose purpose is to find the potential errors or bugs of software as many as possible.

Over the past years several, tools that help programmers quickly create application with user graphical interfaces have dramatically improved programmer productivity. This has increased the pressure on testers, who are often perceived as bottlenecks to the delivery of software products. Testers are being asked to test more and more code in less and less time. They need to dramatically improve their own productivity. In the process of practice, creating test cases, executing test, recording and comparing test results are the major activities of test. It is difficult to go on without the support of tools for all of these activities. Test automation is one way to do this [Bret Pettichord, 2001].

The use of automated test tools to support the test process is proving to be beneficial in terms of product quality, minimizing project schedule and effort and reduced testing costs. Automated testing tools also assist software engineering to gauges the quality of software automating the mechanical aspect of the software testing task. Automated testing tools vary in their underlying approach, quality and ease of use, among other features.

This paper discusses about features of GUI in software testing tools as theoretically mentioned in books, journal, articles and other resources. By referring these features, this paper is to identify the features that such a test tool will need in order to satisfy them. The results are then used in the survey of test tools to determine those that will be suitable for further evaluation.

1.2 Problem Statement

In order to produce quality of software and to save time and cost, during the testing phase, most of organization will be using software testing tools. Chosen the right and quality tools will be determine the successful of testing. There are many type of testing tools offered in current market and Internet and it is difficult to categorize them neatly**.** Some of the testing tools are lack of the graphical user interface (GUI) features. This will influence the usage of the testing tools. In fact, most developers do not use automated testing tools in their testing activities. Therefore, this study was to identify GUI features in software testing tools and to measure the usability of the software testing tools from the users' perspective.

1.3 Objective of The Study

The objectives of this study are:

i. to compare the three selected software testing tools in current market. This discusses about the features of GUI in software testing tools.
ii. to find out the general features of GUI that must be considered to develop software testing tools, which it is not limited to certain organization or software.
iii. to measure the usability of the selected software testing tools from users' perspective.

1.4 Scope of the study

The scope of this study is to analyze the features of GUI in software testing tools for source code/program structures either static analyzer or dynamic analyzer. The software testing tools downloaded from Internet. It is also to measure the usability of tools from users' perspective.

CHAPTER TWO

LITERATURE REVIEW

2.1 Software Testing Tool

C.V. Ramamoorthy and S.F Ho defined software testing tools is the programs that check the presence of certain software attributes which can be program syntax correctness, proper program control structures, proper module interface, testing completeness and so on.

Automated software tools are tools designed to run programs and can also be made to aid programmers in system development. One major problem in dealing with large software system is the size. Automated tools can scan a large volume of source code and indicate questionable features to the human programmer (e.g unreliable program construct), and free the programmer from repetitive tasks. As a debugging aid, automated tools can be use to remove simple coding errors, and allow the programmer to concentrate on advance system check out. As an optimization tool, automated tools can indicate code section that merit detailed examination. Successful application of program correctness proofs to large software systems also depends on the automation generation of assertions by automated tools or theorem provers. It means that, automated tools allow a small programming team to examine a large volume of code which is otherwise impossible [C.V Ramamoorthy and S.F. Ho].

Martin Wieczorek [2001] stated that the requirement for testing tools can be grouped into a number of different areas, including test planning and management, requirement analysis and test case generation, inspection and review, code quality analysis, functional testing, load and performance testing and web application testing.

Bret Pettichord [2001], said that automating tests for a graphical user interface presents significant difficulties not found in character based interfaces, much less command line interfaces or programming interfaces (APIs). Graphical user interfaces tend to be made of complex components and tend to be constantly redesigned during the development process. Significant successes have been made in delivering test tools that are able to identify and manipulate graphical user interfaces.

Professor Ben-Avi in his paper Software Testing Technology was categorized software quality assurance support tools in two categories. There are static and dynamic analysis tools. Most tool functions fall cleanly into one category or the other, but there are some exceptions like symbolic evaluation systems and mutation analysis system (which actually run interpretively). The main tools used in quality assurance static analyzers, code inspectors, standard enforcers, coverage analyzers, output comparators, test file/data generators, test harness and test archiving systems.

When the developers are deciding to use tools during testing phase, they should consider the quality factors to evaluate software testing tools. One of the software quality criteria is usability. Usability is an important aspect of software product. Usable systems are easy to learn, efficiency to use, not error-prone and satisfactory in use [Nielsen, 1993]. Usability brings many benefits. According to ISO/IEC 13407, the benefits of usability include " increased productivity, enhanced quality of work, improved user satisfaction, reductions in support and training costs and improved used satisfaction".

2.2 Features of GUI in software testing tools

2.2.1 Features proposed by Elisabeth Hendrickson

Elisabeth Hendrickson proposed the twelve features that are important in any good software testing tool are scripting language, user interface element identifiers, reusable libraries, outside libraries, abstract layers, distributed tests, file input/output, error handling, the debugger, source control, common line script execution and the user community. Summary of these features is shown in Table 2.1.

2.2.1.1 Scripting language

A prerequisite to all the other features for software testing tool is that the tool must have a scripting language of some kind that contains the usual programmatic constructs. At the very least, it should enable to edit recorded scripts, support variables and data types, support arrays, lists, structures, or other compound data types, support conditional logic (IF and CASE statements), support loops (FOR, WHILE) also enable to create and call functions.

2.2.1.2 User Interface (UI) element identifiers

In order to write test scripts that actually test something, the test tool can identify the elements on tester UI as objects rather than trying to point to them by coordinate. The tool can identify the UI elements in a variety of representative windows. It is true that some UI elements aren't really controls at all, just bitmaps that do something when you click on them. Software that uses UI elements that are bitmaps rather than real controls won't behave well with any automated testing tool.

2.2.1.3 Reusable libraries

Software testing tools need to support reusable library to store function or subroutine that perform the search. The function library created to define the sequence of steps necessary to perform a search for testing an application. Each script calls the function. Any scripts created with the tools can easily call the functions that put in the library and the functions can take parameters.

2.2.1.4 Outside libraries

In addition to creating tester's own libraries, you'll often find it useful to access outside libraries. In Windows, **.dll** files are able to call. As an example, consider a client/server system built to work with a relational database. The software under test uses the database's proprietary API (Application Program Interface). If the automated tests can use the same API, they can be more powerful. They can make checks the user interface doesn't allow. For example, they can check that a changed value was actually written to the database, not just changed on the screen. They can check whether a transaction was correctly and completely logged, even if the UI gives no access to the log.

In general, these tests can determine "pass" or "fail" more accurately than by checking the value through the user interface. If testing is doing on a Windows system, then the Windows API will be accessed. The Windows API enables you to get system information that would be difficult or impossible to obtain in any other way. For example, it's very useful to be able to get or set the value of a registry key from within the automated scripts.

2.2.1.5 Abstract layers

An "Abstract Layer" to define logical names for physical user interface elements. Some tools call this a "test map" or "GUI map" while others call it a "test frame." The purpose of the abstract layer is to make it easier to maintain the tests. As an example, a login dialog box with fields for name and password. Within the program, the programmer named those fields "Name" and "Password." An abstract layer is created to identify the fields as "Name" and "Password" and proceed to use those identifiers in all 500 of the scripts. But with the next version of the software under test, the underlying identifiers of the name and password fields become "username" and "pword." Instead of changing all 500 of the scripts, developer changes the UI identifiers in the abstract layer. Several test tools offer features, such as window recorders, specifically designed to support the creation of an abstract layer.

2.2.1.6 Distributed tests

If application is testing multi-user software, software testing tool need to be able to create tests that involve multiple simulated users. In a distributed test, the automated testing tool enables to specify the machine on which to execute a given command. In launching a test on a remote machine, the remote machine executes that test from beginning to end. However, if the testing needs to coordinate on two different machines, then the testing to do more than launch a test and let it run. Thus, testing tools need to be able to create a test that waits for an action (such as locking a file) to be complete on the first machine before beginning an action (such as attempting to open the file) on the second machine.

2.2.1.7 File I/O

File I/O (input/output) means that the tool provides functions that enable you to open a file on the hard disk (usually an ASCII file) programmatically, read from it, write to it, and close it. File I/O functions are central to “data-driven test automation.” In a data-driven automated test, the script uses test data from a file to drive the test activity. Data-driven testing makes it possible to automate a large number of tests with a minimal amount of test automation code. For example, if the software under test needs to know which server to use, then it’s a good idea to specify the server name in an **.ini** file. Then the test server can changed without having to change the automated scripts.