



Automated Testing of Web Enabled Systems through FSM and Quick Test Professional

¹Sanjeev Dhawan, ²Nirmal Kumar, ³Divya Sethi

¹Assistant Professor, ²Research Scholar

^{1,2} Department of Computer Science and Engineering, University Institute of Engineering & Technology (U.I.E.T), Kurukshetra University, Kurukshetra- 136 119 (K.U.K), Haryana, INDIA.

E-mail (s): rsdhawan@rediffmail.com, nirmal.sirohi@gmail.com

³ GM Conferencing & VSAT Solutions, Enterprise Services, Bharti Airtel, Gurgaon, Haryana 122 001, INDIA.

E-mail: divya.sethi@airtel.in

Abstract: Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Model-based testing refers to the processes and techniques for the automatic derivation of abstract test cases from abstract formal models, the generation of concrete tests from abstract tests, and the manual or automated execution of the resulting concrete test cases. Therefore, the key points of Model-based testing are the modelling principles for test generation, the test generation strategies and techniques, and the concretization of abstract tests into concrete, executable test. HP (Hewlett-Packard) Quick Test Professional is automated testing software designed for testing various software applications and environments. It performs functional and regression testing through a user interface such as a native GUI (graphical user interface) or web interface. Here a simple kind of Model-based testing, which is based on FSMs (finite state machines) is described. Each node of an FSM corresponds to a particular state of the SUT and each arc corresponds to an SUT (system under test) action, so to generate test sequences we can just traverse the FSM. We start with simple FSM models that are suitable for testing passive systems, such as unit testing of an object or class, and for testing deterministic reactive systems, where events occur in a known order.

Keywords: Software Testing; Model based testing; Quick test Professional; Finite State Machine.

I. Introduction

To make error is human tendency which drives the need for software testing. Because manual testers may not execute the test cases correctly. There will be a lot of possibilities for making mistakes. They may give wrong input data due to type, or they may not notice the actual behaviour of the system correctly, or they may not report the test result correctly, or they may miss to execute some test cases, or they may forget to run some preconditions, or they may change the sequence of test case execution in case sequence is important. Another important factor is, Automation test scripts will be used as a way of storing domain/project/task knowledge gained by the test engineers. They can understand the flow/data by seeing the execution of the automation test scripts. Automation tool such as QTP (Quick Test Professional) has feature for storing screenshot of each and every page navigated during the execution. So it can be used as a proof for completion of testing, and also we can refer the screenshots of previous executions if there is any need to refer them. Test report can be automatically written to a customized report page which will ensure accuracy of the report and also it can improve look & feel of the report. The very important advantage of automation testing over manual testing is execution speed. Test execution can be completed quickly and also we can execution the scripts in night time also without human involvement.

The tool such as QTP supports automation object model to achieve this. The functional test automation scripts will be useful for doing performance testing also practically it may not possible for doing it manually. This problem can be easily solved by simple shell script or any other scripts such as vbs, wsh. As the automation test tools support Data Driven Testing, Test execution can be done repeatedly with many different data sets. There are lot of automation test tools available for doing Functional, Regression and Performance Testing. Test complete, SilkTest, SilkPerformer, QARun, QALoad, TestPartner, WinRunner, LoadRunner, QTP, Rational Robot and openSTA are some of them. QTP is most widely used now as it supports vbscript and it can be used for testing many different applications just by adding required add-ins. The present research paper is directed to provide automated testing of Web enabled systems through FSM and Quick Test Professional as well as systems and techniques for developing the same, and is more particularly directed to testing processes where accuracy and authenticity are two main pillars for the success.

II. Quick Test Professional

All HP Quick Test Professional is automated testing software designed for testing various software applications and environments. It performs functional and regression testing through a user interface such as a native GUI or web interface. It works by identifying the objects in the application user interface or a web page and performing desired operations (such as mouse clicks or keyboard events); it can also capture object properties like name or handler ID. HP Quick Test Professional uses a VBScript scripting language to specify the test procedure and to manipulate the objects and controls of the application under test. To perform more sophisticated actions, users may need to manipulate the underlying VBScript. Although HP Quick Test Professional is usually used for “UI Based” Test Case Automation, it also can automate some “Non-UI” based Test Cases such as file system operations and database testing.

A. Basic Features of QTP

This section presents some of the most important features of using QTP: - (i) QTP is mainly used for functionality testing. QTP is user friendly both technical and non-technical users can easily access. QTP has two views: a. keyword view and b. expert view in keyword view when u record a test it generates the script in tree format. It is very easy to understand. If u wants to view the vbScript use the expert view. (ii) QTP has active screen using that you can enhance the script without navigating to the application, like you can insert checkpoints and synchronization points.(iii) QTP has automated documentation. When you start recording it generates the auto documentation in plain English. (iv) Quick Test professional provides supports to many technology like Web, Java(Core and Advanced), Net, WPF, SAP, Oracle, Siebel, PeopleSoft, Delphi, Power Builder, Stingray 1, Terminal Emulator, Flex, Web Services, Windows Mobile, Visual Age Smalltalk, Silver light and mainframe terminal emulators.

B. Testing Process of Quick Test Professional

The Quick Test testing process consists of seven main phases: - (i) Preparing to record: Before you record a test, confirm that your application and Quick Test are set to match the needs of your test. Make sure your application displays elements on which you want to record, such as a toolbar or a special window pane, for example, and that your application options are set as you expect for the purposes of your test. You should also view the settings in the Test Settings dialog box (Test >Settings) and the Options dialog box (Tools>Options) to ensure that Quick Test will record and store information appropriately. For example, you should confirm that the test is set to use the appropriate object repository mode; (ii) Recording a Session for an Application: As you navigate through an application or Web site, Quick Test graphically displays each step you perform as a row in the Keyword View. A step is any user action that causes or makes a change in your application, such as clicking a link or image, or entering data in a form; (iii) Enhancing Test Case: Inserting checkpoints into your test lets you search for a specific value of a page, object, or text string, which helps you determine whether your application or site is functioning correctly. Broadening the scope of your test, by replacing fixed values with parameters, lets you check how your application performs the same operations with multiple sets of data. Adding logic and conditional or loop statements enables you to add sophisticated checks to your test; (iv) Debugging Test Cases: You can debug a test to ensure that it operates smoothly and without interruption; (v) Running Test Cases: You run can a test to check the behavior of your application or Web site. While running, Quick Test opens the application, or connects to the Web site, and performs each step in your test; (vi) Analysing Test Results: You can examine the test results to pinpoint defects in your application; and (vii) Reporting Defects: If you have Quality Center installed, you can report the defects you discover to a database. Quality Center is Mercury Interactive software test management tool.

B1. Sample Flight Reservation System Model

Here in this paper we have taken a useful model of sample “Flight Reservation System” to demonstrate the benefits of automation testing in Model-based Testing. This system is basically designed for Travel Agents to book flights for their customers. It has various modules like Book flight tickets for customers, Delete an order, Update an Order, Generate Agent Order Report and send the fax order to customer. We have design this model on the bases of given below test cases.

• **Table1. Test cases for flight reservation system taken to automate the system.**

Scenario Number	Scenario Description	Pre-Condition	Test Steps	Test Steps	Expected Results	Actual Results
1	User Login to the application with invalid login credentials.	Flight Reservation application should be open.	1. Do not enter user ID and Password and click on "Ok" Button.	1. Do not enter user ID and Password and click on "Ok" Button.	System should display information message "Please enter agent name".	
			2. Click on "OK" button of Information message.	2. Click on "OK" button of Information message.	System should redirect to the login window and cursor should blink at agent name field.	
			3. Enter agent name as "Test"	3. Enter agent name as "Test"	System should display error message	

Scenario Number	Scenario Description	Pre-Condition	Test Steps	Test Steps	Expected Results	Actual Results
			and click on "OK" button.	and click on "OK" button.	"Please enter password ".	
			4.Click on "OK" button of error message.	4.Click on "OK" button of error message.	System should redirect to the login window and cursor should blink at password field.	
			5. Enter "test" in password field and click on "ok" button.	5. Enter "test" in password field and click on "ok" button.	System should display error message "Incorrect password. Please try again ".	
			6. Click on "OK" button of error message.	6. Click on "OK" button of error message.	System should redirect to the login window and cursor should blink at password field.	
2	User close the login window by clicking "cancel" button.	Flight Reservation application should be open.	1. Click on "Cancel" button of Login window.	1. Click on "Cancel" button of Login window.	Login window should get close.	
3	User gets help for password by clicking "Help" button of Login Window.	Flight Reservation application should be open.	1. Click on "Help" button of Login window.	1. Click on "Help" button of Login window.	System should display information message "The Password is Mercury".	
			2. Click on "OK" button of Information message.	2. Click on "OK" button of Information message.	System should redirect to the login window.	
4	User Login to the Flight Reservation Application by providing valid login credential.	Flight Reservation application should be open.	1. Enter "Test" in agent name field. 2. Enter "Mercury" in password field. 3. Click on "Ok" Button.	1. Enter "Test" in agent name field. 2. Enter "Mercury" in password field. 3. Click on "Ok" Button.	Flight Reservation window should display.	
			4. Verify all fields on flight reservation window.	4. Verify all fields on flight reservation window.	All the fields should be in below mentioned state. 1. Date of Flight Should be blank. 2. Fly From and Fly To should be blank. 3. Flights button should be disabled. 4. Flight No., Departure Time, Arrival Time and Airline should be blank and disabled. 5. Name should be blank and editable. 6. Tickets, price and Total should be blank and disabled. 7. Insert Order, Update order and Delete Order should be disabled.	

C. Test Case Generation Using Event-Flow Model

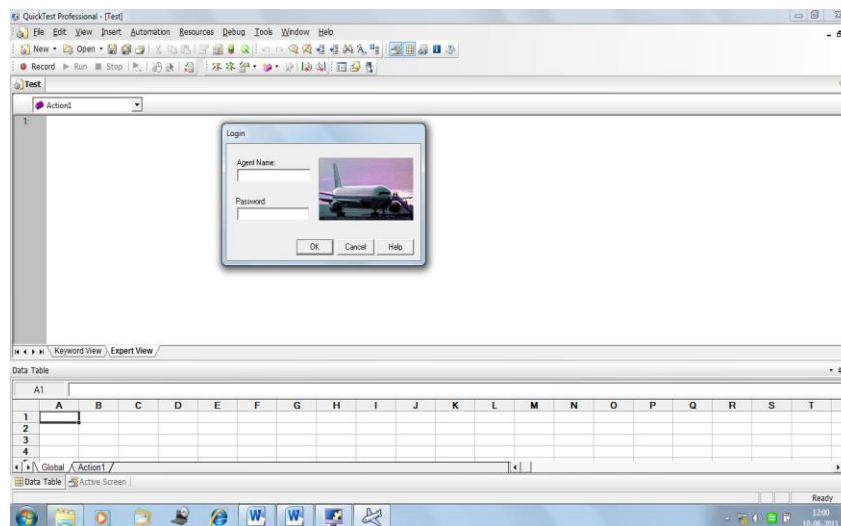
Event Flow Model is a test design technique in Model-based testing. Event Flow Model A popular model that has recently been used extensively for testing software with a graphical user-interface (GUI) front-end is called the event-flow model that represents events and event interactions. In much the same way as a control-flow model represents all possible execution paths in a program, and a data-flow model represents all possible definitions and uses of a memory location, the event-flow model represents all possible sequences of events that can be executed on the GUI. More specifically, a GUI is decomposed into a hierarchy of modal dialogs; this hierarchy is represented as an integration tree; each modal dialog is represented as an event-flow graph that shows all possible event execution paths in the dialog; individual events are represented using their preconditions and effects. An overview of the event-flow model is associated with algorithms to semi-automatically reverse engineer the model from executing GUI software. Because the event-flow model is not tied to a specific aspect of the GUI testing process, it may be used to perform a wide variety of testing tasks by defining specialized Model-based techniques

called event-space exploration strategies (ESES). These ESES use the event-flow model in a number of ways to develop an end-to-end GUI testing process, namely by checking the model, test-case generation.

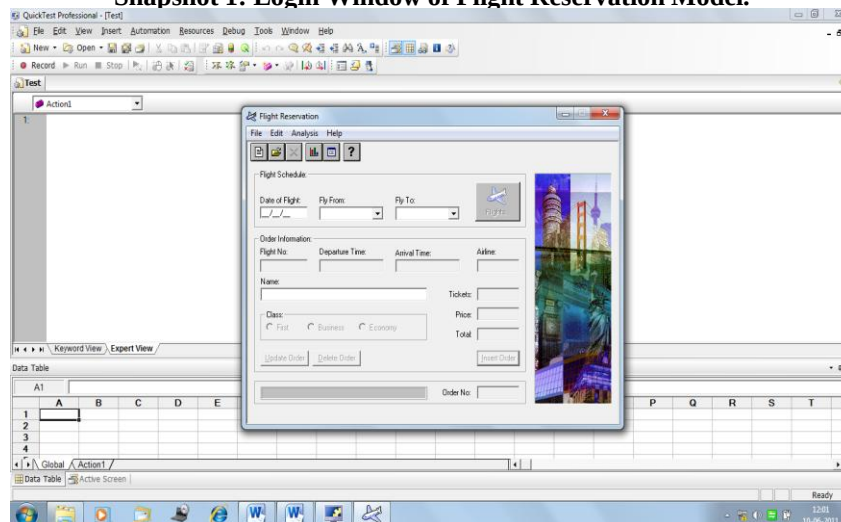
D. Benefits of Automated Testing

If you have ever tested applications or Web sites manually, you are aware of the drawbacks. Manual testing is time-consuming and tedious, requiring a heavy investment in human resources. Worst of all, time constraints often make it impossible to manually test every feature thoroughly before the application is released. This leaves you wondering whether serious bugs have gone undetected. Automated testing with Quick Test addresses these problems by dramatically speeding up the testing process. You can create tests that check all aspects of your application or Web site, and then run these tests every time your site or application changes. As Quick Test runs tests, it simulates a human user by moving the mouse cursor in a Web page or application window, clicking GUI objects, and entering keyboard input; however, Quick Test does this faster than any human user.

Snapshots from 1 to 4 show the results of above taken test cases: (i) Login Window of Flight Reservation system shown below help the user of application to login in to the system for booking of flight for any destination and also help to update and delete order. (ii) Flight Reservation Form: This form is designed to enter the customer information and flight information. (iii) Open Order Form is designed to search the customer record by using Customer Name, Date of Flight and Order Number. An Agent can search the guest record by the following search criteria.



Snapshot 1: Login Window of Flight Reservation Model.



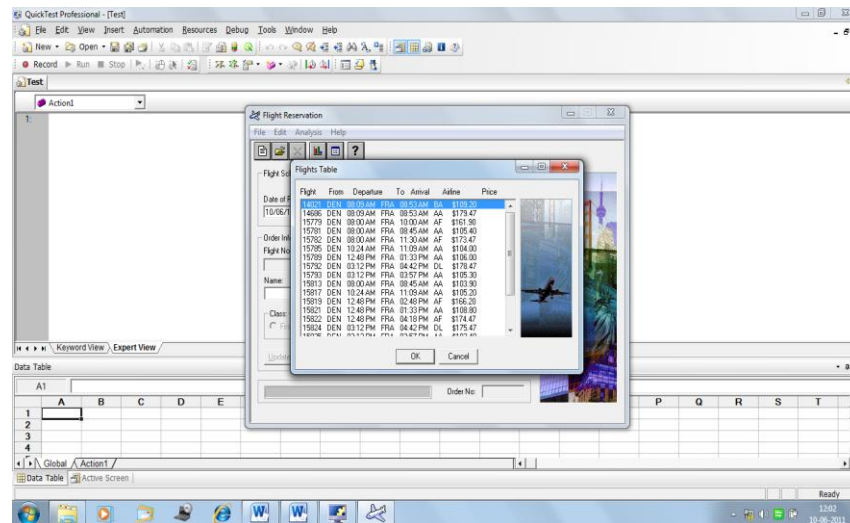
Snapshot 2: Flight Reservation Model.

Open Order Form is designed to search the customer record by using Customer Name, Date of Flight and Order Number. An Agent can search the guest record by the following search criteria.

E. Finite State Machine Model

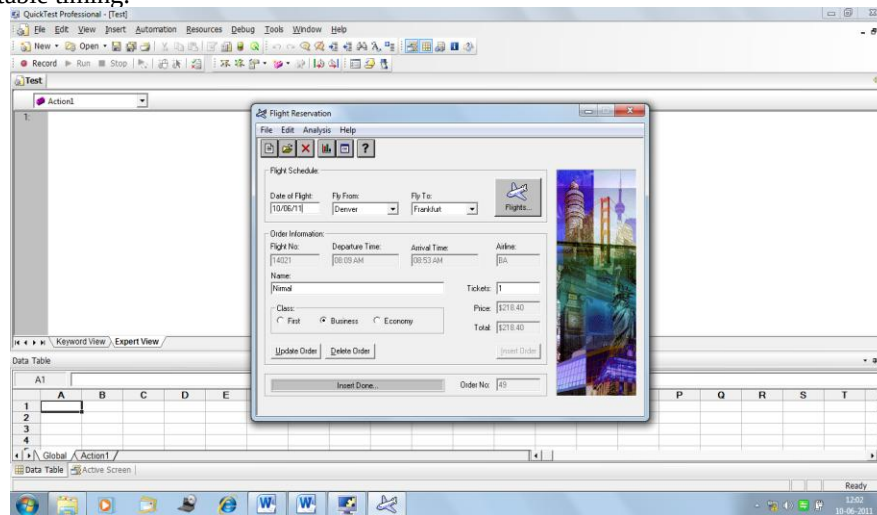
FSM provides a convenient way to model software behaviour and Web enabled systems can be completely modelled with FSM as the Web-based systems have the data. The Web testing is also categorized on the basis of static links, dynamic links, and operational transitions. FSM Model used here is a service provided by France telecom that does the opposite of the white pages - it allows you to enter a telephone number and find out the

name and address associated with the number. It can be very useful in conjunction with caller ID! It is a vocal service, which you use by dialling the telephone number and then listening to the vocal messages and pressing buttons on the keypad of your phone. Let's consider to do some independent testing of the Telecom System in order to check that its behaviour matches its requirements. Since the outputs of the telecom system are all spoken phrases and we want to use a real telephone as our input device, we decided not to try to automate the execution of the tests. Instead, we will aim to choose test generation techniques that produce a small set of tests so that manual execution of the tests is practical.



Snapshot 3: Open Order Form.

In the above snapshot, the search results of flight available from one location to another. We can select flight according to suitable timing.



Snapshot 4: Book Flight.

III. Conclusions

The main contribution of this paper is the development of a framework to support automatic generation of test cases that will show the presence of pre-defined security vulnerabilities. Our approach has been to separate the various concerns in order to model different security issues and specific vulnerabilities. This paper shows that an abstract model of a piece of software can be complemented with implementation details to allow the generation of adequate test cases. In future work we intend to investigate techniques that, given some low-level details of possible implementations, will mimic the behaviour of an intelligent attacker to generate test cases that cover a wide range of known attack variations over defined software vulnerabilities. We will also try to explore the techniques to generate a suitable model of the implementation automatically.

VI. References

- [1] Marius Nita & David Notkin, "White-Box Approaches for Improved Testing and Analysis of Configurable Software Systems" IEEE 2009.
- [2] Girish Janardhanudu, "White Box Testing" in 2009.
- [3] Gregory M. Kapfhammer "Software Testing" ACM 2008.

- [4] [Practical Model-based Testing: A Tools Approach](#), Mark Utting and Bruno Legeard, [ISBN 978-0-12-372501-1](#), Morgan-Kaufmann 2007.
- [5] Apfelbaum, Larry. "Model-based Testing", Proceedings of Software Quality, 1999.
- [6] Beizer, Boris. Black Box Testing: Techniques for Functional Testing of Software and Systems, New York, John Wiley & Sons, 1995.
- [7] Robinson, Harry. "Graph Theory Techniques in Model-based Testing", Proceedings of The International Conference on Testing Computer Software 1999.
- [8] L. du Bousquet, S. Ramangalahy, S. Simon, et al. Formal test automation: The conference protocol with TGV/Torx. In H. Ural, R. L. Probert, and G. von Bochmann, editors, Proceedings of the IFIP 13th International Conference on Testing of Communicating Systems, 221–228. Kluwer Academic Publishers, 2000.
- [9] D. Lee and M. Yannakakis. Principles and methods of testing finite state machines: A survey. Proceedings of the IEEE, 84(2):1090–1126
- [10] J. Hopcroft. An log n algorithm for minimizing states in a finite automaton. In Theory of Machines and Computation, 189–196. Academic Press, 1971.
- [11] G. J. Myers. The Art of Software Testing. John Wiley & Sons, 1979
- [12] M. Guan. Graphic programming using odd and even points. Chinese Mathematics, 1:273–277.
- [13] H. W. Thimbleby. The directed Chinese postman problem. Software: Practice and Experience, 3(11):1081–1096.
- [14] B. Yang and H. Ural. Protocol conformance test generation using multiple UIO sequences with overlapping. In Proceedings of the ACM Symposium on Communications Architectures & Protocols, 118–125. ACM Press, 1990.
- [15] A.M. Davis. Software Requirements: Objects, functions and states. PTR Prentice Hall, 1993.