International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 24 Issue 8 – MAY 2017.

ENHANCED GUI TEST SCRIPT MODEL IN SOFTWARE TESTING PROCESS

Selvakumari. N [#1] and Muthamizharasan. M [*2]

[#] *Assistant Professor, Department of Computer Science, Dharmapuram Adhinam Arts College, Dharmapuram, Mayiladuthurai, India.*

[*] *Associate Professor, Department of Computer Science, A.V.C. College (Autonomous), Mannampandal, Mayiladuthurai, India*

***Abstract*— In order to enhance the usage experience of the Graphical User Interface (GUI) systems, the innovation of the software systems is an important phase. The innovated software should be beneficial for the end users, developers and testers. In some cases, the developers intend to modify the GUI code, according to their usage. Henceforth, the testers should manually inspect the test scripts. This is time consuming and error prone. In this paper, we have proposed a novel test scripts which deals with the self-repairing of the low-level test scripts. An Event-Flow Graph (EFG) has been used for mapping the variant generated scripts. By this framework, we significantly reduced the cost of human interventions and repaired the low-level test scripts. Experimental results have shown the effectiveness of the proposed systems.**

***Index Terms*— Graphical User Interface (GUI), Event Flow Graph, low-level scripts, test scripts, and effectiveness.**

I. INTRODUCTION

Software testing is a process is to identify all bugs that exist in a software product. It is the process of evaluating all the components of a system verifies that it satisfies specified requirements or to classify differences between expected and actual results. Software testing is also performed to achieving quality by using the software with applicable test cases. Testing can be integrated at various points in the development process depending upon the tools and methodology used. Software Testing usually starts after requirements [1]. At a unit level phase, it starts concurrently with coding; whereas at integration level, it starts when coding is completed. Testing process can be performed by two ways that are manual or automation.

Manual testing is a process to test the software manually to find out the bugs. Manual testing is performed without using any automated tool [2]. While performing the manual testing a test plan is used that describe the systematic and detailed

approach of testing a software application. The goal of the testing is to make sure that the software application under test is defect free. Manual testing is not suitable for large projects as it requires more resources and time [3]. Automated testing is a process in which tools execute a pre defined scripted test on software to find defects. Automated software testing is the finest way to increase the effectiveness and efficiency of software testing. Automation testing can does what manual testing does not. Automation testing also improves the accuracy and saves the time of the tester & organization's money. It is best appropriate in the environment where the requirements are repeatedly changing & huge amount of regression testing is required to be performed [4].

Test automation interface is a platform which provides a single workspace for incorporating multiple testing tools. To improve the efficiency and flexibility of maintaining test scripts test automation interface is use [5]. It includes Interface Engine which consist runner to execute the test scripts, the Interface Environment that consists Framework and Project Library and Object Repository are collection of application object data.

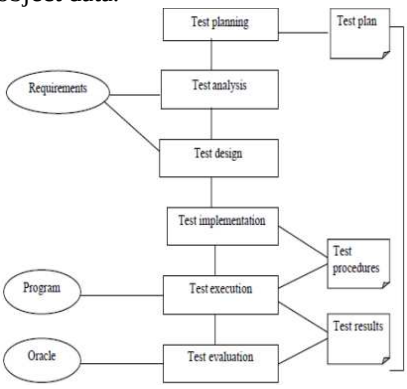

Fig.1. Testing process

The rest of the paper is organized as follows: Section II describes the related work; Section III presents the proposed work; Section IV presents the experimental analysis and atlast concludes in Section V.

II. RELATED WORK

This section presents the prior works of the test script frameworks. Software can be tested either manually or automatically [8]. The two approaches are complementary: automated testing can perform a large number of tests in little time, whereas manual testing uses the knowledge of the testing engineer to target testing to the parts of the system that are assumed to be more error-prone. Auto Test [6] is a testing tool that provides a "best of both worlds" strategy: it integrates developers' test cases into an automated process of

systematic contract-driven testing. This allows it to combine the benefits of both approaches while keeping a simple interface, and to treat the two types of tests in a unified fashion: evaluation of the result is the same, coverage measures are added up, and both types of tests can be saved in the same format.

The test preparation phase includes test plan preparation, test case, test data and test environment preparation. The test plan is the first document to be prepared, which outlines the scope, objectives, features to be tested, features not to be tested, types of testing to be performed, roles and responsibilities of testing team, entry and exit criteria and assumptions[7]. Simultaneously the testing teams start preparing test cases and test data. A test case is a document, which outlines steps required to test any functionality with expected and actual result. If actual result doesn't matches with expected result, then a bug is opened. For each requirement, positive and negative test cases are prepared, which is ensured by requirement traceability matrix (RTM). RTM is a document which maps requirements with test cases to ensure 100% testing is done.

All valid and invalid test data sets are to be prepared for each test case and a test data document is prepared. Test data is also generated based on some algorithm and tools. Test case preparation [8] has various steps which start with Test case generation, Test case selection, Evaluation, and Test case prioritization [9]. There are various algorithms which are used to generate and optimize test cases [10].

III. PROPOSED WORK

This section presents the enhanced test scripts model. It consists of four modules, namely,

A. *System construction:*

Project Management System (PMS) is the simple project that assists to create and manage the projects. Under this framework, the PMS has two views, namely, a) Default screen settings and b) Information of the project. The two views are from the event sequence hfile, Open Projecti under different dialogs. In this perspective, any sorts of members can be added or modified. The Graphical User Interface (GUI) imposes certain constraints based on its members.

B. *Repairing process:*

The repairing process verifies all the testing scripts. By doing so, an events process mapped to its relevant graphs. It generally frames a valid path hCreate project. The same process is executed for all the test scripts projects. Since, it's an event based testing scripts. Before reaching the NULL value, shortest path algorithm is widely used for determining the possible routes. Each event consists of roles with two solutions, namely, no event and role. This information is stored in the mapping to be reused for subsequent repairs. TS2 is still unusable because the EFG model does not contain the edge (Email, Finish Member). The human tester is asked to confirm the existence of this edge.

C. *Ripping:*

Ripping is the process of enhancing the abstraction level of deployed GUI version into an abstract model. The depth-first search algorithm is deployed for automatic detection of the relevant GUI applications. The attributes involved in the GUI is used for EFG construction. Other windows in the GUI that do not restrict the user's focus are called modeless windows; they merely expand the set of GUI events available to the user.

D. *Mapping:*

Mapping is the process involved between low level GUI objects and the logical events. From the ripping process, widget's location is discovered with its container and the created class. Using these ripped features, widgets are located with their event's names. For example, the statement JavaWindow ("PMS"). JavaButton ("Add").click is parsed to obtain two object types: JavaWindow and JavaButton, and their corresponding title: PMS and Add. These are then searched in the mapping from the ripper and represented using their logical form. If a match is not found that a NULL entry is created.

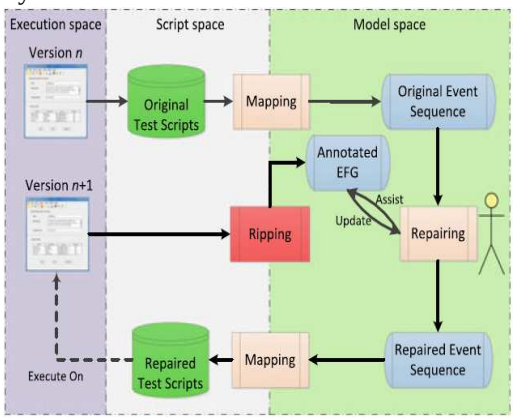

Fig.2. Proposed architecture

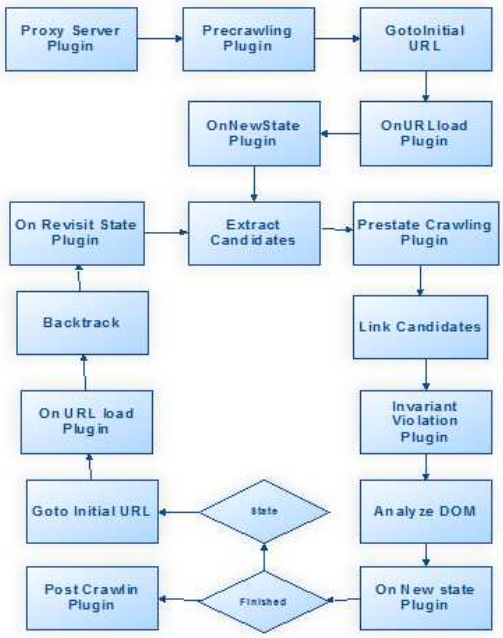

Fig.3. Data flow diagram of the proposed model

IV. EXPERIMENTAL RESULTS

This section represents experimental analysis of the proposed model.

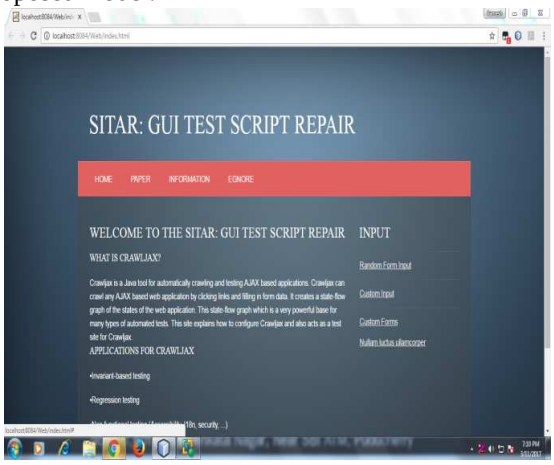

Fig.4.Sample GUI template

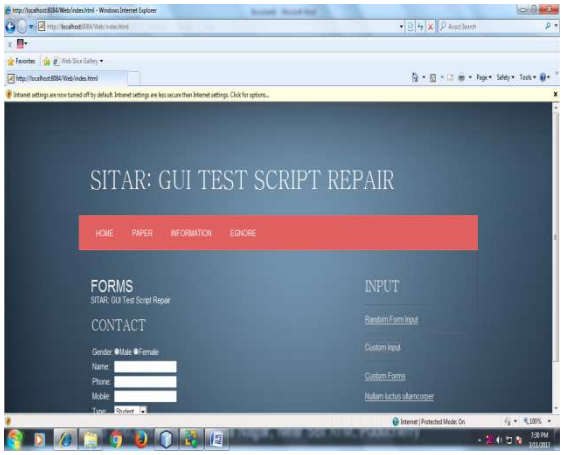

Fig.5. Automatic testing performance over the sample GUI systems.

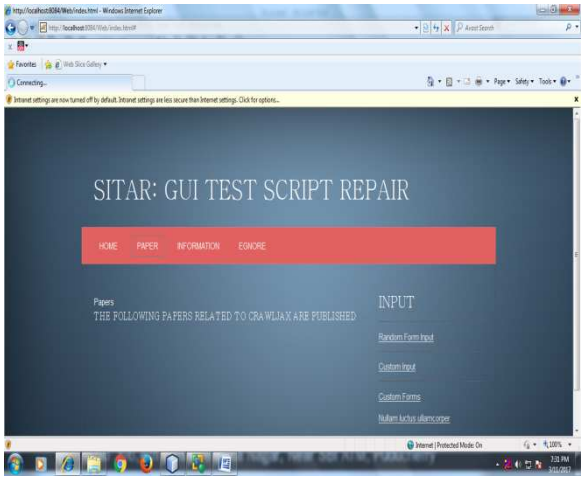

Fig.6. Tester will automatically test each subfield.

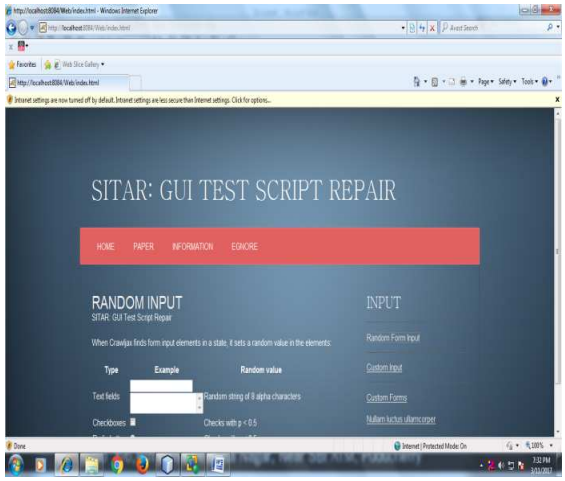

Fig.7. Entering the random inputs for activating the testing functions.

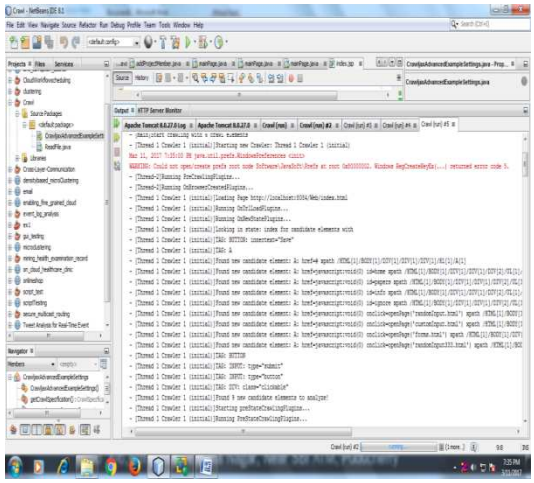

Fig.8. Output of the testing functions

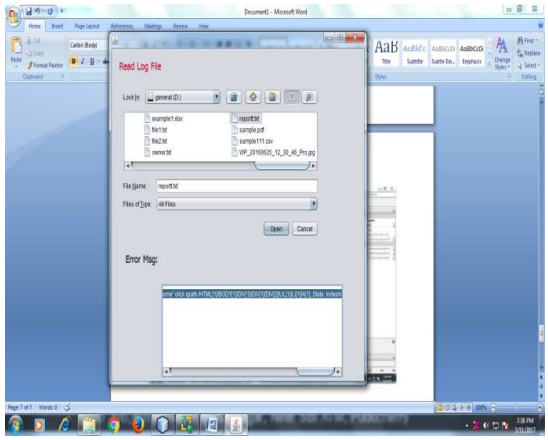

Fig.9. Displaying the error message in Sample GUI template

V. CONCLUSION

Software testing is an important phase in the Software Development Life Cycle (SDLC). This is one of non-functional testing types which test performance of software under all favorable and non- favorable conditions. This includes all time related parameters like Load time, access time, run time, execution time etc. This also includes success rate, failure frequency, mean time between failures and overall reliability of software. In this paper, we have proposed an enhanced test scripts which deals with the self-repairing of the low-level test scripts. An Event-Flow Graph (EFG) has been used for mapping the variant generated

scripts. By this framework, we significantly reduced the cost of human interventions and repaired the low-level test scripts. Experimental results have shown the effectiveness of the proposed systems.

REFERENCES

[1] Zebao Gao et al, "SITAR: GUI Test script repair", IEEE transactions on Software Engineering, 42 (2), 2016.
[2] Swain,Kumar,Santosh.Mohapatra,Durga,Prasad.Mall,Raj ib.2010.Test Case Generation Based on Use case and Sequence Diagram.International Journal of Software Engineering(IJSE). Swain et al.3,2(July 2010).
[3] Akhilesh,Babu, Kolluri.K, Tameezuddin.Kalpana, Guddika dula.2012.Effective Bug Tracking Systems.Theories and Implementation", IOSR Journal of Computer Engineering ISSN:2278-0661 Volume 4,Issue 6(SeptOct 2012), pp 31-36.
[4] Rina,DCSK,KU,Haryana,INDIA,Tyagi,Sanjay.DCSA,K U,Haryana.2013.AComparative Study Of Performance Testing Tools.IJARCSSE. 3,2(May 2013).
[5] Karen ,Scarfone.2012. Intro to Information Security Testing & Assessment. ScarfonecyberSecurity Csr.nist.gov.(7June 2012).
[6] B,Beizer.1990. Software Testing Techniques.Technology Maturation and Research Strategies Carneige Mellon University Pittsburg,USA.
[7] B.Beizer .1995.Software Testing Techniques.2006.Van Nostrand Reinhold,New York.1990.ISBN.0-442-20672- 0.(31.Oct.2006).
[8] A,Bertolino.2001.Chapter 5: Software Testing . IEEE SWEBOK trial version 1.00.IEEE (May 2001).
[9] Khan,Mohd.Khan,Farmeena.2012.A Comparative Study of White Box, Black Box and Grey Box Testing Techniques.2012. International Journal of Advanced Computer Science and Applications(IJACSA). Vol. 3.No.6.(2012).
[10] Tarika, Bindia. Computer Programmer CSE,GNDEC, Ludhiana, Punjab-India.IJRITCC.2,1 .68- 72.(2321-8169).