Graphical Testing of Software Development

Supriya[1], Manish Mahajan[2]
[1] Research Scholar, Department of Information Technology, CEC Landran.,
[2] Associate Professor, Department of Information Technology, CEC Landran.,

ABSTRACT: Graphical software testing is an important process to utilize the easy process for testing. A process of executing a program with the goal of finding errors is known as testing process. So, testing means that one inspects behavior of a program on a finite set of test cases (a set of inputs, execution preconditions, and expected outcomes developed for a particular objective), In these days size and complexity are most essential parts of software to take care of while doing testing. Manual testing becomes more difficult due to limitation of time and resources. As the size and complexity of software is continually growing, manual testing becomes very difficult and tedious task. In this paper we have done experimentation based on the testing cases prioritization process.

Keywords: *Software Development Life Cycle, Software testing, Test case priority, Graphical Interface*

1. INTRODUCTION

GUI testing is an active research area. The open challenge is the judicious generation of event sequences (an event sequence encodes a user interaction). A major advance in this direction is the use of a black-box model to systematically generate event sequences that are executable on the GUI. The black-box model can be, e.g., an Event Flow Graph (EFG) or an Event Sequence Graph (ESG). In this paper we propose a new approach to select relevant event sequences among the event sequences generated by a black box model. We express the relevance of an event sequence by a precisely defined dependency between a fixed number of events in the event sequence. Departing from a pure black box approach we apply a static analysis to the byte code of the application. This allows us to infer a dependency graph, which we call Event Dependency Graph (EDG). We use the EDG together with a black-box model to construct a set of relevant event sequences among the executable ones. We have implemented our approach in a new tool. We evaluate the approach on four open source GUI applications. With the specific choice of a lightweight static analysis, the approach scales to large applications and, at the same time, leads to an informed selection of event sequences. Using our approach we are able to find previously undetected bugs [5].

2. GRAPHICAL FRAMEWORK

We are developing Graphical testing based tool as a complete framework for testing of Java software. The components in the framework, including a test specified a test planner, a test virtual machine (TVM), and a test analyzer. One component is a language, test spec, for specifying a software test process. The specification includes the relevant parts of the program to be tested and the actions needed in the testing process. Testers can either write a specification in test spec or, better, use the GUI, which automatically generates a specification in test spec. A test planner consumes the test spec specification and determines a plan for testing the program given the specification.

3. PROBLEM DESCRIPTION

Our Test cases design is store in the database that is the Excel with the number of the input and the actual value of the output. whichever the value is going to enter by the are going to match from that database the which test case is supposed to be passed will be shown in the graphical user interface. Except from the only the number of the selected input the user can also give the any number of the input it could range from 10 to 1000 we are providing the module of the automate testing in which the user need to upload the file with any number of the input and all the test cases that are pass and fail will be shown in the new interface. The Priority of the test case is also taken as the major module of our work for showing the priority to the test cases the user is provided with the Interface with the number of the test cases shown in the checkbox the user has to select only the number of the test cases that is user want to execute first once the user click on the desired checkbox the .The Particular Test case is executed and the execution of the test case is done as according the user click the checkbox.

4. PROPOSED WORK

This research has focused on providing solution for said problem by enhancing some term of the testing process. For experimentation we have used database with various Test cases entries and with the input and the expected result. Making the suitable GUI is the first process for testing that GUI includes all the text boxes radio button checkboxes. our testing module also include the feature like testing the entire source code and only testing the specific part of the code. Suppose we are entering only the values in the text box ad getting the desired output in that case only the only the action listener of the specific button will be called the rest of the source code will be left unchecked for that we are also performing the entire source code and other part of the source code.

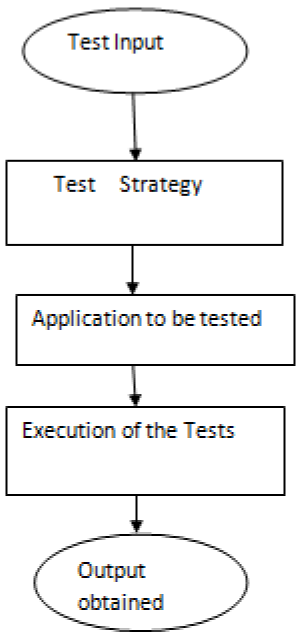


Figure 1: Proposed Test Approach

. The Major Module of the Project includes the Priority of the Test cases in this the user can only test those cases which the user feeling is might be important as compare to the rest of the test cases.so that he can skip the rest of the test case. The output of the cases according to the priority will be displayed .or we can specify that the testing is done at the following ways There are two general approaches to test automation: a. Code-driven testing: The public interfaces to classes, modules, or libraries are tested with a variety of input arguments to validate that the results that are returned are correct. b .Graphical user interface testing: A testing framework generates user interface events such as keystrokes and mouse clicks, and observes the changes that result in the user interface, to validate that the observable behavior of the program is correct.

Below Figure shows the Graphical User interface with the various components that are radio button representing the function of changing the Feet to Inches and Inches to feet option and the button showing the four different modules.

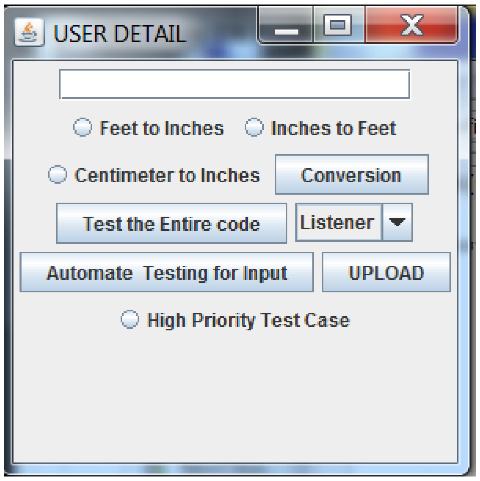
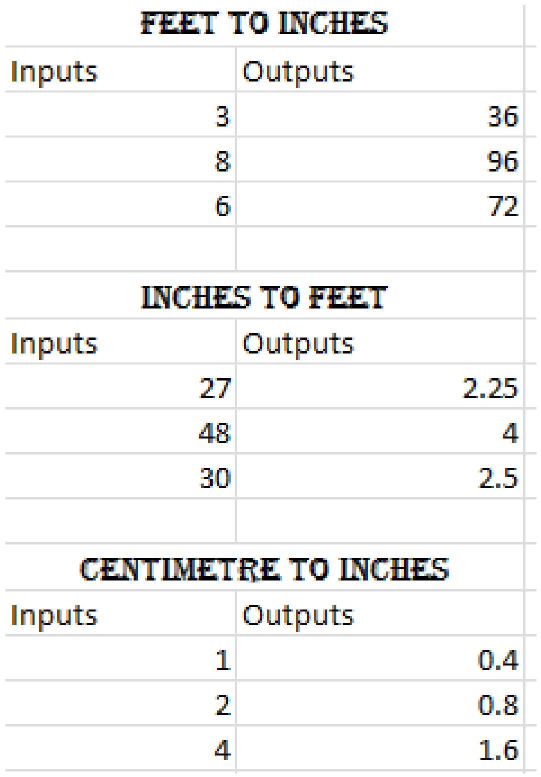


Figure 2: Graphical User Interfaces for the complete software testing module

The Graphical tool developed works from a graphical portal which is shown in figure 2 above. We have different options which could be used for user input; software input and gives results according to the various options available.



FEET TO INCHES	
Inputs	Outputs
3	36
8	96
6	72
INCHES TO FEET	
Inputs	Outputs
27	2.25
48	4
30	2.5
CENTIMETRE TO INCHES	
Inputs	Outputs
1	0.4
2	0.8
4	1.6

Figure 3: Output processing of conversion base on proposed testing

The conversion process of feet to inches with input provided to proposed system and output given by the proposed system based on testing process. Figure 3 provides us good idea of conversion process and testing concerns which need to be take care in software testing process.

5. CONCLUSION

In proposed work, we put more work on making a part of the code in which it is tested for the both testing the GUI Interface and testing of the source code also with providing the priority to the test cases that will be executed according to the choice of the user. In this proposed work, followed approach is different from other technique because it follows the hypothesis of testing screen transitions of a web application through its functional testing and the different functionalities of the application can lie at any layer except the user interface layer and not only on database layer. Also our approach is targeting the limited time a tester has in order to obtain a certain level of confidence about correctness in Testing. We have done some more analysis based on the test cases priority process.

REFERENCES

[1]http://www.webopedia.com/TERM/S/static_routing.html, date last viewed: 2012-10-11.

[2] Routing protocols and concepts, CCNA exploration companion guide. ‘‘Introduction to dynamic routing protocols’’. Chapter three pages 148 to 177.

[3] http://www.ietf.org/rfc/rfc2501.txt , date last viewed: 2012-10-11.

[4] Jing Xie, Luis Girons Quesada and Yuming Jiang, ‘‘A Threshold-based Hybrid Routing Protocol for MANET’’. Department of Telematics, Norwegian University of Science and Technology.

[5] S. Gowrishankar, T.G. Basavaraju, M. Singh, Subir Kumar Sarkar, ‘‘Scenario based Performance Analysis of AODV and OLSR in Mobile Ad hoc Networks’’, Jadavpur University, Acharya Institute of Technology India.

[6] Lijuan Cao Kashif Sharif Yu Wang Teresa Dahlberg, ‘‘Adaptive Multiple Metrics Routing Protocols for heterogeneous Multi-Hop Wireless Networks’’, Department of Computer Science, University of North Carolina at Charlotte, Charlotte, USA.

[7] Ammar Zahary and Aladdin Ayesh, ‘‘Analytical Study to Detect Threshold Number of Efficient Routes in Multipath AODV Extensions’’, Faculty of Computing Sciences and Engineering De Montfort University Leicester, LE1 9BH, UK.

[8] Deep Kaur ,Kirandeep kaur, Vishal Arora, “QoS in WLAN using IEEE802.11e”, 2012 Second International Conference on Advanced Computing & Communication Technologies, IEEE Computer Society, 2012.

[9] Seyed Ali Hosseini, Hamid Farrokhi, PhD,” The Impacts of Network Size on the Performance of Routing Protocols in Mobile Ad-Hoc Networks”, Second Pacific-Asia Conference on Circuits, Communications and System (PACCS), pp-18-22, volume-1,IEEE, 2010.

[10] P.Kuppusam, Dr.K.Thirunavukkarasu, Dr.B.Kalaavathi, “A Study and Comparison of OLSR, AODV and TORA Routing Protocols in Ad Hoc Networks”, pp-143 – 147, volume-5, ICECT, IEEE, 2011.

[11] Sung-Hee Lee, Young-Bae Ko, Youg-Geun Hong, and Hyoung-Jun Kim, ‘‘A New MIMC Routing Protocol Compatible with IEEE 802.11s based WLAN Mesh Networks’’, pp-126-131, ICOIN, IEEE, 2011.

[12] B.Soujanya, T.Sitamahalakshmi,” Study Of Routing Protocols In Mobile Ad-Hoc Networks”, International Journal of Engineering Science and Technology (IJEST), Vol. 3 No. 4, pp 2622-2631 April 2011.