Volume 5, Issue 7, July 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Component Based Testing for GUI

Kanika Rani
Mtech Student(C.S.E)
JMIT Radur, Kurukshetra University
Haryana, India

Vivek Sharma
Assistant Prof.(Mtech)
JMIT Radur, Kurukshetra University
Haryana, India

Abstract: Testing is an important part in software development. When a component based software system are established then to ensure the quality of system, testing must be done.Components have implementation limpidity property andthey are assorted in nature. When the components are integrated to make a system then unexpected result may occur. So integration testing is necessary when components are integrated. In this paper, a student record system is implemented and its components are tested using abbot tool. Abbot tool is used for integration/functional testing. Abbot builds upon the java.awt.Robot class to provide an automated event generation and validation framework for Swing GUI components. The framework can be used to create, record, and execute scripts and programmatic test cases in Java. Abbot also has a script editor called Costello that facilitates the creation of scripts in XML. This paper also describes the features of abbot tool.

Keywords: Script Editor, GUI Component Testing, XML Script, Abbot Tool, Costello.

I. INTRODUCTION

Software testing is one of the most important part in software development procedure. Every programmer knows that testing is essential in the development of software application.Component based software systems are mainly constructed from reusable components such as third party components. Due to these reusable,component based system are developed rapidly, easily and with least resource cost [1]. Component-Based Development (CBD) offers a radically new approach to the design, construction, implementation and development of software applications. Software applications are implemented by assembling the several components; the components may be written in several different programming languages and run on several different platforms. Conventional development is a special case of CBD, which lacks some of the techniques and opportunities that characterize full CBD.To ensure the best supply of component based software, effective and efficient testing is the key process in software development. Component Based Software has implementation transparency property and they are written in numerous different programming languages and run on several different platforms. This heterogeneous nature of components increases difficulties in testing. In software development, several components are reused to build a system. Then the components must be tested effectively. Most of the testing tools are available today for GUI based testing like Marathon, Guitar, Abbot, HTML Unit, HTML Fixture, Selenium. Marathon, Guitar and Abbot are java based tools where as HTML Unit, HTML Fixture, Selenium are web based tools. Junit tool is used for unit testing. Junit is a unit testing framework, which means that junit works by taking the smallest part of testable software, this tool isolate that part from rest of the software and in turn authenticating that it works correctly or not. Aim of this paper is component based integration testing by using testing tool – Abbot that is suitable for integration testing. It checks all the components together. Abbot framework is a java library that provides methods to replicate user actions and examine the state of GUI components. The framework may be invoked directly from java code or accessed without programming through the use of scripts. Abbot tests the java UI. It provides a Costello script editor which is built on Abbot. Costello allows the tester to easily launch, explore, and control an application. Framework may be used with both scripts and compiled code. Costello script editor runs the xml scripts. In this paper, student record system is implemented and all the components are tested. For testing these components abbot tool is used which provides an automated event generation and validation framework for GUI components. In this paper main persistence is to test the functionality of all the components that they are working properly after integrated. The next section presents background information on the test generation. Section 3 describes the proposed work. Section 4 describes the implementation work done and results. Then section 5 conclusion and section 6 future work.

II. RELATED WORK

In the past, many papers have been published to address the component based testing and issues related to component based software engineering. Testing of component based system is different from normal software testing that‟s why the testing techniques are also bit different. There are basically two different methods for testing white box and Black box. The black box testing is more prevalent in the component based systems because in most of the cases the source code is not available with the component; at most only the specification is available. Several techniques are defined to test the component based system. Some of them are defined below.

2.1 Adequate testing

An "Adequate Testing" method for testing component based software was suggested by David S.Rosenblum [2] this technique provide initial basis for testing of component based software. The main result of this technique is the formal definition of the concept c-adequate-for p for adequate unit testing for the components and c-adequate –for m for integration testing for component based system.

2.2 Boundary value analysis

Muthu Ramchandran[4] suggested "Boundary Value Analysis" for testing CBS. It is a software testing method. In this technique test cases are designed in such a way that they include representatives of boundary values. It is based on complex embedded software and it also helps to apply boundary value testing and study on component interfaces which is vital to achieve testability of reusable software components. It uses visual notations to understand component connections/compositions.

2.3 Integrated testing technique

Sami Beydeda and Volkar gruhn et.al [3] proposed "Integration Testing Technique". In this all the features of component based system are tested and it allows more rigorous testing. The main purpose of this method is to give graphical representation which combines the black and white box information. This graphical representation can then be used for test case generation. But this technique has one flaw that is very time consuming approach and there is no provision for automatic test case generation. This technique has one fault that is very time consuming approach and there is no provision of automatic test case generation. Our next part shows the automatic test case generation testing techniques. After this automated testing techniques are projected.

2.4 Automated software robustness testing

The movement shifted towards automatic testing of software components Marcel Dix Holger et al. [5] suggested "Automated Software Robustness testing". Main purpose of this technique is to do the automated robustness testing. Robustness is defined as the components capability to handle invalid input conditions. This technique can generate huge number of test cases with small input values. But there is demand for reducing this large mass of test cases so another technique that is static testing was introduced for test case reduction. It reduces the number of test cases to be executed without affecting test accuracy and reliability but it was unable to reduce the numbers of test case and to generate test cases, human efficient expertise is desirable.

2.5 Self testing of component based system

"Self Testing" was suggested by Fevzi Belli et al. [6]. It is widely accepted that conventional test methods are not necessarily suitable for testing of component based software. Also conventional test tools cause same problem for the automation of the test CBS, because the knowledge about the implementation of the CBS are essential to run the tests. But the component manufacturers are not willing to provide the component source code. This technique is based on black box testing and uses some types of capture/playback tools. It provides a framework for computerization of user oriented component testing which reduces the test cost.

2.6 Regression testing

Chengiying Mao et.al[7] suggested "Regression Testing Technique". It is based on change information. Due to lack of information about source code of externally provided component system tester can"t perform real regression testing on their component based system. Component users don"t know the details of change in components so they are not able to select the proper test cases to test modified components. This technique provides a regression testing method based on greater change information of component version to test the modified components. It wants a joint participation of component developer and the users. Call for graph is used to calculate the change information.

2.7 Modular regression testing

GUI test cases can be unusable, usable or repairable. In RTCM we have followed the principle "do not throw away unusable test cases".Bruce W.Weide [10] proposed "Modular Regression Testing". Whenever the modifications are done in the software system then regression testing must be done. On the release of every new version, new test cases are generated. It is too luxurious to generate the new test cases for every version. Modular regression testing provides an advantage over regression testing. In this RTCM is used. RTCM reduces this cost by providing the concept of reusability. Regression Test Case Modeler (RTCM) is a framework that generates the test cases automatically from event forest and also generates regression test suite. RTCM choose old test suite which represent correct input and is necessary to validate the modified software. When the structure of the original GUI is modified, test cases from the original GUI are either usable or unusable on the modified GUI. Test cases which can"t be used in modified GUI are discarded and test cases which can be used in modified GUI are kept. Test cases which can"t be repeat are known as obsolete test cases.

2.8 Object oriented component testing

Fakhra Jabeen et.al[8] suggested "Object Oriented Testing". The absence of source code overtures inferring standard testing approaches. In this technique object oriented framework is proposed that relies on utilization of discrete descriptors to enable test execution and to enable a uniform information flow. At present it supports component unit testing and partial integration testing. Junit tool is used for unit testing. Abbot tool is a most popular tool and used for integration/functional testing.Its features include:

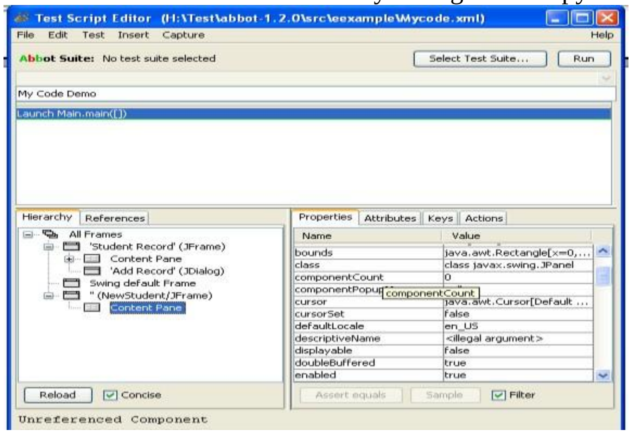
- Abbot is an extension of junit tool and is used for integration/ functional testing.
- Allows to write test cases directly from java code.
- Allows to insert assertions in script easily.
- It runs the XML scripts then there is no need of compilation.
- Provides GUI tests for java AWT/swing application.
- Consists of recorder, player and an editor.
- Records tests script in java.

III. PROPOSED WORK

Software testing is a critical element of any kind of software‟s quality assurance and is extreme important for the credibility of the software system. When it comes to component based software paradigm, testing mechanism and strategies varies from the traditional testing approaches because traditional software system contained source code which can be tested by tester. Automatic testing of software is necessary to save the time and to reduce the labour of programmers. In software development unit testing is the starting phase of testing. First of all unit testing is done. In unit testing a single piece of software/single component is tested by the tester. Junit tool is used for automatic unit testing. This tool works on java platform. Integration testing comes after unit testing and is most important testing in software development. Several components are integrated to make a system. Testing of independent component is simple but when the components are assembled in a system then unexpected results may occur. Then to confirm the quality of the system, integrated testing is necessary. Abbot tool is used for integration testing and it is an extension of junit tool or can be says that a better bot.The final goal of this paper is to completely automate the testing of integrated components. A programmer should be able to perform testing simply by clicking on a button. Abbot tool provides GUI integration/functional testing and test cases are generated automatically by this tool. In this paper, employee record system is implemented and all the components are tested together. Next section describes the implementation and results of the work.

IV. IMPLEMENTATION AND RESULTS

Abbot tool provides a framework to test the GUI components. Abbot builds upon the java.awt.Robot class. It provides an automated event generation and validation framework for Swing GUI components. The framework can be used to create, record, and execute scripts and programmatic test cases in Java. Abbot also has a script editor called Costello that facilitates the creation of scripts in XML. This tool converts the java source code into .jar (java archive files) files which are executable files and path of these files are associated with xml files and give them to abbot then abbot automatically produces test cases. A employee record program is implemented in java. After this xml scripts are created and these scripts are used by Costello script editor to run the applications. To test this program open the Costello (script editor), then LaunchMain.main([]) method occur. With the help of this method tester can run XML script. This method is shown in the Fig.1. This fig also shows the hierarchy, refrences, properties, attributes.etc. If the hierarchy of any component is changed then the script will break. This will produce an error in the system. Then to remove this type of error, script must be restructured after every change in the pyramid of components.



Abbot Suite: No test suite selected

My Code Demo

Launch Main.main([])

All Frames
'Student Record' (JFrame)
Content Pane
'Add Record' (JDialog)
Swing default Frame
" (NewStudent/JFrame)
Content Pane

Name	Value
bounds	java.awt.Rectangle[x=0,...
class	class javax.swing.JPanel
componentCount	0
componentPopup	componentCount
cursor	java.awt.Cursor[Default ...
cursorSet	false
defaultLocale	en_US
descriptiveName	<illegal argument>
displayable	false
doubleBuffered	true
enabled	true

Unreferenced Component

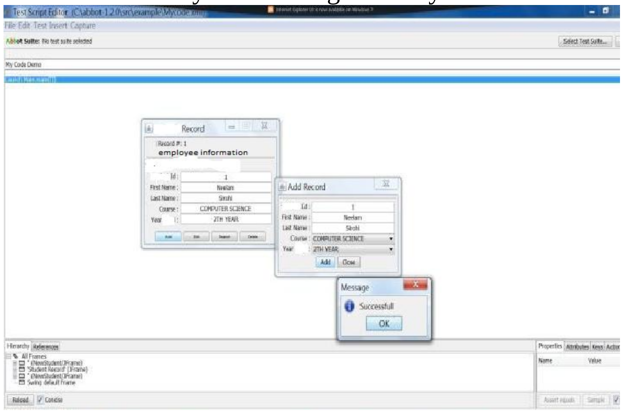
Fig.1 Costello script editor. It shows Launch Main. Main ([]) method which run xml script of employee record.

Example of abbot test code for employee record is shown below. This java code is used for inserting information about student.

```
import java.awt.*; import java.awt.event.*; import javax.swing.*;
public class Add extends AbstractAction
{
        private JDialog dialog;
        private JFrame frame;
```

```
            public Add(String commandAction, JFrame frame)
            {
                        super(commandAction);
                        this.frame = frame;
            }
public void actionPerformed(ActionEvent event)
            {
                        addNewEmployee();
            }
                        public void addNewEmployee()
            {
                        dialog     =     new     JDialog(frame,"Add
Record",true); dialog.setContentPane(new NewEmployee(dialog).getPanel());
                        dialog.pack();
                        dialog.setLocationRelativeTo(frame);
                        dialog.setVisible(true);
            }
}
```

After click on run button employee record window open in front of tester then tester can check all the validation check which put on every component for testing. If all the test cases are passed without any error then result is shown in bottom of costello like invoking test case.. done as shown in fig 2. Tester also able to tell what test failed because it is highlighted in red line and error message appears in bottom of Costello. In this, all the textboxes, dropdown list and command buttons i.e add, close are tested that they are working correctly or not.


Test Script Editor (C:\abbot-1.2.0\src\example\Mycode.xml)

File Edit Test Insert Capture

Abbot Suite: No test suite selected

My Code Demo

Record

Record #: 1
employee information

Id : 1
First Name : Neelam
Last Name : Sirohi
Course : COMPUTER SCIENCE
Year : 2TH YEAR

Add Record

Id : 1
First Name : Neelam
Last Name : Sirohi
Course : COMPUTER SCIENCE
Year : 2TH YEAR

Add Close

Message

Successfull

OK

Hierarchy References

All Frames
"(NewStudent(JFrame)
"Student Record" (JFrame)
"(NewStudent(JFrame)
Swing default Frame

Reload Concise

Properties Attributes Keys Actions

Name Value

Fig.2 Testing of addition component using abbot tool.

```
        import java.awt.*; \
        import java.awt.event.*;
        import javax.swing.*;
        public class Delete extends AbstractAction
        {
Private EmployeeList studentList = newEmployeetList();
private JFrame frame =null;
public      Delete(String     commandAction ,JFrame Frame)
{
                        super(commandAction);
                        this.frame = frame;
                           }
                           public void actionPerformed(ActionEvent event)
                           {
                                    if(employeeList.getList().size() != 0)
                                                                        {
            Int   n    = JOptionPane.showConfirmDialog(frame,"Are  you  sure?  you
want     to  it'","Confirm delete",JOptionPane.YES_NO_OPTION);
if(n == JOptionPane.YES_OPTION)
            {
                         deleteEmployeeRecord();
            }
            }
            }
```

```
          public void deleteEmployeeRecord()
          {
employeeList.getList().remove(employeeList.getRecord(
                    if(employeeList.getList().size() !=
          {
                    Employee employee              = (Employee)employeeList.getList().getLast();
          employeeList.setRecord(employee.getList().size()    -
1);
new EmployeeRecordGUI(employeeList.getRecord() + 1);
                                                                      new
                              EmployeeRecordGUI(employee.getFirstName(),
          employee.getLastName(), employee.getIdNum(), employee.getCourse(),
          employee.getYearLevel());
                                        }
                                        else
                                        {
                    new EmployeeRecordGUI(false);
          new EmployeeRecordGUI(0);
          new EmployeeRecordGUI("","",0,"","");
                                        }
}
```

Similarly delete, search and edit components are tested by this tool. This is an example of employee deletion. After click on delete button of application a message box appears in front of tester into the figure "Are you sure? you want to delete it‟ ". Two command buttons appears, yes or no. This is shown in fig 3.

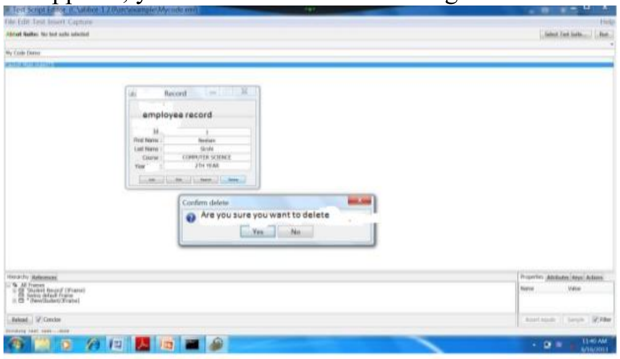

Record
employee record
Confirm delete
Are you sure you want to delete

Fig.3 Testing of delete component of employee record.

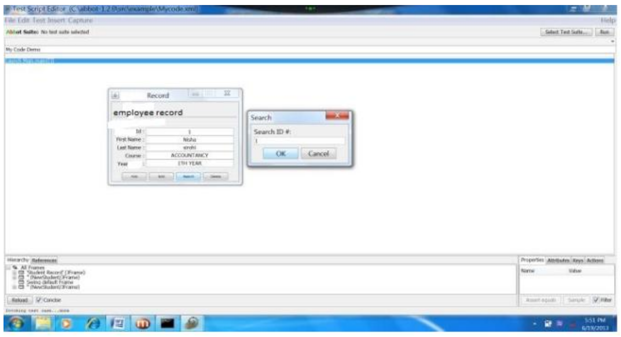

Record
employee record
Id : 1
First Name : Nisha
Last Name : sindhi
Course : ACCOUNTANCY
Year : 1TH YEAR
Search
Search ID #: 1

Fig.4 Testing of search component of employee record

From the above results it is sure that employee record application works correctly. All test of components are passed by abbot.

V. CONCLUSION

This paper presents a systematic integration testing using abbot tool. Abbot tool provides integration/functional testing and it is an extension of junit. It helps writing system/GUI test for java AWT/swing application. Abbot tool automatically generates the test cases. We also describe the role of Costello script editor. Costello allows us to easily launch, explore, and control an application. The framework may be used with both scripts and compiled code. Costello script editor can also run xml scripts and results are shown in the bottom.

VI. FUTURE WORK

This paper discusses the features of abbot testing tool. employee record application is implemented in java and observes that how abbot testing tool can help in testing the components of this application. This paper presents the testing of static components. Dynamic objects are another thing which keeps on changing like we see on some of the website, text logo‟s keeps on changing in the same area. In future some work can be done on dynamic components.

REFERENCE

[1] Jim Q Ning, 1997. Component-Based Software Engineering, Proceedings of U.S National Institute of Standards and Technology‟s Advance Technology Program on Component Based Software, Document number 0-8186-7940-9/97.

[2] David S. Rosenblum, 11 Aug 1997. Adequate Testing of Component-Based Software, Technical Report 97-34 Department of Information and Computer Science University of California, Irvine, CA 92697-3425.

[3] Sami Beydeda and Volker Gruhn, IEEE Computer Society 2001. An Integrated testing Technique for Component-Based Software.

[4] Muthu Ramchandran, 2003. Testing Software Components using Boundary Value Analysis, proceedings of the 29th EUROMICRO conference New Waves in System Architecture" (EUROMICRO 08), Document number 1089-6503/03, IEEE Computer Society.

[5] Marcel Dix. Holger and D. Hofmann, 2002. Automated Software Robustness Testing, proceedings of the 28th Conference (EUROMICRO‟02) IEEE, Document number 1089-6503/02.

[6] Fevzi Belli and Christof J. Budnik, 2005. Towards Self- Testing of Component-Based Software, Proceeding of the 29th Annual International Software and Applications Conference (COMPSAC ‟05), IEEE Computer Society.

[7] Chengying Mao, Yansheng Lu, 2005. Regression Testing for Component-based Software Systems by Enhancing Change Information, Proceedings of the Asia-pacific Software Engineering Conference (APSEC‟05), Document number 0-7695-2465-6/05.

[8] Fakhra Jabeen, Muhammad jaffar-Ur Rehman, 2005. A Framework for Object oriented Component Testing, Document number 0730-3157/05, IEEE press.

[9] Nitin V. Koppalkar, Seshaiah Uppala and Mahesh Madugundu. Testing of Component-Based Software Systems, Philips Research India-Bangalore.

[10] Bruce W.Weide, 2005. Modular Regression Testing, Connections to Component-Based Software, Dept. of Computer and Information Science, The Ohio State University , 2015 eil Ave. Columbus, OH 43210,USA+1 614 292 .

[11] A. M.Memon and M. L.Soffa. Regression testing of GUIs , in Proceedings of the 9th European Software Engineering Conference (ESEC) and 11th ACM SIGSOFT International Symposium on the Foundations of oftware Engineering