A Single Model for Event-Driven Software

Om Kumar C.U., P. Bhargavi & Vinod Kumar. K

Dept. of CSE, Sree Vidyanikethan Engineering college
Rangampet, A.P 517102, India

Abstract **- A widely used class of software that takes sequence of events as inputs changes state and output's new event sequences are called as Event Driven Software (EDS). An event is an incident that is inconsistent, with the ordinary course or expected outcomes. EDS are very diverse such as GUI, Web applications and Embedded Software. We Confine to the first two types of EDS in this paper.GUI is a type of User Interface which allows people to interact by using graphical icon, visual indicators or special graphic elements called "Widgets" along with text labels or text navigations. Web Applications is an application that is accessed over a network such as internet or intranet. All the research work extensively made on GUI & Web application till date is good enough but is disjoint. We overcome it by finding a similarity called parameter values. This paper provides a Single Model by using Test Prioritization Strategies that is generic enough to study develop and test a unified theory for all kinds of Event Driven Software..**

I. INTRODUCTION

Event Driven software are becoming global (from Desktop-GUI chatting to video conferencing-web App).Event Driven Software are classified into Graphical User Interface, Web Application, Embedded Software. These are software's that change state based on incoming events. An Event is a software message indicating that something has happened, such as a Key press or mouse click. An event is an action which is initiated outside the scope of the program but is handled by code snippets inside the program. Events are handled by Event handlers which are in synch with the program flow. Examples of Events are user pressing a key on the keyboard, selections through mouse etc. Another source is a hardware device such as a timer. Event Driven Softwares are nothing but interactive computer programme that responds to Events by changing its behaviour. Event Driven Softwares changes its state by taking in sequence of events and producing new event sequences. So testing Event Driven software's for its correctness is defiant. Since a user can invoke huge number of possible event sequences through a User Interface, tracking and testing it is fractious. The contributions till date made on testing these 2 subclasses of EDS show that they behave similarly [2][3][4]. Despite the above similarities of GUI and Web applications, all the efforts to address their common testing problems have been made separately due to two reasons. First is the absence of a Generic model that captures the event driven nature of the application. This has prevented the development of a shared testing technique that can test any class of EDS. Second is the unavailability of subject application (Web & GUI Together) and tool for researches. This paper provides the first single model that is generic enough to develop study and test GUI and web applications together.

The specific contributions of this work include: the first Single model for testing stand-alone GUI and Web-based applications, a shared prioritization function, and prioritization criteria's. We validate the correctness of the model through a tool.

II. RELATED WORK

A. User Session Based Testing:

An uninterrupted period of time is called a session. Session based Testing aims to provide accountability by creating on the fly a test design and does managing, controlling and metrics reporting. Session-based testing can be used to introduce measurement and control to an immature test process, and can form a foundation for significant improvements in productivity and error detection. Session-based testing can be used when formal requirements are not present, incomplete, or changing rapidly [8][12]. User Session to Test Case transformation (USTCT), transforms each individual user session into a test case. Given m user sessions, U_1 U_2 $U_{3\ldots\ldots}$ U_m, with user session U_i consisting of n requests $r_{1,}$ $r_{2,}$ $r_{3\ldots\ldots}$ r_n , where each r_i consists of url

[name − value], the test case corresponding to U_i is generated by formatting each of the requests, from r_1 to r_n , into an http request that can be sent to a web server. The resulting test suite contains m test cases, one for each user session *Ui*.

B. GUI Test case Prioritization:

A User interacts with a GUI by invoking operations. All these operations are recorded for testing. But testing all the events by tracking and recording is infeasible.

A feasible approach would be by testing a GUI based on the User Interaction. All the interactions made by the user are considered as test cases [12]. A prioritization value is given to a test case based on its interaction. A next test case is prioritized in such way that it shouldn't contain the interactions of the previous test case.

Example:

A File opened and read is given PV 1 based on the action performed. The same File when open again that test case is value is reduced to PV 0.5.The same File when opened again, edited and saved is prioritized as PV 1.2.

C. Model Based Testing:

Model Based Testing is an abstraction or simplification of the behaviour of the application to be tested. A Model is recorded in machine readable format for test generation purpose.

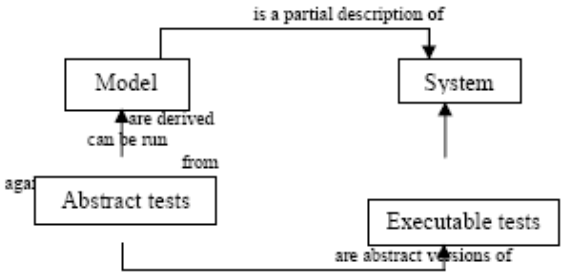


Fig. 1 Model Based Testing

Model based testing can be done in 2 ways [2].

1. On-the-fly
2. Off-line execution.

Typically off-line execution requires the oracle aspect as the test sequences generated through and retained by the model for future execution need to be paired with predictions or estimates of the expected results. A technique to compare the actual application response to the model predicted response needs to coordinate the recording of actual results and the evaluation against the predicted results. A process is required to compare the actual result and evaluate it against the predicted results.

An improvement to this general approach to model based testing is the on-the-fly execution of Model Based Test [12]. In this mode the model generates the test sequences (traces) whilst dynamically interacting with the application under test (AUT) – this in a way mimics user interaction with the AUT. The on-the-fly execution process compares the response with the expectation model dynamically.

III. COMBINATORIAL MODEL

This paper discusses on proposing a combined model that would develop and test both GUI and Web Application. GUI application generally uses event listeners (event handlers) to catch the event caused by the user. For Ex: Scrolling mouse, drag and drop, clicking etc .To all these events explicit event handlers are used that keep listening to these events to occur.. A Web application sometimes listens to action in client side (In Registration form-11 digits are not allowed in mobile number parameter) and sometimes the data is carried to server for testing. They are done using GET/POST method.(In Registration form-availability of user name). All the research till date shows that the tools used for developing and testing these two kinds of EDS have been disjoint. Some of the open and proprietary testing tools available are listed below.

Consider the below example where a Find and Replace editor (Desktop app) and a login form (web App) are considered for testing to perform combinatorial testing.

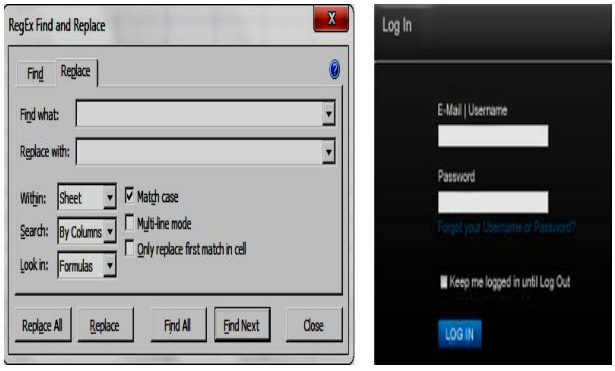


Fig.2 Example of GUI & Web Application

Combinatorial testing can be performed by selecting a field common in both GUI & web-based application. The field that is in common to the above

example is parameter value which is a combination of parameter name and parameter value

Table I : Parameter values of Example

S.NO	Desktop Application	Web Application
1.	<"Find what", drop box, SetText>	<Username TextField, xxx>
2.	<"Find what", drop box ,Left Click>	<Password TextField, xxx>
3.	<"Replace with", drop box ,Set Text>	<Keep Me LoggedIn, Check Box, Leftclick Select>
4.	<"Replace with", drop box, Left Click>	<Keep Me LoggedIn, Check Box, Leftclick UnSelect>
5.	<Match case", Check box, Left click Select>	<login Buton, Leftclick>
6.	<Match case", Check box, Left click Unselect>	<Form Action, Login>
7.	<"Multiline", Check box, Left click Select>	<Form name, Login>
8.	<"Multiline", Check box,Left click Unselect>	
9.	<"Find", Button, Left Click>	
10.	<"Close" Button, Left Click>	

IV. PRIORITIZED MODEL

This model proposes a prioritized solution that produces a sorted sequence of test cases. This model helps in executing the test cases with and without priority. Initially the model executes the test cases without priority and records the values in its database. At times the sorted sequence in the DB may contain 2 or more test cases that have same value. Here to overcome this situation we opt prioritization model.

There are some test cases which end up with same priority. So in that case we need to prioritize those test cases with some other priorities. They are as follows.

1. Customer Requirement Based Priority.

2. Cost criterion.

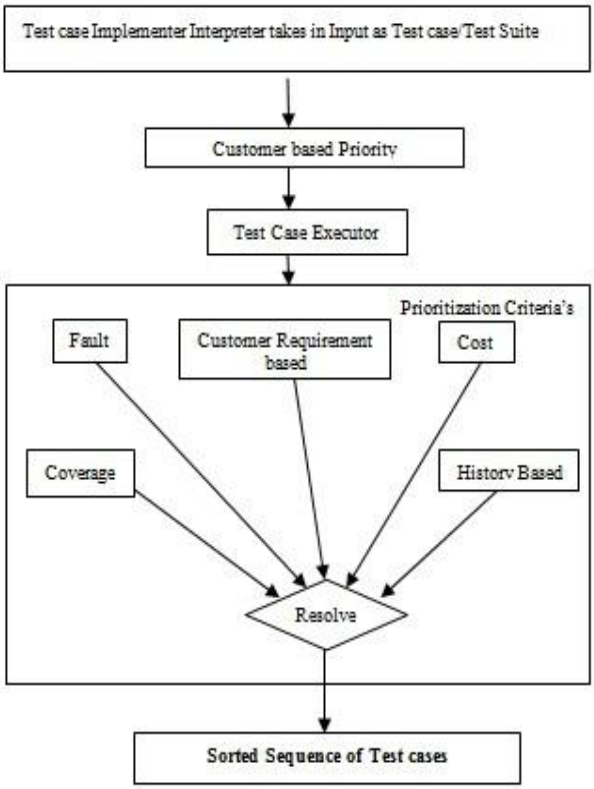
3. Coverage Criterion.

4. History of Test cases.

5. Fault based Criterion.

By using all the above mentioned criteria's a test case will be prioritized. But to get better results we compare our prioritization function with

1. No prioritization.
2. Random prioritization.
3. Optimal Prioritization.(Proposed Technique)



When test cases with equal priority come into picture various criteria's are considered for prioritizing these test cases as shown above. They produce a sorted sequence of test cases.

Prioritization Function:

This function takes in a sequence of test cases prioritizes it and produces a sequence of prioritized test cases. We use this function to prioritize the test cases of GUI as well as Web Application. The Function requires 4 parameters.

1. A Test Suite (TS) which needs to be sorted.

2. A function (f) which takes in input as a single test case prioritizes it based on prioritization criteria's and produces prioritized test case f(x).

3. A function (F) which takes in input a test suite and produces a prioritized test suite F(X).

4. An operation (#) to assign "fitness" value to test cases. They can be "cardinality", "Set Difference".

Algorithm:

Input Parameters:

TS: Test suite for prioritization.

f: For prioritizing a Test Case.

F: For prioritizing a Test Suite.

#: A "fitness" value on Test Case.

Output:

$: Sorted Suite.

Mechanism:

Stack ← Empty

TS ← Test Suite

T ← Test case

REPEAT

T ← Next Best Test Case (TS, T, f, F, #)

Stack ← Insert At End (Stack, T)

TS ← TS-T;

UNTIL (TS==0)

$ ← Stack.

IV EXPERIMENTAL ANALYSIS

We have developed a tool that can test GUI and web application for its correctness. Fig.5 shows a test suite which includes test cases from both GUI & web application. TId 1 is a window from desktop application. TId 2 and TId 3 are pages from web-based application. The test suite containing test cases from both desktop and GUI application are given as input to the tool.

The tool prioritizes test cases in test suite by assigning a maximum priority value 5 initially to it. Whenever the same test cases are again considered for testing the tool reduces the prioritization value of the test case. A threshold value is maintained beyond which a test case is never considered for testing.(For example a test case when tested 3 times the tool records its redundancy as 3 and reduces the prioritization value from 5 to 2. The threshold limit of that particular test case is around 60% .When it further decreases the test case is decreased by 1 and is not considered for testing. Thus the time and cost of testing is reduced).The tool imports major classes from the test cases loaded. Those classes in turn list methods available for testing. By using JUnit the test cases can automatically tested for its correctness. It also provides methods explicitly for testing in which user can write his method for testing.

Prioritization

TId	(W)	(P)	(PV)	(A)	(Pr)
1	Login	UserID	ram	Submit	5
2	Login	Password	password_1	Submit	5
4	Home	UserType	Admin	Request	4

Fig.4 Test Suite Prioritization

The tool uses JWebunit for web application testing. A class is created with methods prepare() and testLogin().web application testing is performed in testLogin method where web login testing happens automatically. If the username is test and password is test123 then the test reports success and records it. Similarly GUI testing can be performed and its result can be recorded for prioritization

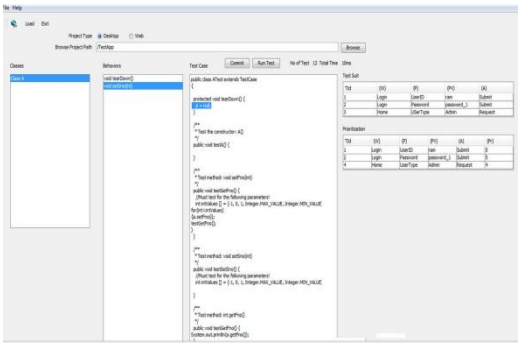


Fig.5 Test Suite testing

Web Application Testing

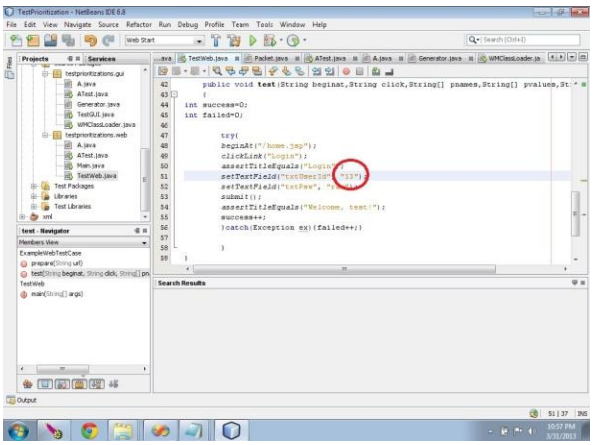


Fig. 6 Web Application Testing

The above snapshot is about web application testing. The UserId used in a web application when entered correctly Logins the account. But when entered wrongly doesn't allow the user to login.

Sample Code:

```
Package testprioritizations.web;
import org.junit.*;
import static net.sourceforge.jwebunit.junit.JwebUnit.*;

public class TestWeb
{
    public static void main(String[] args)
      {
      String baseurl=”http://localhost:9999/jspsite”;
      String beginat=”/home.jsp”;
      String click=”Login”;
      String paramnames[]={“txtUserId”,”txtPsw”};
      String paramvalues[]={“test”,”test123”};
      String expResponse=”Welcome, test!”;
 ExampleWebTestCase t=new ExampleWebTestCase();
  t.prepare(baseurl);
t.test(beginat,click,paramnames,paramvalues,expRespon
se)

      }
}

class ExampleWebTestCase
{
   Before
   public void prepare(String url)
      {
       setBaseUrl(url);
      }
   Test public void test(String beginat,String
click,String[] pnames,String[] pvalues,String
expResponce)
   {
int success=0;
int failed=0;
            try
{
            beginAt(“/home.jsp”);
            clickLink(“Login”);
            assertTitleEquals(“Login”);
            setTextField(“txtUserId”, “1”);
            setTextField(“txtPsw”, “ram”);
            submit();
            assertTitleEquals(“Welcome, test!”);
        System.out.println(“\n\n   success \n\n\n”);
}
catch(Exception ex)
{
failed++;
}
        }
}
```

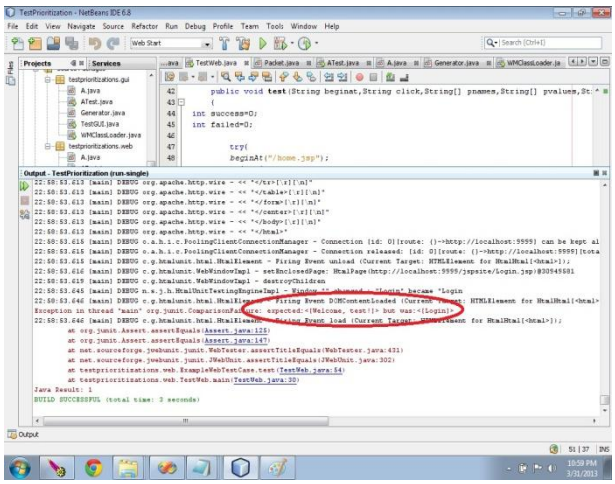


Fig. 7 Web application Negative Test case

The test case showing the negative flow is shown in fig.7.

GUI Testing

The below snapshot shows the execution time of test cases through startTest and endTest. About 3 different set and get methods are tested such as setSno, getSno, setFno, getFno, setResult, getResult. The time taken by each method is shown in milliseconds. Roughly it took around 20 milliseconds.

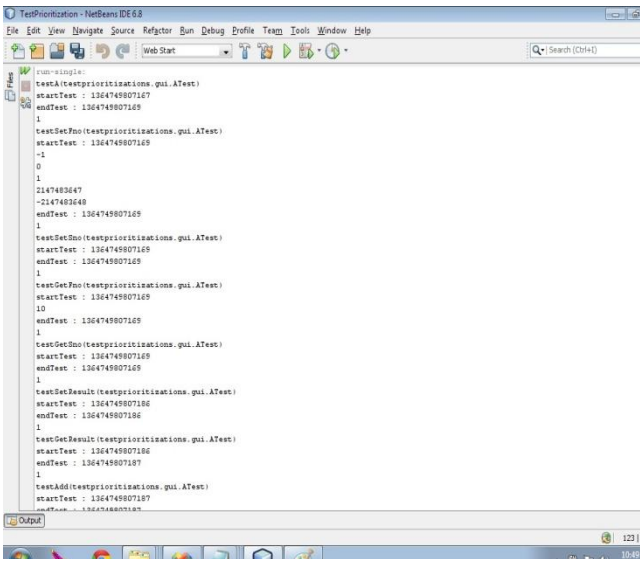


Fig. 8 GUI Testing

V CONCLUSION & FUTURE WORK

Research till date treats GUI & Web-based applications as different entities of research. This paper shows similarities that allow us to create a Single model for testing Event Driven Software. Within the context of our Model we propose prioritization function to prioritize test cases using prioritization criteria's. We also introduce a method to prioritize multiple test cases with equal priority. The prioritization function and prioritization criteria's when used together in our combined model, depicts the usefulness of two kinds of Event Driven Software's for the problem of test prioritization.

Prioritization generally requires a large set of values to produce efficient output. The Future work would be to develop a Generic Model for all kinds of Event Driven Software's by using a minimal set test prioritization.

VI. REFERENCES

[1] Steffen Herbold, Jens Grabowski, Stephan Waack, "A Model for Usage-based Testing of Event-driven Software", 5th International Conference on Secure Software Integration & reliability, 2011, pp.172-178.

[2] Mark Utting, Alexander Pretschner and Bruno Legeard, "A Taxonomy Of Model Based Testing", Working paper by Waikato, April 2006,pp.1-17.

[3] DING Xiao-Ling, DING Chun, Hou Yong hong, "An Approach To Event-Driven Software Testing", Vol .8.No-4 2002, pp.265-268.

[4] Renee C.Bryce, Atif M.Memon, "Test suite Prioritization By Interaction Coverage", Workshop on Domain specific Software Test automation in Conjunction with the 6th ESEC/FSE Joint Meeting, 2007, pp.1-7.

[5] Gregg Rothermel , Roland H. Untch, Chengyun Chu, Mary Jean Harrold, "Test Case Prioritization: An Empirical Study" , Proc. of the International Conference on Software Maintenance, UK, September, 1999, pp.1-10.

[6] Shashank Joshi, Shital Pawar, "Agent Based Testing Tool For Event Driven Software", International Journal of Engineering Research and Applications, Vol.2, Issue 3, May-Jun 2012, pp.2961-2965.

[7] Shashank Joshi, Shital Pawar, "Developing A Testing Tool For Testing both GUI and A Web Applications Together", International Journal of Advances in Engineering & Technology, May 2012, pp.391-394.

[8] Siripong Roongruangsuwan, Jirapun Daengdej, "Test Case Prioritization Techniques", Journal of Theoretical and Applied Information Technology, 2010, pp.45-60.

[9] Praveen Ranjan Srivastava, "Test Case Prioritization", Journal of Theoretical and Applied Information Technology, 2008, pp.178-181.

[10] P. Dileep Kumar Reddy & A. Ananda Rao, "An Empirical Analysis of Single Model Test Prioritization Strategies for Event Driven Software", International Conference on Computer Science,2010, pp.185-188.

[11] J.Praveen Kumar ,Manas Kumar Yogi, "A Survey on Models and Test strategies for Event-Driven Software", International Journal Of Computational Engineering Research Vol. 2 Issue. 4 2012, pp.1087-1091.

[12] Hani Achkar, "Model Based Testing Of Web Applications", Stanz Sydney Australia,2008, pp.1-28.

[13] Anneliese A. Andrews, Jeff out, Roger T. Alexander, "Testing Web Applications by Modeling with FSMs", pp.1-28.

[14] Atif M.Memon, "Developing Testing Techniques for Event-driven Pervasive Computing Applications", International Conference On Testing Techniques,2009,pp.1-10.

◈◈◈