# Web-Based Automation Testing Framework

Anuja Jain M
School of Computing Science and Engineering
VIT University,
Vellore -14

Swarnalatha P*
Muhammad Rukunuddin Ghalib+
*Assistant Professor [S.G]
+Assistant Professor
School of Computing Science and Engineering
VIT University,
Vellore -14

S. Prabu
Professor
Information Technology
P.S.N.A, College of Engineering
Dindigul, Tamilnadu.

## ABSTRACT
Every Software development goes through several phases in SDLC where validation of software plays a important role as it shows correctness of the software. Validation Phases is the most expensive and time-consuming process for QA Engineers as code coverage and version numbering of the software increases the manual testing efforts. To help QA Engineers to utilize there time effectively, several testing automation has been carried out since decade but they result into partially automated or require more tester interventions. Especially for GUI application the automation becomes challenging because of its dynamically changing nature. The paper proposes a cross-platform, code-driven, object-oriented[9] testing framework called as GUI-WAT (Web-Based Automation Testing) Framework which reads HTML source and generates GUIWebObjects. The HTML source of the web based application is parsed into hierarchical structure that represents web elements. The GUIWebObjects defines the properties of each web element and generalization is achieved with the help of Jsoup[5]such that change of UI does not reflect the correctness of the framework. The framework includes libraries, API and test cases repository for performing automation on any web-based application. GUI-WAT uses most promising tool called Selenium [2] for providing Action-events. Hence GUI-WAT is time-efficient, cost-reductive and thereby helps increasing code coverage.

## Keywords
WAT, GUIWebObjects, Jsoup, Selenium, Cross-Platform, time-efficient, cost-reductive.

## 1. INTRODUCTION
The Web GUI applications are becoming more and more complex due to development of internet technology. Recently software development cycle are becoming shorter, which makes GUI application testing[11], especially regression testing more challenging [6]. The manual approach fails to address these issues hence automation testing[12,13,14] is the powerful solution.

The challenges faced during GUI Automation Testing:-

- The most challenging requirement is to handle dynamically changing GUI application.
- With the increasing versions of application, Regression Testing[1] is key requirement
- Cross-platform implementation is also an important requirement, if the actual framework implementation is in Linux but want to run test case by launching browser in Windows.
- There is need for multiple browser support, because some bugs are browser specific.

Hence we analyzed various automation testing tools to find out a potential candidate which will address all our challenges. The following two tools are part of our analysis.

**Testing Anywhere:-**
It is an automation tool which supports record and replay mechanism of creating test cases. It addresses the issue of dynamically changing identifiers and also supports multiple browsers, but the tool fails to address our requirement of cross platform implementation as it supports only windows operating system

**Teleric :-**
It is an automation tool which builds tests for web and desktop application including support for multiple browsers, but it does not support cross-platform implementation and only support Windows operating system.

After analysis we nailed down to selenium tool. Selenium tool provides following features:-

- It is an open source software
- Supports multiple operating systems (Windows, Linux, Mac etc)and cross-platform implementation is possible.
- Multiple browser support
- Problem of dynamically changing identifiers can be solved using xpath.(XML Path Language)
- Provides support for junit ,nunit, ruby, perl tests.
- Allows Regression testing, reporting and parameterizing of test cases.

## 2. WORKING MODEL

### 2.1 Overview
Selenium tool has intern have different types of tools, amongst those tools we selected Selenium RC (Selenium Remote Control). Selenium RC has two components server and client. Server is responsible for launching and killing the browser and also acts as HTTP proxy intercepting and verifying HTTP messages passed between and application under testing[10,11]. Client libraries provides interface

between the programming language and Selenium RC server[2].

The Server receives commands from our test program using simple HTTP GET/POST requests. It means we can use any programming language that can send HTTP requests to automate Selenium tests on the browser. A Selenium client library provides a programming interface (API), i.e., a set of functions, which run Selenium commands from our own program[2]. The client library takes a Selenium command and passes it to the Selenium Server. The client library also receives the result of that command and passes it back to program. The program can receive the result and store it into a program variable and report it as a success or failure, or possibly take corrective action if it was an unexpected error.

## 2.2 Test Case Executer

**Test Case Runner gives the test case execution sequence.**

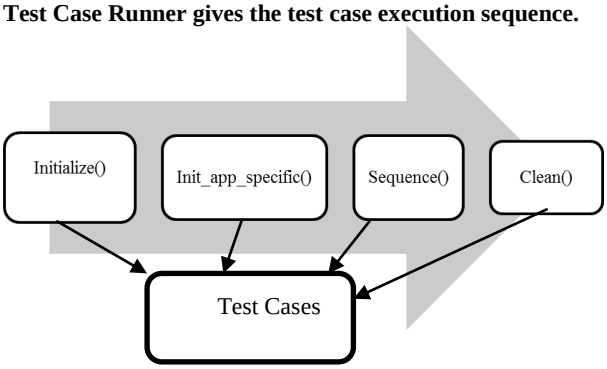



**Figure 1: Test Case Executer**

**initialize()**- Performs Initialization operation and initializes required components for testing.

**Init_app_specific()**-Prerequisites for specific application are executed for example launching of browser.

**sequence()** – Steps for executing specific test scenario are executed.

**clean()** – Cleaning up after test execution so that the next test case will run correctly

## 3. WAT EXECUTION ENGINE

WAT framework is an object-oriented framework developed in java[4] and it also includes different action-events[3] need to perform on the objects which is necessary for performing functional testing[5].

WAT Framework contains the fallowing modules:-

**WebObjects :-**

For GUI Application we are creating web objects of the web elements like button,dropdown,graph ,page etc so that test cases can use the object for generating the test scenario and performing the necessary operation.

**JSoupParser:-**

Each web objects has different html tags on the basis of the tag and the identifier , Jsoup parser helps us to parse the html source code and create the xpath during runtime.

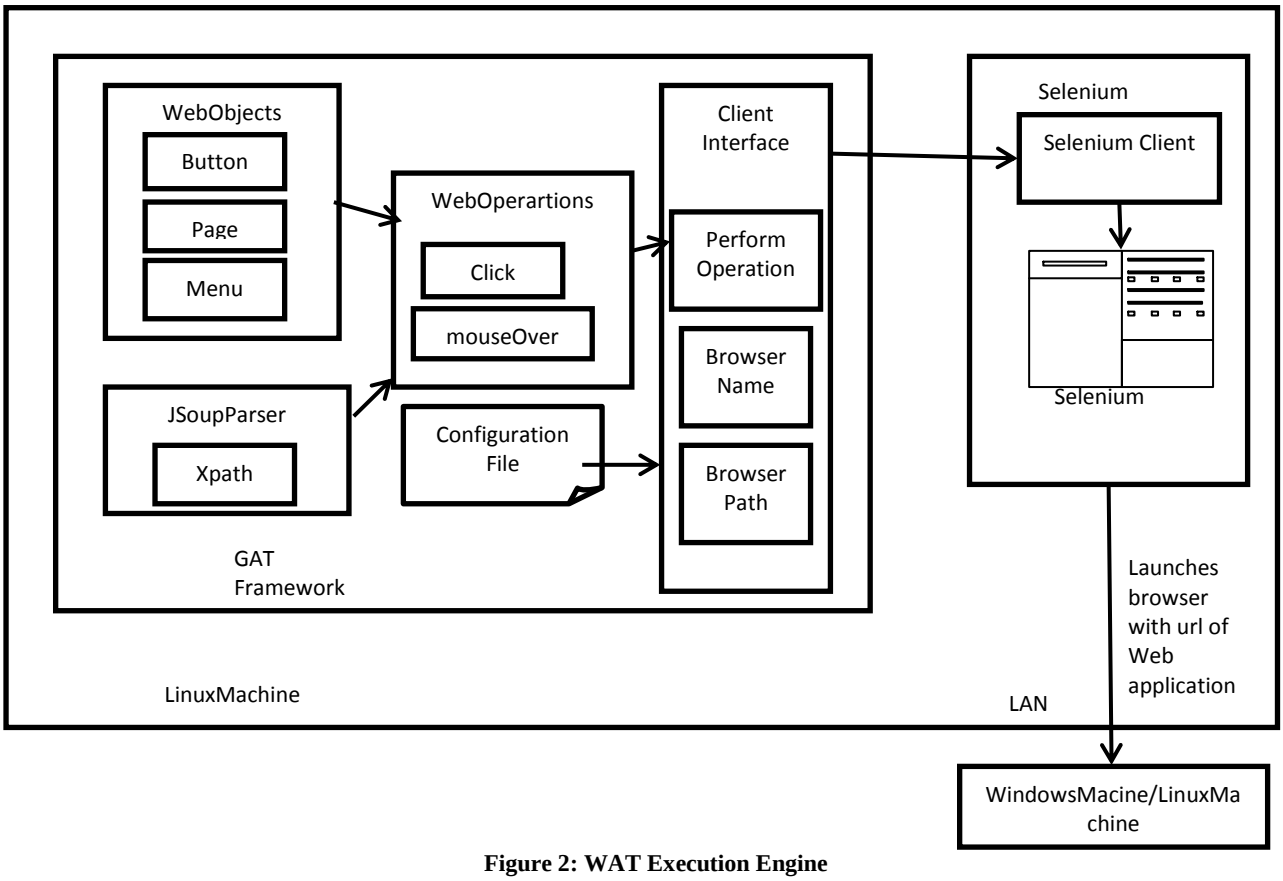



**Figure 2: WAT Execution Engine**

As xpath[6]is created on runtime the dynamically occurring changes due to change in gui application development can also be handled.

The html code shown below contains tags<button> and <img> which represents button and image element. The id attribute of the html tags are not useful as they are

dynamically changing so we need unique locator for identifying element.

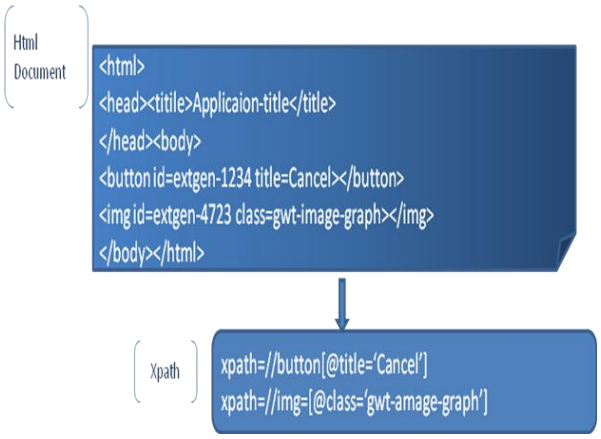



**Figure 3: Dynamically Changing ID**

Xpath for elements as shown in below html code can be represented as given below.

**WebOperation:-**

After specifying the web objects, the Jsoup parser created the locator on the basis of html tag. Once we get the locator we can perform various operations specifying the location of the web element. And the operations include click, mouseOver, type ,select,etc different kinds of operations. While performing the operation we are adding some delay for synchronization between selenium and the web application.

**Configuration File:-**

The Configuration file is used to specify the name of the browser which we are willing to launch for example iexplore, firefox etc. and also we have to specify the path of the browser and machine name. Hence if we want to launch the browser on different machines with different browser we just need to modify the configuration file.

**Client:-**

Client provides the interface to the selenium client and passes the commands to selenium client which are called by test cases and intern selenium client passes it to selenium server which will actually execute the command and return the results back

# 4. ALGORITHMS

There are six modules and detailed implementation of the module is given below.

1. FTIA (Framework and tool Integration algorithm)
2. WOC (Web object creation Algorithm)
3. WSA (Web Scrapping Algorithm)
4. EGA (Event Generation Algorithm)
5. TCA (Test Case Abstraction Algorithm)
6. WA (Windows Agent Algorithm)

## 4.1 FTIA

FTIA is integrates WAT framework and the external tool Selenium and also brings up the web application for testing , providing interface to interact with web application throughout testing and terminating it once testing is done. Presently it provides support for Windows and Linux operating system. There is also an option to specify the type of browser to be launched and the application specific URL. The algorithm is the one which actually interacts with the application and other modules uses it as interface to interact with the application. It also extracts the html source of the web application for further processing.

**INPUT :** webURL, webBrowser ,operatingSystemName
**OUTPUT:** LaunchBrowser, //take commands/events from EGA and pass it to external tool selenium
**ALGORITHM :**

1.OperatingSystemName=System.getproprty("os.name")
2. getBrowserPath(filename)
    a.parse configuration file to extract browser path
3.getMachineName(filename)
    a.parse configuration file and extract machineName
4. launchBrowser(webBrowserPath,webURL,portNumber)
    a.webBrowserPath=getBrowserPath(configFile)
    b.webURL=give URL for web application
    c.portNumber=getRandomPortNumber()
5.executeCommand(String sCommand)
    a.machineName=getMachineName(configFile)
6. startSeleniumServer(portNumber)
    a.executeCommand( "java –jar selenium-serever-standalone-2.1.0.jat –port "+)
7.startSeleniumClient()
  if(operatingSystemName==’Windows’)
    a.Create sockect connection through WA
    b.startSeleniumServer( getRandomPortNumber ())
    c.launchBrowser(getRandomPortNumber,getmachineName(), getBrowserPath, webURL)
    d.Events from EGA passes through WA
  else
    a.startSeleniumServer( getRandomPortNumber ())
    b.launchBrowser(getRandomPortNumber,getmachineName(), getBrowserPath, webURL)

## 4.2 WOC(Web Object Creation)

In Web applications different web objects are categorized based on the html tags associated with it in html source code. WOC algorithm takes html tag as an input and accordingly creates list of web objects. These web objects are then used by WSA for generating the locators.

**INPUT :** html tags
**OUTPUT :** list of web objects

**ALGORITHM :**

1.getHtmlTag()
    a.return obj.tagName()
2.getIdentifier()
    a.returnobj.identifier()
3.if(htmlTag==obj.tag)
    a. create list of list<obj>

## 4.3 WSA (Web Scrapping Algorithm)

Web Scrapping Algorithm fetches html source code for producing locators for uniquely identifying and locating the web objects. The algorithm gets html tag of web objects by WOC and based on the tag it parses the code and generates xpath. Figure below shows the html tag and the xpath created based on the tags. And the xpath is then used by EGA to perform actions on the intended web objects.

**INPUT :** html code, html tags
**OUTPUT:** xpath (object identifier/locator) //passed to EGA
**ALGORITHM :**

1.HashMapInitialLocator(Key=tag,value=initial xpath)
2. HashMapTagAttribute(Key=tag,value=attribute)
3. HashMapFinalLocator(key=tag,value=post xpath)
4.getDocuments(String sSource)

a. Document doc=parse(sSource)
5.getElementsByTag(sTag,doc)
a. return Elements with tag
6.getElememeentAttribute(sAttr)
a. for(Element element: getElementsByTag(sTag)
b. if(sAttr.isNotEmpty)
i. arrayElementsAttr[]=element.attr(sAttr)
else
ii. arratElementsAttr[]=elements.text()
end if
end for loop
c.returnarrayElementAttr
7.parseString(sTag,arrayElements,identifier)
a.While(count<arrayElements.length)
b.if(arrayElements[count].equals("identifier")
i.paredString=arrayElement[count]
ii. break;
end if
end while
8.return parsed String
9.getLocator(String sParsed,StringsTag)
a.returnInitalLocator.get(sTag)+sParsed+FinalLocator.get(sTag)
10.getXPath(WebObject obj,StringsSource)
a.elements=getElements(obj.getHtmlTag,getDocument(sSource))
b.arrayElements=getElementAttribute(TagAttribute.get(obj.getHtmlTag))
c.identifier=obj.getIdentifier
d.sparsed=parseString(obj.getHtmlTag,arrayElements,identifier)
e.returngetLocator(sparsed)

## 4.4 EGA (Event Generation Algorithm)

Every web object is associated with actions such as button is associated with click action, text-box is associated with type action etc. EGA is responsible for generating actions in association with web objects and passing the actions to FTIA algorithm. Along with actions EGA receives xpath from WSA to identify or locate the web objects on which to perform the objects.

**INPUT :** Web objects
**OUTPUT:** // pass the event and xpath to FTIA
**ALGORITHM :**
1.performOperation(WebObject obj,StringsSource)
a.sLocator=getXpathLocator(obj,htmlcode)//calls from WSA
b.m_clientevent.action(sLocator)

## 4.5 TCA (Test case Abstraction Algorithm)

TCA is provides abstraction for test case specific implementation. The Algorithm changes as per the language in which test case changes. For example we will discuss JUnit. JUnit supports annotations like @after and @before. With these annotations we can specify the steps to be executed before test case execution starts and steps to be executed after test case execution ends.
**INPUT :** type of language in which test case is required
**OUTPUT:** setup for writing test cases
**ALGORITHM :**
1.Initialize()
2.AppSpecificSetup()
3.EnvironmentalSetup()
4.tearDown()
5.Cleanup()

## 4.6 WA (Window Agent Algorithm)

The WA algorithm enables cross platform implementation of the system. If the operating system is windows then commands are passed from FTIA to WA algorithm. The WA algorithm establishes connection between Linux machine and windows machine.
The code for launching the browser and starting selenium server is written in Linux machine but actually server gets started on windows machine and the browser is also launched on windows machine.
**INPUT :** commands
**OUTPUT:** //launches browser, executes command on windows machine
**ALGORITHM :**
1.establichConnection(localhost,windowsMachineName)
a. create a scocket on windows machine on a port
b. socket will be in listening state
2. localhost will connect to the windows machine with the socket

# 5. RESULTS

We performed few experiments which give following results shown in the Table-2 average execution time of automated test case exceeds 20 sec from average execution time of manual test case. Even though automation takes some extra time, but unlike manual validation it runs in background, so that tester intervention is not required throughout execution. It is important to consider the time to execute the checklist as shown inTable-1, because as and when new version of the application is released the tester has to execute and validate all the previous checklist, Automation provides the way to run the regression and saves lots of efforts and time of a tester.

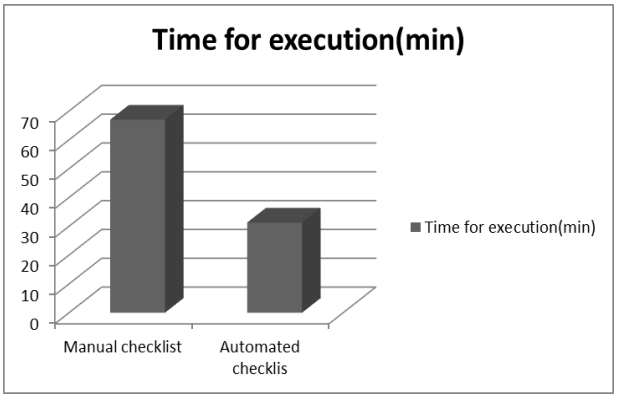
**Table 1: Execution time to run checklist**

| Time required | Manual Execution | Automated Execution |
|---|---|---|
| Create checklist | 32 hrs | 32hrs |
| Execute One checklist | 67min | 31min 30 sec |
| N checklist | N*67min | N*31 min 30 sec |

**Table 2: Manual Testing Vs Automation Testing**

| | Manual Testing | Automation Testing |
|---|---|---|
| Number of test cases | N | n |
| Average execution time for 1 test case | ~40 sec | ~30sec (test execution + ~30sec (synchronization) |
| Time to Execute 1 checklist(n test case) | n*40 | n*60 sec |
| User intervention | Throughout Testing | To start execution and at end to analyze the result |

The manual checklist execution requires more time because tester need to go through the checklist then collect the test sequence and execute the test sequence as mentioned in the checklist also tester need more time trace the objects on which it will perform operations. But in the automated checklist all the test scenarios are ready for direct execution no overhead of going through checklist just need to run the test suit.




## 6. FUTURE WORK

There is scope of reducing the execution time by executing the test cases parallely in multiple browsers on multiple machines. What we need to do is several test cases will execute specifying the machine name and the browser type and accordingly the framework should launch the browser with the specified type and machine. The future implementation picture looks like figure 4.

## 7. CONCLUSION

In a nut shell WAT Framework provides object-oriented, remotely executable code-driven, cost and time efficient framework to improve quality and accuracy of testing for the GUI applications. The advantage of using WAT Framework is the adaptability of WAT Framework with change in GUI application form one version to another version. The most important benefit is that human intervention requirement only for starting test-suit and analysis of results at the end and hence the same resource can be simultaneously involved in other work while GUI validation is running in background.

## 8. REFERENCES

[1] Atif Memon, Ishan Banerjee, Nada Hashmi, AdithyaNagarajan,"DART: A Framework for Regression Testing "Nightly/daily Builds" of GUI".

[2] http://selenium.org – Tool to provide environment for web automation testing in various langauges

[3] Li Feng, Sheng Zhuang,"Action-driven automation test framework for graphical userInterface(GUI) software testing.

[4] Pekka Aho1, Nadja Menz2, Tomi Räty1 and Ina Schieferdecker2,"Automated Java GUI Modeling for Model-Based Testing Purposes", 2011 Eighth International Conference on Information Technology: New Generations

[5] www.jsoup.org – A HTML Parser

[6] www.w3schools.com/xpath - Web Element Locator

[7] ZengWandan, Jiang Ningkang, Zhou Xubo,"Design and Implementation of a Web Application Automation Testing Framework", 2009 Ninth International Conference on Hybrid Intelligent Systems

[8] ZHU Xiaochun, ZHOU Bo, LI Juefeng, GAO Qiu," A Test Automation Solution on GUI Functional Test", College of Computer Science, Zhejiang University, Hangzhou, P.R China.

[9] D. Kung, C. H. Liu and P. Hsia. An Object-Oriented Web Test for Testing Web Applications. In: Proc. of IEEE 24th Annual International Computer Software and Application Conference(COMPSA2000). Taipei Taiwai: IEEE Computer Society Press, Oct. 2000, 111-121

[10] Ji-Tzay Yang, Jiun-Long Huang, Feng-Jian Wang, and William C. Chu. An Object-Oriented Architecture Supporting Web Application Testing. In:Proc. Of IEEE 23rd Annual International Computer Software and Application Conference (COMPSAC2000), Phoenix Arizona USA:IEEE Computer Society Press, Oct 2000,122-127

[11] F.Ricca and P. Tonella. Analysis and Testing of Web Applications. In:Proc. of the International Software Engineering Conference. Toronto Canada: IEEE Computer Society Press, May 2001, 25-34

[12] D. C. Kung, N. Suchak, J. Gao et al. On Object State Testing. In: Proc. of Computer Software and Application Conference. Taipei Taiwai: IEEE Computer Society Press, 1994 222-227

[13] Ye Wu and Jeff Offutt. Modeling and Testing Web-based Applications. GNU ISE Technical, ISE-TR-02-08, November 2002,21-32

[14] J. Li, J. Chen and P. Chen. Modeling Web Application Architecture with UML In: Proc. Of 36th International Conference on Technology of Object-Oriented Languages and Systems (TOOLSAsia' 00). Xi'an, China: IEEE Computer Society Press , Oct. 2000, 265-274 318