COMPARATIVE STUDY OF BROWSER BASED OPEN SOURCE TESTING TOOLS WATIR AND WET

Nisha Gogna

Research Scholar, University Institute of Engineering and Technology
Panjab University
Chandigarh, India

Raj Kumari

Assistant Professor, University Institute of Engineering and Technology
Panjab University
(Corresponding author)

Abstract-Recently, web browser based applications have become very popular in many domains. As an important method to ensure the quality of Web applications, Web testing attracts more and more attentions in the academic community and industrial world. Testing Web applications raises new problems and faces very high challenges. Recently web applications have grown rapidly and have become more and more complex. As web applications become more complex, there is a growing concern about their quality. Web application testing is a challenging work owing to its dynamic behaviors and complex dependencies. As in case what if the end user uses different types of browsers to access the application. So, to deal with Browser compatibility issue this work proposes Browser based testing tools named as Watir (Web Application Testing in Ruby) and WET for the scenario of web-based applications. Starting with the preliminary architectural design, this paper includes the testing scenario for both the tools along with the comparative analysis based on well defined parameters.

Keywords-Browser compatibility;Watir;Ruby;WET;

I. INTRODUCTION

Starting with, web browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web. An information resource is identified by a Uniform Resource Identifier (URI) and which may be a web page, image, video, or other piece of content. Hyperlinks present in resources enable users to easily navigate their browsers to related resources [1]. On the other side, Web testing is the name given to software testing that focuses on web applications. Complete testing of a web-based system before going live or before the system is revealed to the public can help addressing various issues [2]. In recent years, web applications have become important for many companies, as being a convenient and inexpensive way to provide information and services on-line. With the development of internet technology, the Web application becomes more and more complex and the scale of it changes more and more great. The Web application programs testing, especially the regression testing is much more difficult than the traditional [3].

A modern web application typically consists of several client-side components in the form of scripts and resources that are linked from the web page and executed in a web browser. Currently, users have the option to use several web browsers, which introduces problems for web-application developers. Web applications are expected to behave consistently across all of the popular browsers (and across platforms). However, because the standards for client-side technology are still evolving, there is a great deal of inconsistency in how different web browsers behave. These inconsistencies lead to what we call crossbrowser issues–differences in the way a web page looks and behaves in different browsers. Maintaining compliance across these browsers is a crucial task for developers [4]. So browser compatibility testing is needed in order to ensure these applications run normally under the different browsers and user-side configurations. The testing goal is to guarantee the functions of Java Applets, ActiveX Controls, JavaScript, CSS, HTML, etc in all kinds of configurations [5].

By keeping all these views in mind, we propose a novel approach towards Browser compatibility by making use of Browser based testing tools Watir (Web Application Testing in Ruby) and WET.

The remainder of this paper is organized as follows. In section II, background and some basic concepts are summarized. The basic framework of Watir and WET is proposed in Section III and IV. Section V deals with the testing scenario of both tools, followed by implementation observations and results in section VI. Finally, the conclusion of the study is given in Section VII.

II. BASIC CONCEPTS AND BACKGROUND

Web based Applications are increasingly becoming more feature rich, important and also the most popular means for developing commercial systems. Most companies opt for developing web based software wherever possible. This helps in catering to large number of end users and in the deployment of application.

The web based applications are powerful and have the ability to provide feature rich content to a wide audience spread across the globe at an economical cost. Hence it is a daunting task to test these applications and with more and more features testing these apps is becoming even more complex.

Testing web applications is different because of many factors scenarios affecting the performance and user experience. To ensure that the web application works reliably and correctly under different situations some factors need to be accounted for and tested.

Firstly, Test Planning and Test Design during web based testing should be done with extra efforts. Test Cases should be written covering the different scenarios not only of functional usage but also technical considerations like Images may take longer to download for slower networks and the end user perception of the application may not be good.

Also, Due to the design and nature of the web applications it is possible that different users follow different application usage paths as people with varying backgrounds & technical skills may use the application and in last which is common the end users may use different types of browsers to access the application.

Typically for internet based applications users may have different Browsers when accessing the applications. This aspect also needs to be tested. If we test the app only on IE then we cannot ensure if works well on Netscape or Fire-Fox. Because these browsers may not only render pages differently but also have varying levels of support for client side scripting languages such as java-script [6].

As Web applications become more popular and increasingly more complex, the need for test automation of these applications grows. Automating tests through a GUI is difficult and can be costly. The most popular methods of automating using a GUI are to use window co-ordinates to click items or to create and use references to window objects. The first method relies on locations of objects not changing on a page; the second usually relies on a type of proprietary object mapping format where all Web page objects must be captured and saved prior to scripting. Another approach is to seek out and use testable interfaces that are built into Web browser applications and provide the ability to directly access the objects on the page as they appear in the Web browser.

In the case of Internet Explorer, which is the basis of our study, there is a published, maintained testable interface in the form of the Component Object Model (COM) interface. This interface provides almost full control of the Internet Explorer Web browser and allows access to the objects in Web pages presented in the Web browser [7].

III. WATIR (WEB APPLICATION TESTING IN RUBY)

Developer(s)	Bret Pettichord, Charley Baker, Angrez Singh, Jari Bakken, Jarmo Pertman, Tom Copeland
Stable release	1.7.1 / January 9, 2011
Development status	Active
Written in	Ruby (programming language)
Operating system	Cross-platform
Type	software testing framework for web applications
License	BSD license
Website	http://watir.com/

Watir is built on the object-oriented scripting language Ruby. Its developers have used it for large-scale system testing, functional testing, and for automating user acceptance test. Watir uses a real programming language, is a free open source product, and allows direct control over objects such as HTML and JavaScript in a Web page. There are no layers between the HTML and test scripts—one can create its own if required and no license fees [7]. The basic specifications of Watir are depicted in Fig. 1 as below:

A. Working Principle

Watir makes use of the fact that Ruby has built in OLE capabilities. As such it is possible to drive the Internet Explorer programmatically. Watir operates differently than HTTP based test tools, which operate by simulating a browser. Instead Watir directly drives the browser through the Object Linking and Embedding protocol, which is implemented over the Component Object Model (COM) architecture.

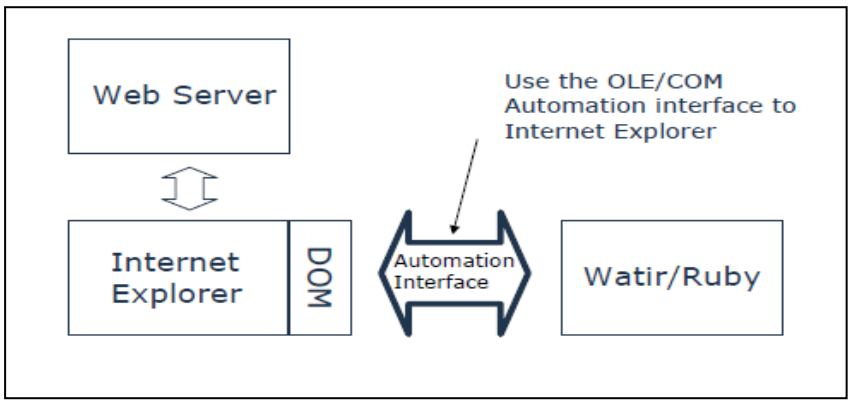


Figure 2: Watir architecture

As shown in Fig. 2 COM permits interprocess communication (such as between Ruby and Internet Explorer) and dynamic object creation and manipulation (which is what the Ruby program does to the Internet Explorer). Microsoft calls this OLE automation, and calls the manipulating program an automation controller. Technically, the Internet Explorer process is the server and serves the automation objects, exposing their methods; while the Ruby program then becomes the client which manipulates the automation objects [8].

B. Ruby (Scripting language)

Ruby is a dynamic, general purpose object-oriented programming language that combines syntax inspired by Perl with Smalltalk-like features. Ruby supports multiple programming paradigms, including functional, object oriented, and imperative. It also has a dynamic type system and automatic memory management.

The standard 1.8.7 implementation is written in C, as a single-pass interpreted language. As of 2010, there are a number of complete or upcoming alternative implementations of the Ruby language, including JRuby, IronRuby, Rubinius, and MacRuby. Each takes a different approach, with IronRuby, JRuby and MacRuby providing just-in-time compilation and MacRuby also providing ahead-of-time compilation. The official 1.9 branch uses YARV, as will 2.0 (development).

The basic specifications for Ruby are listed below in Fig. 3.

Paradigm	multi-paradigm
Appeared in	1995
Developer	Yukihiro Matsumoto
Stable release	1.9.2-p180, February 18, 2011
Typing discipline	dynamic, strong
Influenced by	Smalltalk, Perl, Python
OS	Cross-platform
License	Ruby License, GNU General Public License
file extensions	.rb, .rbw

Ruby is object-oriented where every data type is an object, including classes and types that many other languages designate as primitives. Every function is a method. Named values (variables) always designate references to objects. Though Ruby does not support multiple inheritance, still classes can import modules as mixins. Ruby has been described as a multi-paradigm programming language: it allows procedural programming, with object orientation (everything is an object) or functional programming [9].

IV. WET

WET Web Tester is a web based testing tool that drives an IE Browser directly and so the automated testing done is equivalent to how a user would drive the web pages. The tool allows a user to perform all the operations required for testing web applications – like automatically clicking a link, entering text in a text field, clicking a button etc. One may also perform various checks as a part of the testing process by using Checkpoints. The latest version of WET is 1.0. WET sits on top of Watir, an automated test tool which uses Ruby scripting language. WET retains all the features that Watir has and adds many usability related functionality [10].

WET started off as a small extension library for Watir – **W**atir **E**xtension **T**oolkit. It has come a long way and is now bundled with many compelling features. As the WET code base has grown multiple times, it now uses Watir just as the library to drive IE. WET not only offers all that Watir offers but many more powerful features to make test automation effort a success. With the increased features bundled into the code base, now WET no longer referred to be as Watir Extension Toolkit. It is simply called as **WET**.

A. Features/Components of WET

WET can be broadly classified as consisting of two layers: Wet Core and Wet UI.

The WET core layer is like the heart of WET with regard to test automation. This is the *engine* that drives automation using WET. The WET core is a library written using the Ruby scripting library. Ruby is a powerful object oriented scripting language and therefore WET itself has a great scripting ability. The WET library can be used to drive an Internet Explorer browser and perform various operations like clicking buttons, setting text, etc. on the IE browser. Wet UI is a collection of various UI utilities that assist the creation of scripts. Typically, each of the UI utility works as a front end for a corresponding WET core feature. There are various UI utilities that assist the tester to create WET test suites that may be called as front end UI for the various features of WET Core. The WET UI utilities have all been written using C# [11].

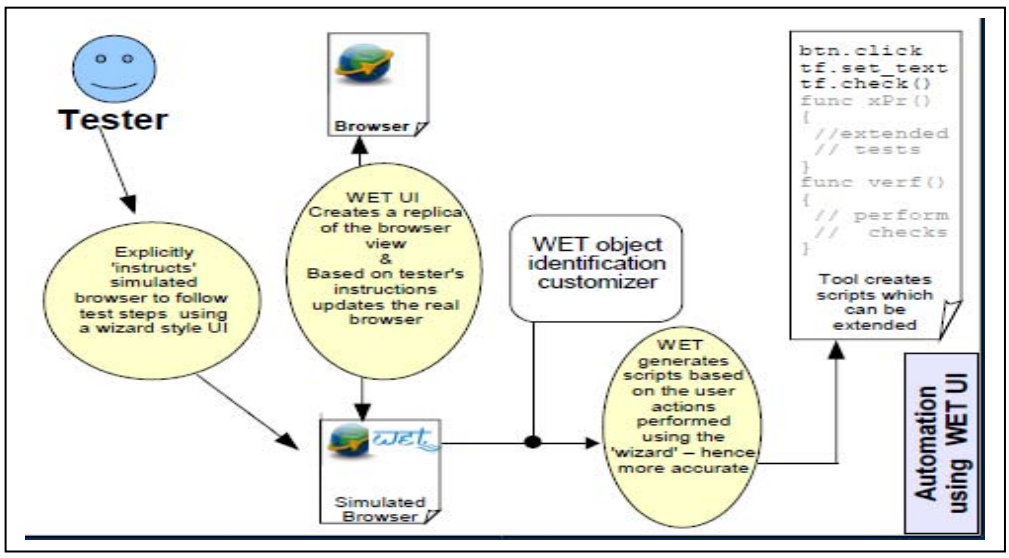


Figure 4: Automation using WET UI

V. TESTING SCENERIO

A. Evaluation Criterio

Before starting up the testing process for both tools, we included several criteria or desired functions for the evaluation purpose to see how both tools supports that criterion. Following are the criterion against which the tools evaluated:

1) Automation – Record and playback feature
2) Object parameterization
3) Ability to run multiple scripts
4) Object Repository
5) Script Reusability
6) Integrated Data Driven support
7) Ease of creation of scripts
8) Learning curve

For the evaluation purpose, we tested web based application – *Google* and its various utilities against above defined criteria, and generated the results for same.

B. Testing Steps

First step is to write a test case to automate functionality of Google using Watir and WET. Scripts for the same are written using SciTE editor or Notepad which are saved using .rb, .rbw file extension. For the testing process, we used IE (Internet Explorer) as a browser and the specified versions of tools - Watir 1.6.5, WET 1.0 and Ruby 1.9.2.

VI. IMPLEMENTATION OBSERVATION AND RESULTS

1) *Automation* - Watir is not a Record/Playback Tool because it is a family of ruby libraries. There is no User Interface of 'Watir' for Record and Play scripts. A user can write scripts to perform automated actions in ruby but there is no built-in feature of Record and Play in 'Watir' as shown below:

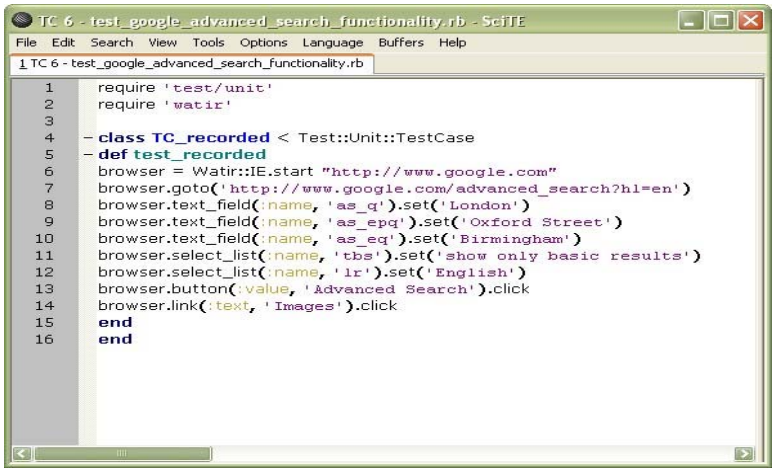


Figure 5: Test case to Automate Advanced search option of Google

Step 1: Script written above will Open IE instance, redirects to google.com and Click on the “Advanced Search” link located next to the search bar as shown below:

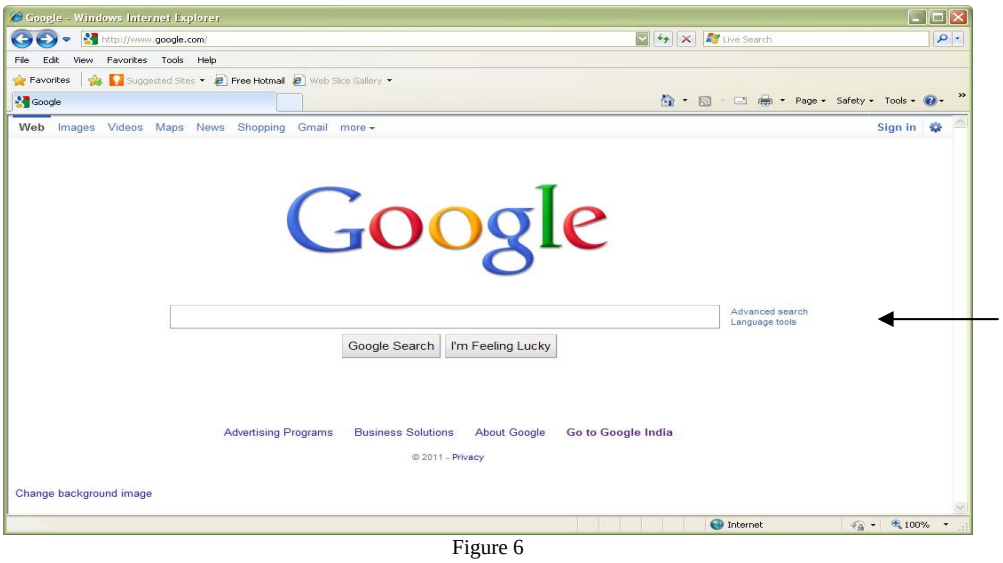


Figure 6

Step 2: Advanced search page opens, fills the text in the fields and automatic click on the “Advanced Search” button.

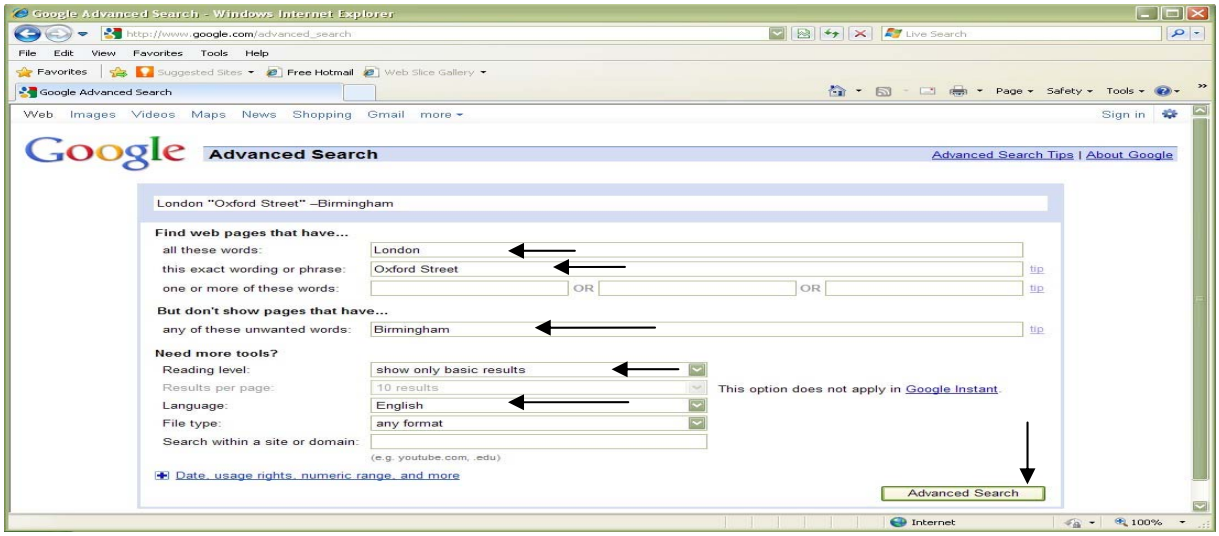


Figure 7

OUTPUT

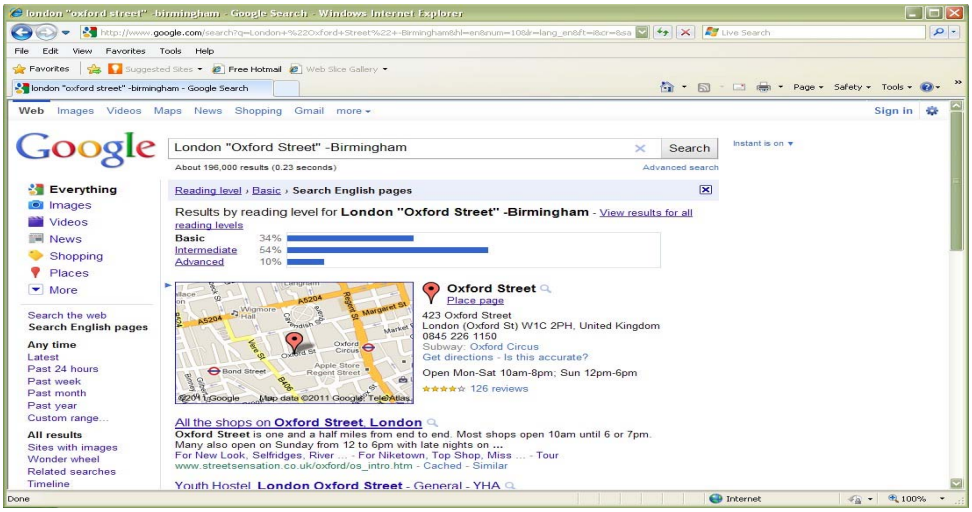


Figure 8

On the other side, WET is a Record/Playback Tool. There is a User Interface known as WET UI with different components which are used for record and play test scripts. User Interface Test automation is conventionally done by either using a record and playback technique or by scripting completely. Same test case as for Watir applied for WET as shown below:

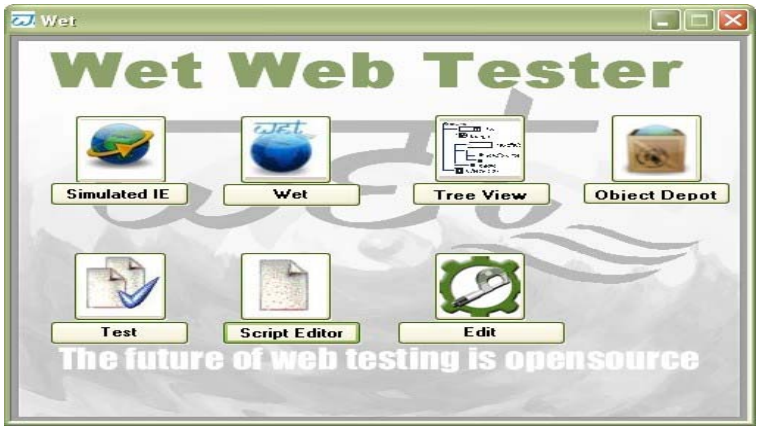


Figure 9: WET UI

Now click on Simulated IE, it will test a site which is already open in IE using *Sync with open IE* option:

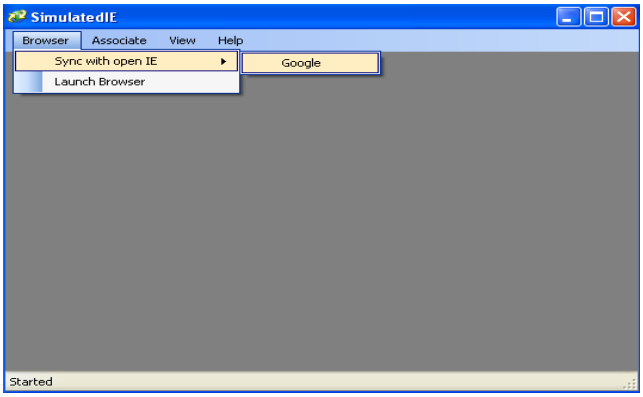


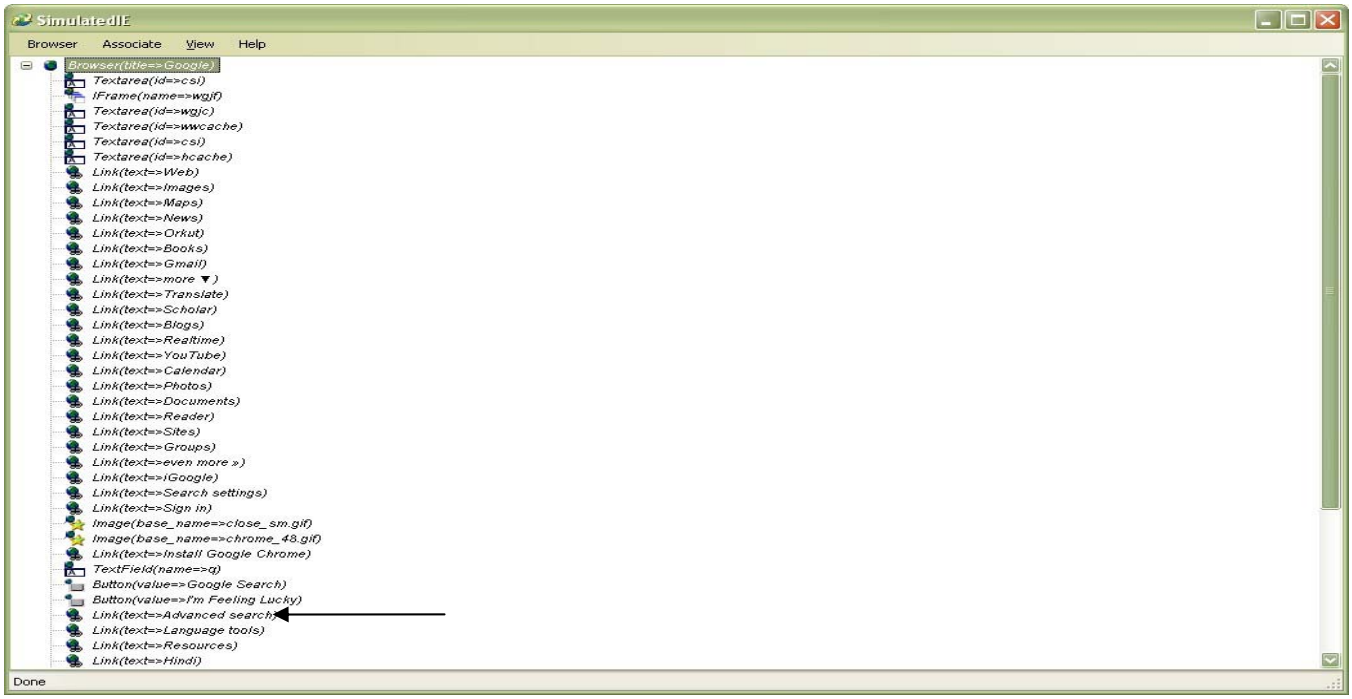
Figure 10: Simulated IE

Now click on Editor Button, a small window will come up on the screen. When we click on the dropdown list, it will show an untitled script which we opened earlier. Choose that script and click OK.

Figure 11: AssociatedScriptEditor

Next click on Tree View on WET UI which will open up Tree view of the objects used in this webpage as shown below. As we can see an object as “Link (text=>Advanced search)”, this is the link “Advanced search” located next to the Search textbox shown in google.com. As we need to click on this link to proceed further, so we will right click on “Link (text=>Advanced search)” link and click on “click” as shown below:

Tree view



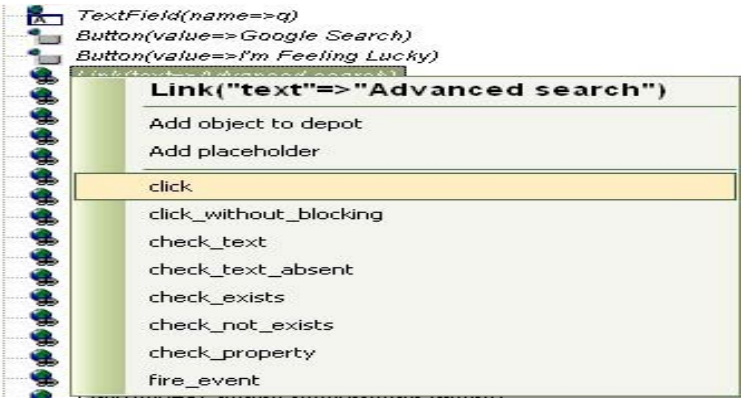


Figure 12

Script Editor - On clicking "click", it will automatically record this step in the script editor window being opened and associated earlier. Similarly, we can click all the respective fields and fill the respective values that need to be tested functionality. The final automatically generated script will look like as shown below:

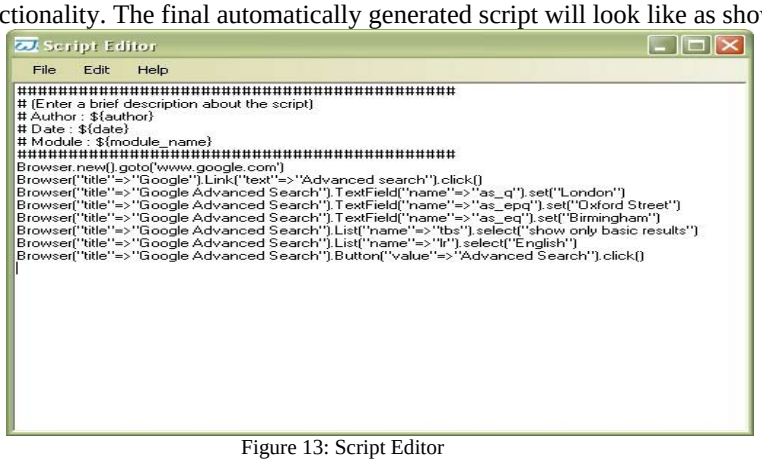


Figure 13: Script Editor

2) *Object Parameterization* - For example, there may be text fields with the label 'name' – one may be for the User's name while the other may be for the Developer's name. WET allows a tester to identify even these kinds of objects by letting to search for objects using multiple parameters. So, WET allows an easy way to use the same test script to test against multiple data values but object Identification through multiple parameters is not supported by Watir. Watir has library of Classes and Modules which are use to parameterize for Object Identification as shown in the following script to automate Google web search functionality:

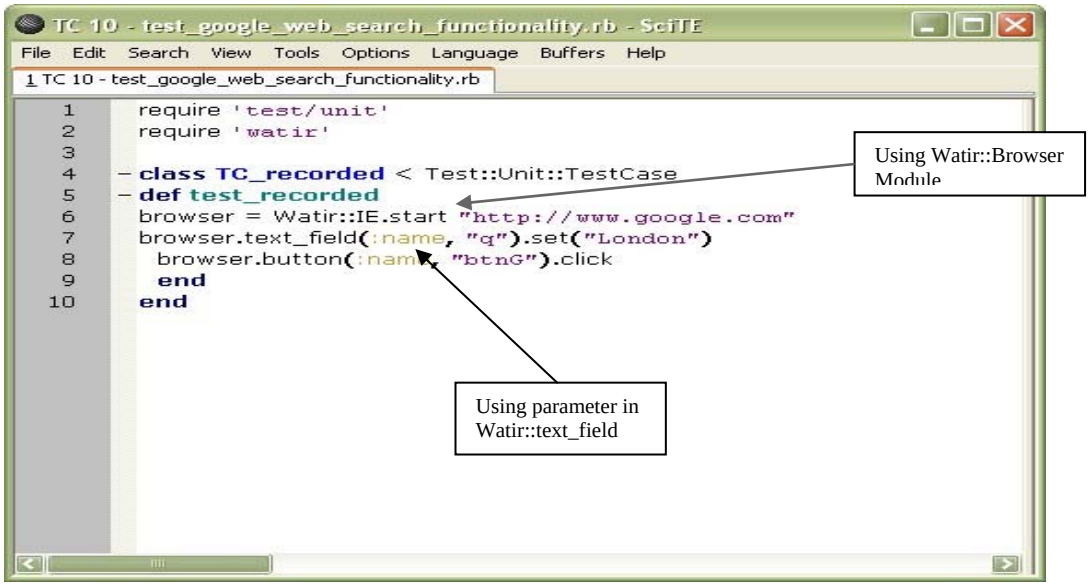


Figure 14

3) *Ability to Run Multiple Scripts* - Watir uses the Ruby (programming language) scripting language to drive Internet Explorer. Consider the test case for concurrent search, here we can write different scripts to perform different tests and those scripts can run at the same time but with different instances as depicted below:

```
# demonstrate ability to run multiple tests concurrently

require 'thread'
require 'watir'

def test_google
  ie = Watir::IE.start('http://www.google.com')
  ie.text_field(:name, "q").set("pickaxe")
  ie.button(:value, "Google Search").click
  ie.close
end

# run the same test three times concurrently in separate browsers
threads = []
3.times do
  threads << Thread.new {test_google}
end
threads.each {|x| x.join}
```

Figure 14

OUTPUT

Figure 15: Concurrent Search

On the other side, WET provides the facility to run multiple scripts at the same time in same instance via WET UI.

4) Object Repository - Watir makes use of the fact that Ruby has built in OLE capabilities. As such it is possible to drive Internet Explorer programmatically. Watir operates differently than HTTP based test tools, which operate by simulating a browser. Instead Watir directly drives the browser through the Object Linking and Embedding protocol. So there is not Object Repository with Watir where as an object repository is a very essential entity in any UI automation tool. A repository allows a tester to store all the objects that will be used in the scripts in one or more centralized locations rather than letting them be scattered all over the test scripts which WET provides through Object Depot.

5) Script Reusability - Watir uses Ruby which allows different scripts to perform different tests and those scripts can run at the same time but with different instances. Scripts written can be easily reused if we have Ruby interpreter is pre-installed on the system.Using Watir, we can write reusable classes/methods for library reusability but cannot directly reuse tests but WET provides support for plug-in in libraries where Reusable tests can be created and every test can define preconditions / teardowns using Ruby.

6) Integrated Data Driven Support - The concept of data tables is used to test more than one set of data for the same test scripts. For example if we have a script that tests registration, then we can test the script for different categories of data without having to rewrite the scripts for each test. The most common way to use data tables is to retrieve data from a table or spreadsheet. WET supports data table through Microsoft Excel or by using a XML based data table.

For e.g. consider the spreadsheet for employee, While running the tests, to retrieve data, we can use the utility method,
datatable().item(field, category)

Item	id	fname	lname	dob	doj
first_emp	001	Joe	Black	01/02/69	01/01/01
new_emp	0921	Raju	Rao	03/08/82	10/10/06
young_dude	079	Cool	Guy	05/09/85	05/06/03
elderly_sir	085	Robert	Gross	06/12/45	01/12/02

Table 1

Our script to create an employee could then look like:

```
1. all_data=["first_emp", "new_emp", "young_dude", "elderly_sir"]
2.
3. all_data.each do |dt|
4.     Browser(...).TextField("name:=empId").set datatable().item("id", dt)
5.     Browser(...).TextField("name:=fname").set datatable().item("fname", dt)
6.     ..
7.     ..
8. End
```

Clearly the above utility provides a very simple yet organized way of testing the same scenario using multiple data.

7) Script creation - Watir uses the Ruby (programming language) scripting language to drive Internet Explorer browser. Since Ruby is a full-featured modern scripting language so we need to have knowledge and information about writing the programming code. It's fairly easy creating of scripts provided the user have knowledge of Ruby scripting language. Graphically, it can be shown as below with knowledge (on the scale of 0-10) on the vertical axis and time on the horizontal axis.

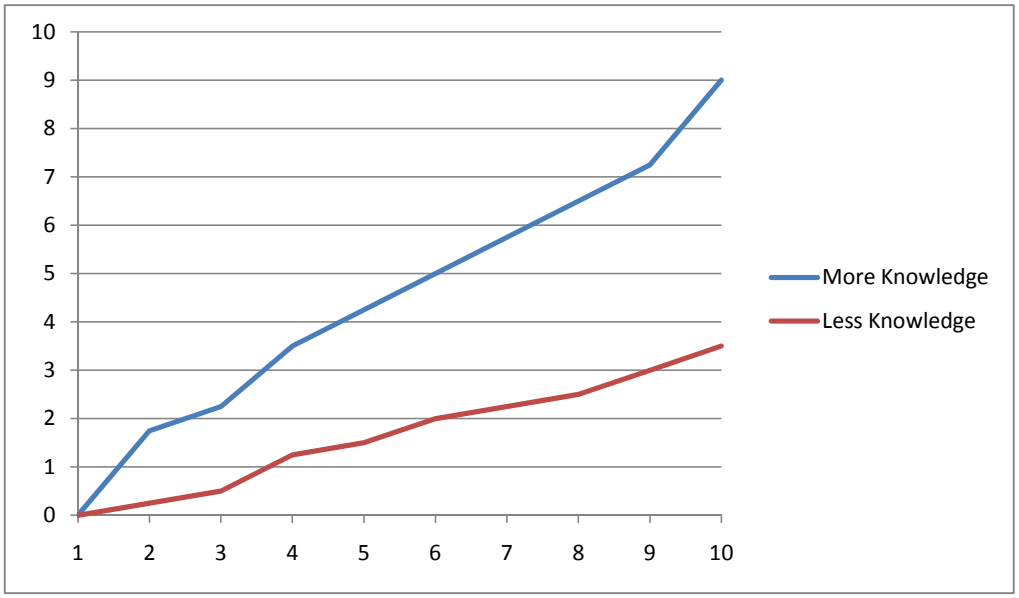


Figure 16: Graphical Representation for Watir

WET also uses Ruby for scripting but it supports three different views for script creation Simulated IE, Treeview, WET Simulation. Out of these, WET Treeview considered to be the best as the simulation mode to create accurate WET scripts which acts as script assistant. Graphically it can be shown as below with knowledge (on the scale of 0-10) on the vertical axis and time on the horizontal axis.

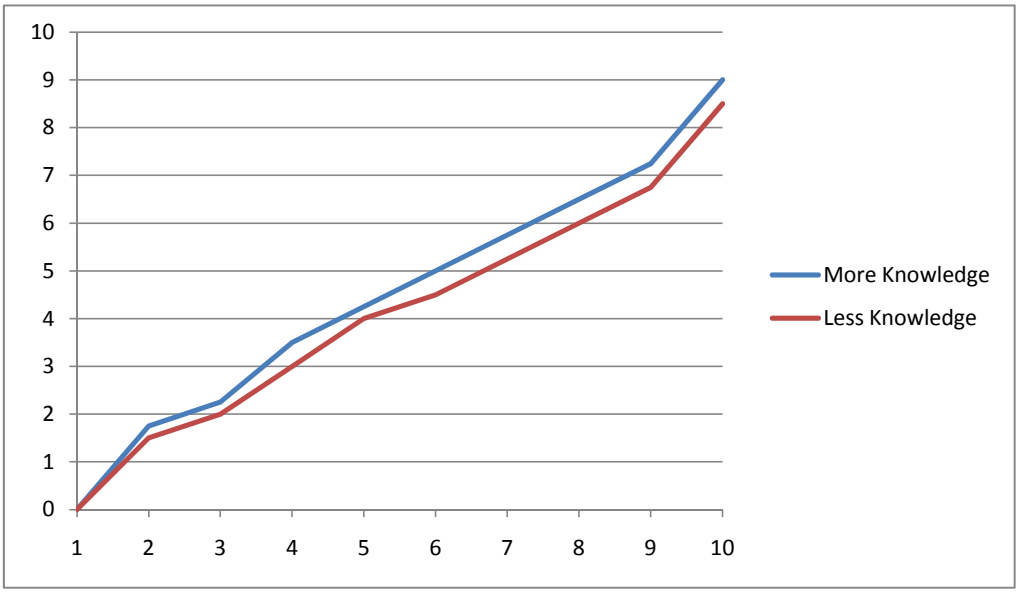


Figure 17: Graphical Representation for WET

8) Learning curve - Watir uses Ruby Programming Language so it has tremendous learning curve as far as programming methodologies are concerned. Ruby is a dynamic, reflective, general purpose object-oriented programming language that combines syntax inspired by Perl with Smalltalk-like features. So more we know about Ruby the more benefit we can take out of Watir.

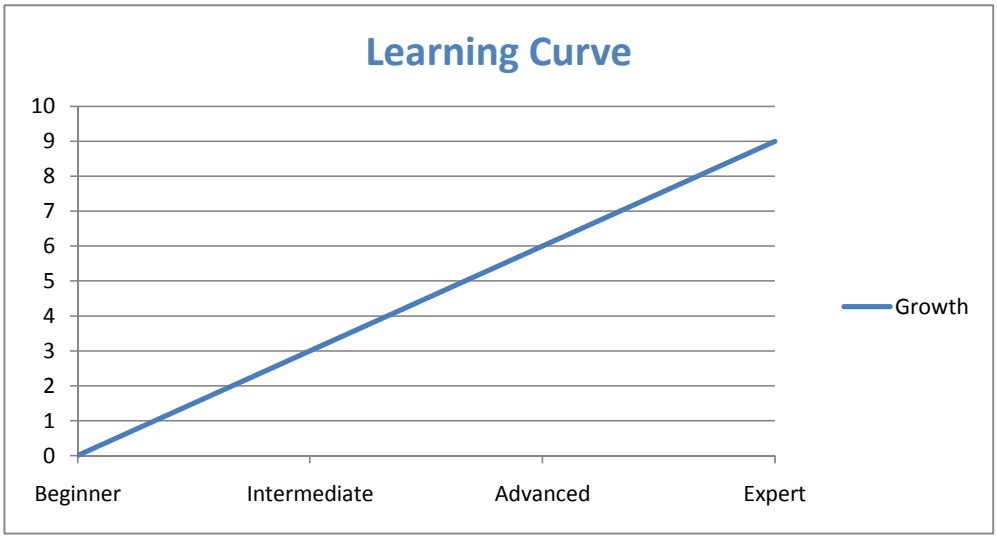


Figure 18: Learning Curve for Watir

As compared to Watir, WET is a record and playback automation tool. There is very good scope of learning as one can see the script without having much knowledge of Ruby. The automatically generated script can give the idea to the user how the script is written and what is the syntax for the particular event to fire and perform the respective action. So using the WET UI, one can start developing wet scripts quite easily.

Features	Watir	WET
Record and Playback scripts	No	Yes
Object Parameterization	No	Yes
Ability to run multiple scripts	No	Yes
Object Repository	No	Excellent
Script Reusability	Fair	Excellent
Integrated Data Driven Support	No	Yes
Script creation	No	Yes
Learning curve	High	Very Good

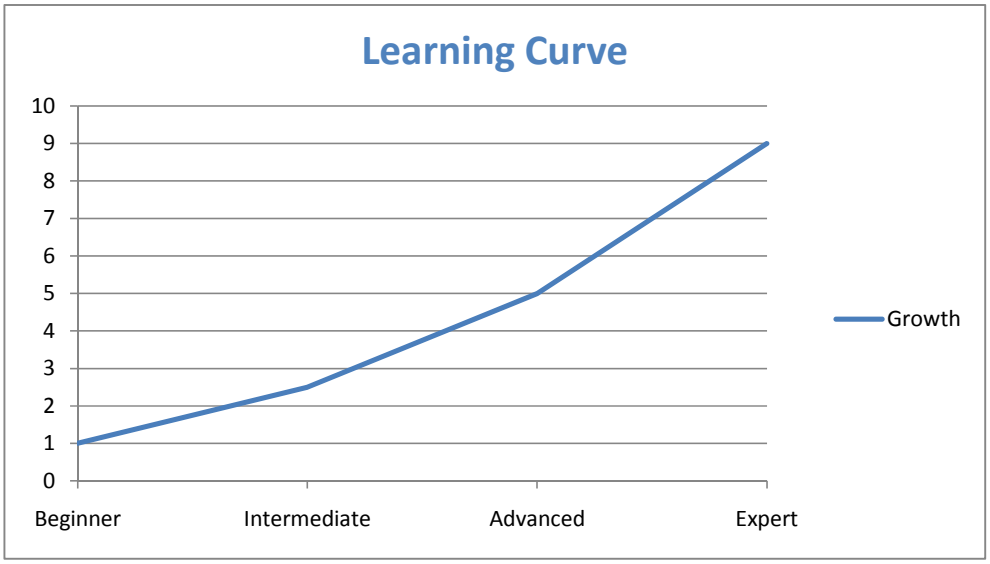


Figure 19: Learning Curve for WET

So, after evaluating Watir and WET against above defined criterions, we obtained following Results:

VII. CONCLUSION

In this study, we tested two open source Browser based testing tools Watir and WET. Results obtained after testing depicts that while using Watir, tester needs only to learn "keywords" required. The detailed test cases can be written in spreadsheet format containing all input and verification data. Also the transfer of knowledge is easier along with bug reporting and its application independent providing multiple browsers support. But initial time and effort to develop framework for Watir is quite high. Also, it does not support plug in applications like Java Applet, Macromedia Flash. On the other side, WET is a web based automated testing tool which uses Watir as the library to drive web pages. The only limitation with WET is that it currently supports Internet Explorer (IE) only. WET drives an IE Browser directly and so the automated testing done using WET is equivalent to how a user would drive the web pages. It extends the scripting abilities of Watir and also offers the convenience of recorders under LGPL and BSD style open source licenses. Future work includes testing the same web application across different available Browsers other than Internet Explorer (IE) like Firefox, Safari, and Chrome etc.

REFERENCES

[1] http://Web browser - Wikipedia, the free encyclopedia.htm
[2] http://Web testing - Wikipedia, the free encyclopedia.htm
[3] Zeng Wandan, Jiang Ningkang, Zhou Xubo, "Design and Implementation of a Web Application Automation Testing Framework", 978-0-7695-3745-0/09, 2009 IEEE
[4] Shauvik Roy Choudhary, Husayn Versee, Alessandro Orso, "WEBDIFF: Automated Identification of Cross-browser Issues in Web Applications", 978-1-4244-8628-1, 26th IEEE International conference on Software Maintenance, 2010
[5] Baowen Xu, Lei Xu, Changhai Nie1, William Chu C. H. Chang, "Applying Combinatorial Method to Test Browser Compatibility", 0-7695-2031-6/03, Proceedings of the IEEE Fifth International Symposium on Multimedia Software Engineering (ISMSE'03)
[6] http://Challenges in Testing Web Based Applications.htm
[7] BETTER SOFTWARE, APRIL 2005 by Jonathan Kohl and Paul Rogers www.StickyMinds.com
[8] http://Watir - Wikipedia, the free encyclopedia.htm
[9] http://Ruby (programming language) - Wikipedia, the free encyclopedia.htm
[10] http://WET Web Tester - Wikipedia, the free encyclopedia.htm
[11] http://WetForWebTesting.html