7th International Scientific Conference
Technics and Informatics in Education
Faculty of Technical Sciences, Čačak, Serbia, 25-27th May 2018

Session 2: IT Education and Practice

UDC: 004.43

Benchmark of Web Browsers with Automated Testing Tool

Srđan Nogo [1*], Zoran Škrkar [1]
[1] University of East Sarajevo/Faculty of Electrical Engineering, East Sarajevo, Bosnia and Herzegovina
[*] srdjan.nogo@gmail.com

Abstract: *The paper investigates method of benchmark of four web browsers against open source Automation Testing tool Selenium web driver. We will present two test scenarios and in both of them it is necessary to generate an automated test using the C # programming language, in combination with the Selenium web driver. The aim of this research paper is to evaluate and compare execution time for automated test setup against four web browsers to determine their usability and effectiveness. Based on the presented scenarios and described procedures, we will show that Microsoft has seriously approached resolving the deficiencies that existed on Internet Explorer, and that Edge has become a competitive browser, at least when we are talking about test executing, which has not been the case with Internet Explorer so far.*

Keywords: *Benchmark; Selenium web driver; Automated test; C# programming language*

1. INTRODUCTION

According to Glenford et al. in [1], "Software testing is a process, or a serial of processes, designed to make sure computer code does what it was designated to do and, conversely, that it does not do anything unintended". From previous statement we can conclude that the main objective of testing is to find bugs in the computer code and to fix them to improve quality of software. For example Srinivas and Jagruthi in [2], give an assessment that the process of testing consumes 40-50 % of development cycle time and more effort for software requiring more reliability as well. From this statement we can see that a significant number of quality assurance team working hours have been allocated for testing software tools for purpose of web browsing. According to Li et al. in [3], "Seeking information on the Web has become an important learning activity in current learning environment". This assertion points out that it is very important for users to choose a particular type of web browser that will save their time spent on searching large datasets. Thus, based on the correct selection of the web browser, they will avoid the situation of being exposed to disorientation and cognitive overload, and thus simplify their Information Gathering task to finding an answer or a Website.

In the present work, we planned to study methods of automatic testing of the response of four types of web browsers, with which we can measure the load time of a particular web site.

There are two scenarios. In both scenarios, it is necessary to generate an automated test using the C # programming language, in combination with the Selenium web driver.

This paper is structured as follows: After introductory section where the general definitions of automated testing are given, there is Section 2. describing the methodology used to start two different test scripts for automatic testing using Selenium Web Driver that supports four types of Internet browsers (Mozilla Firefox, Google Chrome, Internet Explorer and Microsoft Edge). Methodology Section was developed, as a starting basis for the proposed evaluation study for automated testing outlined in Section **3**. when the usage is concerned. Section **3.** presents a good practice case and explains the main focus of this survey paper. In this section, the final result of the research is presented in the form of the time difference in the performance of tests on different web browsers (the worst, best and average time of execution) of the test scenarios for each browser. Based on these research results, we can evaluate Benchmark for web browsers using an automated test tool and provide the visual means to confirm our summary and conclusions outlined in section **4**.

2. METHODOLOGY

According to Ieshin et al. in [4], use of automation test tool for program code testing increases the test execution speed and software become more reliable, repeatable, programmable,

comprehensive, and reusable. In the present work, we have created a test using the C # programming language, in combination with the Selenium web driver. Inderjeet and Tarika in [5], state that Selenium is one of the efficient open-source automated testing tools which provides a nice testing framework for testing wide variety of applications exporting scripts in almost every language including java, .net, c#. Selenium Web Driver supports all browsers for execution. With this automated testing tool, we can run more tests on different types of web browsers. Two tested scenarios were launched on 4 (four) different web browsers:

- Mozilla Firefox,
- Google Chrome,
- Internet Explorer,
- Microsoft Edge

Both of the tested scenarios are based on calculation of the response time required for the test to be performed using each web browser. The first test scenario was designed to measure the time required to open **google.ba**, then to search for the term **Automated test** and to check if the search results were loaded. The second scenario opens the **ibusiness.ba** page and, by clicking on each menu, checks if they are available and clickable.

The time in both scenarios is measured using the Stopwatch method in the following way, first the Stopwatch class object is created, followed by the Start method, as shown in "Fig. 1".

```
Stopwatch stopWatch = new Stopwatch();
// Start the stopwatch
stopWatch.Start();
```

Figure 1. *Creating object of Stopwatch class*

The next step is to call a method that performs the complete test, so that the desired URL opens and performs all necessary operations. Calling a method that performs all of the necessary test steps from Scenario 2 is shown in "Fig. 2".

```
//open ibusiness page and check menu
string message = LoadIBPage();
```

Figure 2. *Calling LoadIBPage method*

As you can see from the "Fig. 2", it is the string method which has some return value, and subject to its value, the results that arrive at the email, as a final report depend as well.

The last step is used to stop the stopwatch to get the final test time.

The stopwatch stops by calling the Stop method, as shown in "Fig. 3".

```
// Stop the stopwatch
stopWatch.Stop();
```

Figure 3. Stop the stopwatch

The LoadIBPage method is used to load the page you want and to check if the menus that exist on that page are available and whether they are functional. The basic idea is to somehow count the menus and the number of menus to be the upper limit of the *for loop*, in this way avoiding the possible "hard coding" in which you should know in advance how many menus page there are. The tendency of today's web pages is that there are always some changes, so it can change the number of menus. If the testing was made so that the number of the menus is "hard coded", any change in this number would failed the test. If this does not happen, the code is implemented by pre-counting the menus by using a simple java script function that is executed using **IjavaScriptExecutor** and which as a return value, has the number of desired elements. JavaScript is a powerful scripting language to develop cross-browser compatible software libraries. In combination with HTML5 or HTML6 in modern browsers, JavaScript is the language of choice to ensure portability and wide applicability interactive web-facing tools [6,7]. The entire process is shown in "Fig. 4".

```
string jsCode = "function rows(){return JSON.stringify($(\"#u155-18 a\").length);} return rows();";

IJavaScriptExecutor jsExecutor2;

jsExecutor2 = Driver.Instance as IJavaScriptExecutor;
string values = (string)jsExecutor2.ExecuteScript(jsCode);

int c = int.Parse(values);

for (int i = 1; i <= c; i++)
{
```

Figure 4. *Using the java script code to count the page's menu*

As given in "Fig. 4", all **the** elements within the parent element labelled with u155-18 are counted, where # indicates that it is an element in which the u155-18 is **id**.

The Java script code is written in such a way that the return value is in the JavaScript Object Notation-JSON format, after executing the code, returned value must be converted into **string** format, and then **into** the int format because the return value is required as a numeric value within the loop. By this approach, we have resolved a problem if menus are created dynamically, because their number are no longer important to us.

3. EVALUATION STUDY

Today, there is a large number of Internet browsers in use on the web market. Web browsers have become a major component of the routine human-computer interaction, with some operating systems entirely based on browsers (e.g., ChromeOS by Google [8]). They all have almost the same functionality and offer almost the same services, but they are not used equally by users.

Some web browsers come in a package with an operating system that is used on a local machine,

others can be downloaded from the Internet (may be commercial or written in open source technology) and the user can decide which type of web browser they want to use. This is true for most internet users, but when we talk about the business world, some other rules may apply. Some companies, due to certain security clauses in contracts signed with various partners and other companies, decided to use only a specific type of browser to access the Internet. This approach is a challenge for teams that test web applications. If testing is performed exclusively on a single browser, that kind of testing may be considered incomplete.

Because of this two test scenarios are presented in this paper, both of them are tested on 4 (four) different browsers, (Mozilla Firefox version 59.0.1 (64-bit), Google Chrome version 64.0.3282.186 (64-bit), Internet Explorer version 11.309.16299.0 and Microsoft Edge version 41.16299.248.0.). All tests are executed on machines with installed Windows 10 operating system, test code is written in C# programming language using automated testing Selenium WebDriver version 3.11.0.

The final result will present the difference in test run times on different web browsers, with the worst, best, and average test run times for each browser individually.

For both scenarios testing comparison between these four web browsers is made on the basis of the following:

- A concrete browser starts,
- Measurement of time begins,
- Opening the appropriate website,
- Test scenario is executed,
- Closing the browser,
- measuring time stops,
- A report will be sent to the email with the time of the test and information about the used internet browser.

Tests are directly run from VS (Visual Studio) environments as shown in the "Fig. 5".

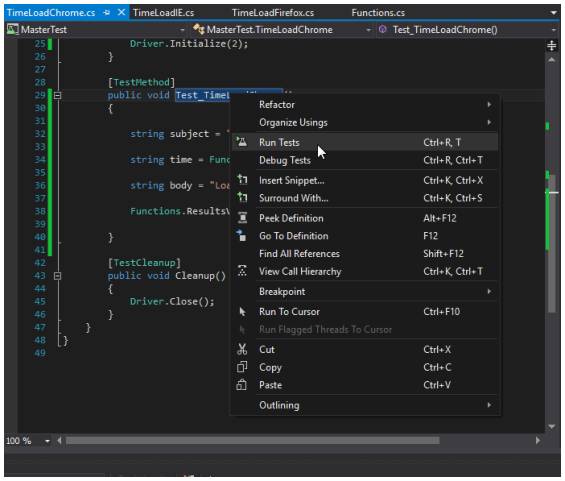

Figure 5. *Run test from Visual Studio*

3.1. Automatic test no.1

In this automated test scenario, we have a few steps to open a web site that is commonly known as "google.ba", to enter the term Automation test in the search box and click on the search after the result is displayed, it is necessary to check whether the first result is available in a row. After the completion of the test, on the test engineer's e-mail results with the time required for the execution of this scenario expressed in seconds will be sent. The test will be performed 10 times in a row on each examined browser, registering the best and worst performing times, as well as the average time for all of them. After completing all of the above test steps, the comparative results for 4 browsers are given in Chart 1. Time is expressed in seconds.

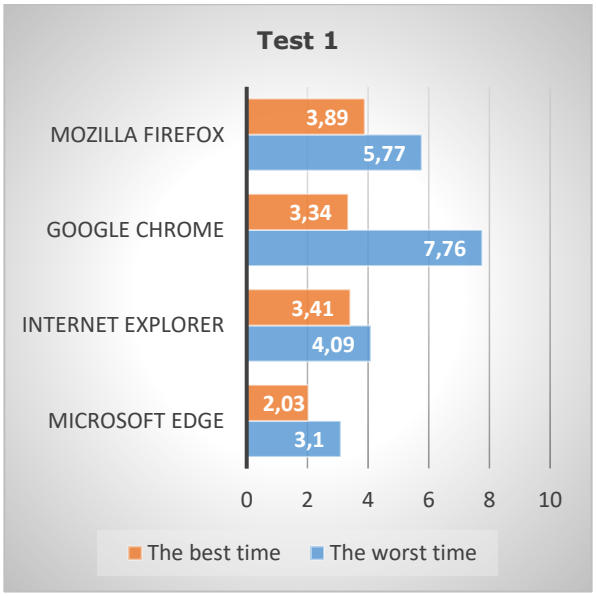

Chart 1. *Showing comparative results for 4 browsers*

The average time for performing the 10 reps for test 1. is given in Chart 2.

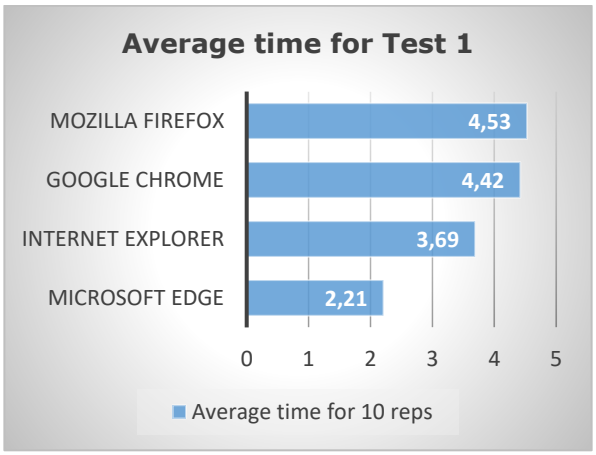

Chart 2. *Display the average time for 10 reps*

The results provided indicate that a Firefox browser had the slowest time, while on the other hand, the best results were achieved with Microsoft Edge browser.

3.2. Automatic test no.2

In the following automated test, the **ibusiness.ba** website opens and after the displayed result, it is necessary to check that 5 menus are active and accessible. After completing all of the above test steps, the comparative results for 4 browsers are given in Chart 3.

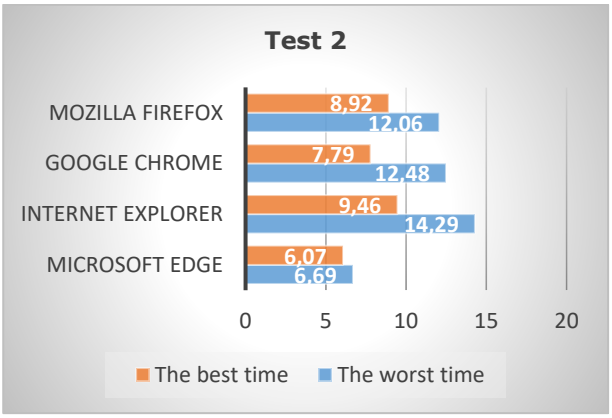

Chart 3. *Showing comparative results for 4 browsers*

The average test run time for 10 reps for each individual web browser is given in Chart 4.

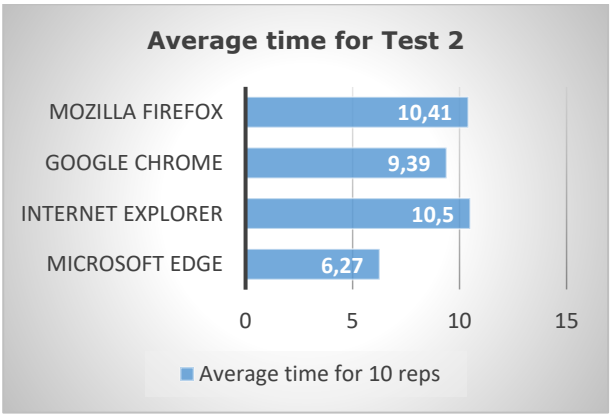

Chart 4. *Display the average time for 10 reps*

The average results for each of the individual web browsers are different from the results in test no.1. The Chart. 4, shows that the best average time as in test no.1, was achieved with Microsoft Edge, but based on the test result 2 in a more complex scenario, Internet Explorer has the worst result.

From the graphs shown, it can be seen that the test run rate differs from browser to browser, and that the difference between the worst and the best average time is reduced as the testing scenario becomes more complicated. In both scenarios, the best time has been achieved by Microsoft's next generation search engine (Microsoft Edge, which rightfully inherited an old and pretty obsolete version of Internet Explorer).

However, in the second test scenario, which is more complex than the first one, the difference between the best and the worst time is decreasing, and it can be concluded that by completing the scenario, the average execution time is approaching each other. That situation for future testing scenarios is more than good, because testing teams are given the option (if they are not conditioned by running tests on a specific browser), to select the one which offers the easiest way for creating the tests. It should be noted that at least two browsers are included in the testing process, while at least free would be optimal, but right approach is that at least one of the browsers should be Microsoft's one.

4. CONCLUSION AND FUTURE WORK

Based on the presented scenarios and described procedures, it is concluded that Microsoft has seriously approached to resolving the deficiencies that existed on Internet Explorer, and that Edge has become a competitive browser, at least when test execution is concerned, which has not been the case with Internet Explorer so far.

It can be said that it is on the test designer himself to adjust to the browser as desired, and subject to his experience, to select which browser to use for testing. We should bear in mind that most of the development teams have the most problems with the older versions of Internet Explorer that are still in use, and if necessary, testing in any of the versions of this browser would be desirable. This research work can be extended to more experiments with more tools and different comparative parameters parameters as i.e. tests should be executed on computers with Open source operating system.

REFERENCES

[1] Glenford J. Myers, Corey Sandler, Tom Badgett, (2011). *The art of software testing. 3rd edition.* ISBN: 978-1-118-03196-4.

[2] Srinivas Nidhra and Jagruthi Dondeti. (2012). *Black box and White box techniques-A Literature review*. *International Journal of Embedded Systems and Applications*, 2 (2).

[3] Li, L.-Y., & Chen, G.-D. (2010). A Web Browser Interface to Manage the Searching and Organizing of Information on the Web by Learners. *Educational Technology & Society, 13* (4), 86–97.

[4] Nevin, A. (1990). The changing of teacher education special education. Teacher Education and Special Education: The Journal of the Teacher Education Division of the Council for Exceptional Children, 13(3-4), 147-148.

[5] Inderjeet Singh and Bindia Tarika. (2014). Comparative Analysis of Open Source Automated Software Testing Tools: Selenium, Sikuli and Watir, *International Journal of Information & Computation Technology*. ISSN 0974-2239, Volume 4, Number 15, pp. 1507-1518

[6] Bienfait B, Ertl P.(2013) JSME: *a free molecule editor in JavaScript. Jcheminformatics*.5:24.

[7] Earley CW. CH5M3D: an HTML5 program for creating 3D molecular structures. *J cheminformatics.*2013;5:46.http://www.ncbi.nlm.nih.gov/ pubmed/24246004. (Accessed: March, 2018)

[8] *http://www.chromium.org/chromium-os;.(*Accessed: June 1, 2017)