Testing Responsive Web Pages Using the Consistency of Automated Web Pages

Yulia Samoylova

Dissertation 2014

Erasmus Mundus MSc in Dependable Software Systems

NUI MAYNOOTH

Ollscoil na hÉireann Má Nuad

Department of Computer Science

National University of Ireland, Maynooth

Co. Kildare, Ireland

A dissertation submitted in partial fulfilment

of the requirements for the

Erasmus Mundus MSc Dependable Software Systems

Head of Department : Dr Adam Winstanley

Supervisor : Dr. Stephen Brown

June, 2014

Declaration

I hereby certify that this dissertation, which is approximately 16 000 words in length, has been composed by me, that it is the record of work carried out by me and that it has not been submitted in any previous application for a higher degree. This project was performed by me at the National University of Ireland, Maynooth towards fulfillment of the requirements for the module CS645: Erasmus Mundus Dissertation in Dependable Software Systems. The work was conducted under the supervision of Dr. Stephen Brown.

In submitting this project report to the Nation University of Ireland, Maynooth, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Acknowledgements

I would like to thank my supervisor Dr. Stephen Brown for his suggestions, patience and motivation for this dissertation. He helped and guided me throughout this dissertation project. His positive attitude towards the project encouraged me during the whole semester. I would also like to thank Dr.Rosemary Monahan for her constant support during the two years of Erasmus Mundus MSc in Dependable Software Systems.

Abstract

The introduction of mobile devices with smaller screens motivates the need for web pages that work correctly across many different devices—referred to as *responsive web design*. Mobile access is a key feature for companies: both to reach new customers, and also to provide an enhanced service to existing customers. Testing the correct appearance of a responsive web page on different devices is not a trivial task because there are no standard rules for responsiveness, and the layout may need to be significantly rearranged in order to fit on smaller screens.

This dissertation describes an investigation into the automated test case generation for responsive web pages. The aim of the project is to develop a novel approach to test responsive web pages for the consistency of their appearance. This thesis develops the idea of consistent web pages which are not identical but provide essentially the same interface to the user. The goal of this project is to develop approach on how to do automated test case generation of the appearance of responsive web page and to evaluate this approach by building a test tool that automate the testing of consistency of responsive web pages against the master copy of the page. Several rules were implemented for comparing web pages displayed on different screens. The results show that the approach taken is an effective way of automating testing for consistency. This dissertation also identifies a number of unanswered research issues to be addressed by future work.

Contents

1. Introduction

The research project for “Testing Responsive Web Pages” was performed in the context of the dissertation project for the Erasmus Mundus MSc in Dependable Software Systems. The aim of the project is to investigate an approach on how to do automated test case generation of the appearance of responsive web page which helps to test the display of the same web page on different device screens. The development of the approach will be explained and the background of the web testing will be shown. After the rules and heuristics for automated testing were determined, I developed a program to evaluate the technique. All stages of development of the program are shown.

Currently, there are some existing applications for testing responsive web pages which provide the functionality to show the webpage on different devices screens but do not add any functionality for automated comparing the appearance of these web pages. The implemented approach is an alternative to manual testing and semi-automated testing.

1.1 Motivation

The project is motivated by Jennifer Keane, a Senior Developer at SAP Business Object in SAP Ireland. This project is designed to meet the real needs for developers, testers and users. With the rapid growth of smartphones and tablets, lots of people are starting to access the web pages from different devices with various screen sizes. Many companies, including SAP need to ensure that their web pages are readable and displayed correctly on any devices.

1.2 Research Question

The research question for this project is:

- How to do automated test case generation of the appearance of responsive web pages.

Automated testing of responsive web pages using, for example Selenium, is possible but very time consuming. We are not working on how to do automated testing of the web pages as these tools already exist. Using these tools for testing a responsive web page on different screen sizes is inefficient as the tester is required to generate test cases manually.

We are looking at the specific case where there is a master copy of web page
a full-size 'reference' version of each page exists, which is displayed using responsive web techniques on smaller screens (such as mobile devices).

Since screen sizes are not all the same, the interfaces on numerous devices can be different as web page should be adjusted for the appropriate screen size. To fit a smaller screen, the content of the web page must be reorganized as, for example, text in particular will be unreadable if it just shrunk. To ensure that the web pages are shown correctly on different devices, the displayed interface must be tested on each device.

Testing can be divided into two types: manual and automated. Initially, automated testing can be more expensive, but over time it starts to bring significant benefits compared to manual testing. The main benefit of automated testing is that they will show if something breaks for free as in automated testing there is only extra cost in developing the tests and the tests could be executed without additional cost.

The key idea behind this project is to develop rules for the “consistent”(foot node -> say that it is not identical but consistent) display of a responsive web page. Imagine that if we have program that would check if a web page properly displayed on the desktop is shown suitably on the mobile screen. By doing this, we can significantly reduce the amount of manual testing and this is a direct impact on company profits.

1.3 Project Aim & Objectives

The aim of your project is to develop a technique for automated generation of test cases for responsive web pages based on using a master copy.

The goal is to complete several objectives:

- model how a web page is displayed
- understand the principles of Responsive Web Design (RWD)
- develop RWD mapping of the model
- develop rules for consistency
- develop a test tool to validate the technique
- evaluate the results

1.4 Evaluation Criteria

For selected test cases I am going to demonstrated automated generation and execution of the tests of both pass and fail scenarios. The discussion of the rules selection will be presented in Chapter 3.

1.5 Dissertation Outline

The work consists of six chapters. The first chapter is *Introduction* where the problem definition, project motivation, aim and goals are presented. The rest of this paper is organized as follows:

Section 2 *Related Work and Technical Background* gives a brief introduction to web applications and the challenges that affect the analysis and testing of web applications. Also this section describes related work and gives the technical background overview. A major part of the chapter presents RWD, its principles and existing tools for the testing.

Section 3 *Solution* describes the technique by expanding the idea of consistency, defining model and choosing the rules for comparing.

Section 4 Implementation explains the design solution and implementation aspects of the program.

Section 5 *Evaluation* gives the evaluation of the developed approach and show automated generation of test cases for the selected rules for both pass and fail scenarios.

Finally, the conclusion and suggestion of some open problems are reported in Section 6.

2. Related Work and Technical Background

2.1 Introduction

This chapter describes the relevant context survey and the work related to the project. The first section gives the overview of the addressed problem and the mobile web. Also an introduction of software testing, web application and different methods of testing are described.

Second section describes the Responsive Web Design: its principles, benefits and downsides. Then, next section shows testing tools that identify if the web site is correctly responsive and describe challenges and open problems in automated test case generation for responsive web pages.

The final section gives an overview of the technologies that are used for the program realization that evaluates the approach.

2.2 Problem Definition

A modern website is an essential tool for a company trying to expand their visibility to future customers. Many organizations, institutions and companies are using websites to get a bigger audience and obtain more clients. Today, people are spending a significant amount of time online. Nonetheless, at this time it is not enough to have a web page that is available through search engines. Nowadays, more and more people are using electronic devices to simplify their lives and gain access to the Internet, so websites need to be improved for all these devices to provide a great user experience.

PC users have smoothly flowed into laptops and now smartphones. Major companies, for example, Apple, Dell and Microsoft want to be in a smartphone market because at the end of 2011, worldwide sales of smartphones beat global PC sales [1].

Moreover, smartphones and tablets sales are growing every day and there is an increased attention to web interfaces for the mobile and tablet devices. How to satisfy all user requests that are becoming more and more difficult? How to make so that people could see the site on a Full HD monitor, or on a mobile phone? Also besides different screen sizes and resolutions, various platforms and web browsers, there are some differences on how users interact with the devices: using a mouse, or using touch screen capabilities.

Internet is becoming accessible to lots of people all over the world. Tim Berners-Lee stated that the universality is the primary design principle of the World Wide Web [2]. The Web as it is open by its nature should be reachable from any browser and device that have is connected to the Internet [3].

First web pages were simple HTML pages that could be automatically updated to suit the width and height of the browser. Long time before smartphones and tablets the very first pages were mobile ready and responsive. In 2003 this web page was republished to celebrate 20th anniversary of the World Wide Web [4]. This page consists of structured plain text with hyperlinks and this makes it simple to view on different devices [5].

Although the total number of websites is increasing, there are a big number of pages that are not optimized for viewing from different devices and browsers. According to Gartner's research report on worldwide shipment for smartphones, tablets, ultramobiles and PCs from 2012 to 2017 (Table 1) [6], the total shipment is expected to grow by 16% in 2017 from 2014.While the most significant grow is predicted for the tablets (69% increase from 2012 to 2017), the shipment of mobile phones will be about 71% of the whole worldwide device shipment by 2017. For this reason, the main focus of our project is the mobile web.

Device Type	**2012**	**2013**	**2014**	**2015**
PC (Desk-Based and Notebook)	341,273	299,342	277,939	268,491
Tablet (Ultramobile)	119,529	179,531	263,450	324,565
Mobile Phone	1,746,177	1,804,334	1,893,425	1,964,788
Other Ultramobiles (Hybrid and Clamshell)	9,344	17,195	39,636	63,835
Total	**2,216,322**	**2,300,402**	**2,474,451**	**2,621,678**

Table 1. Worldwide Device Shipment by Segment

2.3 Mobile Web

It is important to understand why people actually need a mobile version of the site. Someone might think that only a small percentage of users who visit the web site use mobile phones. But according to statistics from Facebook, more than 250 million active users currently access Facebook through mobile devices

[7]. Facebook has more than 500 million active users, so that 50% of them are mobile users. 75% of YouTube users report that mobile is their primary way to access the content [8]. More statistics on the mobile web, commerce and consumers can be found in [9], [6].The reason behind this statistics is that people cannot always carry a computer or laptop and the mobile phone is always close by.

Currently, many developers after creating a website, test the in different browsers (IE, Firefox, Chrome, Opera and Safari), but do not check how the web page will look on mobile devices.

There are several possibilities to create an application for smartphones. One way is to develop one application using Objective-C for iPhone and iPad, another application in Java for Android, BlackBerry and one more .NET application for Windows Phone 7. This is not the easiest solution, so another option for mobile is a web application.

These are the main advantages of creating a web application instead of using native apps:

- Cross-platform application
- Ability to update the web page at any time
- Simplicity to alter one web site than to change the design for each native application
- Users prefer to use browsers and not download special applications. [10]

2.4 Software Testing

Software Testing is an activity performed to assess and improve software quality. This activity is generally based on the detection of defects and problems in software systems [11].

The testing process is an integral part of any software development process and plays an important role at all stages of development. Fault detection in the early stages of the life cycle of software development can reduce the cost of the software and the probability of failure in the future.

Testing as an independent discipline of software development started in the early 90s of 20th century, when the clients, quality and development time became more important than before. The software testing is especially important in critical systems. For example, radiation therapy machine Therac-25 between 1985-1987 gave overdose of radiation to 6 people due to lack of checking the status of the

unit in the code and validation [12]. Also Mars rover in 1999 brought a loss of over $125 million due to the different measurement system that was used between developer teams [13].

Software testing covers a wide range of activities including specifying test objectives, writing and running test cases and studying the test results [14].

The developer is working on the implementation of the application using its specification. This implemented application is tested with test cases, or test data. A test case consists of execution conditions, set of inputs and expected results [15].

Creating a strategy for test cases design is the most important testing activity [14]. There are two main testing philosophies: black-box or functional testing and white-box or structural testing.

Black-box testing is when the test cases are developed by analyzing specifications and the test data is generated from the specification, whereas white-box testing introduces test data generation using both the implementation and the specification.

Testing is performed at three crucial levels of the software development life cycle [15]. Individual software units such as classes or methods of the application are tested using unit testing. Interrogational testing focuses on how multiple components of the application are functioning together. System testing is used to test the complete application if the implemented application has specified capabilities and characteristics.

2.5 Web Applications

Web applications are becoming more common: enterprise management systems and drivers for network printers, online shopping and communication switches and all these is only a small part of the applications that have web interface. In this work, we will call web-based applications those applications that have web-based interface.
Unlike a conventional graphical user interface, web-based interface is displayed by web-browser and not by application itself. Web-browser takes care of all interaction with the user and accesses the web-application when it is necessary [16].

Figure 1 shows the process of a typical Web-based application. The user interacts with the application through the web-browser, which, if necessary, makes requests to the web- application to perform a particular operation [17].

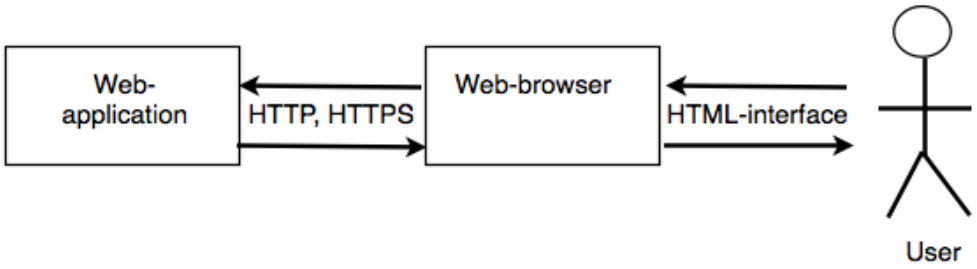


Figure 1. The process of a typical Web-based application

When the browser accesses the web application, it sends a request using one of the access protocols (HTTP, HTTPS , or others). The web application processes the request and returns a description of the updated browser interface.

A web application is primarily characterized in that the blocked user interface has a standardized architecture in which [17]:

1) for user interaction using web browser;

2) the interaction with the user is clearly divided into stages, during which the browser works with a description of the interface;

3) these stages are separate appeals uniquely distinguished from the browser to the application;

4) the interface described by a standard representation (HTML);

5) communication between the browser and the application is made by the standard protocol (HTTP).

Web-based applications may be viewed as a client/server application in which the functionality is implemented both on a server and on client side. Functionality implemented on the client side is usually input validation and implementation of additional interface features that are implemented using HTML built in scripting capabilities. (JavaScript, VBScript and etc.)

2.6 Testing Web Applications

There are several quality aspects of a web application that should be addressed in the testing processes [18]. These aspects are:

- Structural Quality: All parts of the website should be connected and all links inside and outside should be working.
- Content: The content of critical pages should be as it supposed to be and HTML code should be valid.
- Timeliness: As the web page could change often and rapidly, the changes should be highlighted and properly tested.
- Accuracy and Consistency: The content of the web page should be consistent and the presented data should be accurate.
- Response Time and Latency: Website server respond to a browser request should be within certain performance parameters.
- Performance: performance should be adequate under different usage loads.
- Underlying complex architecture: There is a need to perform testing with elements of the architecture including web, database and application servers.
- Usability: A web page should be easy accessed by different types of users.

In this paper we focus on the interface of the web application, leaving functional testing of responsive web pages on the server and client-side for future research. For this project, the main interest is the representation that is done by browser.

2.7 Manual and Automated Testing

Testing of the user interface can be performed either manually when the operator is involved, or it can be performed automatically with the use of various tools.

Manual testing of the user interface is beneficial because the person checks the correctness of the interface. This means that the verification is done by the major “consumer” of the system. In some cases, human perception is more important than strict adherence to the rules as a tester can notice some minor changes in the interface that are not specified in the requirements of the system [18].

The main disadvantage of the manual testing is that it requires significant human time resources. This disadvantage is especially noticeable during regression testing or if there is a need in frequent re-testing. Due to these drawbacks, in the last decade there were developed several tools for UI automation that are reducing the load for testers. These tools soon became popular.

With **record and playback testing**, the tests are recorded during the manual testing of the program. This allows performing tests without direct human intervention for a long time, significantly increasing productivity and eliminating the repetitive actions during manual testing. At the same time, any small changes in the software under test require rewriting the manual tests [18].

UI test automation uses software tools to simulate the tester behavior during manual testing of the user interface. There are several methods to determine the expected state of the user interface: the first is to perform comparison and analysis of screen shots with the reference and, another method, is to access interface elements from the operating system (for example, access to all elements on the screen using their id or filed names).

When the information is passed to the interface tests and the information is collected after the tests for the analysis, the interface element can be accessed in two ways:

- By position, when the element can be approached by specifying its relative (relative to the window or absolute (relative to the screen) position and its size;
- By identifier, when each element has its unique identifier within the window.

When testing user interface using the manual or record and playback testing, there are high chances that after a few minor changes in interface, many tests will fail, as the script will not work correctly. In this **automated testing** method it is necessary to change a significant part of the system test scenarios accordingly to the changes in interface. This method is suitable for automated testing for systems that have well established and rarely changing interface [19].

The second test automation method, where the access is performed by using identifiers, is more resistant to changes in locations of elements in the interface.

2.8 Responsive Web Design

To conduct extensive testing, there is a need for a number of different environments. The tester should configure a representative sample of all possible operating systems, browser, and different security settings. This is applicable for any type of application.

There is an emerging need to test the web page in different browsers to check if the content is displayed properly. The webpage should be optimally displayed on different screen sizes as it is difficult to predict what devices the visitors will use. Responsive Web Design approach can be used to create a universal web application for various devices, including smartphones and tablets.

Responsive web design is a new web design approach that sets up the flexibility of the website to accustom to any device. A number of articles [20] [9], that discuss web trends, declared 2013 the year of responsive web design. These articles state that responsive web design was one of the biggest marketing trends in 2013.

Responsive Web Design was introduced by Ethan Marcotte in 2010 [21]. “Responsive Web design” is the approach that suggests that design and development should respond to the user’s behavior and environment based on screen size, platform and orientation” [14].

As the term “Responsive web design” (RWD) was coined only about 4 years ago, this topic is relatively new and there is not many publications on this topic. A search of the ACM Digital Library and IEEE conference publications gives a total of 5 papers connected to responsive web design in a title and meta-data. In order to represent a current state of art the latest materials that non-peer reviewed were used.

Web pages using responsible web design approach can have only one web page layout that will run on devices ranging from phones to huge TVs. Examples of these kinds of web pages could be yiibu.com or alistapart.com. The aim of responsive web design is to achieve readability and navigation on any device with minimal scrolling and resizing.

Key features of responsive web design are: fluid grid, flexible images and media queries [22].

2.8.1 Media Queries

Traditionally, users, based from which device they accessed the web page, were redirected to different subsites. These subsites have the same content but with the different layout and information design. For example, users who visit BBC.com could see the traditional PC web page or get a specific mobile version by being redirected to http://m.bbc.com.

Redirection requires multiple separate engineering efforts, this approach is optimized only for two screen layouts and this does not provide a great user experience if visitors are using significantly larger screens or tablets.

There is a need in displaying optimized layout according on the device the visitor is using at the moment. Media queries are a module from CSS3 specification. CSS3 helps web developers to create separate content and markup for different screen sizes using CSS with media queries. Each media query has two parts: a

media type and the query. Although all modern browsers support media queries, some older browsers might lack of media queries support. However there are some alternative solutions that could be used for a particular browser.

Below there is the example of CSS file using media queries. In this CSS min-width is a key word in CSS3 that associates a particular style with a particular screen size. Width is a media feature that is the most commonly used. The min-width property sets the minimum width for an element. The style layout and proportions of the web site content are changing according to the web browser width. Styles for the appropriate media device would be automatically identified and associated with the web page elements.

Figure 2 shows a responsive website on several devices.

```
/ * Start css * /

The basic styles for 'simple' browsers. For example, here will be styles for not high-end level phones (pocket Internet Explorer for Windows Mobile 6.5 here).

@ media only screen and (min-width: 480px) {

Here are the styles for more advanced mobile devices (for example: Android, iPhone and etc.)
}

@ media only screen and (min-width: 768px) {
Tablets in portrait mode.
}

@ media only screen and (min-width: 992px) {
Tablets in the mode of landscape, netbooks, notebooks , desktop.
}

@ media only screen and (min-width: 1382px) {
Desktop higher resolutions Television.
}

/ * End css * /
```

Figure 2. CSS example for responsive website

2.8.2 Fluid Grid

Fluid grid consists of columns that are expressed in relative widths [23]. The proportion is expressed in the percentages of their containing element. The grid resizes to match the window size. The basic idea for a fluid grid is to use a formula for calculating the proportions in percentage: «target / context = result». For example, if we have a psd layout with width of 1000px and it has two sections: section on the left has 270px in width; the right section has width of 730px. In the CSS by specifying the width 27% of the left column, the width of this column will be automatically resized. We can set this as following:

```
. leftcolumn {
width: 27% ; / * 270px / 1000px = 0,27 * /
float: left;
}

. rightcolumn {
width: 73% ; / * 730px / 1000px = 0,73 * /
float: right;
}
```

Figure 3. CSS example for Fluid Grid with 2 columns

If inside of the left column there is another block with 170 px width, then the width should be changed:

```
. leftcolumn. some-div {
width: 62,962963%; / * 170px / 270px = 0.62962963 * /
float: left;
}
```

Figure 4. CSS example for Fluid Grid with 3 columns

2.8.3 Fluid Images

Fluid images adjust their sizes according to their parent unit. The idea behind fluid images is to use the property {max-width: 100%}. Image with this property will never be bigger than its parent [24].

If the parent block is smaller than the dimensions of image inside, the image inside is proportionally decreased. This principle is applied to images, embed, objects and videos.

Additionally to using fluid images for resizing images according to the screen, there is an opportunity to create different versions of images for the screens that have different resolutions.

2.8.4 Benefits of Responsive Web Design

Using these techniques for responsive web design helps to provide the user with high quality navigation. All benefits from implementing responsive web design can be divided in 3 main categories: benefits for end users, benefits for developers and benefits for web masters.

Main advantage of using responsive web design is one device-independent website, giving time savings and easy maintenance. Implementation of responsive web design gives better usability and more consistent user experience [25].

Websites with implemented responsive web design have a single URL for all devices. This makes the updating and maintenance of the content easy. Changes and editing the content is made in the same HTML instead of updating mobile and desktop versions separately. It is important to have one URL for sharing on various social networks. The users will be directed to a website regardless of the device the user is using [23].

Responsive web design can help in maintaining better flexibility and user experience on an increasing number of different devices and platforms.

2.8.5 Downsides and Potential Problems

There are several problems related with using responsive web design (RWD). The first issue is extra resources and time that is needed for RWD. RWD requires the developer to cut and replace the current content of the site in order to make it responsive. If the site was not developed with RWD guidelines, it would be difficult to transition it into a responsive site. According to J. Mazzei [25], responsive web design project cost will increase by 10-20% compare to regular websites. Nonetheless, in this case the company does not require a separate mobile site.

Another important issue is the time that a web page loads. Most of the users are expecting the page to load within 4 seconds. The long load time of the web page on mobile devices is a big problem for many responsive sites. This is an obstacle because most responsive web pages need to download extra CSS, and pictures to hide or shrink them. As reported by Wisniewski, the developers should think about web performance while designing, developing and deploying processes [26].

Furthermore, some older browsers do not recognize the CSS media queries and are not prepared for the use of Responsive Web Design. There should be an alternative solution developed for those devices. Also responsive images are still unsolved problem of responsive web design. The article [27] shows some attempts in solving it.

Another important issue is to know the users. If the users only assess site from mobile devices, the developers might just develop a few mobile applications. Also, suitability plays an important role. If one web page requires high resolution content, the mobile version might not be appropriate. For example, for advertisements it is hard to place banner advertisement in a screen of a fixed width.

2.9 Tools for testing if the web site is correctly responsive

Tools for testing if the web page is responsive are widely used to ensure that the website content is displayed on different devices without sacrificing user experience. These tools work a bit like HTML and CSS Validation.

Responsive Design Checker, Responsive Design Simulation and Galen Framework are discussed in more detail. Basic functionality and the main advantages and disadvantages of the tools are described.

2.9.1 Responsive Design Checker

Responsive Design Checker [25] provides features to show how site visual elements respond to different screen sizes. This allows the entire team to keep track of events during the development process.

Responsive Design Checker is a tool that is easy to use: user enters web address, click the “Go” button and select any device from the list. Responsive Design Checker can support the following devices: 27 inch monitor, 17 inch

workstation, 15 inch Macbook Pro, 11 inch Macbook Air and IPAD, Nexus 7 and iPhone both in landscape and in portrait mode.

Responsive Design Checker is easy to understand and it can be used by different users in the project.

Responsive Design Checker does not aim to test a mobile site version as it will not show if the web page is redirected to a mobile version. Also some devices, including iPhone, rescale sites to fit the screen.

The main limitation of Responsive Design Checker is the small number of different devices supported. The tester has to choose between the set of devices and the tester might not find suitable option. Also the tools do not provide the functionality to test world web page. These tools that perform automated display of the web page but the testing is manual.

Furthermore, this tool does not provide any analytics on the displayed on different sites web pages and there is no service to specify customer rules for web site look on various screens.

2.9.2 Responsive Design Simulation

Responsive Design Simulation[1] is another tool for testing RWD. It is a free resource developed by Hollingsworth Web Solutions. Responsive Design Simulation can be used by different members of the development team. This tool fulfills its purpose: to help the designers and audiences to visualize their responsive sites on most popular devices. The tool supports only limited set of devices: desktop, iPhone5 in portrait mode and retina iPad both in landscape and in portrait mode.

Responsive Design Simulation has similar to Responsive Design Checker restrictions: it is not developed to perform the replacement for actual mobile testing as the mobile browsers can render CSS and HTML individually.

The main constrains of Responsive Design Simulation are similar with Responsive Design Checker: limited number of supported devices, testing only world web pages and there is no functionality for the analysis of the displayed pages.
In this section other tools for testing responsive web design, such as “Responsive Web Design Testing Tool by Matt Kersley”, Resposinator and

[1] Responsive Design Simulation - http://responsivedesignsimulator.com/

others are not discussed in details as they have similar functionality with the discussed tools.

2.9.3 Galen Framework

Galen Framework[2] is a special language and a tool for testing the display with a web site in a browser. It allows a tester to perform cross-browser testing and also to test for responsive design.

Galen Framework uses Selenium to open a page in the selected browser, resize the browser window and get the information about the items on a Web page.

Also Galen can be used in the front-end approach TDD (Test Driven Development) development: at first tests should be written using Galen syntax, then the developers implement front-end and later run tests and do refactoring. Unfortunately, Galen has not been fully tested and is only introduced to a public not long ago.

Galen Framework helps to optimize the site for different devices. User can identify several types of devices: mobile, tablet, desktop and specify how the same elements will be displayed differently on various devices.

First step for using Galen Framework is to write Galen Specs, where we define the names of all elements we are going to work with on the page. In Galen we can use three way of getting the elements by id, css and xpath.

```
header              id   header
header-logo         css  #header .logo
header-caption      css  #header h1

menu                id   menu
menu-item-*         css  #menu li a

comments            id   comments

footer              id   footer
```

Figure 5. Example of Galen Framework script where names and elements are defined

[2] Galen Framework - http://galenframework.com/

Next step is to describe the properties depending on the size of the browser. There are 4 tags: mobile, tablet, desktop and all. The last tag is used for convenience to avoid listing all devices.

```
@ all
-----------------------------------
header
    width: 80% of screen/width
    height: 90px
    above: menu 10px

menu
    width: 70% of screen/width
    below: header 0px
    above: content 0px

footer
    width: 80% of screen/width
    height: > 150px


@ desktop, tablet
----------------------------------
menu
    height: 10px

footer
    below: content 0px


@ mobile
----------------------------------
footer
    below: side-panel 0px
```

Figure 6. Example of Galen Framework script for different devices

After the tests and the html document are ready, the developer can start tests from the command line. As a result, Galen will create html report, underlying all checks and errors.

Using Galen Framework, developers can test almost any layout of the page layout. The main difficulty is to choose the right way of specifying the check, so that it would be easy to maintain.

The main benefits of using Galen are the ability to check the relative location of the blocks and the visibility of the text. Also Galen works with Selenium Grid and runs custom javascript on the test page (for example, to open a drop-down menu and check the items). Galen Framework interacts with the browser via Selenium, in order to get to find right elements on the site if there is no direct link.

Galen can perform component testing. Galen can run individual files checks for these objects. Thus, we can realize the verification of complex repetitive elements on the page (for example: search results, comments, etc.) There are blocks conditions in Galen Framework. They are useful when it is impossible to know exactly which item will be displayed on the page (for example: banners).

There is a parameterization tests in test cases; for example, the same test case could be used for different screen sizes and various browsers.

There are also some limitations for Galen Framework due to Selenium restrictions and early stages of the project.

With Galen Framework it is not possible to check the location of the visible text and not of the whole unit. Developers plan to implement the detection of text in the screenshot but this will work only in Firefox as Selenium is experiencing problems of taking screenshots in IE and Chrome browsers.

Galen Framework cannot be run in real mobile browsers. This is Selenium's problem, because Selenium does not allow getting information about location of elements in the Android browser.

Galen also cannot compare screenshot and compare the colors of particular areas. There is a limitation of arithmetic operations that Galen can perform but in developer plan to support more complex arithmetic checks.
Although Galen Framework is a powerful tool, it requires the user to specify rules for every web page for every window size.

2.10 Challenges and Open Problems

Even though there has been some work done in testing the responsive web design, there are still many unaddressed issues. The major open problems in the testing of responsive web design are:

- Testing responsive web design is difficult due to its nature. The testing of the interface requires a tester to check the web page on different devices and compare with the prototype. There is no single model of the web page because if there was, the testing could be done against that model. Every page has its own design and it is hard to check whether something is correct or not. The chosen way is a rule based approach rather than parametric approach.
- Thus, automated testing of the responsive web design is exceptionally difficult to perform for web applications. In addition, current testing techniques require user to generate view prototypes for each of the screens.
- Some existing tools for testing the responsive web pages have only a small set of screen devices as a result the look of the web page is not tested on adequately large set of screens.
- There is a lack of an automated testing for the content of web-based applications.
- There is a lack of implemented approaches to test responsive web pages.

2.11 Focus of this Dissertation

This dissertation focuses on presenting an approach for automated testing of web interfaces. The approach involves extracting DOM model of visible elements from desktop a web page and from a mobile web page. This dissertation empirically sets the rules for web pages with the respect to responsive web design. A program is developed to enable automated testing of responsive web design and there is an evaluation of the strategies performed in this dissertation.

2.12 Alternative Approaches

There are two possible approaches for the goal of automated generation of test cases for responsive web pages: compare DOM or use image processing for matching the displayed web pages. The first approach was chosen because of several benefits over image processing. The first reason is that image processing

approach does not take advantage of the information that is already available as DOM model is a part of the whole web. Another disadvantage of doing an image comparison is the difficulty with analyzing how the elements are changing and working with the scrolling bar.

2.13 Context Survey and Technical Background

In the following section, we will discuss some of the technologies that are used for the realization of our approach.

Currently sharing languages HTML, CSS and JavaScript is a standard for creating web pages [28]. HTML is a basic markup language which is used to describe the content in the web page. CSS identifies the appearance of HTML contents.

2.13.1 HTML

HTML (HyperText Markup Language) - is hypertext markup language used to create web pages and viewed using a browser. Its story began in 1989, when Tim Berners-Lee published an article [29], in which he suggested to use the concept of hypertext for the exchange of information between employees within the CERN (CERN) - the European Organization for Nuclear Research. Later, he developed the first specification for the language and the software to use it.

HTML language consists of HTML- elements, which are based on tags. Tag has a structure in either (div tag name enclosed in angle brackets) or </ div> form. The tag in first form is called a opening tag and in the second - the closing tag. Most items have both with the opening and the closing tags. In between there is the content of the element, which can be a text and other tags. Some elements consist only of the opening tag. Also each opening tag might have attributes in a form: key-value that is defined as following:

<div key1="value1" key2="value2" ...>.

Browser interprets this code and based on it displays the web page that the user can see on the screen. «HTML-element" is usually identified with the word "tag" because it is convenient to see two tags as a single structure. Thus, this structure: ... is usually called a "div tag".

2.13.2 CSS

CSS (Cascading Style Sheets) - a technology that is used to describe the appearance of a document written using a markup language. It is mostly used to design web pages written using HTML [30]. Code CSS is a list of consecutive rules. Each rule has the following form:

```
selector {
      key1: value1;
      key2: value2;
        ...
}
```

The first part, selector, sets a list of tags to which the rule should be applied. The body of the rule consists of a “key-value” pairs, where each pair sets different properties. These properties can specify margins, page positioning, color text, background and etc. CSS provides a big range of ways to design the web page and as this is described in [4], this was the main reason why the previously approach of using pure HTML failed.

2.13.4 Java

Among various programming languages, Java (http://www.java.com/en) was chosen for the implementation of this project. Java is owned by Sun Microsystems and was established in 1995. The list below shows main reasons behind the choice of using Java:

- Platform-Independent. Different machines are able to run Java.
- Available libraries. There is huge variety of open-source libraries that are available for use.
- Industry adoption. Since the project was proposed by real industry company, it was crucial to use language that is widely used in the industry.
- Object-Oriented Paradigm. Java is an object-oriented programming language that promotes re-use of the code.
- Previous experience. During the course, Java was used as a primary language for the taken modules.

2.13.5 DOM TREE

The Document Object Model (DOM) is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents. Elements in the DOM model may be manipulated. The public interface of a DOM is specified in its application programming interface (API) [31].

The elements in HTML, XHTML and XML can be viewed as a node in the DOM tree structure and each of these nodes is a object. DOM standard greatly simplifies the documents structure in the programming environment. For example, we will take a sample of HTML code and convert into a DOM tree. HTML source code is following:

```
<TABLE>

<TBODY>
<TR>

<TD> Element1 </TD>
<TD> Element2 </TD>

</TR>
<TR>
<TD> Element3 </TD>
<TD> Element4 </TD>

<TR>

</TABLE>
```

Figure 7. A graphical representation of the DOM tree of the HTML code.

A graphical representation of the DOM tree of this HTML code is shown on Figure 8 [31].

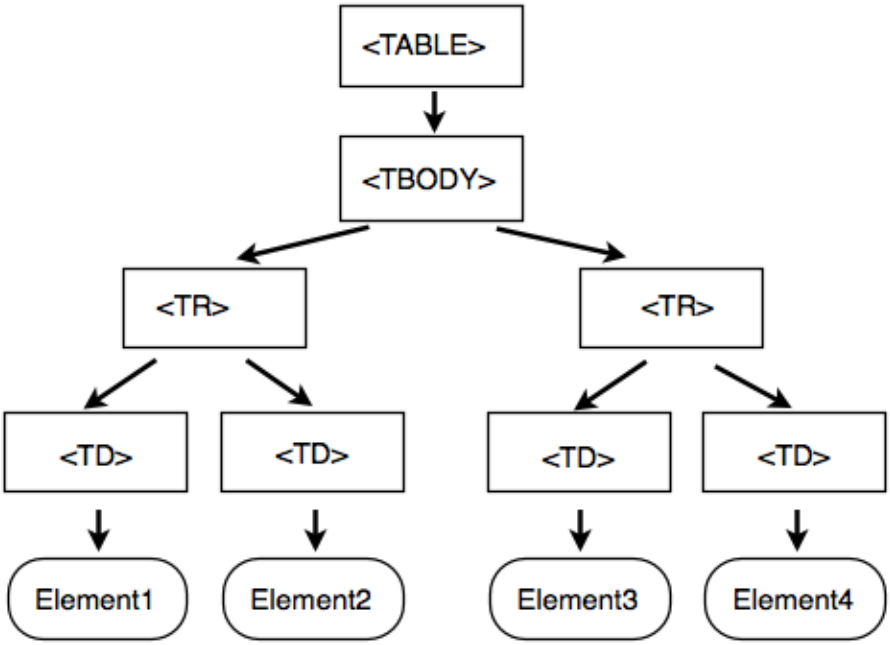


Figure 8. Graphical representation of the DOM tree for HTML sample.

The DOM structure of HTML documents is a tree, or if we want to be more precise, the document can contain several trees and could have a logical structure like a “forest” or “grove”. Each document has one root element node, that serves the element tree for the document. Although, the DOM does not state how the relationships between elements should be specified, it is a logical model that could be implemented in the convenient for the user manner.

For this project, we use DOM structure to make a tree-like representation for the documents. An important property of DOM for this project is structural isomorphism. This means that if we create two DOM implementations for the same document, the Document Object Model structure will be the same.

2.13.6 HTML DOM

There is a standard defined by the DOM for accessing HTML and XML documents [32].

There are 3 different levels in the DOM separation:

- Core DOM - a standard model for structured documents
- XML DOM - a standard model for XML documents
- HTML DOM - a standard model for HTML documents

HTML DOM is a platform and language independent, standard interface for HTML. In HTML DOM each node is HTML element. All nodes could be accessed via walking through the tree; their content also can be deleted or modified. The tree starts with the root node and branches out to the next children nodes at the lower level [32].

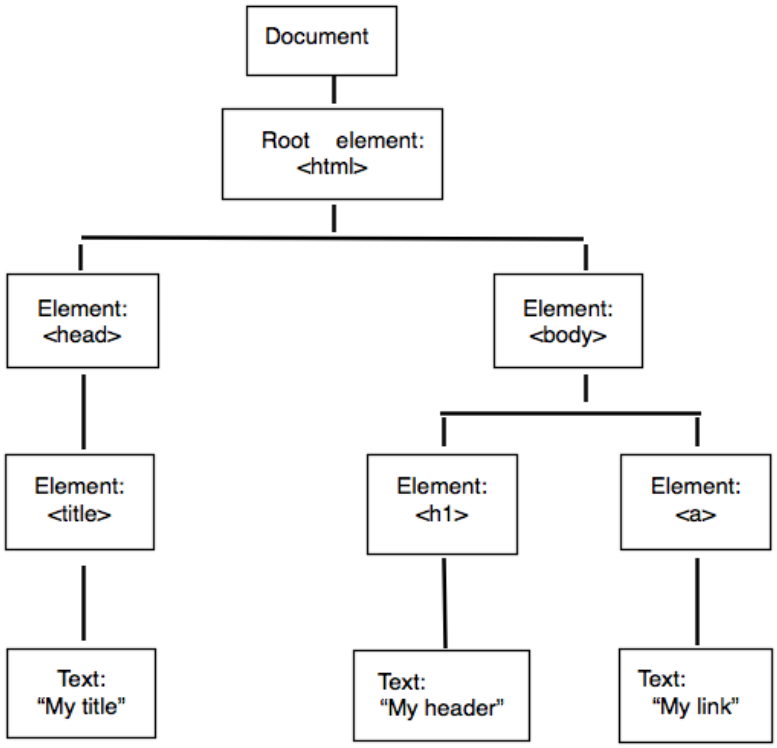


Figure 9. HTML DOM example

The nodes have a hierarchical relationship to each other. To describe relationship the terms parent, child and siblings are used. Parents node have children, siblings are the children being on the same level.

- The top node is called the root

- Every node has exactly one parent node. (except the root node)
- Every node can have any number of children
- A leaf is a node without children
- Siblings are nodes with the same parent

2.14 Conclusion

This chapter describes several technologies that will be applied in this project. Fundamental knowledge about this material is important, such as the appreciation of software testing evolution, the reason behind the RWD and several technologies. The DOM model was used to develop the approach of automated testing of responsive web pages.

3. Solution

3.1 Introduction

This chapter focuses on defining an approach for automated test case generation of the appearance of responsive web page and introduces the development of the idea of consistency. The approach is evaluated by developing a test tool that automates the testing of consistency of responsive web pages against the master copy of the page.

The fundamental underlying principle of the developed approach is a consistent web page that allows to automatically comparing a responsive web page against another web page that is designed to be a master web page. So the technique requires one thing: a master copy of the web page what the developer is happy with. This master web page is a representation of the web page with which the target web page needs to be consistent with.

The rules for consistency should be expressed in the terms of the model that is being used. Also there is an element of subjectivity in developed approach. The reason why this approach is subjective is based on how the evaluation works. The consistency can be expressed with set of rules but various people might come with different rules. However once there is agreement on consistency behavior, the users could choose specific rules for appearance of the responsive web pages. Moreover, within the company, people can come with a standard set of rules.

3.2 Overview of Technique

Responsive designs do contain layouts for different devices: mobile, tablet, laptop, desktop and etc. One of the fundamental choices, when these layouts are being designed, is to decide which layout to design first. This choice will affect the structure and content as well as the strategy of the web site.

There are two principle options for a web design: “mobile first” - the narrowest design and “desktop first” the widest design. [35]

Mobile First was created by Luke Wroblewski (2009). Mobile First philosophy outlines the prioritization of the use of mobile devices. Using this technique the simple content is first built for the screen with a limited screen size and bandwidth so that the mobile and simple browsers will receive the easiest

content. After developing this version, media queries and other features will be added to be displayed by more advances devices and browsers.

The second option is “desktop first” approach. Using this approach the version for the desktop is designed first and later it is scaled down, although when the site is designed it should be kept in mind that it will be used by different devices.

These two options on designing the layouts have a direct correlation with the chosen approach on testing. We are two possible choices for this project on how to test responsive design of the web pages. First choice is similar to “desktop first” - to start with viewing web page on the device with big screen and moving down to the device with smaller screen.

Second choice is to work with the web page viewed on a small screen device to compare that view with the same web page view on big screen device. This option has similar ideas with “mobile first” design approach.

The project is based on desktop first approach: we start with viewing web page on the device with big screen and moving down to the device with smaller screen. The reason for choosing "desktop first" approach was that although mobile and tablet usage is growing fast, mobile screens are getting bigger and this makes them closer to desktops screens. Also, many websites already have been developed for desktop version and there is a need to test them on different devices.

I want to highlight that it is a separate problem to looking at small screens first and later to work with big screens. Although, these approaches are related, they are different. This distinction is quite important. By following chosen approach starting from desktop version, we can say that all elements on the desktop screen is a super set of all possible elements that could be on the screen. Our aim is to make sure that the appropriate subset from the desktop set is shown on the smaller screen.

Using this concept we could talk about different subsets for various devices with different screen sizes. There might be a reasonable subset for a particular tablet, a next subset is for a mobile phone in portrait mode, and another will be a subset for the same mobile device that is in landscape mode.

An Important step is to have rules to map the final model from the superset to another subset for a particular device.

The terms used for automated testing of responsive web pages are not completely unified. It is necessary to clearly set a terminology that we are going to use.

- *Reference Web Page* - The web page that is shown for device with 800 x 800 pixels resolution which is the appearance-correct master model.

- *Target Web Page* - The web page that is shown for desktop having resolution of 480 x 800 pixels which needs to be tested.

- *Responsive display is a model as displayed on a different screen size.*

- *Transformation* describes the changes occurring to the interface of the web page when it is shown on the mobile screen after the desktop screen.

- Consistent Web Pages - Pages are called consistent if the reference and target web page and reference web page of the same site, are follow the set of rules.

We used a group of activities in Figure 10, to test responsive web pages.

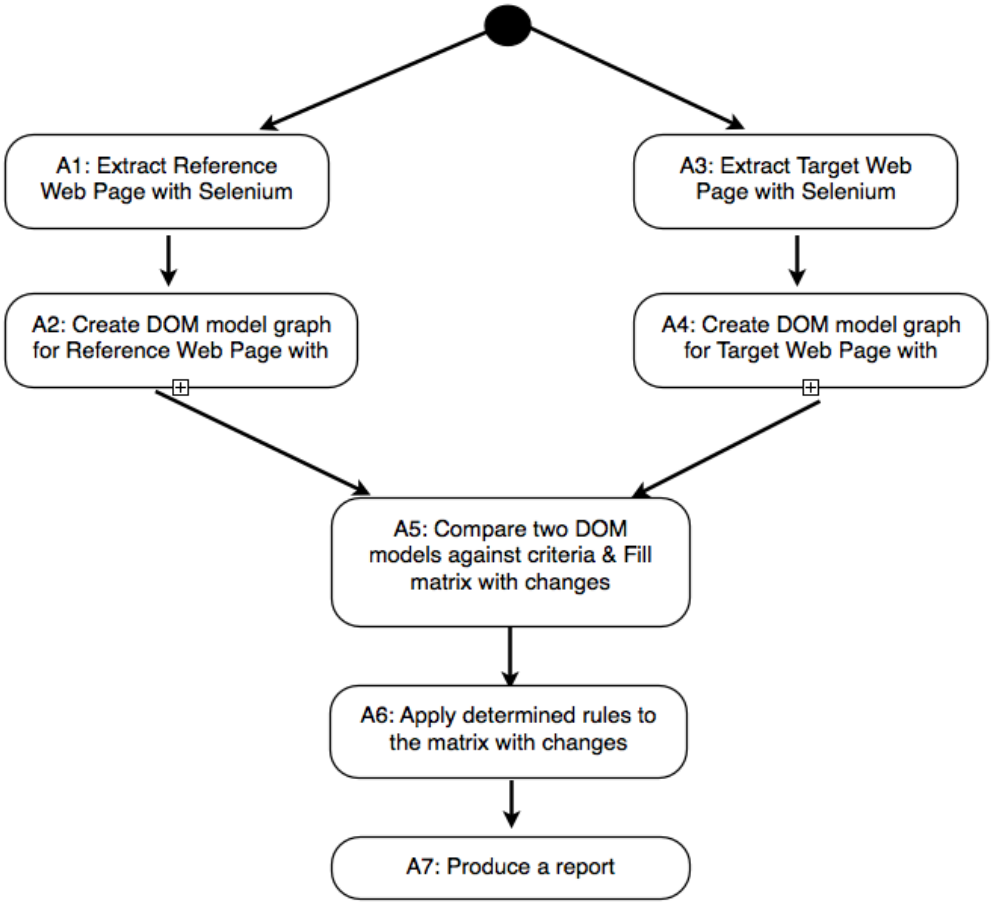


Figure 10. Flow graph of activities

Below we will discuss each of the activities in more detail.

During the first activity A1: Extract Reference Web Page with Selenium, we run the script that opens the provided web page with the desktop size screen. I chose the dimensions of the web page for the tool to be 980*800 pixels. This was needed because this is the usual page size used in websites, so most of the web pages have their web sites optimized for this size so that all information needed is displayed on the screen. After the web page was open with the proper screen size, in A2 activity the program parse HTML document of the web page into DOM Tree. Each node of the tree is an object. For our visualization of DOM Tree, use only visible elements on the web page are used. Having parsed web page into our adjusted DOM tree brings convenience to extract website information and to visually compare DOM trees of the reference and target web page.

Activities A3 and A4 for target web page are similar to A1 and A2 activities for reference web page. For the target web page the resolution of 480* 800 pixels was chosen as this is a common screen size for mobile devices Companies including HTC, LG, Nokia, Samsung have mobile phones models with these

screens. The examples of these models are HTC Touch HD, LG Optimus White, Samsung Galaxy S, Google Nexus S. After A3 and A4 activities, there is a DOM Tree of visible on the web page elements for the mobile screen size.

After A1.. A4 activities, there are 2 graphs of visible Document Object Model elements.

During the A5 Activity the matrix of changes is filled. This table contains the changes that were occurred in the interface of web page when it was shown on the mobile screen after desktop screen.

After this table is completed, in action A6 the user chooses the rules from the list. Depending on the chosen rules the matrix of changes is analyzed. The final action A7 shows the report with the results of chosen in the actions in A6 rules.

In the next 3 subsections, the actions A5, A6 and A7 will be discussed in more details.

3.2.1 Matrix of changes

Our DOM tree graph can be analyzed as whole or in sections, providing a range of flexibility for implementation of extraction algorithm.

As every node in the graph is an element, we can get the attributes of the particular element. To compare two graphs we perform search though two graphs and compare elements against specified criteria. Different sorts of transformation can occur. Based on different layout and information design the key criteria which reflect three concepts of responsive web design are deletion, resizing, physical movement on the screen, logical movement in the DOM, replacement.

For the program there were chosen 3 main criteria are: visibility, moving and scaling. There are the key criteria to do with the responsive web design: whether something is visible on the screen or not which reflects the earlier information, if the element moved or not and where the element was scaled or not.

So we check if each element is visible, where it moved and the scale from the original.

While going though graphs, we fill the matrix with the changes between two versions against specified criteria.

id	visible	moved	scale
body	yes	X = 1%, Y = 17%	No change
a	yes	No change	-100% on X, 0% on Y
hideme	no	Tag is not Visible	Tag is not Visible
h1	no	Tag is not Visible	Tag is not Visible
h2	no	Tag is not Visible	Tag is not Visible

Table 2. Matrix of changes

- For the first criteria “visible” we check if the element appears on the target web page.

- The movement of the elements is calculated in the following way. First the percentage of x and y axis is calculated with the respect to the browser window size. This calculation is done for both desktop and mobile version. After the computations are performed the difference between x axis desktop with x axis mobile and y axis desktop with y axis mobile is calculated and put into the table.

- To calculate the scaling value the difference between element size on the desktop and on mobile was calculated in percentage.

3.2.2 Rules

One of the most challenging tasks in the project was to identify the rules based on which the web page representations are considered consistent. Four critical, high level rules were chosen for evaluating the approach. I selected the current set of rules because they represent the major difference that might take place in the RWD. Each rule aims to achieve a specific goal.

As explained in the previous paragraphs, the program is responsive to execute defined rules. Presented rules correspond to the structural properties of the system. However, the system is implemented in order to promote extensibility of the rules.

For this project 4 basic rules were chosen. First rule is whether all elements that are visible on reference web page are also visible on the target web page. Second rule checks if all images are shrink proportionally. The third rule reviews if there is a horizontal scroll bar. Last forth rule checks if the image fits it parent container.

3.2.3 Report

The report shows the result obtained by applying the selected rules in the program. For each selected rule, the report shows if particular rule is passed or not and provides further detail. The figure (Fig. 6) shows the example of report for a web page.

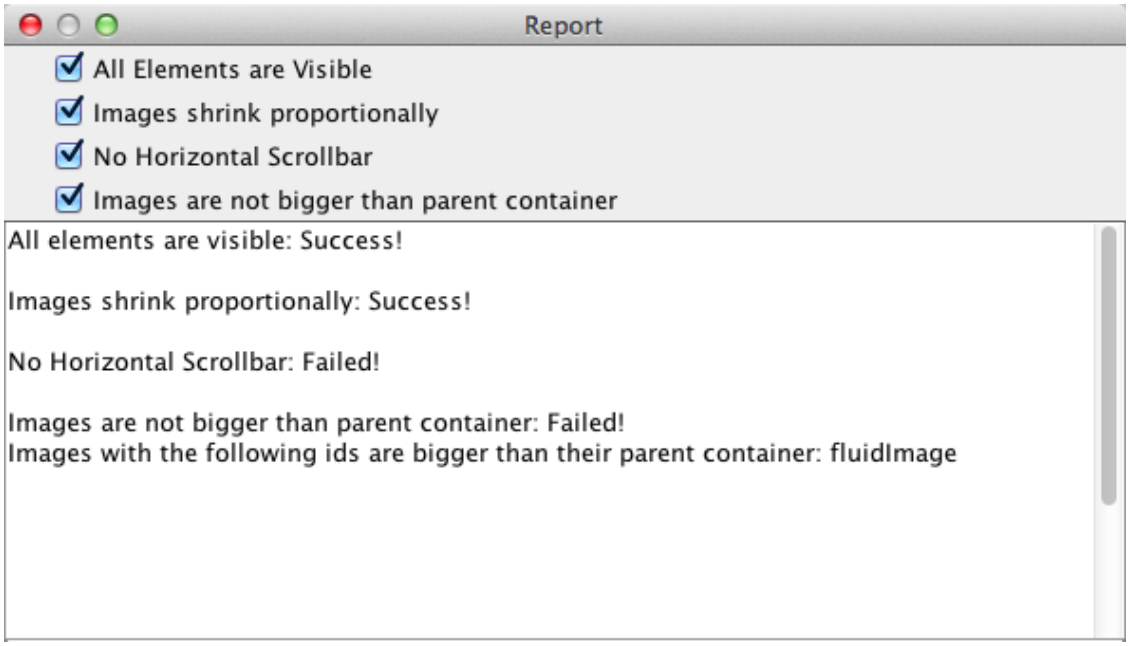


Figure 11. Report page

3.3 Conclusion

A well-defined set of rules for consistency were presented in this chapter. This chapter focused on defining an approach for automated test case. Once these rules are successfully implemented, the approach could be evaluated. The implementation of the program is presented in the next chapter.

4. Implementation

4.1 Introduction

This section discusses how the approach is developed is built and core technologies are described in detail. The implementation of the program performed to evaluate the technique. It also touches upon additional tools and libraries that were used for the project. Finally, the following sections give an in-depth explanation of program development and design and also show its interface. The right choice of tools ensures a reliable and robust application.

4.2 Development Strategy

Because the development phase was constrained with the time constrains, several principles from Extreme Programming were adopted for software development. Extreme programming leads a software project in five ways: communication, simplicity, feedback, respect and courage [19]. Two of these principles: principle of simplicity and communication was used for the development of the dissertation project.

The principle of communication announces that programmers should constantly communicate with their customers. During the work on the dissertation project, every week I either send the progress report to the supervisor we there was a meeting with my supervisor. This guarantees that my supervisor knew about the progress of the work and all decisions were made after discussion.

The principle of simplicity declares programmers should archive goal with small and simple steps while keeping the design clean and simple. This principle was adopted and the design remains clean and plain.

Furthermore, manual functional testing was chosen over unit testing due to the specifics of the project and to the short time frame.

4.3 Implementation Resources

The program was developed using the following versions of software:

Operating System: Mac OS X Lion 10.7.5 (11G63b)
Development framework: Eclipse Classic 3.7.2
Graphical Library: JGraph Library version 2.5.1.1
Selenium Client & Web Driver: Client Version 2.41.0

4.4 Tools and Technologies

4.4.1 Eclipse

Eclipse was chosen as Integrated Development Environment (IDE) to implement Java. (http://www.eclipse.org) Eclipse is platform-independent open-source software framework which is written in Java [33]. Furthermore, Eclipse has a community of users that extend the capabilities of this IDE. Moreover, there is a support for different operating systems such as Linux, Mac OS and Windows [33]. Eclipse is widely used in industry as with this reliable and scalable platform applications can be quickly and easily designed, developed and deployed.

4.4.2 JGraph Library

JGraphX Library was chosen for displaying graphs of visible DOM elements. JGraphX is a graph visualization library that is licensed under the BSD license [34].

This library provides capabilities for visualization and interaction with node-edge graphs. JGraphX provides functionality for visualization and interaction with node-edge graphs.

The benefits for using JGraph comparing to other libraries are great documentation, many manuals and examples with intuitive commands.

4.4.3 Selenium

Selenium is a set of different open source software tools. Each tool has its own application to support agile process automated testing and web application test automation and [35] Selenium Core is a fundamental part of Selenium that is written in JavaScript. Figure shows the relationship between different Selenium tools. The relationship among different Selenium tools can be depicted as shown in following Figure 1.

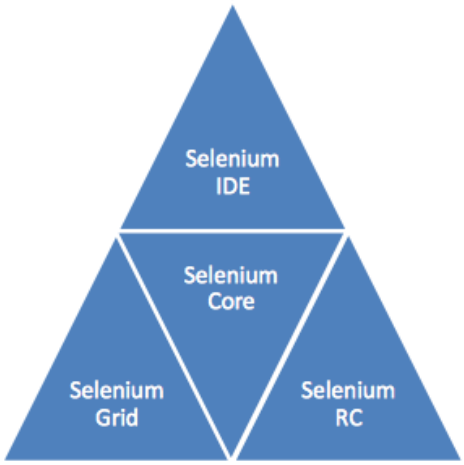


Figure 12. Relationship between different Selenium tools

Framework Selenium [36] is widely used in test automation. It allows describing the test cases, record and playing user actions through the browser extensions and interacting with the browser from the code using different programming languages such as Java, C #and other. Selenium Webdriver [37] is a layer between the application and the browser. Interaction occurs through specific modules for each browser drivers. There are different realizations of Selenium Webdriver written in different programming languages. Currently supported libraries are written in Java, C #, Python and Ruby, there are not official implementations in Perl and PHP.

4.4.4 Version Control - Git

Git is a tool for version control which is created by Linus Torvalds. Git is a distributed revision control and source code management system recommended by Google. The reason behind choosing git was the school suggestion and also, git has a code repository website for hosting git projects. For Eclipse IDE there is a Git plug-in called “EGit” that can be installed from this URL: http://download.eclipse.org/egit/updates.

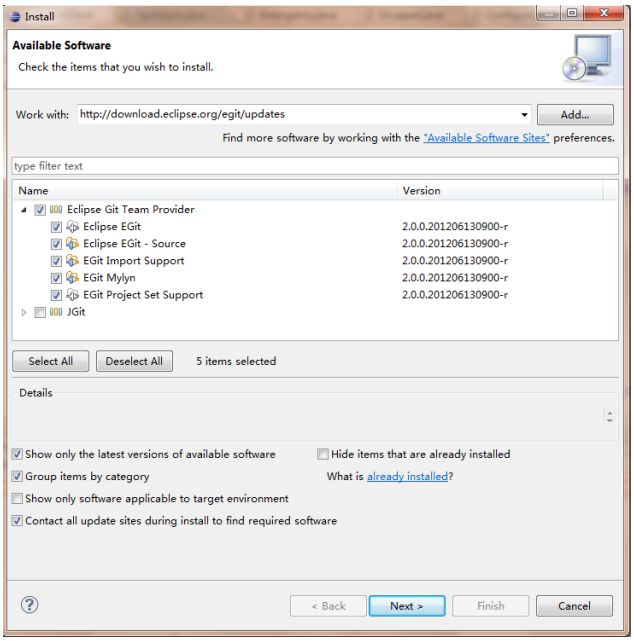


Figure 13. Install Git

4.5 Application Design

This section shows the design and implementation solution for the program however it only describes the latest version of the program as during the work on the project the changes were made in the implementation and the design had to be renewed to meet the new requirements.

While working on design of the system's architecture, the following design decisions were prioritized:

- Encapsulation and Abstraction. Application has been structured strictly according to Object Oriented Programming (OOP) principles. Wrapping underlying logic together with wrapping information is a good practice in OOP as these helps to store the state of objects locally and it is easy to develop implementation further without affecting the objects that are using it.

- Inheritance and Interfaces. I have used inheritance using various classes and interfaces where need. This will make the code structure less repetitive and easier to manage.

- Modularity. Application design is made modular wherever possible so that it becomes easier to change the application according to requirement.

- Flexibility. Flexibility is a driver for the program. It is important to accommodate preferences of different user's.

- Extensibility. It is easy to enhance the application in future: add additional rules and extend the functionality.

- Re-usability. Certain parts of the program are used more than once, and therefore, in a good program design it should be possible to access existing functionalities from different classes of the program.

4.6 Application Development

The full structure of the project is presented on Figure 14.

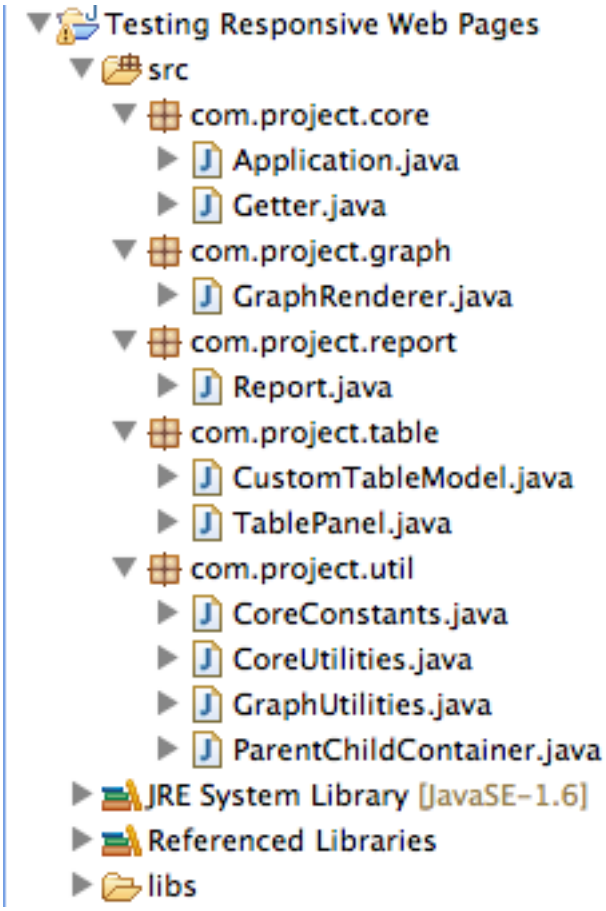


Figure 14. Project structure

com.project.core package
Application.java - the starter class
Getter.java - A fundamental class which invokes browser, creates graph, and table for the provided URL.

com.project.graph package - drawing graphs for DOM model
GraphRender.java - A dedicated Graph Rendering class which renders graph inside a panel.

com.project.report
Report.java – Class which displays report of various rules performed on the extracted webpage elements.

com.project.table package
CustomTableModel.java - An implementation of table model used in displaying statistics table.
TablePanel.java - A Table panel which displays various statistics.

com.project.util package
CoreConstants.java - Various constants used throughout the code.
CoreUtilities.java - This class contains mathematical operations needed to perform on extracted elements from webpage.
GraphUtilities.java - graph utilities including tags, width and height
ParentChildContainer.java – Class for holding image element and its immediate parent element data.

Figure 15. High level explanation of classes and packages

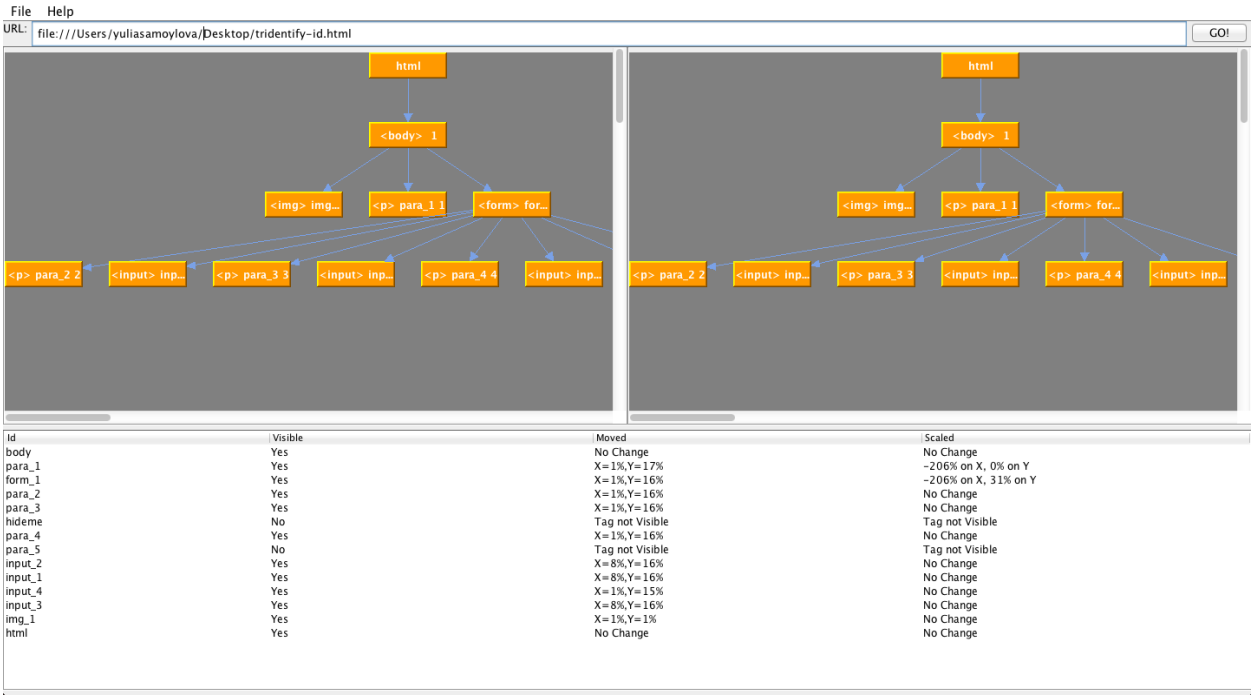
4.7 User Interface Design

After gathering the requirements, the next step was to design the basic user interface of the system.

The graphical design for the program should be simple. The main priority was on implementing the functional interface without spending much effort on designing the graphical design. The improvements of the graphical design could be considered in future versions of the program.

The objective is to design user interface that is easy to understand and to use. Moreover implementing a plain interface will help to focus on developing the functionality without spending much time on building the interface.

• Main page



Id	Visible	Moved	Scaled
body	Yes	No Change	No Change
para_1	Yes	X=1%,Y=17%	-206% on X, 0% on Y
form_1	Yes	X=1%,Y=16%	-206% on X, 31% on Y
para_2	Yes	X=1%,Y=16%	No Change
para_3	Yes	X=1%,Y=16%	No Change
hideme	No	Tag not Visible	Tag not Visible
para_4	Yes	X=1%,Y=16%	No Change
para_5	No	Tag not Visible	Tag not Visible
input_2	Yes	X=8%,Y=16%	No Change
input_1	Yes	X=8%,Y=16%	No Change
input_4	Yes	X=1%,Y=15%	No Change
input_3	Yes	X=8%,Y=16%	No Change
img_1	Yes	X=1%,Y=1%	No Change
html	Yes	No Change	No Change

Figure 16. Main page of the program

The main page is divided into 3 sections. These sections are: field for putting the address of the web page, DOM graphs of the visible elements for reference and target web pages and the matrix of changes.

The top section consists of the field where user can type the address on the computer of the web page.

The middle part of the screen shows two DOM graphs of the visible elements on the screen of particular size and the bottom has a matrix of changes.

The top part consists of two sections. The right section provides the graph for DOM model of the reference page and the left section shows the the DOM model of the target page.

• Matrix of changes

Id	Invisible	Moved	Scaled
body	no	No Change	No Change
para_1	no	X=1%,Y=17%	-206% on X, 0% on Y
form_1	no	X=1%,Y=16%	-206% on X, 31% on Y
para_2	no	X=1%,Y=16%	No Change
para_3	no	X=1%,Y=16%	No Change
hideme	yes	Tag not Visible	Tag not Visible
para_4	no	X=1%,Y=16%	No Change
para_5	yes	Tag not Visible	Tag not Visible
input_2	no	X=8%,Y=16%	No Change
input_1	no	X=8%,Y=16%	No Change
input_4	no	X=1%,Y=15%	No Change
input_3	no	X=8%,Y=16%	No Change
img_1	no	X=1%,Y=1%	-206% on X, -165% on Y
html	no	No Change	No Change

Figure 17. Matrix of changes

On Figure 17 there is a section from the program that shows the summary of the performed changes.

The table consists of 4 columns: “id” for the id of the element, “invisible” to determine if the tag is visible on the target page, “moved” and “scaled” columns shows the modifications for the elements.

• Determine Rules & Report

The screen for choosing rules and producing the report is presented on Figure 18. The user can be chosen the rules from the provided options. The user is also able to select a preferred set of rules to execute.

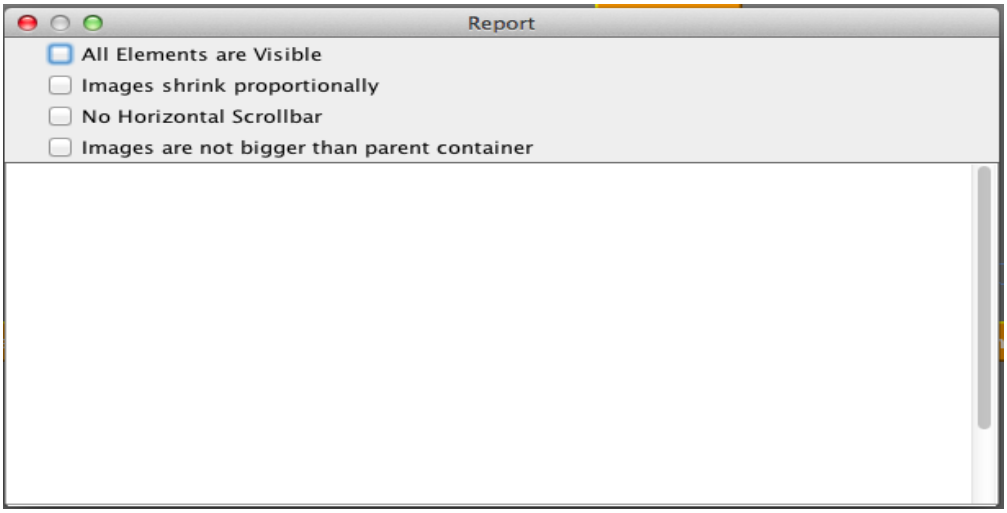


Figure 18. Selection of Rules & Report

4.8 Data Entry

For the purposes of developing the prototype, I have manually entered the tested webpages into the program. In a real life scenario, an appropriate functionality should be added to allow user to test any number of web pages.

For this project, from analysis we determined a set of rules identifying the consistent web pages. An essential characteristic of program tool would be to provide extra functionality to user to describe their own rules.

4.9 Dom Tree visualization

The DOM tree for reference and target web page is implemented with JGraph Library. The library provides a simple way for filling and showing the tree.

JGraph Library was chosen because it is a powerful library that can be easily used in Java project in Eclipse. This library provides the way to interact with nodes such as moving them on the screen and grouping them in a way the user needs.

The library provides three ways of displaying unclosing tree tags. The design with simple arrow was chosen.

4.10 Problems Encountered

No major problems were encountered during the implementation. A couple of minor problems were because it took some time so start using the JGraph tool for the visualizing the DOM however the issues were solved relatively quickly and did not affect the project much.

4.11 Conclusion

In this chapter, the system design was presented, starting from an abstract view of the system and overview of the used tools to a more detailed explanation of the most important parts of program, such as the organizing the project and user interface design.

In this chapter the implementation details involved in the project were highlights. Due to space limitation, only the most high level details were discussed here.

5. Evaluation

5.1 Introduction

This section presents the evaluation of the developed approach against the four selected rules which map on the written test cases. For each of these rules there are two test cases for the pass and failure scenario. Also this chapter presents a critical analysis of the developed approach. Furthermore, this section describes a comprehensive set of test cases that were made to ensure the developed approach works as it was specified in earlier chapters.

The time required for testing the program for one web page is about half a minute. The required time to view report is the same - about half a minute, but this time, of course, it depends on the web page and the number of detected changes. This method of testing is mostly suited for continuous testing of the web page or for making refactoring. The program does not require a large computational power and lots of memory, as a result it can be easily can be run on a developers local machine. Moreover, if the developer runs the program often enough, one look at the report can ensure that the latest changes do not cause new errors.

To carry out testing 2 web pages were chosen. The program was run with each of the testing web pages. The first test page that is displayed on the desktop screen is shown on the Figure 19. The same web page that is displayed on a mobile screen is shown on the Figure 20. When this page is redrawn on the smaller screen the image shrinks proportionally and the text disappears.

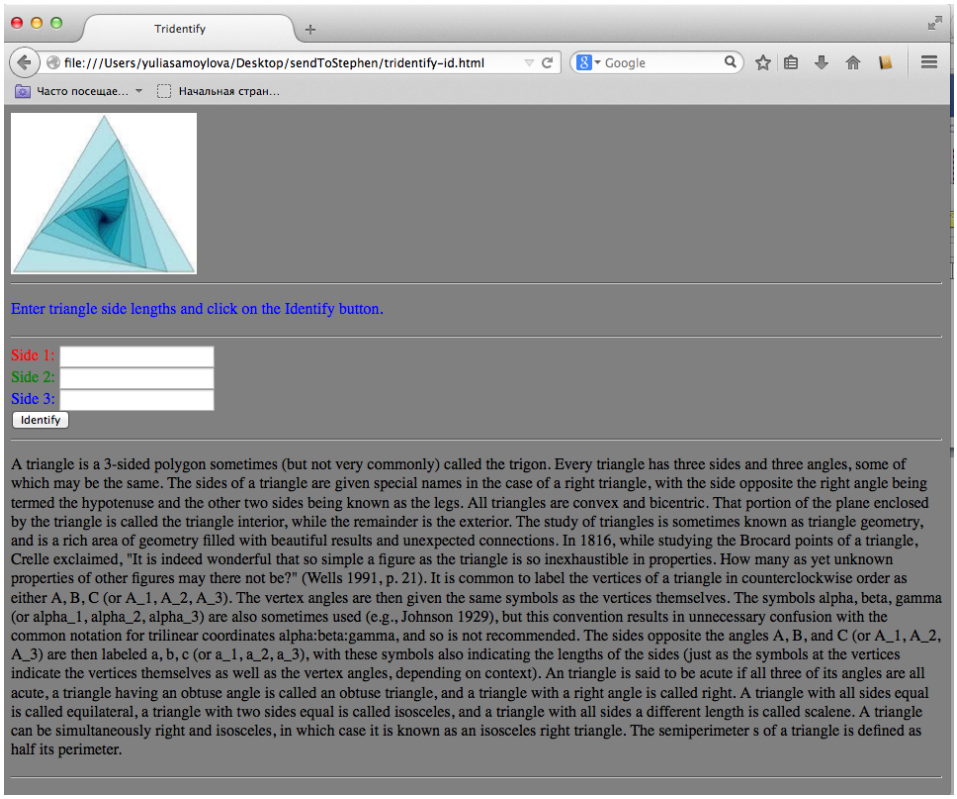


Figure 19. First testing page displayed on desktop size screen.

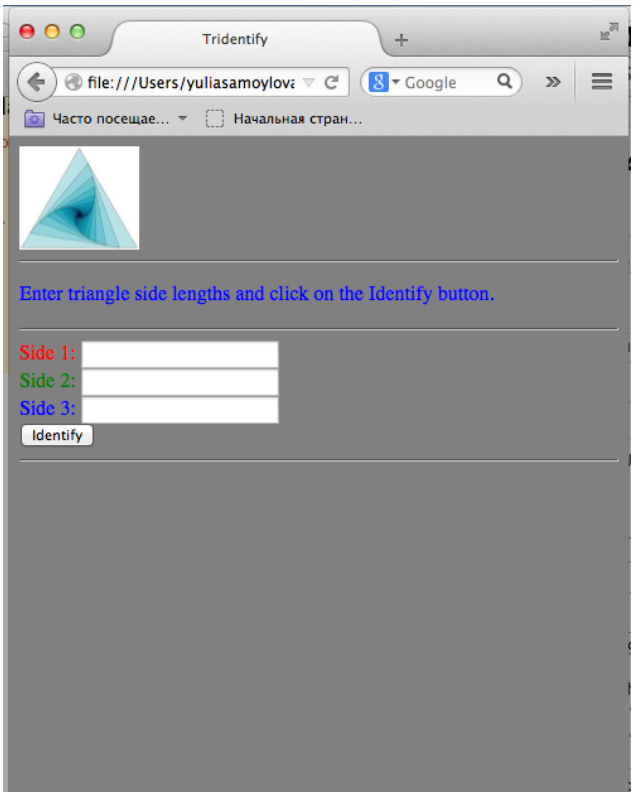


Figure 20. Second testing page displayed on mobile size screen.

The second web page for testing is displayed on the desktop size screen is shown on the Figure 21. The same web page that is displayed on a mobile screen is shown on the Figure 22. When this page is redrawn on the smaller screen the image becomes bigger than parent container. Also for the web page on smaller screen there is a scroll horizontal bar that is not visible on the screen shot.

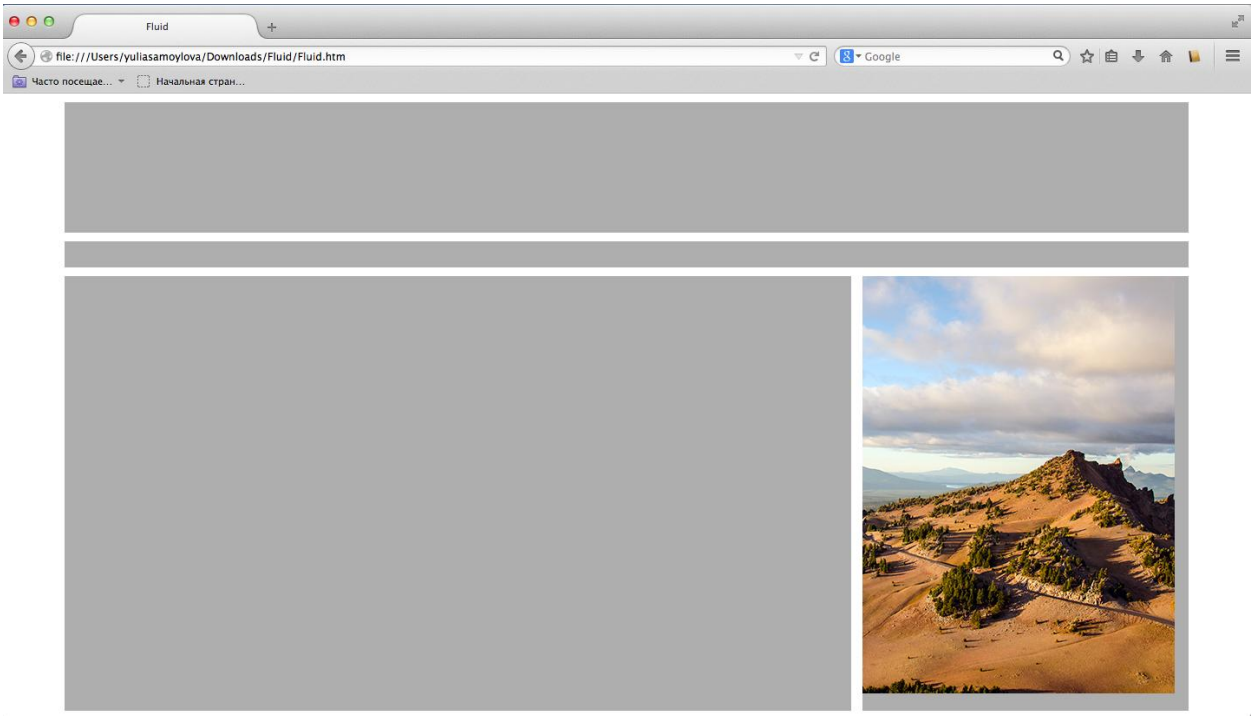


Figure 21. Second testing page displayed on desktop size screen.

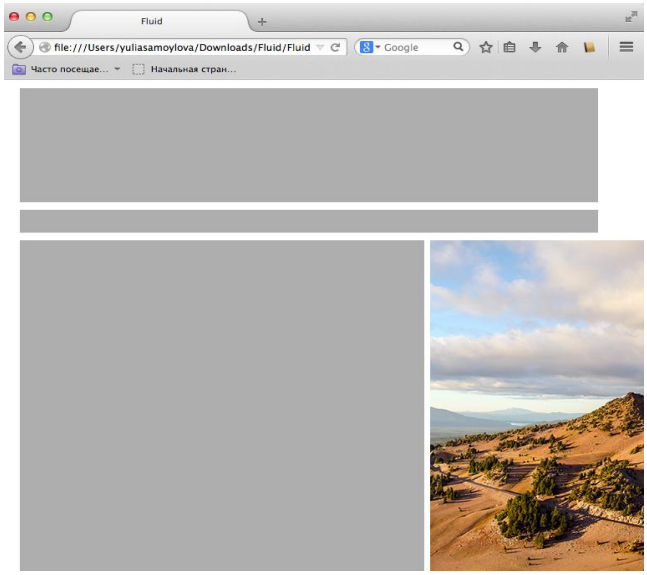


Figure 22. Second testing page displayed on mobile size screen.

5.2 Test Cases

The coming list of test cases outlines the most important testing and conclusion that were carried out while authenticate that the system satisfies the requirements.

In test cases different behaviors will be showed and all rules will be evaluated. Each test case that is shown outlines the features that this test is verifying, as well as describes inputs, the expected output, the actual output, and if the test has failed or passed.

5.2.1 Visibility

The first significant requirement is to verify whether the first rule specifying if all elements are visible on the mobile screen. First, we will test with the web page that has all elements that are on desktop version are visible on mobile version as well. Then, for the second test case, we will modify web page so that in mobile version of the web page some elements are hidden and run the program once again to compare the outcomes.

Test No.	**1**
Features Tested	Checking whether all the elements are visible.
Input	The web page that has some hidden elements when it is displayed on a mobile device.
Expected Output	The program identifies that all elements on desktop version are visible on mobile version. The success report is returned.
Actual Output	Same as expected.
Pass/Fail	Pass

Table 3. Test Case 1 - Visibility check

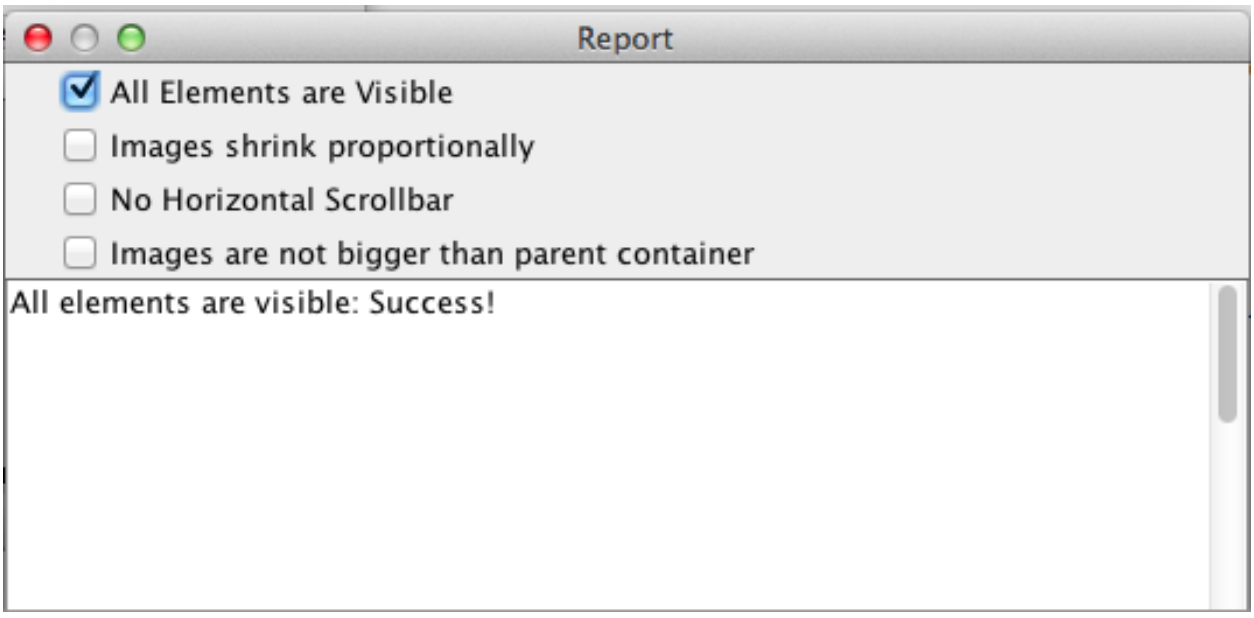


Figure 23. Select a “All Elements are Visible” rule and report

Test No.	**2**
Features Tested	Checking whether some elements on the web page are invisible.
Input	The web page that have the same set of visible elements on the desktop and mobile device.
Expected Output	The program identifies which elements are visible on desktop version and not visible on mobile version. The fail report is returned.
Actual Output	Same as expected.
Pass/Fail	Pass

Table 4. Test Case 2 - Visibility check

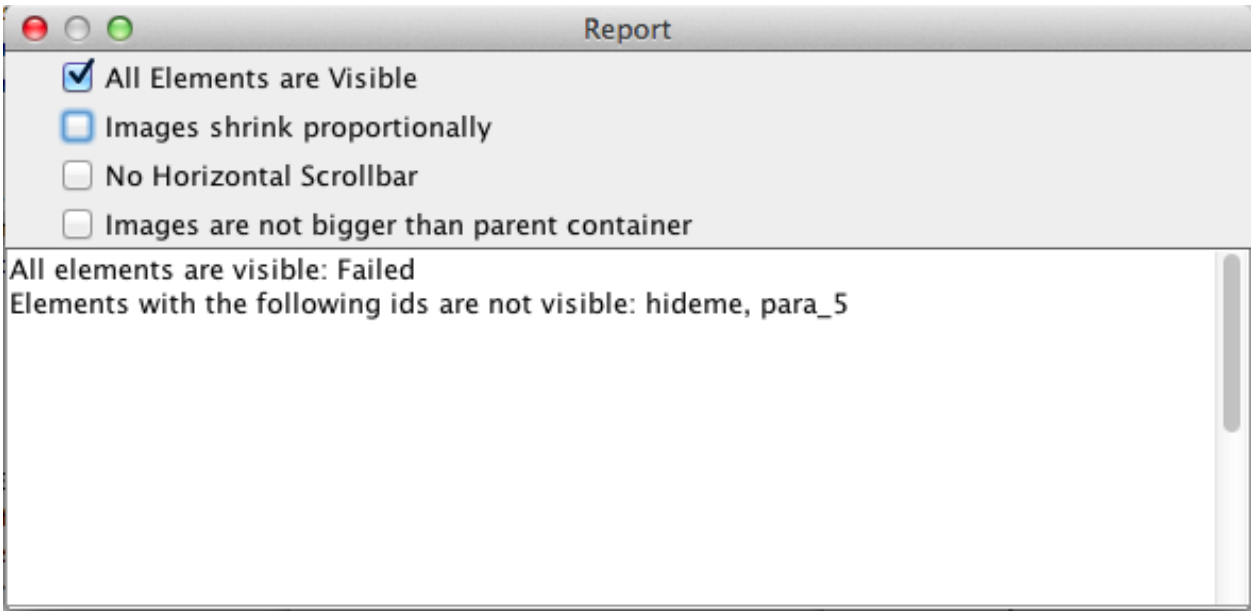


Figure 24. Select a “All Elements are Visible” rule and report

Test Case 1 and Test Case 2 show that the first rule is working correctly and the implemented program is suitable to automatically test the visibility of elements.

5.2.2 Images shrink proportionally

Another significant requirement is to verify whether second rule of the system works and images shrink proportionally.

First, we will test when on the web page all images are shrinking proportionally. Then, in another web page we will add some violations, run the application once again and compare the outcomes.

Test No.	**3**
Features Tested	Checking whether all images on the web page shrink proportionally.
Input	The web page that have images that shrink proportionally on mobile device comparing to desktop.
Expected Output	The program identifies that images shrink proportionally. The success report is returned.
Actual Output	Same as expected.
Pass/Fail	Pass

Table 5. Test Case 3 - Proper shrinking of images check

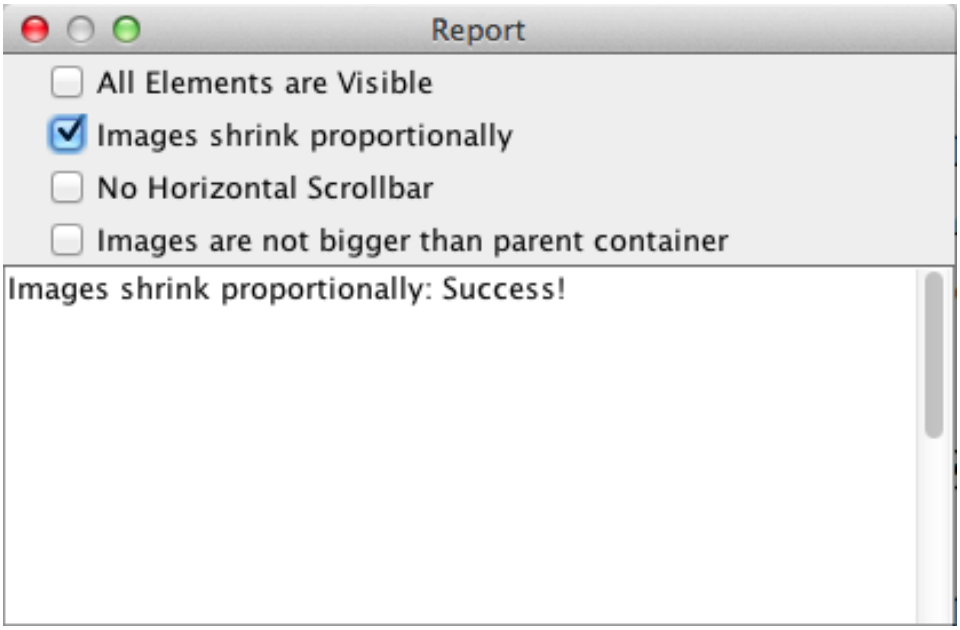


Figure 25. Select an “Images shrink proportionally” rule and report

Test No.	**4**
Features Tested	Checking whether some images on the web page do not shrink proportionally.
Input	The web page that have images that do not shrink proportionally on mobile device comparing to desktop.
Expected Output	The program identifies that images shrink proportionally. The fail report is returned.
Actual Output	Same as expected.
Pass/Fail	Pass

Table 6. Test Case 4 - Proper shrinking of images check

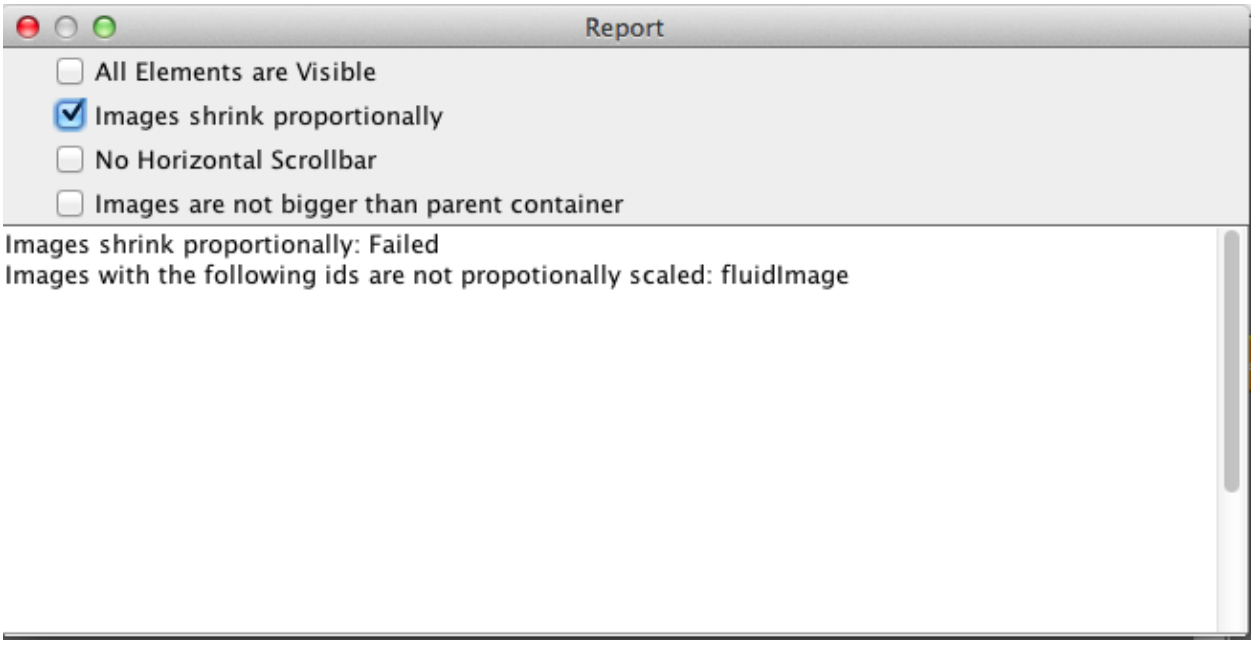


Figure 25-2. Select a “Images shrink proportionally” rule and report

By checking whether the system can identify that there are no violations, means that the program has a correct identification of the shrinking of images on the web page.

5.2.3 Horizontal Scrollbar

In order to verify if the program identifies correctly if the mobile version of the web page has a scrollbar two test cases were performed.

Test No.	**5**
Features Tested	Checking whether web page has a scroll bar on mobile device.
Input	The web page that does not have a scroll bar on mobile device.
Expected Output	The program identifies that there is no scroll bar on the mobile device. The success report is returned.
Actual Output	Same as expected.
Pass/Fail	Pass

Table 7. Test Case 5 - Horizontal Scrollbar check

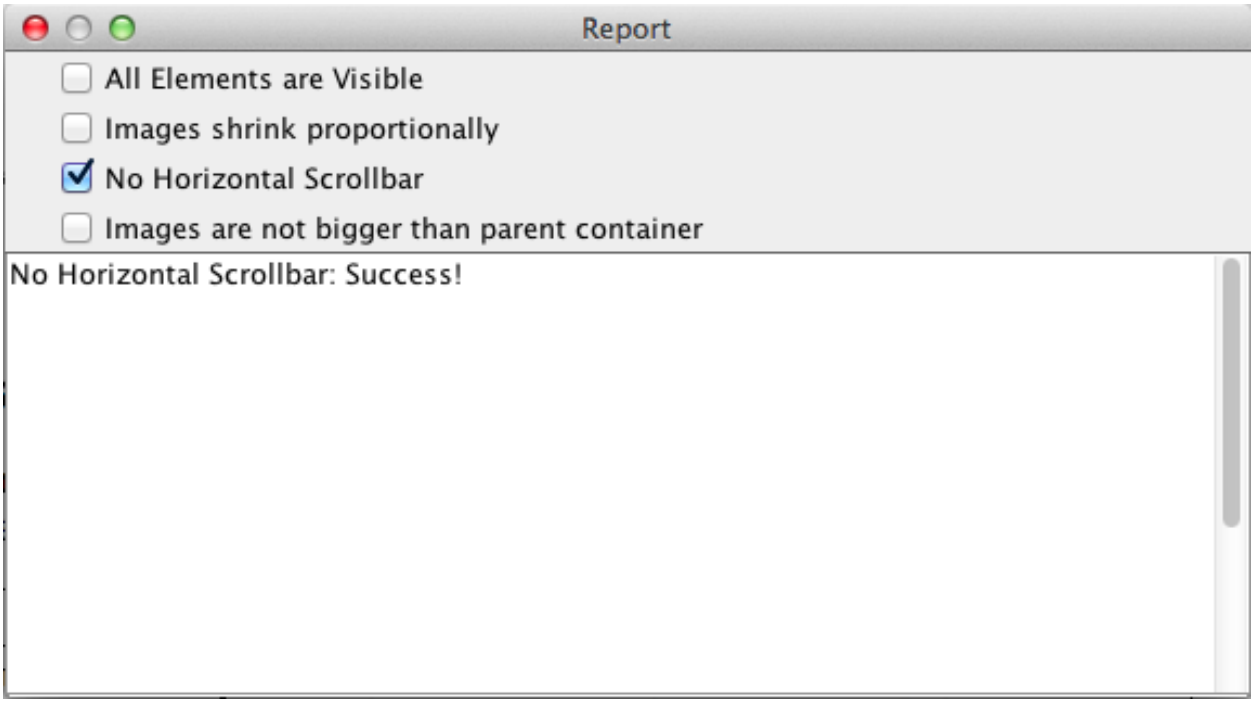


Figure 26. Select a “No Horizontal bar” rule and report

From the table shown above, it can be inferred that the program correctly identified the existence of the scroll bar on the mobile version of web page.

The next scenario will present a case when a web page does not have a scroll bar on mobile device.

Test No.	**6**
Features Tested	Checking whether web page has a scroll bar on mobile device.
Input	The web page that does have a scroll bar on mobile device.
Expected Output	The program identifies that there is a scroll bar on the mobile device. The fail report is returned.
Actual Output	Same as expected.
Pass/Fail	Pass

Table 8. Test Case 6 - Horizontal Scrollbar check

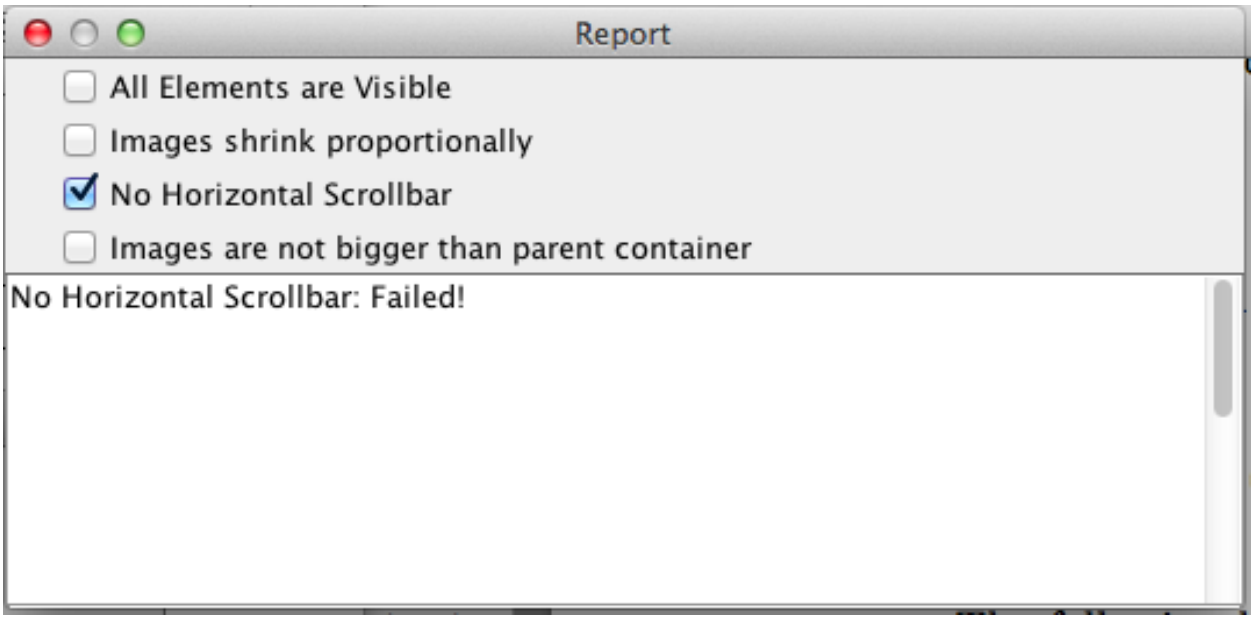


Figure 27. Select a “No Horizontal Scrollbar” rule and report

5.2.4 Images inside the parent container

The next set of test cases involves checking whether the image fit its container.

Test No.	**7**
Features Tested	Checking whether all images on the web page fit the parent container on mobile device.
Input	The web page images are fit the parent container.
Expected Output	The program identifies that all images fit parent container. The success report is returned.
Actual Output	Same as expected.
Pass/Fail	Pass

Table 9. Test Case 7 - “Images inside the parent container” check

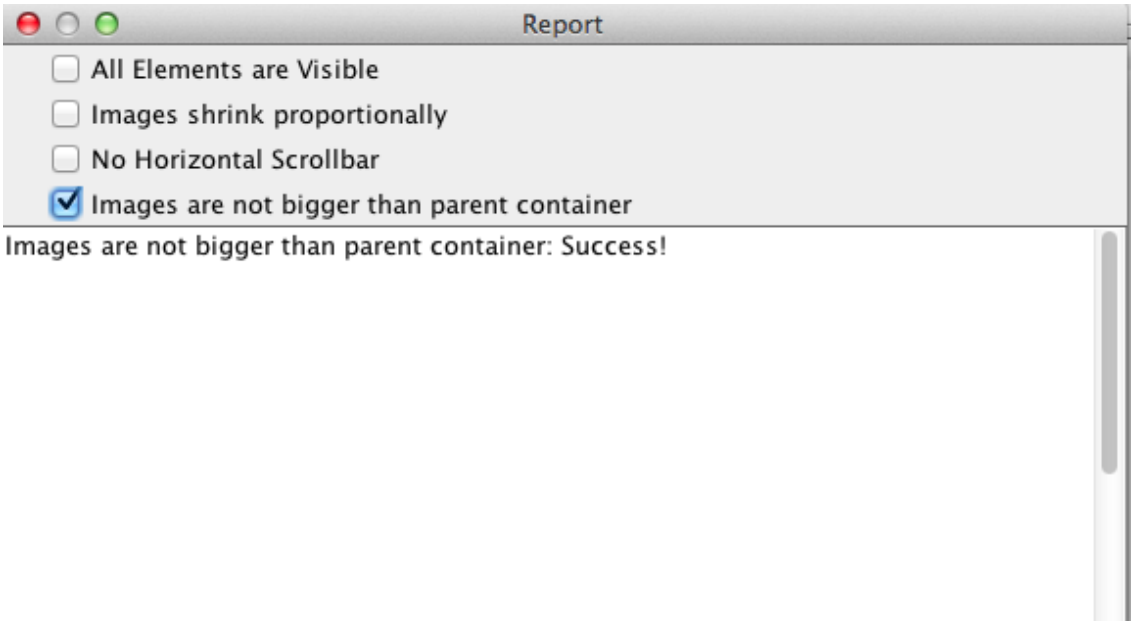


Figure 28. Select a “Images are not bigger than parent container” rule and report

In the example shown above, we can see a case when the image fits its container and in the example below the image does not fit its container on the mobile version.

Test No.	**8**
Features Tested	Checking whether some images on the web page do not fit the parent container on mobile device.
Input	The web page images are fit the parent container.
Expected Output	The program identifies that at least one image does not fit a parent container. The fail report is returned.
Actual Output	Same as expected.
Pass/Fail	Pass

Table 10. Test Case 8 - “Images inside the parent container” check

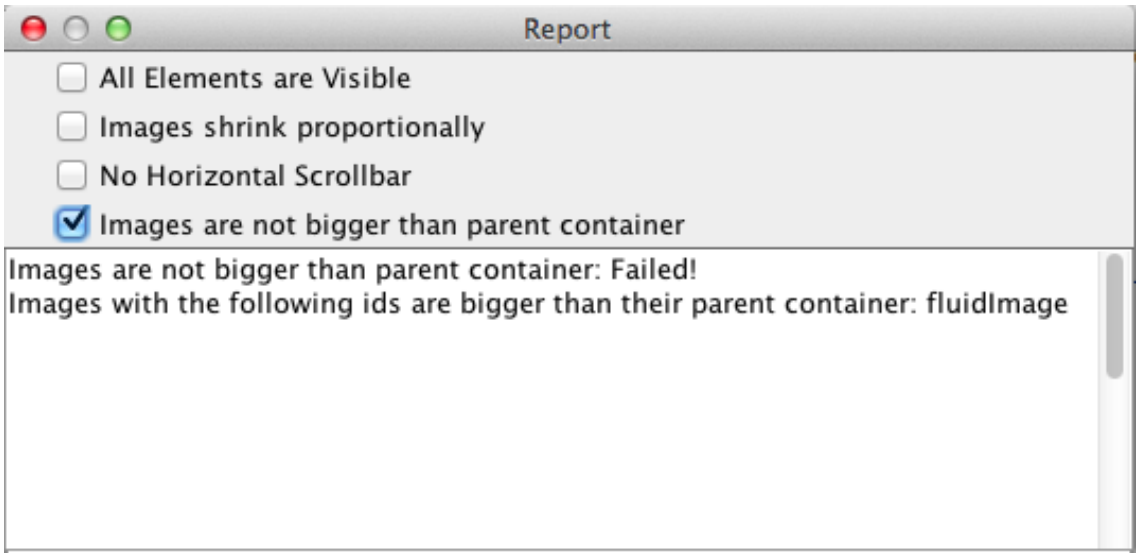


Figure 29. Select a “Images are not bigger than parent container” rule and report

5.2.5 Rules Selection

The user is also able to choose a set of rules to be executed.

Figure 30. Selection of Rules

Thus, if the user wishes to validate for a specific rule, this can be done by checking the checkbox in the Preferences section (as shown in Figure N). This customization feature allows better flexibility for the users. Test case 9 will have a full set of rules and Test case 10 will have a smaller set of rules.

Test No.	**9**
Features Tested	Checking if all the rules from the set have been executed.
Input	The web page that passes all rules except of first: some elements become invisible on a mobile version.
Expected Output	The violations of the rules that exists in the mobile version of the web page were detected.
Actual Output	Same as expected.
Pass/Fail	Pass

Table 11. Test Case 9 - All Rules Selected

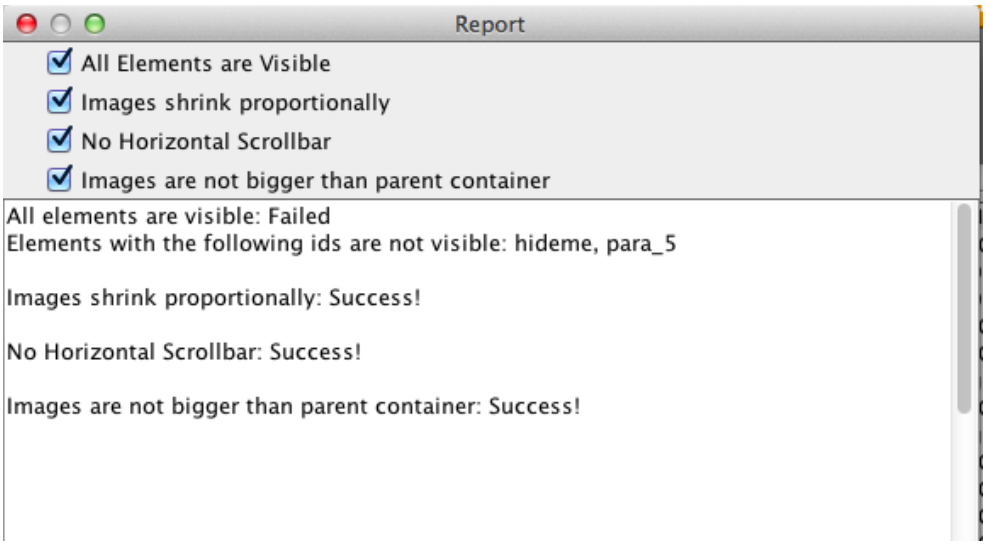


Figure 31. All rules are selected and report

From the table shown above, it can be assumed that all the rules have been executed as expected. Thus, this test case has successfully passed.

The next scenario presents a case where the user only chose several rules to be executed while other rules are omitted.

Test No.	**10**
Features Tested	Checking if a program executes only the rules that were selected.
Input	The same web page as for Test 9 that that passes all rules except of first: some elements become invisible on a mobile version.
Expected Output	Some violations that the rules are not selected are not showed to the user.
Actual Output	Same as expected.
Pass/Fail	Pass

Table 12. Test Case 10 - Some Rules Omitted

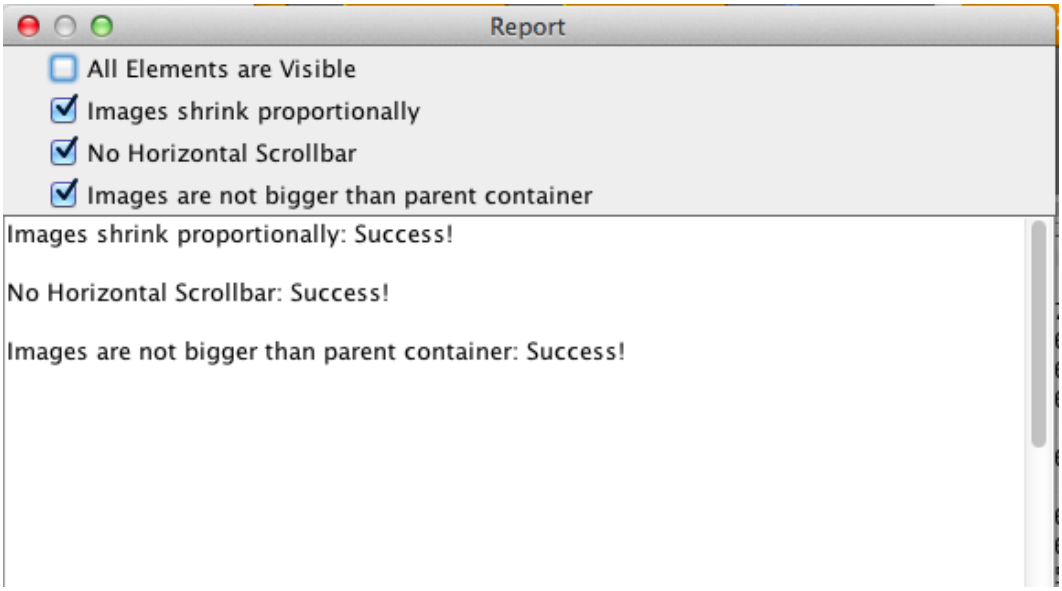


Figure 32. Set of rules is selected and report

After making a comparison between the results from the runs in Table N and N, it can be seen that only the checked rules were executed. As a result and as it is expected these errors are not detected.

5.3 Functional Testing

The aim behind building any software is to fulfill some business needs and requirements. Functional testing guarantees that the requirements are met as a group and individually.

To ensure that the system is implemented and is working as expected the testing was performed as an iterative process at each stage of the development. When a part of the system structure or functionality was developed it was checked on the developer's laptop to ensure that it was working without errors.

When most of the implementation was almost done, I went through the whole functionality of the application to make sure that the tool is working correctly.

5.4 Non-functional requirements

All non-functional requirements were fulfilled: the project was submitted on time, the development of the program was done using the stated tools and the performance was checked tested and evaluated.

The non-functional requirements are the following:

2. All the work is performed by student (Yulia Samoylova, student No: 12250250)

3. Project has to be submitted before 14th of June 2014.

4. The project is implemented using declared technology.

5. The graphical design of the program remains simple.

5.5 Correctness and Usability

By correctness, we assume that for each selected rule if there is successful result that means that the chosen web page fulfills the rule and there are no actual violations on the web page. Also if there is a failure in return rule has been detected as a violation, then it means that there was actually a violation.

As far as possible, we tried to achieve completeness, but since the program is used for finding bugs and errors, it is difficult to prove such claim. Furthermore, this project is a proof of concept, and thus, the objective is not to achieve perfectionism, but to show that chosen approach works.

Usability attribute was verified throughout the testing of the program. It is important to develop user-friendly system. Thus while designing and implementing the program, this goal was prioritized over every added feature.

5.6 Comparison with the project «Galen Framework»

In the chapter "Related Work and Technical Background" was mentioned project Galen Framework, the purpose of which is also automated testing of the web page content, albeit using a different approach with a different set of rules.
Galen Framework has a finite set of screen sizes that are available for the tester. Galen allows testers to specify rules to a particular screen size. Therefore Galen Framework does not allow comparing web pages on different screens and setting rules for all screen sizes.

5.7 Comparison with Responsive Design Simulation and Responsive Design Simulation

In the chapter "Related Work and Technical Background" was mentioned a few tools for testing responsive web design including Responsive Design Checker and Responsive Design Simulation.

The important limitations of the tools are that they do not provide the functionality for automated testing of the world web pages. User can see the displayed web page on limited amount of supported devices but the testing should be performed manually. The implemented tool shows a novel approach to test responsive web pages for the consistency of their appearance. In our program, the developed rules are implemented and our approach generates test cases automatically.

5.8 Limitations

The limitation of the implementation is connected with the performed choice of software tools including the use of Selenium. Selenium does not support some versions of the browsers and some aspects that the user can see were difficult to find with Selenium, for example, the scroll bar.

The performed evaluation has its own limitations. The approach was evaluated using 4 high level rules. The other rules and parameterizing of existing rules were not developed and implemented because they are of secondary importance and also, these new rules are essentially the same with the rules that were evaluated although have different details. The aim of the evaluation was not to automatically generate test cases for all possible rules but to show that the technique works by picking out just enough of representative rules without creating rules for the entire set of the domain model.

Next, the limitations of the performed approach are identified. This limitation of the technique could prevent to automate generation of test cases. Currently our approach is based heavily on the fact that every tag has its unique id, so to perform the rules there is a mapping between the objects with the same unique identifiers.

Another limitation of the developed approach is that it is assumed that the DOM structure is essentially the same. Although in some cases the structure of the DOM model can be changed.

Moreover, there is no clearly defined default behavior on how the web page should be displayed on the smaller screen compare to a master copy of the

same web page, for example, for the for the picture we can assume that it should be scaled down, but what is the default behavior for the peace of text. To define the text behavior there could be several options including text matching: if the text is the subset of another text or checking if the font size changed or not. For different tags and their properties it is not clear what should be the default consistency rules.

In terms of record and play back testing, current technique is record and play back with no record. The recording is automated because we are not testing the functionality, we are testing the appearance. The approach only requires the user to fire the web pages as we are currently working on assumption that you can go on particular web page. However in real application the tester might not able to do this. The tester might need to go through an application to get a particular web page. It might be a case that for somebody to use this tool, there will be a need in doing a recording session with Selenium: going through all web pages in order to get to the next one to get a master copy of each web page.

The proposed solutions for the described limitation are presented in *Future Work* section.

5.9 Conclusion

This chapter presents the several test cases covering the main functionality of the program. Each test case shows different aspect of the system and outlines if the requirement is satisfied. Also this section presents the comparison of implemented program with already existing programming solutions.

6. Conclusion

6.1 Completed Work

The project was based on the problem that was proposed by SAP and formed into a basic idea. Later on, after making the extensive research, it was concluded that the given problem probably does not have any software solution for automated testing solution yet, but there is a huge need for such program for the companies presented in a web. After conduction the research, the project objectives and specification were formed.

This thesis has also explored three other tools for testing responsive web design and outlines the advantages and disadvantages of them.

This thesis also presents a demonstration of the implemented program on automated testing of responsive web pages, which determined that while it includes some interesting functionality and easy use that is accessible with existing tools, the program requires some upgrade before it will become an efficient tool.

6.2 Usage

Regardless of the use of implemented program, it is worth mentioning two cases. The first use could be for the developers who can configure the program to the url of the application, which is under development at the moment, to automatically test the application on different devices of the screen after the changes in a code or before launching each new version.

The second use of the application is to use it with any available crawler to check linked pages. On each webpage we can run the program to test it. Crawler linked pages - and run checks on the layout of each page reached. If the robot will be on each web page for a couple of minutes than even for a big webpage, all process will be quite fast and without spending any human resources.

6.3 Future Work

The designed and implemented program achieves objectives that were set in the beginning of the paper. However certain aspects in the dissertation project could be improved during further work. Future work will focus on overcoming the limitations of the developed approach and expanding the current approach.

Overcoming identified limitations

The current approach is heavily based on the assumption that every tag has a unique identifier. Future work may overcome this limitation by automatically generation of the tags. Furthermore, the tags should be generated and put to the master copy only once and they will be automatically in the web page when it is accessed from different devices. So the tags do not need to be replicated across all different web pages and we can do it only once for a master web page because the web page has the same source code.

Another limitation that was mentioned in previous chapter is that current model is based on the fact that the structure of the web page is essentially the same. However, we have ids for each tag and this will allow us to identify the changes in the structure and map the elements correctly. Also for this approach, it is not important where the element is the model but it is important where it is on the screen.

Another dimension of further work on automated testing of responsive web pages could be providing extra functionality for the user to enter and encode rules and requirements into an executable structure. This is a challenging problem and could be a project itself. The program should be able to provide the users with proper outcome and explanation as on how the particular rule was applied. This extension has a particular importance as for some elements there are no clearly defined default behavior and their behavior might vary depending on the context. By providing a scripting support that allows user to write in the language of their choice the consistency rules, testers can make consistency rules for entire web page and run the tool against the entire web page. This approach significantly reduces a work required for automated test case generation. The amount of manual work is very limited and the tester does not need to look at every web page as long as there are consistency rules.

We are working with assumption that if the web page is displayed with the particular screen size it will be displayed the same on the device with these screen size. To validate this assumption in future work, we need to run implemented program with a Selenium or equivalent tool on a different devices just to confirm how the web page is displayed.

Adding new functionality

• Touch Screen

Due to the time constrain, we were not look at touch screens elements. Touch screens do not work as well as when you have a keyboard. Sometimes when it works correctly on the desktop, on mobile phone there is something wrong, we do not know if the problem was within the browser or in the web page.

• Application Design

Due to the time constrains and the subject of the dissertation, it was chosen not to have the prime focus on constructing complicated design solutions. This is the reason why the minimalistic design was adopted. However, for commercial release, there should be done improvement of the design of the program. After improving of the visual appearance, new design should not cover the functionality.

• Run-time tool

Implementing a run-time tool that would perform testing of displayed of responsive web pages might improve development. Assuming that the end-users of the program will be software developers, these features are crucial for the application that is used by multiple users on different devices.

6.5 Personal Experience Carrying Out Project

The experience that I gained working on the project has been priceless for me because I was exposed to the sphere of Dependable Requirements Engineering that is new to me, and also I was learning the technologies with which I have never worked before.

6.6 Final Conclusion

My contribution is primarily the novel idea of consistency. The idea of consistency is defined in terms of mapping between the master web page and the one that we are comparing against the master web page. Using this idea we could automate entire process of generation of automated test cases for responsive web pages. The pages are checked for consistency by applying rules which a user can activate by selecting checkboxes or eventually putting one's own rules by writing the script. In the work I showed that the basic idea is valid with the help of implemented tool with four high level rules. The developed program demonstrates how the current approach works.

The working time was distributed between exploring the topic with related materials and tools and the development part. All the sections were completed within the determined time boundaries. Hence, from the organizational point of view, the project is successful.

The implemented program manages to include the simple design that does not distract the user with the working core functionality. The implemented user interface is simple and plain.

Finally, the whole project was completely tested and evaluated. In addition some recommendations of what could be done in future were mentioned. Overall, the project fulfilled the set of expectations and provides a great basis for the potential future development of a commercial product.

Taking into account the advantages summarized in this paper of the current approach and increasing demand of generation of automated test cases for testing responsive web pages, I believe that in the nearest future, this technology will be brought to the industry. Undoubtedly, lots of work should be done in future development to increase the functionality and effectiveness, but I am certain that these costs will pay off soon.

7. Bibliography

[1] Ann Armstrong, Joseph J. Mueller, and Timothy D. Syrett. The Smartphone Royalty Stack. http://www.wilmerhale.com/uploadedFiles/Shared_Content/Editorial/Publications/Documents/The-Smartphone-Royalty-Stack-Armstrong-Mueller-Syrett.pdf (accessed April 15. 2014).

[2] Paul Robert Lloyd. The Web Aesthetic. http://alistapart.com/article/the-web-aesthetic (accessed 19 April 2014).

[3] Tim Berners-Lee. Long Live the Web: A Call for Continued Open Standards and Neutrality. http://www.scientificamerican.com/article/long-live-the-web/ (accessed 20 April 2014).

[4] Zach Epstein. World's first web page brought back from the dead for 20th anniversary. http://bgr.com/2013/04/30/first-website-anniversary-cern/ (accessed 20 April 2014).

[5] Stephen Hay. Responsive Design Workflow. 1st ed. . New Riders; 2013

[6] Digby. Mobile Commerce and Engagement Stats. http://digby.com/mobile-statistics/ (accessed 2 May 2014).

[7] Facebook. Statistics. www.facebook.com/press/info.php?statistics (accessed 25 April 2014).

[8] Johanna Werther. 75% of YouTube Mobile users report that mobile is their primary way of accessing YouTube content. http://googlemobileads.blogspot.ie/2010/11/75-of-youtube-mobile-users-report-that.html?m=0 (accessed 25 April 2014).

[9] Joanna Krenz-Kurowska. Web design trends for 2013. http://99designs.com/designer-blog/2013/02/21/web-design-trends-for-2013/ (accessed 1 May 2014).

[10] Emarketer. Mobile Users Prefer Browsers over Apps. http://www.emarketer.com/Article.aspx?R=1008010 (accessed 25 April 2014).

[11] Orlik S. Software Engineering. Software Testing/ Introduction to software engineering and software life cycle. 2004. http://sorlik.blogspot.com (accessed 28 April 2014).

[12] Nancy Leveson. An Investigation of the Therac-25 Accidents. http://courses.cs.vt.edu/professionalism/Therac_25/Therac_1.html (accessed 28 April 2014).

[13] Jakuba,S.; Letters: Measure of risk. (NASA Mars spacecraft loss) Mechanical Engineering, v.122(3), 2000-Mar, p.6, 10

[14] Glenford J. Myers. Software Reliability. Principles and Practices. A Wiley-Interscience Publication, 1976.

[15] R. Binder. Testing Object-Oriented Systems. Addison Wesley, 2000.

[16] D. Raggett, A. Le Hors, I. Jacobs. HTML 4.01 Specification. http://www.w3.org/TR/html40/ (accessed 20 April 2014).

[17] A.A.Sortov, A.V. Choroshilov,Function testing of Web-application with UniTesK technology,Institute for System Programming, 2004

[18] Edward Miller. WebSite Testing . http://www.e-valid.com/Technology/White.Papers/wpaper.testing.pdf (accessed 10 June 2014).

[19] Srinivasan Desikan, Gopalaswamy Ramesh. Software Testing: Principles and Practice. 1st ed. India. Pearson Education; 2006

[20] J.A. Sanchez, Infographic: 2013 The Year of Responsive Design 2012

[21] Ethan Marcotte. Responsive Web Design. http://alistapart.com/article/responsive-web-design (accessed 12 May 2014).

[22] Biljanović, Petar. 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) : proceedings : May 20-24, 2013, Opatija, Croatia. Rijeka, Croatia: Croatian Society for Information and Communication Technology, Electronics and Microelectronics, 2013. Print.

[23] Marcotte, Ethan. Responsive web design. New York: A Book Apart, 2011. Print.

[24] Kadlec, Tim. Implementing responsive design : building sites for an anywhere, everywhere web. Berkeley, CA: New Riders, 2013. Print.

[25] Media Genesis. How responsive is your website? http://responsivedesignchecker.com/ (accessed 10 June 2014).

[26] Jeff Wisniewski. Responsive Web Design from the Trenches. http://librarianinblack.net/librarianinblack/2013/10/il2.html (accessed 10 June 2014).

[27] T. Kadec, Implementing Responsive design, New riders, Berkley, 2013

[28] W3C. The web standards model - HTML CSS and JavaScript. http://www.w3.org/community/webed/wiki/The_web_standards_model_-_HTML_CSS_and_JavaScript (accessed 14 May 2014).

[29] W3C. A history of HTML. http://www.w3.org/People/Raggett/book4/ch02.html (accessed 1 June 2014).

[30] W3C. What is CSS?. http://www.w3.org/Style/CSS/ (accessed 1 June 2014).

[31] W3C. What is the Document Object Model?. http://www.w3.org/TR/DOM-Level-2-Core/introduction.html (accessed 1 June 2014).

[32] W3Schools. The HTML DOM (Document Object Model). http://www.w3schools.com/js/js_htmldom.asp (accessed 1 June 2014).

[33] Diana Cerbu. UML2 with Eclipse, May 2007.

[34] Github. jgraph/jgraphx. https://github.com/jgraph/jgraphx (accessed 1 June 2014).

[35] SeleniumHQ. Selenium Documentation. http://docs.seleniumhq.org/docs/index.jsp (accessed 28 April 2014).

[36] SeleniumHQ. Selenium. http://docs.seleniumhq.org/ (accessed 18 April 2014).

[37] SeleniumHQ. Selenium WebDriver. http://docs.seleniumhq.org/docs/03_webdriver.jsp (accessed 18 April 2014