AN APPROACH TO AUTOMATE A MODIFIED HEURISTIC EVALUATION METHOD FOR ASSESSING USABILITY OF WEBSITES

J.S.S. Wijesundare

139190 E

Faculty of Information Technology

University of Moratuwa

March 2016

AN APPROACH TO AUTOMATE A MODIFIED HEURISTIC EVALUATION METHOD FOR ASSESSING USABILITY OF WEBSITES

J.S.S. Wijesundare
139190 E

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Degree of Master of Science in Information Technology.

March 2016

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

….....

Jeevanthe Sulakshan Samaradiwakera Wijesundare

Supervised by

Dr. C.D. Gamage

…...….....

Signature

Date:

Mrs. G.T.I Karunarathne

…...….....

Signature

Date:

Dedication

I dedicate this work to my parents.

Acknowledgement

I would like to acknowledge the continuing guidance and support by my supervisors, who are Dr. C.D. Gamage and Mrs G.T.I Karunarathne, which has made the conduct and completion of this study possible. I would like to thank my parents for the support they gave to complete this work on time. I would also like to acknowledge the support provided by the staff of the department of Computer Science and Engineering, University of Moratuwa, who generously offered their help in numerous ways.

Abstract

There has been a growing tendency towards greater usage of websites over the last few decades. This has resulted in a need to ensure better usability of these web sites. However, as some of the usability tests on web sites were either technology dependent, such as conformity to a particular web standard or application dependent, such as web sites built for news organizations, it was difficult to conduct a more general form of usability tests on websites. As a solution to this problem, researchers have developed various algorithms for assessing usability of websites in a more generic manner. However, as these evaluation schemes required the manual application of such algorithms, this exercise has turned out to be a tedious and time consuming task.

The research presented in this thesis had been conducted to develop an automated solution for the evaluation of usability of websites. It is hypothesized that automation of heuristic evaluation of web sites can be done by web page parsing with CssParser and Jsoup libraries. The solution takes web pages and CSS files saved on disk as inputs and produces a report of usability issues as the output.

Once the input set of web pages is provided, the automated solution developed in this research extracts certain attribute values from the saved CSS files and checks them against a set of predefined values. Based on the results, the usability problems are identified, and displayed as a report. The system developed is intended to be used by user interface designers, during the software interface design phase.

By using the web site usability solution developed in this research, web designers will be able to improve their designs after identifying the usability problems in user interfaces. The overall design of the solution include three modules, namely

the user interface module, evaluation engine module and a database of evaluation parameters. These modules were developed using the Java language and the overall system has been implemented to work in a platform independent manner.

The proof-of-concept of the automated solution for web site usability evaluation has been tested by considering 8 sample websites. The evaluation results shows that the developed scheme can evaluate the websites with an average accuracy of 67% while taking less than 30 seconds for evaluation. This clearly shows the utility of the developed system as an initial filter to identify web site designs with significant problems in their usability.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Prolegomena

The Internet has shown a continuous growth over the past few decades [1] [2]. This indicates that the amount of material available on the web has similarly increased. The huge amount of web material created by social media networks [3], has also contributed to this matter. The tendency for using the web for entertainment purposes, has further increased the amount of web pages available on the Internet.

This trend has created a research challenge to assess the usability of websites. Currently such analysis is done using manual techniques, giving results relatively inefficiently and cost ineffectively than software systems. Cognitive walk-through, heuristic evaluation are such manual methods of usability evaluation [4]. Both of these methods require an expert to conduct cognitive tasks for evaluating websites. This process can take up to hours depending on the complexity of the website.

Since assessing the usability of websites with the heuristic evaluation method requires expert knowledge, conducting evaluation sessions can be relatively costly than using an automated system for usability evaluation. This problem can be solved by incorporating professional expertise of usability experts into the evaluation logic of an automated usability evaluation system.

In a study done by Ivory & Hearst [5], they have identified that automation can expedite usability evaluation methods such as heuristic evaluation method. We have conducted a research to develop a automated solution for a modified heuristic evaluation method for assessing the usability of websites. Our solution has shown 67% accuracy in evaluating

the usability of websites. This system can be used as an initial filter for identifying websites with usability issues.

1.2 Background and motivation

In 2002, Palmer [6] predicted that the number of web pages in the world wide web will rapidly increase. By now popular search engines like Google has made selecting information from a very large volume of on-line content quite easy. When a product is plentiful it's consumers tend to be very selective in using it. This phenomena equally applies to the web. Thus, websites may not be used by Internet users unless effective efforts in usability evaluation are made [7].

Heuristic evaluation is a widely used usability evaluation method. However in a study done by Ivory & Hearst [8], they identified that the current usability evaluation methods, are not systematic and does not produce consistent results when different evaluators assess an interface using the same evaluation method. They have proposed automation as method of improving the efficiency and effectiveness of usability evaluation methods.

1.3 Problem definition

The large number of websites on the Internet has given rise to the need for efficient and effective usability evaluation methods for web sites [7], [4]. A heuristic evaluation can take several hours to perform and requires expert knowledge. As a solution to this problem, automation of the manual method can be considered. This study aims to automate a modified form of the heuristic evaluation method [9], by considering the knowledge of usability experts.

1.4 Objectives

The aim of this study is to develop a software system, which automates the heuristic evaluation method. The objectives of the study for achieving this aim can be given as follows.

1. Critically review the developments and issues in usability evaluation.

   As a first step in the research process, existing literature related to usability evaluation is examined and reviewed to identify research gaps.

2. Develop a prototype to automate the modified heuristic evaluation method.

   A software system which can evaluate usability of websites is developed. This tool is developed as a prototype, which serves as an instrument to test the hypothesis of this research.

3. Evaluate the automated solution.

   Sample websites are evaluated by the system and all of the evaluated websites are heuristically evaluated by usability experts. Then the results are compared and the hypothesis is either accepted or rejected.

1.5 Hypothesis

By developing a software based website usability evaluation system, the heuristic evaluation process can be automated.

1.6 Structure of the Thesis

The rest of the thesis is structured as follows. Chapter 2 is on critical review of the area of heuristic evaluation of web usage. Chapter 3 presents technology adapted toward an automated solution for evaluation of web usage. Chapter 4 provides the over picture of our novel approach to automated solution for assessing web usage. Chapter 5 discusses the design of the solution. Chapter 6 is about the hardware, platform, software, and algorithms related to implementation of the design. Chapter 7 reports on the evaluation of the proposed solution. Chapter 8 concludes the thesis with a note on further work.

1.7 Summary

This chapter gave an overall picture of the thesis. As a part of this discussion, the hypothesis, the objectives of the research were highlighted. The next chapter presents recent technological developments in usability evaluation. It aims at deriving a research gap from the existing literature.

Chapter 2

State of the Art in Usability Evaluation

2.1 Introduction

Chapter 1 presented an overall description of the thesis. The hypothesis, the objectives of the study and the problem addressed by the research were some of the important areas of the research presented in Chapter 1. This chapter presents a critical review of the recent developments in usability evaluation.

2.2 Recent developments in usability evaluation

In their study, Allen et al. have developed a quick, simple, and inexpensive variation of the heuristic evaluation method, which can find usability issues in web pages [10]. This evaluation method is based on Nielsen's 10 heuristics [11], and Shneiderman's eight golden rules [12]. The method by Allen et al. is faster than the standard heuristic evaluation approaches, because this method avoids direct website interaction, and the review of the identified usability problems. Avoidance of above two factors amounts to simplicity and inexpensiveness of the proposed heuristic method. However, this heuristic evaluation method has not been formally tested to date.

Research by Seffah et al. have reviewed existing usability standards and consolidated them into a hierarchical model of usability measurement [13]. This model is based on the usability attributes identified by Constantine & Lockwood [14], Schneiderman [15], Nielsen [16], Preece et al. [17], Shackel [18], and ISO 9241-11[19]. This model can be used as a tool for developing usability measurement plans, which are used for collecting data from websites for calculating metrics that represents it's overall usability. Usability evaluation has been time consuming due to large number of metrics in Seffah's work. In

comparison with the research by Allen et al. [10] this method is less efficient and it is done manually.

Nielsen & Molich have examined four heuristic evaluation methods [20]. They have reviewed the the set of heuristics developed by Smith & Mosier [21], and have identified that designers prefer relatively less number of heuristics to be used in usability evaluations. For these experiments they have used ten heuristics identified in their early research [22]. The results of the current study shows that when the evaluations are done by multiple evaluators, it leads to the identification of a large number of usability issues if aggregation of evaluation results is performed. It has also been found that the number of evaluators between three and five appears to deliver best results. Two of the experiments in this study have been conducted in the paper based format. Probably this results may be different from the ones generated though a software interface.

Carta et al. have developed a software tool that allows remote usability evaluation of a websites [23]. In remote usability evaluation, the users and system evaluators are either spatially or temporally separated [24]. The software tool by this research supports accumulation many types of data related to user interaction with the evaluated websites, and selecting the type of tasks users can perform during interaction. The tool also enables visualization of the collected data so that experts are able analyze the data, to identify usability issues in the websites considered. The tool requires a usability expert to participate in the evaluation, so that an optimal sequence of user actions for performing the selected tasks to be used in the evaluation can be defined in the software system. The necessity of involvement of the usability experts, constrains the applicability of the software tool introduced in this research.

Oslina & Rossi have developed an evaluation method, named as web quality evaluation method [25]. This research is based on Web Quality evaluation method [26] and the WebQEM software tool [27]. They have implemented a software tool (WebQEM), which can compute a website quality score by considering various metrics pertaining to

website quality. This computed value can be used to compare the quality of different web sites. This model can be used to measure the quality of a website with reference to standards, such as ISO standards for website quality. However we have note found a comprehensive study which investigates the correlation between quality standards and website success in general.

Akincilar & Dagdeviren have developed a model for evaluating the quality of hotel websites [28]. This model is based on the models developed by Saaty [29], and Brans et al. [30]. As Akincilar & Dagdeviren have identified, the new model produces reliable and robust results. The model can identify the quality of a web site using a hierarchy of metrics. However, evidence of the validity of this model has not been reported to date.

Oztekin & colleagues have developed a software tool, which can evaluate the usability of e-Leaning systems [31]. The evaluation algorithms this tool uses is based on a number of prediction models [32], [33], [34], [35]. These machine learning methods ensure that the accuracy of the results generated by the software system increases with the number of systems evaluated. Also, they have combined the results produced by different prediction models, which further improves the accuracy of the results. This software tool is capable of suggesting design improvement strategies for the assessed user interfaces. The data set required for the tool is created through an on-line checklist [36]. The data collected through the questionnaire is then analyzed to identify possible usability issues with e – Learning systems. However, it may not be easy to incorporate evaluation method as proposed into the software development process, since it requires considerable time for the data collection.

In a study done by Hong et al. [37], a software system that can support the identification of website usability problems was developed. This tool implements proxy based logging, and can track user interactions in detail [38]. Further, it is compatible with a range of operating systems, and since it uses a proxy, the tool overcomes many of the disadvantages of server side and client side logging. However, this software system

requires manual intervention and is only an analysis of the quantitative data derived from user interactions.

Atterer et al. have developed a software tool, which can track user interactions such as navigation between pages and actions that a user performs during the navigation of a page, non intrusively [38]. This is made possible by a proxy server [39], which embeds JavaScript code in the HTTP responses from the server with the requested web pages. The tool is designed to be compatible with existing server and browser setups. However, this usability evaluation method has not been formally tested to date.

In a study done by Botella et al., they have reviewed existing interaction design patterns and usability heuristics have developed a framework for improving the quality of heuristic evaluation reports [40]. These design patterns are based on the work done by Wellie [41] and Alexander [42]. By presenting the heuristic evaluation results using design patterns, the proposed framework can improve the designer's understanding of the identified usability problems. Further, solutions to the identified usability issues (from a heuristic evaluation) can be done with the help of the design patterns. However, they have not given clear evidence of the validity of the heuristics used for this method of usability evaluation.

Dingli & Cassar have developed a software agent that can evaluate the usability of websites [43]. This software system is based on the work done by Misfud & Dingli [44]. The tool developed by Dingli & Cassar can capture website data, analyze the websites and suggest improvements in its' design with minimum human intervention. Further, it can produce evaluation results as easy to understand reports. However, the usability heuristics used for developing this automated tool probably may not be valid for all websites.

2.3 Problem definition

The above study shows that numerous limitations of different usability evaluation methods and models used. Among other issues, the requirement for expert knowledge and skill, even for automated systems, inefficiencies of the evaluation models, and lack of formal validation can be highlighted. Theses issues are summarized in Table 2.1.

It is evident from the literature that, a cost effective, efficient and an effective heuristic evaluation mechanism remains a research challenge. We intend to solve this problem by automating the heuristic evaluation methods using a software system.

Table 2.1 Limitations of recent developments in usability evaluation

Author	Limitation
Allen et al.	Potential ineffectiveness of the evaluation method
Seffah et al.	Probable inefficiency incurred when the model is used for usability evaluation
Nielsen & Molich	Two of the experiments have not considered real user interfaces for evaluation. Thus they may not identify usability errors in the functionalities of the two systems considered.
Carta et al.	Software tool developed needs expert knowledge for operation.
Oslina & Rossi	Lack of empirical validation of the software tool developed.
Akincilar & Dagdeviren	The model is not tested to date.
Oztekin et al.	Potential inefficiency of software interface evaluation.
Hong et al.	Software tool needs manual intervention in usability evaluations.
Atterer et al.	Evaluation tool not formally tested.
Botella et al.	Lack of validity of the the usability evaluation method
Dingli & Cassar	Heuristics used for the evaluation tool may not be valid for all types of websites. (certain heuristics target only certain types of users such as older adults or physically challenged users.)

2.4 Summary

This chapter highlighted the limitations of the recent developments in usability evaluation. Also a cost effective, efficient, and effective heuristic evaluation method was identified as a research challenge. The next chapter discusses the technology used in implementing the automated system.

Chapter 3

Technology Adapted: HTML and CSS style-sheet analysis

3.1 Introduction

Chapter 2 highlighted the limitations of researches related to heuristic evaluation of software, as well as usability evaluation methods in general. Further, an effective and efficient heuristic evaluation method was identified as a research challenge. This chapter presents the technology used for implementing the automated heuristic evaluation method.

3.2 Programming Language

The software system developed in this research is implemented using the Java language. During runtime the system executes on a virtual machine (Java Virtual Machine). The JVM is available for a variety of operating systems. Once the JVM has been installed on a laptop or a PC, the system can be successfully executed after installation. Thus this automated system is platform independent.

Since Java is an object oriented programming language, the the E-Validator software system was developed based on object oriented principals. This enabled the automated system to be designed in a way that allows any number of 'error' classes, related to usability errors, to be integrated into the system architecture.

3.3 HTML scanning and data extraction

During web page analysis, in certain situations, it may be required to extract information from the web page (from the HTML encoding) before any CSS code analysis is being done. For example, menus on web pages can be created with div tags in HTML and CSS code. When evaluating the usability of web pages, one of the errors that the system checks is whether cascading menus have a small target area to click on. In order to check for this error, the automated system, more specifically the evaluation engine, first has to analyze the div tag hierarchy such as the one shown below.

```
<div>
     <div class = “inner”>
     </div>
</div>
```

Figure 3.1 Nested unit of div tags

(The above code fragment shows how to nest div tags. The innermost tag element has the attribute called class with value inner.)

During this analysis it is required to extract the class attribute values from all the div tags in the web page. This task is done using methods from the Jsoup library. The relevant source code for extracting this information is given below.

```
/*Load web page*/
File input = new File("/home/input.html");
Document doc = Jsoup.parse(input, "UTF-8",
"http://example.com/"); /*Parse the page*/

Element link = doc.select("div").first();    /*Navigate to
the first div tag*/
```

```
String data = link.attr("class");       /*Extract attribute
value of type class*/
```

Figure 3.2 Extraction of web page data from div tags with Jsoup
(The above source code fragment shows how to extract attribute values from div tags in a web page. Here the class attribute value from only the first div tag is extracted.)

The Jsoup library is a collection of objects, methods and related documentation for navigation and information extraction from on-line and off-line web pages. It was released under multiple versions, where the latest version is Jsoup 1.8.3.

3.4 CSS data processing

Mostly, the identification of each usability error in a web page is done through CSS (Cascading Style Sheets) processing. During evaluation, the system reads the CSS files saved on disk, which are saved along with the web page in a separate folder. The CSS instructions for the display of web pages is identified using methods in the CSSParser library. It provides objects and methods to extract information on CSS style-sheet settings for web pages from CSS files saved on disk.

3.4.1 CSS Selectors

A CSS selector denotes a setting that applies to a particular HTML element such as a anchor tag (which represents hyper link) or a div tag (organizes elements in a HTML document for display). The selector can be a generic entity or be named with a user defined string. Figure 3.3 shows a generic selector while Figure 3.4 shows a selector defined by a user.

```
a : visited {
color :  red;
text-decoration : none;
}
```

Figure 3.3 A generic selector

(This figure shows how a generic selector is used to define the behavior and state of a typical hyper-link.)

```
.zn-body__paragraph { /* selector name */
    color: #262626; /* property : value */
}
```

Figure 3.4 A selector named with a user defined string

(The above code fragment shows how a selector can be named by a user defined string of characters. It contains the CSS commands to set the color of the paragraph text to #262626 (the color specification is in hexadecimal).)

3.4.2 CssParser Library

The CSS parser library provides objects and methods for extracting the selector name, property name (such as text-decoration for hyper-links), it's value and the selector priority. The source code for extracting these values is shown below. The CSS parse library is written in Java language.

```
InputSource source = new InputSource(new
StringReader(linkData));
CSSOMParser parser = new CSSOMParser(new SACParserCSS3());
```

```
// parse and create a stylesheet composition
CSSStyleSheet stylesheet =
parser.parseStyleSheet(source,null,null);

 //ANY ERRORS IN THE DOM WILL BE SENT TO STDERR HERE!!
                // now iterate through the dom and inspect.

          CSSRuleList ruleList = stylesheet.getCssRules();

            for (; i < ruleList.getLength(); i++)
                  {
                  CSSRule rule = ruleList.item(i);
                   if (rule instanceof CSSStyleRule)
                  {
             CSSStyleRule styleRule=(CSSStyleRule)rule;

CSSStyleDeclaration styleDeclaration=styleRule.getStyle();
CssDataBundleList[] databundlelist=new
CssDataBundleList[500];

/*Data structure of Css settings for current selector. Each
css file contains multiple selectors */

int j;

CssDataBundleList temp;

for ( j = 0; j < styleDeclaration.getLength(); j++)
{
```

```
 String property = styleDeclaration.item(j);

temp=new CssDataBundleList (property,
/* Get selector name*/
styleDeclaration.getPropertyCSSValue(property).GetCssText()
, /* Get selector value */
styleDeclaration.getPropertyPriority(property));
/* get selector priority */

databundlelist[j] = temp;   /* Add results to the data
structure*/

}
```

Figure 3.4 Extracting data from CSS style-sheets

(The above code fragment shows how to extract details pertaining to selectors from CSS style sheets using the methods and objects provided by the CSSParser library.)

3.5 Limitation of the static CSS code analysis

Even though this method of information extraction from web pages supports the identification of usability issues, it cannot be used to identify certain usability issues in web pages, such as the ones pertaining to web page content. For example, it cannot check usability problems related to the heuristic - minimization jargon or technical terms in textual information [9] . Handling such cases may involve additional technologies and different system design.

3.6 Summary

This chapter discussed the technology which was used to automate the heuristic evaluation process. The core technology used for this purpose is the CSSParser library. To-date, almost all web pages are displayed according to CSS style-sheets. Thus using the CSSParser library it was possible to analyze web page state and behavior quite effectively. This information is used in identifying the usability issues in websites. However, it should be noted that it is not possible to detect all information related to web page state and behavior via analysis of CSS style-sheets. The next chapter presents the approach used for developing the heuristic evaluation method.

Chapter 4

A novel approach to Automate heuristic evaluation

4.1 Introduction

Chapter 3 discussed the technology for developing a software system which automates heuristics evaluation of websites. Further, limitations the identified technology was highlighted. This chapter presents our approach for developing the heuristic evaluation method under several headings, namely, hypothesis, input output, process, users and features. The chapter highlights how the novel approach offers a cost effective and efficient solution for assessment of websites.

4.2 Hypothesis

The hypothesis of this research can be stated as follows.

By developing a software based website usability evaluation system, the heuristic evaluation process can be automated.

This statement is either supported or rejected based on the results of the evaluation step of the research. This result is documented in Chapter 8, which presents the conclusion of the study.

4.3 Input

The system takes HTML 5 web pages and, their related CSS files, saved on a disk, in a PC or a laptop, as input. The end-user initially selects the web pages. The system then searches for the relevant CSS files on disk for processing. Identifying usability errors in a

web page requires the analysis of CSS style-sheet information.

4.4 Output

After processing HTML and CSS data, the system produces a report, containing the usability errors in a web page. The usability errors are printed on a text area on a graphical user interface. The user can also view all the web pages added to the system for processing, as well as the amount of files added, which is displayed as a percentage. The output can be filtered so that usability problems related to older adults are removed from the report dynamically, and usability errors pertaining to only the average user is displayed.

4.5 Process

The automated system initially extracts the instructions from the CSS files related to the web pages designated by the user. The fetched CSS instructions are then stored in a data structure. Then it's contents are examined for identifying the usability errors in web pages. The descriptions of these errors, which are stored as phrases are then inserted into a report, which is displayed to the end user.

As web pages are added to the system, an integer counter is incremented. A maximum of twenty web pages can be added to the system. The percentage load on the system in terms of number of web pages added is displayed to the user by considering the value of the counter. The process of identifying usability issues in web pages is shown as a flow chart in Figure 4.1.

4.5.1 Identification of usability errors

The system analyzes the HTML and CSS code by checking for certain values, which conforms to professional experience of usability experts, and converts the resulting flags

into phrases. These phrases (usability issues) are displayed as a report.

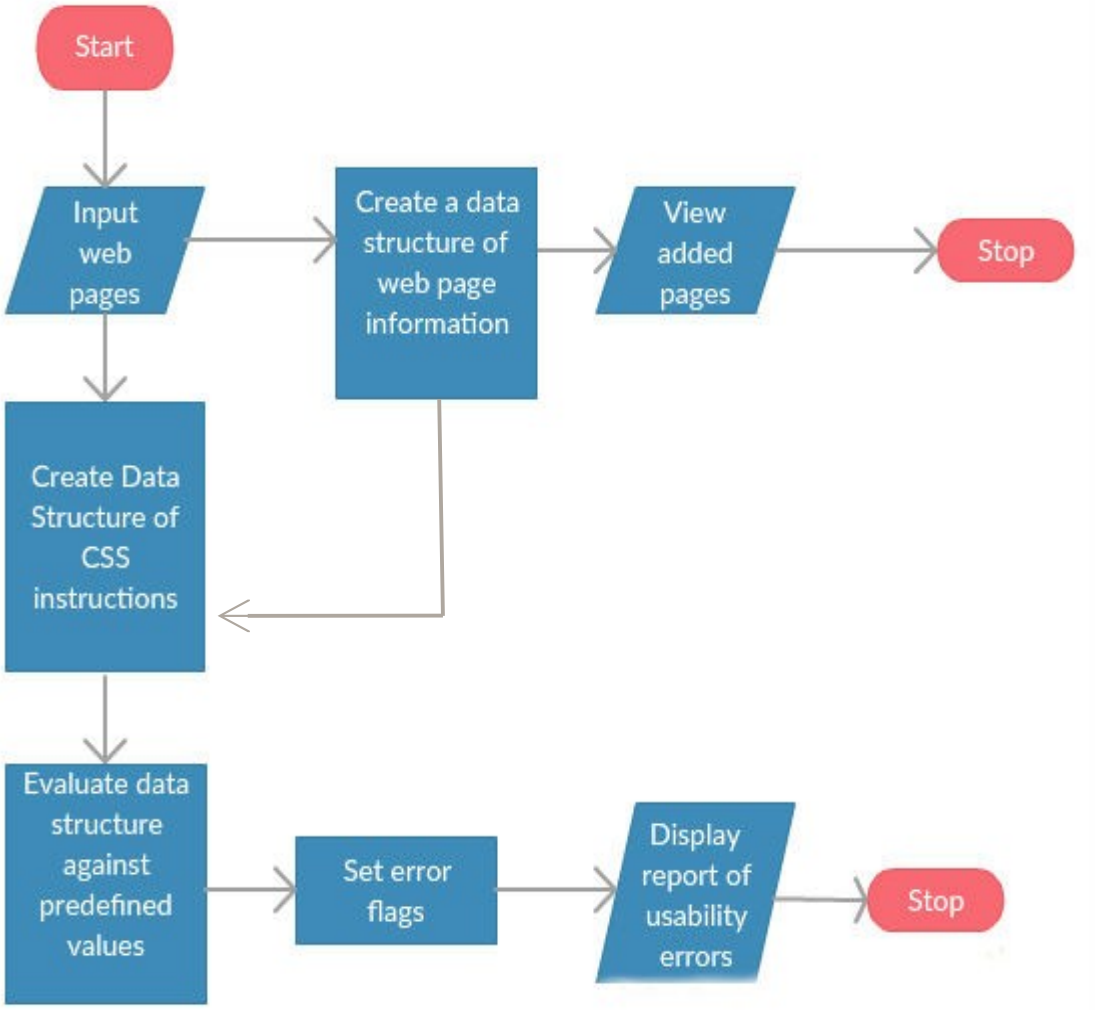


Figure 4.1 Identification of usability errors by the automated system

(This shows the process of identifying usability errors in web pages. After setting the error flags, the user has to give a command to view the report of usability issues. This is not explicitly shown in the diagram. This diagram shows a typical instance of usability evaluation by the E-Validator software system.)

4.5.2 Filtering specific results

The system identifies usability errors of two types. Generic usability errors and errors specific to older adults. The filtering feature takes all the usability errors identified by the parsing process, and extracts only the general usability issues, and displays them on the screen.

4.6 Features

The E-Validator software system has the following features.

i. Display usability issues in a web page as a report.
ii. Enable the user to view a list of files added to the system.
iii. Filter out usability errors specific to older adults.

4.7 Users

Usability evaluation should ideally, be integrated the designing phase [4]. The automated system can be used in this phase, where the it is used by user interface designers. Thus the designers can quickly get a usability report on a web site as soon as a design decision has been realized on a web page.

4.8 Summary

This chapter presented our novel approach to develop an automated solution for heuristic evaluation. In this sense, it is pointed out how the novel approach offers a efficient and accurate solution for assessment of websites. The next chapter shows the design of the automated solution.

Chapter 5

Design of the E-Validator System

5.1 Introduction

Chapter 4 presented the approach to develop an automated heuristic evaluation system for assessing the usage of website. The hypothesis of the research, the automated system's inputs, outputs, the process that converts the inputs into output, the softfware system's features were included in this chapter. This chapter elaborates the approach, and describes the architecture of the solution. The top level architecture of the solution includes three modules, namely, interface, the heuristic engine, database of evaluation parameters.

5.2 Top Level Architecture

The top level architecture of the automated solution is shown in Figure 5.1. Within this architecture, the evaluation engine constitutes the core of the solution. All the other modules are connected and coordinated by the engine. Currently, the database has not been integrated to the system. The evaluation parameters are embedded in the evaluation engine but these values can be easily migrated into the database. This can be done after modification of the evaluation engine. Next we briefly describe the function of each module.

5.2.1 User Interface

The user interface offers facilities to interact with the system for designers. This interface can select web pages for evaluation, view currently added pages in the system, supports commands for the evaluation of the selected web pages, and the filtering of evaluation results specific to older adults. The user interface of the automated system consists of two

reports and a menu driven interface.

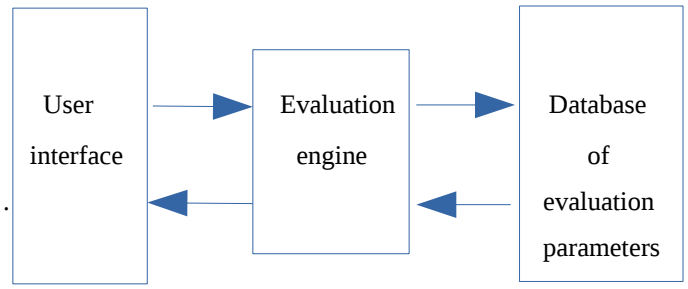


Figure 5.1 Top level architecture of the E-Validator system

(The above figure shows the main components of the automated solution.)

Table 5.1- The table below shows a mapping between the heuristics selected for the study and the WCAG 2.0 usability guidelines

Heuristic	WCAG 2.0 usability guideline	Usability errors identified by the E-Validator System
Use conventional interaction elements.	Links 2.4.4 Link Purpose (In Context): The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general. (Level A)	1. Browser links are not in purple 2. Links are not underlined or dynamically underlined
Make click-able items easy to target and hit.	Guideline 2.4 Navigable: Provide ways to help users navigate, find content, and determine where they are.	Menus have small targets to click on

5.2.2 Evaluation Engine

The evaluation engine maintains a data structure of CSS instructions, originally used for web page display. The examination and analysis of the CSS style-sheet information of web pages in the system, and the identification of usability errors is done using this module. This is done by transforming CSS style-sheet information for web pages into a composite data structure, comparing the values in this data structure against a set of predefined values and identifying whether the pages in a website contains the specific usability issues.

The evaluation engine includes five important sub-modules, which are WebPageParser, sub classes of PageError sub module, and CssParser. This engine is connected to the user interface for obtaining details of web pages designated for evaluation.

The WebPagParser sub-module constitutes the core of the evaluation engine. It has an array of WebPage objects which corresponds to the web pages added to the software system for evaluation. Also, this object contains a two dimensional array of error objects. An error object encapsulates the information related a particular usability error which the system can identify. It has a 'parse' method which examines a composite data structure of CSS style-sheet information. Each row in this array contains the error objects that correspond to a web page added to the system. Once the user clicks on the view results button, the WebPageParser module invokes the parse method of each error object in the array and evaluates the usability of all the web pages added to the system.

The CssParser is another important element of the evaluation engine. It is responsible for creating a data structure of CSS style-sheet information. The CssParser object contains methods for extracting CSS instructions from the HTML inside the style tag of a web page and from CSS style-sheets saved on disk.

5.2.3 Database of evaluation parameters

The database module can ensure that web page data can be stored on disk so that evaluation results of websites can be retrieved later. However this module has not been integrated into the system yet. The automated solution is fully functional without this module.

5.3 A mapping between the selected heuristics and the WCAG 2.0

The WCAG (Web Content Accessibility Guidelines) 2.0 guidelines ensure that web pages implementing them are more accessible by individuals having disabilities. They also ensure that the designs adapting these guidelines are usable for the general website user [45]. These guidelines are recommendations for website design, by the World Wide Web Consortium (W3C). The usability heuristics used for the implementation of the E-Validator system was based on the work done by Chisnell & Redish [9]. Table 5.1 shows a mapping between the usability heuristics considered for implementing the automated system, and the WCAG 2.0 usability guidelines.

5.4 Summary

This chapter described the design of the automated system. The details of the top level architecture of this software system and it's modules was also presented. Further, a mapping between the WCAG 2.0 usability guidelines and the heuristics used for this study was presented. The next chapter provides the implementation details of the automated solution.

Chapter 6

Implementation of the E-Validator Software system

6.1 Introduction

In chapter 5, the top level design of the solution has been described in terms of what each component does. This chapter describes the implementation of each component regarding hardware, software, algorithms, flowcharts etc. In that sense this chapter is about how the system is implemented.

6.2 Implementation details

E-Validator has been implemented to run in a platform independent manner. This was possible since it was developed using the Java language. The following presents the implementation details of the individual components in the system. Figure 6.1 shows the essential sub-modules in the system and their connections.

6.2.1 User interfaces

The desktop version of the user interface has been designed using Netbeans 8.01. This interface has three screens. The main interface is menu driven and has commands to invoke the evaluation report interface and the interface which shows the system load in terms of the added web pages. However, until an evaluation of a page has taken place, the system does not considerably load web page data into the system memory.

6.2.2 Evaluation engine

The evaluation engine contains ten sub-modules. Each of them are discussed in the following subsections. In the discussion, during certain places, the term sub-module is used to refer to an object in the system. However, it is important to note that, the term sub-module refers to software entity, whereas an object refers to a portion of memory allocated in the system.

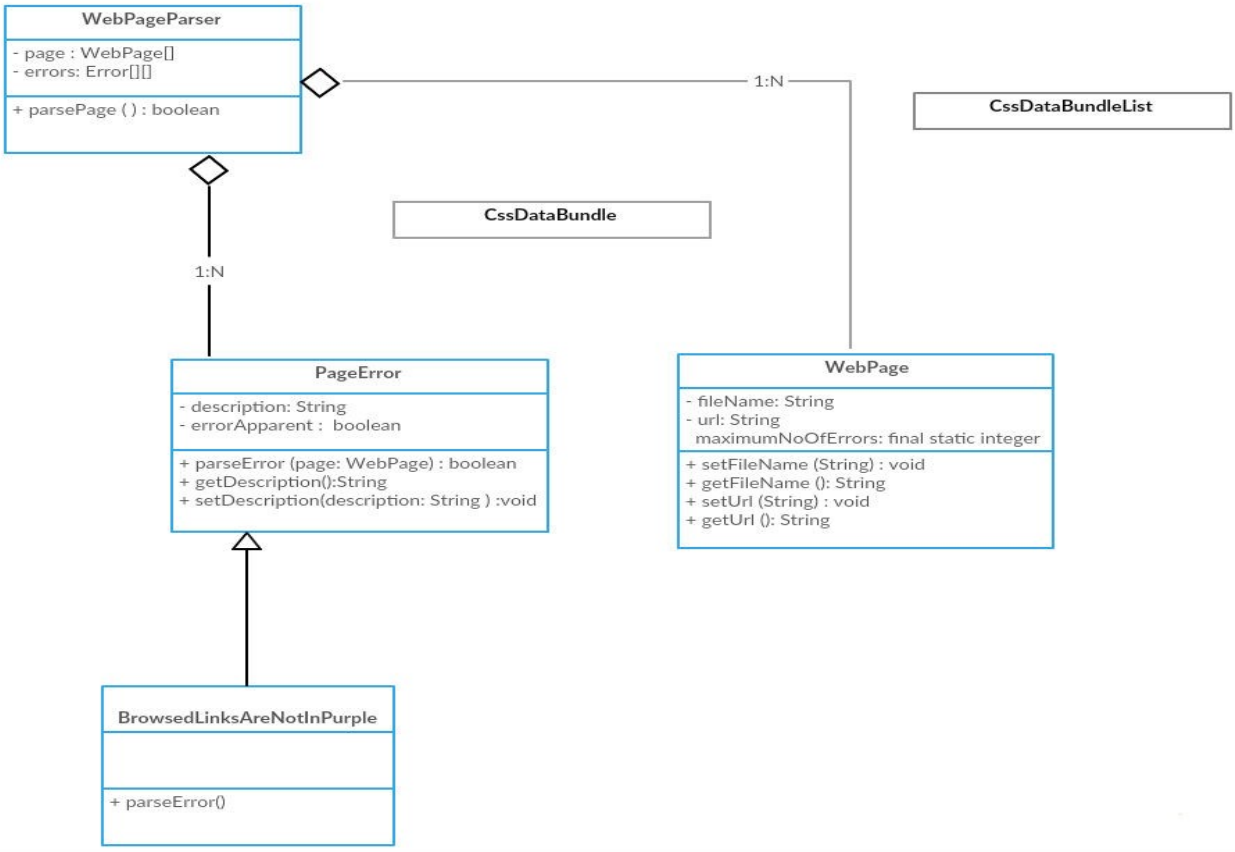


Figure 6.1 Essential sub-modules of the E-Validator software system.

(The above diagram shows the essential classes of the system. For clarity all the members of the classes are not shown.)

6.2.2.1 WebPageParser

This is the core of the evaluation engine. It contains details of all the web pages in the system and results of usability evaluations. The latter type of data is retained in a two dimensional array where the former is stored in a single dimensional array. When the user commands the system to assess a page, the WebPageParser object performs the essential processing for identifying the usability errors in web pages in the system. When the user adds web pages to the system, the pages array is populated with data about the added web pages. All the details of the status of each of the errors in web pages is stored in the 'error' array. When the end user clicks the evaluation button on the graphical user interface, this array is updated, which finally holds the evaluation results.

6.2.2.2 WebPage

This sub-module encapsulates a web page in the software system. Essential details of a web page, such as its name and maximum number of errors in a page, is included as the state of the this object. This sub-module is used by the WebPageParser class. The WebPage object contains getter and setter methods for accessing its state.

6.2.2.3 CssDataBundle

Specifying CSS instructions for web pages is done basically using selectors. Each selector has a set of related data, such as attributes, their values and priorities. A CssDataBundle object in the system contains the data produced by the CssParser sub-module. For the former type of object, the selector name relevant for a particular HTML element, and an array of CssDataBundleList objects represents it's state. Both of these objects are created by the CssParser object during run time, when creating the data structure of CSS style-sheet information.

To better understand the state of the CssDataBundle object consider the CSS instructions included in Figure 3.3. These display instructions can be in a CSS style sheet, as a text file or inside style tags in the HTML of a web page. After CSS data extraction the corrosponding CssDataBundle object will have a:visited as the selector name, and two CssDataBundleList objects as its state (assuming that there are no other CSS instructions). The first CssDataBundleList object will have its property attribute set to color and the value attribute set to red. Similarly, the next CssDataBundleList object will have it's property and value attributes set to text-decoration and none respectively.

6.2.2.4 CssDataBundleList

This sub-module contains a list of values under a selector as it's state. These values are the selector's attribute, value and priority. The CssParser object identifies usability issues in a web page by comparing the values in a CssDataBundleList sub-module against a set of values stored in the evaluation engine.

6.2.2.5 CssFileOfRules

This object contains the styles in a CSS file in a form understandable by the evaluation engine. During the processing of a web page, information in the CSS files are converted into an array of CssFileOfRules objects. It is the main data structure handled by the objects such as BrowsedLinksAreNotInPurple, which encapsulates a specific usability error. Each of these 'error' classes have a parse method which examines this fairly complex data structure, and identifies any usability issues in a web page. This sub-module contains the name of the web page, the number of CSS rules associated with a page and an array of CssDataBundle objects as its state. Collectively these information forms an abstraction of a CSS style sheet.

6.2.2.6 CssParser

The CssParser sub-module prepares a data structure of CSS instructions. In order to do this, this object reads each CSS file pertaining to an evaluated web page and converts the CSS style-sheet data into an array of CssFileOfRules objects. When formulating this array, the CSS instructions embedded inside the style tag of a web page is also considered by the CssParser object. For example, consider the following HTML encoding of a web page with CSS instructions embedded inside the style tag [46].

```
<!DOCTYPE html>
<html>
<head>
<style>
/* inside style tag */

/* unvisited link */
a:link {
    color: red;
}

/* visited link */
a:visited {
    color: green;
}

/* mouse over link */
a:hover {
    color: hotpink;
```

```
}

/* selected link */
a:active {
    color: blue;
}
</style>
</head>
<body>

<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order to be effective.</p>

</body>
</html>
```

Figure 6.2 How to include style-sheet information in a web page

(The above HTML code shows how style-sheet information can be included in a web page. The CSS code is included within the style tag.)

6.2.2.7 PageError

The PageError is a super class of all errors. For this object, errorApparent is an attribute of type boolean, having protected access. In the Java language, protected access for an

attribute implies that objects in the subclasses inherits that attribute. When this class is inherited, the errorApparent attribute in the sub-class object indicates whether the error in the selected web page is present or not. Figure 6.2 shows the source code of the PageError class.

```
public class PageError {

private String description; /*Description of the error */

protected boolean errorApparent; /*Is the error present in the web page */

protected int severityRating ; /*related to a functionality currently not implemented*/


    public int getSeverityRating(){
     return severityRating;
    }
    public PageError(String aDescription){
    this.description=aDescription;
    this. errorApparent=false;
    }

public void setDescription(String aDescription) {this.description=aDescription;}

    public String getDescription(){
    return description;
```

```
    }

public boolean parseError(CssFileOfRules[] cssData,int
numberOfFiles)
{
    return true;/*Overriden by sub classes*/
    }
  public boolean getErrorApparent(){return errorApparent;}
}
```

Figure 6.3 PagerError class

(The above code fragment is the class definition of the PageError class. It is the parent class of all specific usability errors.)

6.2.2.8 BrowsedLinksAreNotInPurple

This sub-module is a subclass of the PageError class. The errorApparent attribute of this class indicates whether there are is any hyper-link in the page where its color appears in a color other than purple. This object overrides the parseError method of the PageError super-class. This was done so that usability error identification by the evaluation engine can be done through run time polymorphism, which makes the source code more simple and readable. The below code fragment from the WebPageParser shows how this was done.

```
public boolean parsePages(){ /* Method of WebPageParser
class*/

    if(numberOfWebPages==0)
    {
```

```
JOptionPane.showMessageDialog(new  JFrame(),  (String  )"No
pages added. Please add a page to analyze.","Information",
JOptionPane.INFORMATION_MESSAGE);
    return false;
    }
    parsedOnce=true;
    CssParser cssParser = null;
    CssFileOfRules[] allCssNew = null;
/* Main data structure prepared by  CssParser*/

for(int i=0;i<numberOfWebPages;i++){

cssParser=new CssParser();
/*  Create new CssParser object*/

allCssNew = cssParser.searchAllPossibleCss( pages[i]);

/* Populate the primaray data structure by processing CSS
style sheets*/

for(int j=0;j<WebPage.maximumNoOfErrors;j++){

/*Set error flags*/
errors[i][j]. parseError (allCssNew,
cssParser.getNumberOfCssFiles() +1);
}
}
```

```
        return true;
}
```

Figure 6.3 parsePages() method of WebPageParser

(This code fragment from the WebPageParser class shows how the web pages were evaluated while making use of run time polymorphic behavior.)

6.2.2.9 LinksAreNotUnderlined

This module represents a subclass object of PageError object. When the errorApparent attribute of this class is set to true, it indicates that a hyper-link in a web page is not underlined or is not dynamically underlined when the user hovers over the link. This object contains a parseError method as one of its members, which overrides the PageError classes parseError method. This enables run time polymorphism which is used by the WebPageParser object.

6.2.2.10 MenusHaveSmallTargets

The MenusHaveSmallTarget object is also a subclass of the PageError object. This class encapsulates a usability flaw of the instance where small target area is allocated for menu items. This usability issue applies users in all age groups and cognitive and physical abilities.

6.3 Summary

This chapter presented the implementation details of the E-Validator software system. In this discussion, the major modules in the system and the sub-modules of the evaluation engine, were presented. The next chapter describes the how the software solution was evaluated.

Chapter 7

Evaluation

7.1 Introduction

Chapter 6, presented the implementation details of the E-Validator system. Details relating to all the sub-modules and the user interface was presented in this chapter. This chapter describes how the software solution was evaluated to determine an overall system accuracy.

7.2 Data collection

During this stage of the research, usability evaluation of sample websites by industry experts and via the automated system was done, and the results were compared to calculate a measure of system accuracy.

For this purpose ,eight websites were selected for system validation. The questionnaire used for collecting expert reviews on the selected websites which contains the URLs of the websites is included in Appendix C. These URLs are also shown in Table 7.1. This questionnaire also includes the instructions for the evaluation. The assessments were performed by QA engineers, UX engineers etc. Some of the web pages were designed (with specific usability issues) for the study, and had to be hosted in a server. This website is shown in Table 7.1 as w8.

7.2.1 The evaluated websites

The web sites considered for the evaluation is included in the Table 7.1. For simplifying data analysis, a label is assigned to each URL which is shown in a separate column of this

table.

7.2.2 System generated output

Table 7.2 shows the usability errors identified by the system. Also the labels assigned to each of them is shown as a column in the table. These errors represents the range of usability errors identified by the current release of the system. The first usability error (1) included in table 7.2 occurs during website interaction when the visited hyper-links do not appear in purple color. The second usability error (2) indicates that hyper-links in a web page are not underlined. The third usability issue (3) occurs in a web page when hyper-links are not underlined when a user hovers the mouse pointer over them. The final usability error indicated by 4, occurs in a web page when menus have a small target area to click on.

Table 7.1: This table shows the websites considered for evaluation and their assigned labels.

URL of the website	Label assigned
http://edition.cnn.com/EVENTS/1996/year.in.review	w1
http://edition.cnn.com/2016/01/14/us/possible-powerful-supernova/	w2
http://www.barnesandnoble.com/b/textbooks/_/N-8q9	w3
http://www.bbc.com/future/story/20160124-are-paper-books-really-disappearing	w4
http://www.bbc.com/news/technology-35706730	w5
http://news.nationalgeographic.com/news/2014/01/140103-music-lessons-brain-aging-cognitive-neuroscience/	w6
https://www.longtermcarelink.net/a8profiles.htm	w7
http://www.hci.midmaas.com	w8

By running the automated solution on the websites given in Table 7.1 the usability issues in the web pages were identified. The results are summarized in Table 7.3.

7.2.3 Results of expert evaluations

The websites shown in Table 7.1 were assessed by industry professionals. As Nielsen & Molich [20], has found, heuristic evaluations provide best results when the usability issues identified are aggregated. For validating the automated system, the assessment results are aggregated before being used for analysis. The aggregated evaluation results for the selected websites is shown in Table 7.4.

Table 7.2 This table shows the different usability errors and their assigned labels. These labels were mapped to simplify referencing inside the text.

Usability issue identified by the system	Label
Browsed links does not appear in purple	1
Links are not underlined	2
Links are not Dynamically underlined	3
Menus have small targets to click on	4

7.3 Method of analysis

The data in Table 7.3 and Table 7.4 were combined into a single table and then were analyzed to determine the overall system accuracy as a score. The calculated score indicates to which extent, the automated method produces a similar result as the manual method. The method of calculation of the score of accuracy, with reference to Table 7.5, is shown in Figure 7.2.

$$Accuracy = (\sum_{i=1}^{n} value_i \in column\,4 \div Total\ number\ of\ errors\ found\ by\ experts) \times 100$$

Figure 7.1 Formula for calculating accuracy.

(The above figure shows the mathematical formula for calculating the system accuracy with reference to table 7.5)

Table 7.3 This table shows the usability errors identified by the system. The system inputs are the websites included in Table 7.1.

Website considered	Quantified usability issues
w1	1
w2	1, 2
w3	1, 2
w4	1,2
w5	1
w6	1
w7	1, 2
w8	1, 4

7.4 Results

An accuracy of 67 % was identified after computation of results using LibreOffice Calc software tool (a spreadsheet application). The information used for this calculation is shown in Table 7.5.

Table 7.4: This table shows the aggregated results of the heuristic evaluations done by industry experts. w1 to w8 refers to the websites included in Table 7.1. Labels of the usability problems included in the second column refers to Table 7.2.

Website considered	Quantified evaluation results
w1	1
w2	1, 2,3,4
w3	1, 3
w4	1,2,3
w5	2,3
w6	1,3
w7	1,3
w8	1, 3

7.5 Analysis of the test-case with the lowest accuracy

According to table 7.4, the test-case that corresponds to the lowest accuracy is assessment done for the website w3, w6 and w5. The reasons underlying this mismatch in results are given below.

1. Human error

   The industry experts rated the website w3 (refer to Table 7.1) as having usability errors 3 and 4 (refer to Table 7.2) as shown in Table 7.5. These errors were not detected by the system. However, when inspecting the web page w3 it was identified that usability flaw 2 was present in the web page, which was not detected by the experts. Further, after examining w3, it was identified that all the hyper-links in the page were dynamically underlined during browsing which indicates another error made by human experts.

2. Limitations of CSS data analysis

   When referring to Table 7.5, it is clear that for the w5, the automated system did not detect the usability flaws 2 and 3 despite the fact that these usability errors were present in the web page. It is believed that this could be the result of limitations in the CSS data analysis process. In certain instances, CSS instructions for a particular web page can be given in a number of ways. Thus overcoming such limitations of the system requires more in-depth study of the CSS language, and designing the system at larger scale.

Table 7.5 The table below shows the table used for the calculation of the accuracy of the E-Validator software system.

Website	System generated result	Expert assessment	Match (1=Yes,0=No)
w1	1	1	1
w2	1	1	1
	2	2	1
		3	0
		4	0
w3	1	1	1
	2	3	0
		4	0
w4	1	1	1
	2	2	1
		3	0
w5	1	1	1
		2	0
		3	0
w6	1	1	1
		2	0
		3	0
w7	1	1	1
	2	2	1
		3	0
		4	0
w8	1	1	1
		2	0
		3	0
	4	4	1
		Overall Score	66.6666666667

7.6 Summary

This chapter presented the details of evaluation of the E-Validator software system. A review of the test cases with the lowest level of accuracy was also presented as a part of this discussion. The next chapter describes the conclusions and further work of the study.

Chapter 8

Conclusion and Further work

8.1 Introduction

Chapter 7 described the evaluation of the E-Validator software system. Details of the websites evaluated, the system generated results, and the expert reviews on these websites were presented in the discussion. A total of 8 websites were evaluated by the system and by expert evaluators. The results were compared and an overall accuracy score of 67% was computed as a result. This chapter concludes the results of the research. Also limitations of this research and further work is highlighted.

8.2 Conclusion

Table 7.5 indicates an overall system accuracy of 67%. This measure indicates the level of conformance of the automated method to the manual method. The accuracy of the system generated results was reduced due to lack of comprehensiveness in the system design and occasional errors made by human experts. Further, it is possible to accept the hypothesis of the research and conclude that with regard to the selected heuristics in the study, the heuristic evaluation process can be automated by using a software system.

The automated system can identify the programmed usability issues in web pages in less than 30 seconds. This performance was evident when the automated system was being executed on a Core i-3, 1.8 GHz processor with 4 GB of main memory. Since this is a quite common configuration, it can be concluded that the E-Validator system gives acceptable performance for the given features. Further it is justifiable to say that the system can give much higher performance on a processor with more computing resources such as memory and the number of processors.

One of the limitations of the system is that prior to evaluation web pages has to be saved on disk through a web browser like Mozilla Firefox. Thus it is not possible to evaluate a website directly by connecting to the Internet.

8.3 Further work

The automated system is fully functional without a data storage module indicated in Figure 5.1. This module is required if the system is extended to accommodate a large number of usability errors, to store evaluation results and website information on disk. The already existing error classes can be modified easily, and integrated with the database module. Further, by integrating a database module, the system will be capable of storing evaluation results as well as details of web pages and its related data so that evaluations can be done later.

The error detection can be done using machine learning algorithms. By using such algorithms, it is possible to improve the accuracy of system generated results so that it can assess websites associated with different methods of CSS encoding. For this purpose, supervised machine learning algorithms should be used in this respect.

8.4 Summary

This chapter concluded the work done in the research. It was identified that the automated system can be used as an initial filter to the heuristic evaluation process.

References

[1] D. C. Mowery and T. Simcoe, “Is the Internet a US invention?—an economic and technological history of computer networking,” *Res. Policy*, vol. 31, no. 8, pp. 1369–1387, 2002.

[2] R. Pastor-Satorras, A. Vázquez, and A. Vespignani, “Dynamical and correlation properties of the Internet,” *Phys. Rev. Lett.*, vol. 87, no. 25, p. 258701, 2001.

[3] J. Choi, R. Pearce, D. Poland, B. Thomee, G. Friedland, L. Cao, K. Ni, D. Borth, B. Elizalde, L. Gottlieb, and C. Carrano, “The Placing Task: A Large-Scale Geo-Estimation Challenge for Social-Media Videos and Images,” 2014, pp. 27–31.

[4] A. Dix, J. Finlay, G. D. Abowd, and R. Beale, *Human Computer Interaction*, 3 edition. Harlow, England ; New York: Prentice Hall, 2003.

[5] M. Y. Ivory and M. A. Hearst, “The state of the art in automating usability evaluation of user interfaces,” *ACM Comput. Surv. CSUR*, vol. 33, no. 4, pp. 470–516, 2001.

[6] J. W. Palmer, “Web site usability, design, and performance metrics,” *Inf. Syst. Res.*, vol. 13, no. 2, pp. 151–167, 2002.

[7] M. Matera, F. Rizzo, and G. T. Carughi, “Web Usability: Principles and Evaluation Methods,” in *Web Engineering*, E. Mendes and N. Mosley, Eds. Springer Berlin Heidelberg, 2006, pp. 143–180.

[8] M. Y. Ivory and M. A. Hearst, “The State of the Art in Automating Usability Evaluation of User Interfaces,” *ACM Comput Surv*, vol. 33, no. 4, pp. 470–516, Dec. 2001.

[9] D. Chisnell and J. Redish, *Designing web sites for older adults: Expert review of usability for older adults at 50 web sites*, vol. 1. AARP, 2005.

[10] M. Allen, L. M. Currie, S. Bakken, V. L. Patel, and J. J. Cimino, “Heuristic evaluation of paper-based Web pages: A simplified inspection usability methodology,” *J. Biomed. Inform.*, vol. 39, no. 4, pp. 412–423, Aug. 2006.

[11] J. Nielsen, “Finding usability problems through heuristic evaluation,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1992, pp. 373–380.

[12] B. Shneiderman, “Designing the User Interface.,” *Inc Read. MA*, 1998.

[13] A. Seffah, M. Donyaee, R. B. Kline, and H. K. Padda, “Usability measurement and metrics: A consolidated model,” *Softw. Qual. J.*, vol. 14, no. 2, pp. 159–178, Jun. 2006.

[14] L. L. Constantine, L. A. Lockwood, and L. Wood, “Software for use: A practical guide to the models and methods of usage-centered design,” *SIGCHI Bull.*, vol. 32, no. 1, p. 111, 2000.

[15] B. Shneiderman, *Designing the user interface: strategies for effective human-computer interaction*, vol. 3. Addison-Wesley Reading, MA, 1992.

[16] J. Nielsen, *Usability engineering*. Elsevier, 1994.

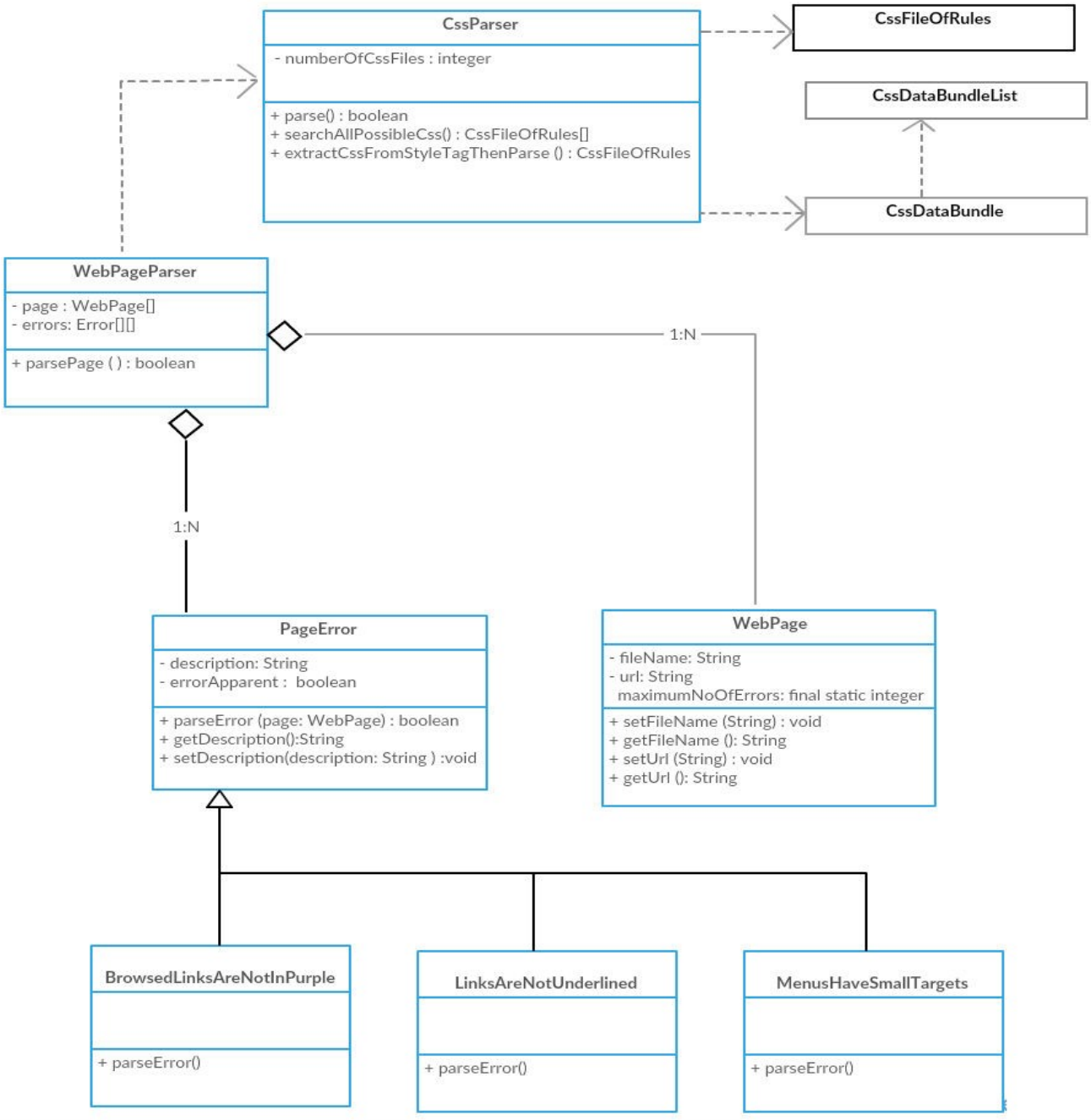
[17] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey, *Human-*

computer interaction. Addison-Wesley Longman Ltd., 1994.
[18] B. Shackel, “Usability-context, framework, definition, design and evaluation,” *Hum. Factors Inform. Usability*, pp. 21–37, 1991.
[19] I. O. for Standardization, *ISO 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Part 11: Guidance on Usability*. 1998.
[20] J. Nielsen and R. Molich, “Heuristic evaluation of user interfaces,” presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 1990, pp. 249–256.
[21] S. L. Smith and J. N. Mosier, *Guidelines for designing user interface software*. Mitre Corporation Bedford, MA, 1986.
[22] R. Molich and J. Nielsen, “Improving a human-computer dialogue,” *Commun. ACM*, vol. 33, no. 3, pp. 338–348, 1990.
[23] T. Carta, F. Paternò, and V. F. De Santana, “Web usability probe: a tool for supporting remote usability evaluation of web sites,” in *Human-Computer Interaction–INTERACT 2011*, Springer, 2011, pp. 349–357.
[24] H. R. Hartson, J. C. Castillo, J. Kelso, and W. C. Neale, “Remote evaluation: the network as an extension of the usability laboratory,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1996, pp. 228–235.
[25] L. Olsina and G. Rossi, “Measuring Web application quality with WebQEM,” *Ieee Multimed.*, no. 4, pp. 20–29, 2002.
[26] L. Olsina, G. Lafuente, and G. Rossi, “E-commerce site evaluation: a case study,” in *Electronic Commerce and Web Technologies*, Springer, 2000, pp. 239–252.
[27] L. Olsina, M. F. Papa, M. E. Souto, and G. Rossi, “Providing automated support for the Web quality evaluation methodology,” in *Fourth Workshop on Web Engineering, at the 10th International WWW Conference, Hong Kong*, 2001, pp. 1–11.
[28] A. Akincilar and M. Dagdeviren, “A hybrid multi-criteria decision making model to evaluate hotel websites,” *Int. J. Hosp. Manag.*, vol. 36, pp. 263–271, 2014.
[29] T. L. Saaty, “The analytic hierarchy process: planning, priority setting, resources allocation,” *N. Y. McGraw*, 1980.
[30] J.-P. Brans, P. Vincke, and B. Mareschal, “How to select and how to rank projects: The PROMETHEE method,” *Eur. J. Oper. Res.*, vol. 24, no. 2, pp. 228–238, 1986.
[31] A. Oztekin, D. Delen, A. Turkyilmaz, and S. Zaim, “A machine learning-based usability evaluation method for eLearning systems,” *Decis. Support Syst.*, vol. 56, pp. 63–73, 2013.
[32] K. Pearson and A. Lee, “On the generalised probable error in multiple normal correlation,” *Biometrika*, vol. 6, no. 1, pp. 59–68, 1908.
[33] S. Weisberg, “Applied Linear RegressionJohn Wiley,” *N. Y.*, p. 283, 1980.
[34] J. R. Quinlan, “Induction of decision trees,” *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.
[35] S. Haykin, *Neural Networks and Learning Machines, 3a ediçao*. Prentice Hall. (Citado na página 37.), 2008.
[36] A. Oztekin, Z. J. Kong, and O. Uysal, “UseLearn: A novel checklist and usability evaluation method for eLearning systems by criticality metric analysis,” *Int. J. Ind. Ergon.*, vol. 40, no. 4, pp. 455–469, 2010.

[37] J. I. Hong, J. Heer, S. Waterson, and J. A. Landay, “WebQuilt: A proxy-based approach to remote web usability testing,” *ACM Trans. Inf. Syst.*, vol. 19, no. 3, pp. 263–285, 2001.
[38] R. Atterer, M. Wnuk, and A. Schmidt, “Knowing the user’s every move: user activity tracking for website usability evaluation and implicit interaction,” in *Proceedings of the 15th international conference on World Wide Web*, 2006, pp. 203–212.
[39] Y.-H. Wu and A. L. P. Chen, “Prediction of Web Page Accesses by Proxy Server Log,” *World Wide Web*, vol. 5, no. 1, pp. 67–88, Mar. 2002.
[40] F. Botella, E. Alarcon, and A. Peñalver, “A new proposal for improving heuristic evaluation reports performed by novice evaluators,” in *Proceedings of the 2013 Chilean Conference on Human-Computer Interaction*, 2013, pp. 72–75.
[41] “Welie.com - Patterns in Interaction Design.” [Online]. Available: http://www.welie.com/. [Accessed: 22-Mar-2016].
[42] C. Alexander, S. Ishikawa, and M. Silverstein, *A pattern language: towns, buildings, construction*, vol. 2. Oxford University Press, 1977.
[43] A. Dingli and S. Cassar, “An intelligent framework for website usability,” *Adv. Hum.-Comput. Interact.*, vol. 2014, p. 5, 2014.
[44] J. Mifsud and A. Dingli, *USEFul: A Framework to Mainstream Web Site Usability Through Automated Evaluation*. LAP LAMBERT Academic Publishing, 2012.
[45] “Web Content Accessibility Guidelines (WCAG) 2.0.” [Online]. Available: https://www.w3.org/TR/WCAG20/. [Accessed: 10-Mar-2016].
[46] “CSS Styling Links.” [Online]. Available: http://www.w3schools.com/css/css_link.asp. [Accessed: 19-Apr-2016].

Appendix A- Detailed Design

The following diagram shows the detailed design of the automated system. Some of the members in classes are hidden for clarity.



Appendix B – Selected Source code

The error classes and the CssParser class can be considered as the major innovative elements of the system implementation. Sample error classes from the source code and the CssParser class in included below.

(1) BrowsedLinksAreNotInPurple Class

The following code fragment is the class definition of the BrowedLinksAreNotInPurple sub-module.

```
package evalidator;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.LinkedList;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
/**
 *
 * @author jeevanthe
 */
public class BrowsedLinksAreNotInPurple extends PageError {
public BrowsedLinksAreNotInPurple(){

super("Some visited links does not appear in purple after
```

```
visiting");
    super.errorApparent=true;
}

public boolean parseError(CssFileOfRules[] cssData,int
numberOfFiles){

        String selectorInList = null;
        String property=null;
        String value=null;
        CssDataBundle cssDataBundle=null;

  outer: for(int i=0;i< numberOfFiles;i++){
           if(cssData[i] != null){
               selectorInList=null;
               cssDataBundle=null;

  for(int j=0;   j<    cssData[i].getNumberOfRules() ;j++){

     if(cssData[i].getData()[j]   != null){
     selectorInList=     cssData[i].getData()
[j].getSelector();      //get the selector from data
                                        //bundle

if(selectorInList.trim().indexOf(",a:visited,")!=-1||
     selectorInList.trim().indexOf("a:visited,")!=-1||
     selectorInList.trim().equals("a:visited")){
     cssDataBundle=cssData[i].getData()[j];
```

```
for(int k=0;k<cssDataBundle.getListLength();k++){

property=cssDataBundle.getSelectorsList()[k].getProperty();

value=cssDataBundle.getSelectorsList()[k].getValue();

if(!(property.equals("color") && value.equals("rgb(153, 0,
153)"))){

errorApparent=true;
return true;
}
                              else errorApparent=false;
                              break outer;
                           }
                        }
                     }
                  }
               }
            }
         return false;
         }
      }
```

(2) CssParser class

The source code for CssParser class is included below.

```
package evalidator;
```

```
import java.io.*;
import com.steadystate.css.parser.CSSOMParser;
import com.steadystate.css.parser.SACParserCSS3;
import java.io.IOException;
import java.io.StringReader;
import java.util.ArrayList;
import java.util.LinkedList;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.select.Elements;
import org.w3c.css.sac.InputSource;
import org.w3c.dom.css.CSSRule;
import org.w3c.dom.css.CSSRuleList;
import org.w3c.dom.css.CSSStyleDeclaration;
import org.w3c.dom.css.CSSStyleRule;
import org.w3c.dom.css.CSSStyleSheet;
/**
 *
 * @author jeevanthe
 */
public class CssParser {
    private int numberOfCssFiles;
    //private int arrayjIndex;

public CssParser(){
numberOfCssFiles=0;
}

public int getNumberOfCssFiles(){
```

```
return numberOfCssFiles;
}

    public void resetCounters(){
numberOfCssFiles=0;
}

public ArrayList<String>
getCssFileListInSavedFolder(WebPage page){

    resetCounters();
    ArrayList<String> cssFiles = new ArrayList<String>();
    String  webPage=
page.getFileName().substring(0,page.getFileName().indexOf("
."));
    File dir = null;
    dir = new File(webPage+"_files");// css savedfolder

    if(dir !=null);

 for (File file : dir.listFiles()) {
 if ( file.getName() != null &&
file.getName().endsWith((".css"))) {
        System.out.println(file.getName());
        cssFiles.add(file.getName());
        numberOfCssFiles++;
       }
  }
   return cssFiles;
```

```
}

//provides an ArrayList for all web pages and css files
    public CssFileOfRules[] searchAllPossibleCss(WebPage
page){

        CssFileOfRules temp=null;//holds css data bundle
        //composite data structure
        CssFileOfRules[] allCssRules = new
CssFileOfRules[20];

     temp = extractCssFromStyleTagThenParse(page);
        int i=0;
        if(temp != null){
             temp.setWebPageName(page.getFileName());
             allCssRules[i]=(temp);
             i++;
        }

        else{
            System.out.println("Web page does not have any
embedded css.");
               }

        ArrayList cssFiles=null;
        cssFiles = getCssFileListInSavedFolder(page);
        CssFileOfRules tempDataResult;

    if(cssFiles!=null){
```

```
    for(int j=0;j<cssFiles.size() ;j++){

    String currentFile =  cssFiles.get(j).toString();

    if(currentFile != null){
    tempDataResult=parse(page.getFileName().substring(0,
    page.getFileName().indexOf(".html"))
+"_files/"+currentFile);

     if( tempDataResult!= null ){
     tempDataResult.setWebPageName(page.getFileName());
     allCssRules[i]=tempDataResult;
     i++;
        }
       }
     }
   }

    return allCssRules;
    }

    //css rules from web page
public CssFileOfRules

extractCssFromStyleTagThenParse(WebPage aPage){

     CssDataBundle[] cssRules = new CssDataBundle[10000];
     String linkData=null;
     CssFileOfRules file = new CssFileOfRules();
```

```
      File input =  new File(aPage.getFileName());
      try{
      Document doc = Jsoup.parse(input,"UTF-
      8","http://www.thisisatest.lk");
             Elements styles=null;
             styles= doc.select("style");
             if(!styles.isEmpty()){
                 int i=0;
                 linkData=styles.first().html();
                 InputSource source = new InputSource(new
                 StringReader(linkData));

      CSSOMParser parser = new CSSOMParser(new
      SACParserCSS3());
                 // parse and create a stylesheet
                 //composition
      try{
CSSStyleSheet stylesheet =

parser.parseStyleSheet(source,null,null);

     //ANY ERRORS IN THE DOM WILL BE SENT TO STDERR HERE!!
     // now iterate through the dom and inspect.

CSSRuleList ruleList = stylesheet.getCssRules();
try{
   file = new CssFileOfRules();
     file.setNumberOfRulesInFile(ruleList.getLength());
```

```
                for (; i < ruleList.getLength(); i++)
               {
                 CSSRule rule = ruleList.item(i);
                 if (rule instanceof CSSStyleRule)
                 {
                 CSSStyleRule styleRule=(CSSStyleRule)rule;
                 CSSStyleDeclaration styleDeclaration =
                 styleRule.getStyle();
/**********************rule list for this selector
**********************/
CssDataBundleList[] databundlelist=new
CssDataBundleList[4000];int j;
CssDataBundleList temp;

for ( j = 0; j < styleDeclaration.getLength(); j++)
     {
    String property = styleDeclaration.item(j);
     temp= new
CssDataBundleList(property,styleDeclaration.getPropertyCSSV
alue(property).getCssText(),styleDeclaration.getPropertyPri
ority(property));

     databundlelist[j] = temp;
     }//add the selector name and list
     CssDataBundle tempDataBundle = new

CssDataBundle(styleRule.getSelectorText(),databundlelist);

tempDataBundle.setListLength(styleDeclaration.getLength());
```

```
      cssRules[i]= tempDataBundle;
      }// end of StyleRule instance test
      } // end of ruleList loop
      cssRules[i] = new CssDataBundle("####",null);
      }catch(IndexOutOfBoundsException e){
}
      }
      catch (Exception e)
              {
                 System.err.println ("Error: " + e);
               }
               }
               }
             catch(Exception e ){}
File.setData(cssRules);

           return file;
    }
 //css rules from css file
    public CssFileOfRules parse(String cssfile)
     {//main data structure with all the css rules
CssDataBundle[] cssRules = new CssDataBundle[10000];

CssFileOfRules file =new CssFileOfRules();
PrintStream ps = null;
         int i=0;

         boolean rtn = false;
```

```
        try
        {

// cssfile accessed as a resource, so must be in the pkg
(in src dir).

InputStream inStream = new FileInputStream(cssfile);

InputSource source = new InputSource(new

InputStreamReader(inStream,      "UTF-8"));
                CSSOMParser parser = new CSSOMParser(new
SACParserCSS3());
                // parse and create a stylesheet
composition
                CSSStyleSheet stylesheet =
parser.parseStyleSheet(source,null,null);

                //ANY ERRORS IN THE DOM WILL BE SENT TO
STDERR HERE!!
                // now iterate through the dom and inspect.

CSSRuleList ruleList = stylesheet.getCssRules();

CssDataBundleList[] dataList=new CssDataBundleList[4000];
                CSSStyleRule styleRule;

                CSSStyleDeclaration styleDeclaration;
                 if(file != null)
```

```
file.setNumberOfRulesInFile(ruleList.getLength());
                for ( ; i < ruleList.getLength(); i++)
                {
               CSSRule rule = ruleList.item(i);
               if (rule instanceof CSSStyleRule)
                     {
                              styleRule=(CSSStyleRule)rule;
                              styleDeclaration =
                              styleRule.getStyle();

                         int j;
     for (j = 0; j < styleDeclaration.getLength(); j++)
                                  {
               String property = styleDeclaration.item(j);
               dataList[j]=new

CssDataBundleList(property,styleDeclaration.getPropertyCSSV
alue(property).getCssText(),styleDeclaration.getPropertyPri
ority(property));

 }//add the selector name and list

CssDataBundle temp= new CssDataBundle
(styleRule.getSelectorText(),dataList);

     temp.setListLength(styleDeclaration.getLength());
                       cssRules[i]=temp;
```

```
                  }// end of StyleRule instance test
                } // end of ruleList loop
                rtn = true;
            }
            catch (IOException ioe)
            {
                System.err.println ("IO Error: " + ioe);
            }
            catch (Exception e)
            {
                System.err.println ("Exception at Css
parser: " + e);

            }
            finally
            {
                if (ps != null) ps.close();
            }
          file.setData(cssRules);
               return file;
   }
}
```

Appendix C - Questionnaire

Heuristic evaluation of websites

Dear Colleague,

I am Sulakshan Wijesundare, a postgraduate student in the MSc in IT program of the Faculty of Information Technology, University of Moratuwa, and as part of my studies I am conducting a research on automating the heuristic evaluation method for usability of software systems.

For this study, I have developed a software tool which can identify certain usability problems in websites. In order to validate this tool I am collecting evaluation results of a websites from experts including UX engineers, QA engineers, architects, etc.

Please read the evaluation instructions carefully and provide your assessment on the selected websites included in the form.

Instructions:

The evaluation of the websites is performed using a modified heuristic evaluation method. For this method, two user profiles are considered, which have four types of ratings.

Description of rated parameters

Age : User's age
Ability : Indicates cognitive and physical ability
Aptitude : Internet and computer literacy.
Attitude : positive (forward looking, risk-taking, and experimental) or negative (fearful or

diffident), confidence levels, and emotional need for support

The evaluation should be performed by assuming that a typical user belongs to either of the following two profiles.

Profile 1:

Age : 73
Ability : little above average (55%)
Aptitude : below average (40%)
Attitude : positive (60%)

Profile 2:

Age : 52
Ability : High (90%)
Aptitude : High (90%)
Attitude : positive (80%)

The steps in this evaluation method are as follows.

1. Explore the web pages.
2. Record observations in the web page, with regard to the following heuristics.

Heuristic 1. Use conventional interaction elements.
Evaluation questions

1.1 Does the site use standard treatments for links?
1.2 Is link treatment the same from section to section within the site?

Heuristic 2. Make click-able items easy to target and hit.

Evaluation questions

2.1 Is there enough space between targets to prevent hitting multiple or incorrect targets?

3. Using the questions included in step 2, identify usability issues which can affect a typical user belonging to Profile 1 or Profile2.

Thank you for your assistance.

Consider the following usability errors (a) browsed links are not in purple (b) links are not underlined (c) links are not dynamically underlined (d) Menus have only small area to click on

[ ] (a)-(d) applies for any website user
[ ] (a) - (c) applies mainly for older adults (adults above 50)
[ ] (d) applies to all users

url: http://edition.cnn.com/EVENTS/1996/year.in.review/ Q: What are the usability issues in this page with regards to the above heuristics?

[ ] Browsed links does not appear in purple
[ ] There are no usability problems

url: http://edition.cnn.com/2016/01/14/us/possible-powerful-supernova/ Q: What are the usability issues in this page?

[ ] Menues have small targets to click on
[ ] Links are not dynamically underlined
[ ] Links are not underlined
[ ] Browsed links does not appear in purple

url: http://www.barnesandnoble.com/b/textbooks/_/N-8q9 Q: What are the usability issues in this page?

[ ] Links are not dynamically underlined
[ ] Links are not underlined
[ ] Browsed links does not appear in purple
Menus have small target area to click on

url: http://www.bbc.com/future/story/20160124-are-paper-books-really-disappearing Q: What are the usability issues in this page?
[ ] Links are not dynamically underlined
[ ] Links are not underlined
[ ] Browsed links does not appear in purple
[ ] Menus have small target area to click on

url: http://www.bbc.com/news/technology-35706730 Q: What are the usability issues in this page?

[ ] Links are not dynamically underlined
[ ] Links are not underlined
[ ] Browsed links does not appear in purple
[ ] Menus have small target area to click on

url:http://news.nationalgeographic.com/news/2014/01/140103-music-lessons-brain-aging-cognitive-neuroscience/ Q: What are the usability issues in this page?

[ ] Links are not dynamically underlined
[ ] Links are not underlined
[ ] Browsed links does not appear in purple
[ ] Menus have small target area to click on

url: https://www.longtermcarelink.net/a8profiles.htm Q: What are the usability issues in this page?

[ ] Links are not dynamically underlined
[ ] Links are not underlined
[ ] Browsed links does not appear in purple
[ ] Menus have small target area to click on

url: http://www.hci.midmaas.com Q: What are the usability issues in this page?

[ ] Links are not dynamically underlined
[ ] Links are not underlined
[ ] Browsed links does not appear in purple
[ ] Menus have small target area to click on