Exploring Automatic CSS Accessibility Evaluation

Amaia Aizpurua, Myriam Arrue, Markel Vigo, and Julio Abascal

Laboratory of Human-Computer Interaction for Special Needs, Informatika Fakultatea, University of the Basque Country, Manuel Lardizabal 1, 20018 Donostia, Spain
amaia.aizpurua@ehu.es, myriam@si.ehu.es, markel@si.ehu.es, julio@si.ehu.es

Abstract. Automatic evaluation tools are crucial for helping designers to develop accessible web content. However, most of the existing automatic tools are focused on evaluating the accessibility of (X)HTML code and do not consider style sheets. CSS provides mechanisms for separating content from display which is a requirement for accessible web documents. Although the use of CSS has become essential, sometimes its powerful mechanisms and functionalities may lead to a misuse. This paper presents an accessibility evaluation framework for verifying the correct application of CSS techniques. For this purpose, a flexible accessibility evaluation framework was selected and adapted to incorporate CSS test cases defined in WCAG 2.0. As a result of a detailed analysis, 6 different types of test cases were identified and a total number of 92 test cases were accommodated into the framework. This process has lead to obtain a flexible framework which performs more comprehensive automatic evaluations.

1 Introduction

One of the most important issues for improving accessibility of web content is to ensure that its structure is separated from the presentation. According to Paciello [1] *"by separating presentation from structure, specialized technologies used by people with disabilities can easily interpret the structure and render it properly to users"*. Therefore, web designers should use appropriate mechanisms for this purpose. In this sense, the World Wide Web Consortium (W3C)[1] recommends using Cascading Style Sheets (CSS)[2].

In recent years, the use of CSS has significantly increased. Designers are supposed to consider structure and presentation as two different aspects of web development by properly structuring web documents using only (X)HTML mark-up and defining its presentation in a separated CSS file. This process facilitates the development of web documents which are more accessible, easier to maintain, possible to navigate with screen readers, better indexed by search engine [2], etc. In addition, the use of CSS has other advantages regarding accessibility [3]:

[1] http://www.w3.org/
[2] http://www.w3.org/Style/CSS/

M. Gaedke, M. Grossniklaus, and O. Díaz (Eds.): ICWE 2009, LNCS 5648, pp. 16–29, 2009.
© Springer-Verlag Berlin Heidelberg 2009

- Allows designers to control the spacing, alignment, positioning, etc. of components without using (X)HTML structural elements for stylistic effects.
- Reduces required download time by preventing the use of images for positioning content such as invisible images.
- Provides control over font size, color and style avoiding the use of images to represent text.
- Allows users to override designers' styles.
- Provides techniques for including orientation mechanisms such as numbers or contextual clues, etc.

However, the use of CSS does not guarantee accessibility of web documents. For instance, when the designer defines rigid styles may disturb with the ones defined in users' personal style sheets. Therefore, mechanisms for evaluating the use of CSS are needed in order to ensure accessibility of web documents.

Web Content Accessibility Guidelines 1.0 (WCAG 1.0) [4] specify several CSS techniques [5] which are necessary for developing accessible web documents. In addition, the new version of this set of guidelines, WCAG 2.0 [6], defines evaluation procedures for ensuring CSS techniques [7] are correctly used. Automatic tools able to evaluate the correct use of these CSS techniques would be very useful as they are an essential help for web developers. Ivory and Hearst [8] highlight some advantages of using automatic tools:

- Evaluation process becomes less time demanding and consequently there is a reduction in costs.
- The detected errors are more consistent.
- Possibility for predicting the effort needed in the process in terms of time and economical costs.
- Spreads evaluation scope as it is possible to analyse diverse aspects of the interface in less time.
- Facilitates the process to evaluators with little experience in usability and accessibility evaluation.
- Facilitates comparing the adequacy of different user interface design alternatives.
- Facilitates incorporation of evaluation tasks during the development process.

Many automatic web accessibility evaluation tools exist though most of them focus on (X)HTML mark-up evaluation. The existing automatic evaluation tools for CSS are based on simple syntax verifications such as checking that relative units of measurement are used. The main objective of this work is to extend the evaluation of style aspects by adapting a flexible accessibility evaluation framework to incorporate WCAG 2.0 CSS techniques. For this purpose, a thorough analysis of CSS techniques has been done. It has been useful in order to detect similarities of these techniques with respect of (X)HTML ones. The paper is structured as follows: section 2 is devoted to the analysis of the CSS techniques evaluation coverage of existing accessibility evaluation tools; section 3 describes the analysis process of CSS techniques proposed in WCAG 2.0, in this process test cases are identified and classified; section 4 presents the evaluation process of CSS performed by the adapted evaluation framework; section 5 points out the limitations of current CSS evaluation procedures and conclusions are drawn in section 6.

2 Related Work

There are numerous accessibility evaluation tools. Diverse ways for classifying them can be found in the literature [8, 9, 10, 11]. For instance, they can be classified in two groups, remote or local, based on the location they are executed, on the local computer or on a server respectively. Other [12] studied the coverage of several evaluation tools in all the stages of the development process: specification, design, implementation and post-implementation.

However, the most relevant aspect of tools considered for this research work is their coverage of CSS techniques when evaluating web accessibility. W3C-Web Accessibility Initiative (WAI)[3] maintains a complete list of available evaluation tools[4] and it is possible to search for tools with specific characteristics, for instance their coverage of CSS techniques. According to this list there are 18 tools which meet the specified search criteria: they are free software and evaluate the accessibility of CSS based on WCAG 1.0. Some of them are specific tools which are focused on CSS evaluation while others are accessibility general tools which check some CSS techniques in addition to (X)HTML techniques. Nevertheless, most of them only incorporate a few aspects of CSS; for instance the W3C CSS Validation Service[5] is a specific tool which checks style sheets against the grammar, properties and values defined in the corresponding CSS specification. The CSS Analyser[6] tool by Juicy Studio checks the validity of the given CSS as well as the color contrast and the use of relative units. Hera[7] is an online general accessibility tool based on techniques defined in WCAG 1.0 and also checks some CSS related techniques from checkpoints such as "*3.2: Create documents that validate to published formal grammars*", "*3.3: Use style sheets to control layout and presentation*" or "*3.4: Use relative rather than absolute units in markup language attribute values and style sheet property value*". The evaluated aspects regarding CSS are basically related to checking the use of style sheets in the evaluated web page, validating its syntax against the corresponding formal grammar and verifying the use of relative units.

The aim of this research work is to extend the CSS techniques verified by automatic tools incorporating them into general accessibility evaluation tools in order to perform more comprehensive evaluations. There are several accessibility evaluation tools which are interesting to incorporate new techniques as they do not have to be recoded. They are based on flexible guidelines definition language which provides mechanisms for specifying testing cases. AccessEnable [13] and Kwaresmi [14, 15] are two examples. AccessEnable is a commercial tool which is not longer supported, whereas the GDL guidelines definition language used by Kwaresmi has been recently revised [16]. The locally executable version of TAW[8] offers several functionalities for defining personalized tests but they are limited to some regular expressions not sufficiently complete for accommodating CSS techniques testing

[3] http://www.w3.org/WAI/
[4] http://www.w3.org/WAI/ER/tools/Overview
[5] http://jigsaw.w3.org/css-validator/
[6] http://juicystudio.com/services/csstest.php#csscheck
[7] http://www.sidar.org/hera/
[8] http://www.tawdis.net

cases. More recently, Leporini et al. [17] have developed a new evaluation tool MAGENTA which is based on Guidelines Abstraction Language, GAL. Abascal et al. [18] proposed in 2004 the evaluation tool EvalAccess. Recently, the language for guidelines definition used by the tool, UGL (Unified Guidelines Language), was extended and revised in order to accommodate different types of sets of guidelines [19]. This framework has been selected for this research work since its proved flexibility. As far as the evaluation logic of evaluation tools is concerned, there is a growing trend towards using XML technology. XML query languages are very powerful due to their expressiveness and flexibility. Takata et al. [20] proposed a pseudo-XQuery language for accessibility evaluation purposes and XPath/XQuery sentences are defined to check WCAG guidelines in [21]. The use of this technology makes the implementation of the evaluation logic easier and, as a result, many lines of source code are saved.

3 Incorporating CSS into Accessibility Evaluation Process

The process described by Vanderdonckt in [22] has been taken as the basis of this work. The principal steps of the process are the following:

- Gather, combine and compile accessibility guidelines from different sources in order to develop a complete set of guidelines.
- Classify and order the obtained set of guidelines in one organizational framework.
- Develop a computational representation of the set of guidelines so guidelines can be specified and manipulated by software components.

As it can be observed, this process is focused on a complete analysis of sets of guidelines in order to obtain a computational representation flexible enough to accommodate different types of guidelines. In this case, different sets of guidelines have been analysed in order to detect the CSS techniques defined. This process is not simple as guidelines may have different formats, may contain different information and may be described with different level of detail [23, 24]. However, it has been simplified as the recently released WCAG 2.0 determines the exact evaluation procedure for each CSS technique. Depending of their evaluation procedure CSS techniques can be classified in three groups:

- Automatic tests: these problems should not require human judgment to check their validity. Therefore, their evaluation can be completely automatic.
- Manual or semi-automatic tests: human judgment is necessary to check potential problems associated to particular fragments of code implementing the page.
- Generic problems: human judgment is necessary to check potential problems that cannot be associated to any code fragments.

EvalAccess framework can manage these three types of tests. However, the most important types of tests are those that its evaluation can be totally or partially automated.

As a result of the analysis, 22 CSS techniques have been detected. 13 of those can be automatically or semi-automatically evaluated whereas 8 specify generic problems

independent of any CSS code fragment and require human judgment. In addition, there is one CSS technique[9] which has no available tests. For the automatic and semi-automatic techniques a number of 92 test cases have been identified. Those test cases can be classified in 6 different types shown in Table 1.

Table 1. The detected different types of test cases for CSS techniques in WCAG 2.0

Id.	Test case name	Description	Example
1	Selector warning	Using a selector may cause accessibility problems and have to be tested manually	*WCAG 2.0 – C15 technique* `A :focus`
2	Property warning	Using a property may cause accessibility problems and have to be tested manually	*WCAG 2.0 – C25 technique* `*{color, background-color}`
3	Determined value	The value of a property has to be one of some specifically defined	*WCAG 2.0 – C13 technique* `*{font-size: xx-small, x-small, small, medium, large, x-large, xx-large, larger, smaller}`
4	Determined Part of Value	The value of a property must contain a determined value	*WCAG 2.0 – C12 technique* `*{font-size: *%}`
5	Value between two values	The value of a property must be between to values	*WCAG 2.0 – C21 technique* `P{line-height: 150% - 200%}`
6	Avoid determined part of value	Avoid a determined value for a property	*WCAG 2.0 – C20 technique* `DIV {width: cm, mm, in, pt, pc, px }`

The detected test cases can be easily represented by UGL, the guidelines specification language used by EvalAccess. This ensures a straightforward process for accommodating the new test cases into the evaluation framework simply by defining each one and incorporating this definition in UGL to the repository of EvalAccess. However, each test case type requires the specification of one XQuery template. The evaluation engine of EvalAccess will match the UGL document of each test case with the corresponding XQuery template. These templates will be completed with the necessary information contained in the UGL document. These completed XQuery templates will be those directly applicable evaluation queries which will be used by the evaluation engine.

[9] http://www.w3.org/TR/WCAG20-TECHS/css.html#C18

Figure 1 shows the XQuery template corresponding to test case no. 6 completed with the necessary information for the evaluation of the test case included as an example.

```
let $r:=//rule return(
if($r/selector='div') then
(let $p:= $r/declaration_block/declaration/property return(
if($p='width') then(
for $v in $s/descendant::value return(
if( (contains($v/text(), 'cm') or contains($v/text(), 'mm') or
contains($v/text(), 'in') or contains($v/text(), 'pt') or
contains($v/text(), 'pc') or contains($v/text(), 'px'))) then
('warn')
else ())) else ())) else ())
```

Fig. 1. XQuery template for CSS test-case no. 6 "*Avoid determined part of value*"

Note that some details of the XQuery sentence related to the results are omitted in order to enhance readability.

4 Evaluation Process of CSS Techniques

Figure 2 depicts the evaluation process of CSS techniques included in EvalAccess framework. Each component block in Figure 2 is described below:

1. The CSS Code Retriever obtains the content of a style sheet from the WWW. The obtained CSS code is then converted into XML. This pre-processing of CSS code is necessary since EvalAccess framework is prepared to evaluate (X)HTML code parsed into XML. For this purpose, a XML Schema for parsing CSS code has been developed.

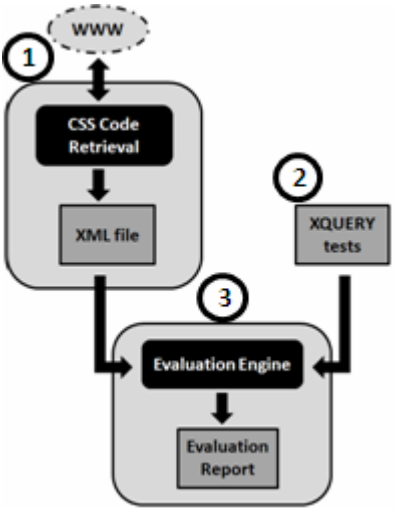

Fig. 2. Evaluation process included in EvalAccess for verifying CSS techniques

2. The necessary XQuery templates are matched and completed with the information contained in UGL. This process leads to obtain all the XQuery sentences to be applied in the evaluation process.
3. The code of the style sheet in XML format is evaluated against the XQuery sentences. As a result, detailed evaluation report, which contains information regarding errors, warnings, etc., is obtained. Since reports are formatted according to a specific XML Schema, they can be also exploited by external applications.

Figure 3 shows the defined XML Schema for representing the content of style sheets. As it can be appreciated, style sheets consist of a set of style rules which contains a selector and a declaration block. The last one will gather the attributes and the corresponding values that are defined for the selector element.

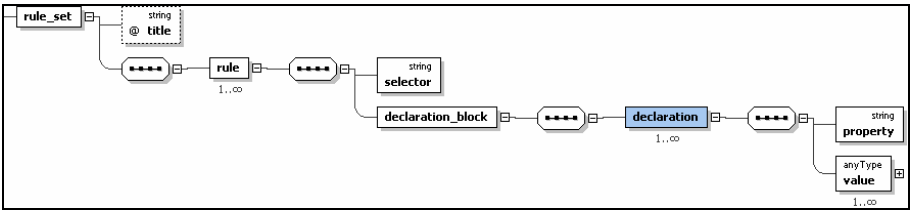

Fig. 3. XML Schema developed for parsing CSS code into XML

4.1 Example of the Evaluation Process

This section presents a detailed description of each necessary step in the evaluation process. For this purpose, a CSS code fragment has been selected as an example for illustrating the developed evaluation process. Figure 4 shows the CSS code fragment.

```
div{
      width: 350px;
      margin: 1em 0 1em 0;
}
#tag p{
      text-align: justify;
      font-size:10px;
      font-weight: bold;
      background-color: #FFFFFF;
}
```

Fig. 4. An example of CSS code

As mentioned above, EvalAccess framework transforms the CSS code into XML based on the developed XML Schema. The XML file corresponding to the example CSS code (Figure 4) can be found in Figure 5.

```
<?xml version="1.0"?>
<css xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file:///c:/schemaCSS.xsd">
     <rule_set title="C:\style">
        <rule>
                <selector>div</selector>
                <declaration_block>
                        <declaration>
                                <property>width</property>
                                <value>350px</value>
                        </declaration>
                        <declaration>
                                <property>margin</property>
                                <value>1em</value>
                                <value>0</value>
                                <value>1em</value>
                                <value>0</value>
                        </declaration>
                </declaration_block>
        </rule>
        <rule>
                <selector>#tag p</selector>
                <declaration_block>
                        <declaration>
                                <property>text-align</property>
                                <value>justify</value>
                        </declaration>
                        <declaration>
                                <property>font-size</property>
                                <value>10px</value>
                        </declaration>
                        <declaration>
                                <property>font-weight</property>
                                <value>bold</value>
                        </declaration>
                        <declaration>
                                <property>background-color</property>
                                <value>#FFFFFF</value>
                        </declaration>
                </declaration_block>
        </rule>
     </rule_set>
</css>
```

Fig. 5. XML representation of the CSS code in Figure 4

As it can be observed, the CSS code is analysed in order to detect the selectors and all the rules in terms of attributes and values applied to them. This information is inserted in a XML file. This XML transformation facilitates the application of the CSS test cases defined in the evaluation framework.

The 92 CSS test cases identified in WCAG 2.0 document are stored in UGL format in a repository. In this way, incorporation of new test cases or new versions of existing ones is quite straightforward.

Next figures, Figure 6, 7 and 8, show the UGL definition of different CSS test cases applied in the evaluation process and the generated XQuery sentences based on

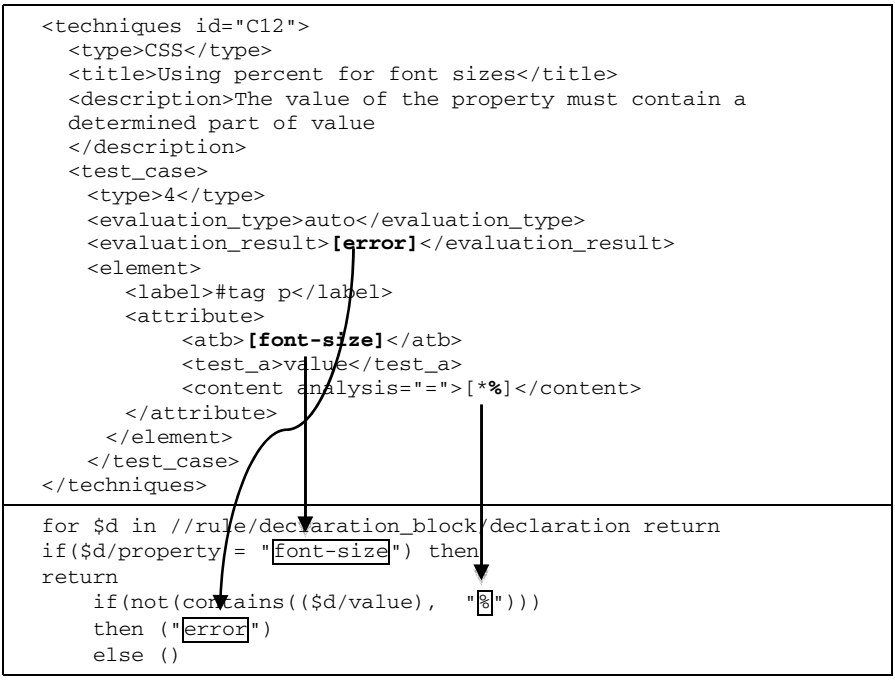

Fig. 6. One test case for verifying WCAG 2.0 C12 CSS technique specified in UGL and the corresponding XQuery sentence

appropriate XQuery templates directly applied to the XML file containing the CSS code fragment.

Figure 6 contains the UGL specification of one test case for CSS technique C12 defined in WCAG 2.0. All the necessary information for completing the corresponding XQuery template is included in this specification. UGL provides mechanisms for determining the correspondence of one CSS test case with its XQuery template (*type* attribute of the *test_case* element). In this case, the corresponding XQuery template is the one defined for CSS test case type no.4 *"Determined Part of Value"* (see Table 1).

Figure 7 shows the UGL file and the generated XQuery sentence for the evaluation of one test case of WCAG 2.0 C19 CSS technique. The XQuery sentence is generated by including the necessary data in the XQuery template corresponding to CSS test case type no.3 *"Determined value"* (see Table 1).

Figure 8 shows UGL specification and XQuery sentence for verifying one of the test cases of WCAG 2.0 C25 CSS technique. XQuery sentence is generated by including the necessary data in the XQuery template corresponding to CSS test case type no.2 "*Property warning*" (see Table 1). In this case, a warning will be created if the XQuery sentence is proved to be true as the test case is of semi-automatic type. This is defined in UGL by *evaluation_result* attribute of *test_case* element.

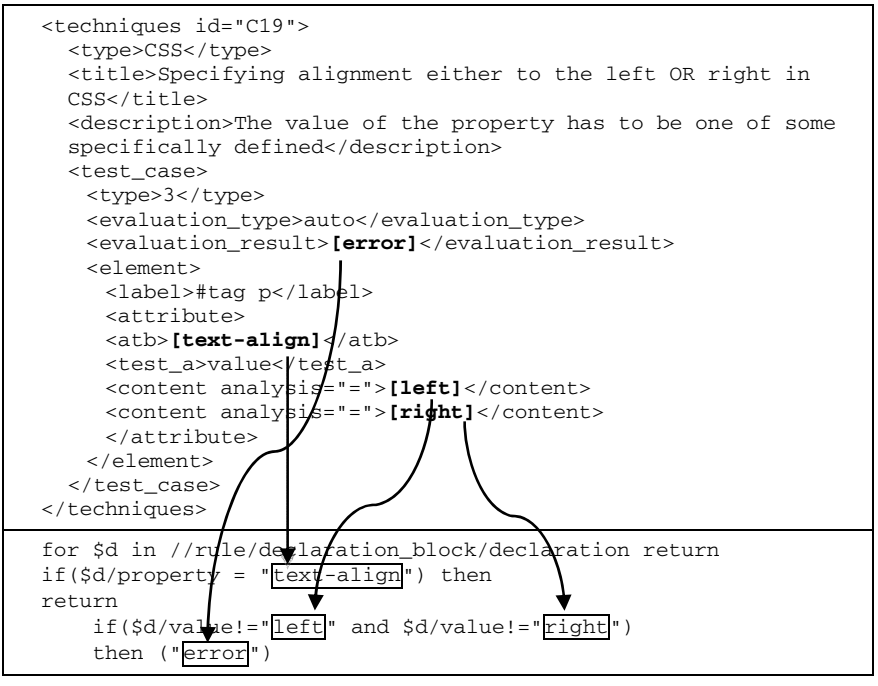

Fig. 7. One test case for verifying WCAG 2.0 C19 CSS technique specified in UGL and the corresponding XQuery sentence. CSS test-case no. 3 "*Determined value*"

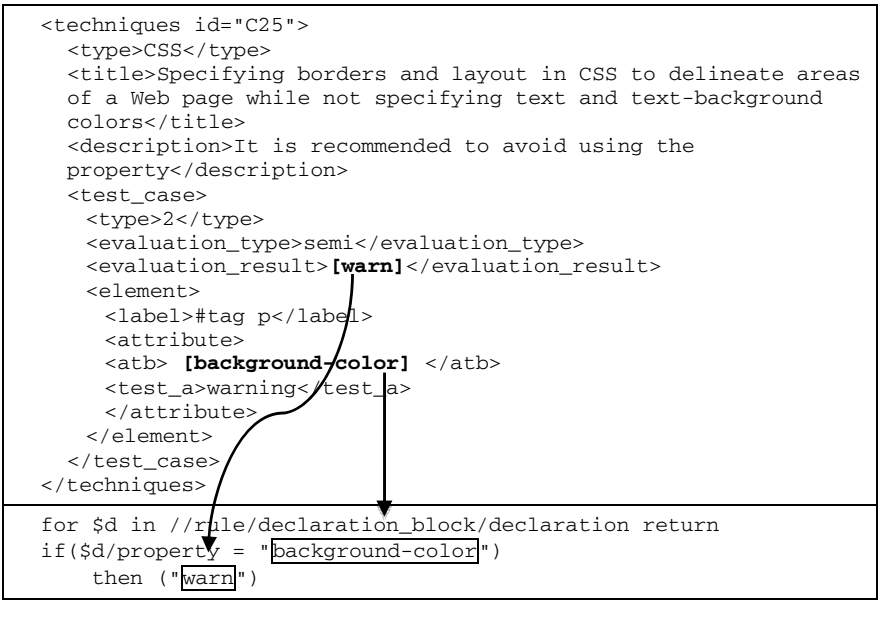

Fig. 8. One test case for verifying WCAG 2.0 C25 CSS technique specified in UGL and the corresponding XQuery sentence

Table 2. Evaluation results of the example CSS code fragment based on the described three test cases

Type	Selector	Attribute	Description	Technique Id	CSS code
Error	#tag p	font-size	The value of the property must contain a determined part of value	C12	`font-size:10px`
Error	#tag p	text-align	The value of the property has to be one of some specifically defined	C19	`text-align: justify`
Warning	#tag p	background-color	It is recommended to avoid using the property	C25	`background-color: #FFFFFF`

UGL specification of CSS test cases also provides useful information for creating the evaluation results report as the description and title of the test case is also available. The results obtained evaluating the example CSS code fragment according to the described three test cases are described in Table 2.

5 Limitations of CSS Evaluation

The main advantage of using style sheets is that they allow the separation of content from design, as recommended by the W3C. Separating markup and presentation is a crucial part of universal design paradigm. This architectural principle is the key for the evolution of the web in a wide range of aspects, such as accessibility, device independence, ubiquity and so on. Currently, it would be very difficult to fulfill the WCAG accessibility guidelines without using CSS.

However, CSS provides powerful mechanisms and functionalities which can lead to a misuse. In this sense, some practices may lead to confuse specialized technologies used by people with disabilities. For instance, the use of misusing (X)HTML structural elements for its expected visual effects, such as the TABLE element which is often used for stylistic purposes like positioning or alignment. On the contrary, style sheets may be used to imitate proper (X)HTML Markup. Elements such as headings, paragraphs and inline elements (STRONG, EM, etc) are sometimes replaced with inappropriate tags which are styled to simply look like markup elements. In order to avoid confusing specialized technologies, it is crucial to verify that CSS techniques are applied appropriately.

Even though this paper describes a useful evaluation framework for CSS techniques, there are several issues which have to be necessarily considered in order to perform comprehensive evaluations. For example, the presented framework evaluates the accessibility of a style sheet itself but it is more interesting to evaluate the result of applying the styles on a specific web page. In addition there are some

aspects of CSS which make more difficult to foresee the final display of a web page, but they should be considered for ensuring accessibility of web documents:

- **Inheritance**[10]**.** CSS allows some properties applied to determined elements, to be inherited from those specified for the parent elements. Although all CSS properties cannot be inherited, the latter CSS specification introduced the *inherit* property value. This value allows the property to be inherited from a parent element in the document tree.
- **Cascading**[11]**.** Style sheets may belong to different agents: author, user and user-agent. The cascade is the property which allows having multiple styles from different sources merged together into one definitive style. It consists of a set of rules to determine the interaction among conflicting styles from different origins. Conflicts among styles happen when for the same element in a document a determined property is assigned contradictory values by different style sheets. Priority levels have been determined in order to solve these conflicts. They are based on three main factors: weight and origin, specificity of selectors and order of appearance. However, there are some mechanisms which can override the established priorities such as *!important* style rules, *@import* statement. Therefore, it is a complex task to foresee which style rules will be finally applied to the document.
- **Media selection**[12]**.** Media Types allow specifying how documents will be presented on different media, such as speech synthesizers, braille devices, printers, etc. The design of a web page to be displayed on a normal desktop screen may not be suitable for a printer, or a handheld device. There are several media types but by default, style sheets apply to all media types. Different ways can be used to make styles apply only to specific media types. The most commonly used methods are the use of the media attribute of the link or style tag, and the *@media* rule. Most of the styles are available to all media types, but some CSS properties are only designed for certain media. For example, the *font-size* style property does not make any sense in speech media. This means that the rules of the style sheets must be applied depending on the selected media.
- **Browser implementation differences**[13]**.** Although most browsers support style sheets, not all of them provide the same level of implementation[14]. Moreover, there are implementation differences among versions of the same browser. There are several mechanisms[15] to solve the CSS related browser bugs. Nevertheless, if those solutions are not applied and a design for a given web page is made for a determined browser, the content of the page can be inaccessible for persons using other browsers.

All these aspects should be considered in order to perform a more adaptive evaluation but it requires gathering more information about final users' environment, such as the

[10] http://www.w3.org/TR/CSS21/cascade.html#inheritance
[11] http://www.w3.org/TR/CSS21/cascade.html#cascade
[12] http://www.w3.org/TR/CSS21/media.html
[13] http://www.webreference.com/authoring/style/sheets/browser_support/
[14] http://www.quirksmode.org/css/contents.html
[15] http://websitetips.com/css/solutions/

browser model and version, access device used, existence of user defined style sheets, etc. Otherwise, evaluating accessibility of CSS for all possible interaction schemas becomes an excessively complex task.

6 Conclusions

In this paper we have presented a framework to evaluate accessibility of style sheets according to the CSS techniques specified in WCAG 2.0. For this purpose, a detailed analysis of CSS techniques has been performed. The framework itself has not been developed from scratch since one flexible accessibility evaluation framework was selected to accommodate the new CSS techniques. This allows extending the efficiency of the framework so that more comprehensive accessibility evaluation can be performed.

Unified Guidelines Language (UGL) is the basis of the framework. The use of this language guarantees that new CSS techniques will be easily incorporated into the framework. A total number of 92 CSS test cases have been defined in UGL and incorporated to the framework for their automatic verification.

This work involves an important step towards considering the web design in the accessibility evaluation process. However, the proposed framework only deals with the evaluation of CSS files and it does not consider some significant aspects inherent to the use of style sheets such as inheritance, cascading, differences in browser implementation, etc.

Comprehensive accessibility evaluations involve considering more aspects than only the CSS or the (X)HTML code. In this sense, it is necessary to predict the resulting display of combining an (X)HTML file with the applicable style sheets in a determined context of use (specific browser and version, access device, users' preferences, etc.). Nevertheless, this is a complex task and future work will be focused on trying to find better solutions in order to improve the accessibility evaluation process.

References

1. Paciello, M.G.: Web Accessibility for People with Disabilities. CMP books (2000)
2. Pemberton, S.: Accessibility is for Everyone. ACM Interactions 10(6), 4–5 (2003)
3. Jacobs, I., Brewer, J. (eds.): Accessibility Features of CSS. W3C Note, (August 4, 1999), http://www.w3.org/TR/CSS-access
4. Chrisholm, W., Vanderheiden, G., Jacobs, I. (eds.): Web Content Accessibility Guidelines 1.0, W3C Recommendation (May 5, 1999), http://www.w3.org/TR/WCAG10/
5. Chisholm, W., Vanderheiden, G., Jacobs, I. (eds.): CSS Techniques for Web Content Accessibility Guidelines 1.0, W3C Note (November 6, 2000), http://www.w3.org/TR/WCAG10-CSS-TECHS/
6. Caldwell, B., Cooper, M., Reid, L.G., Vanderheiden, G. (eds.): Web Content Accessibility Guidelines (WCAG) 2.0, W3C Recommendation (December 11, 2008), http://www.w3.org/TR/WCAG20/
7. Caldwell, B., Cooper, M., Reid, L.G., Vanderheiden, G. (eds.): Techniques for WCAG 2.0. CSS Techniques for WCAG 2.0, W3C Working Group Note (December 11, 2008), http://www.w3.org/TR/WCAG20-TECHS/css.html

8. Ivory, M.Y., Hearst, M.A.: The state of art in automating usability evaluations of user interfaces. ACM Computing Surveys 33(4), 470–516 (2001)
9. Ivory, M.Y., Mankoff, J., Le, A.: Using Automated Tools to Improve Web Site Usage by Users with Diverse Abilities. Information Technology and Society 1(3), 195–236 (2003)
10. Brajnik, G.: Comparing accessibility evaluation tools: a method for tool effectiveness. Universal Access in the Information Society 3(3-4), 252–263 (2004)
11. Abou-Zahra, S. (ed.): Selecting Web Accessibility Evaluation Tools (2006), http://www.w3.org/WAI/eval/selectingtools
12. Xiong, J., Farenc, C., Winckler, M.: Analyzing Tool Support for Inspecting Accessibility Guidelines During the Development Process of Web Sites. In: Weske, M., Hacid, M.-S., Godart, C. (eds.) WISE Workshops 2007. LNCS, vol. 4832, pp. 470–480. Springer, Heidelberg (2007)
13. Brinck, T., Hermann, D., Minnebo, B., Hakim, A.: AccessEnable: A Tool for Evaluating Compliance with Accessibility Standards. In: Automatically Evaluating the Usability of Web Sites, CHI Workshop (2002)
14. Beirekdar, A., Vanderdonckt, J., Noirhomme-Fraiture, M.: Kwaresmi - Knowledge-based Web Automated Evaluation with REconfigurable guidelineS optimization. In: Forbrig, P., Limbourg, Q., Urban, B., Vanderdonckt, J. (eds.) DSV-IS 2002. LNCS, vol. 2545, pp. 362–376. Springer, Heidelberg (2002)
15. Beirekdar, A., Vanderdonckt, J., Noirhomme-Fraiture, M.: A Framework and a Language for Usability Automatic Evaluation of Web Sites by Static Analysis of HTML Source Code. In: Proceedings of 4th Int. Conf. on Computer-Aided Design of User Interfaces CADUI 2002, ch. 29, pp. 337–348. Kluwer, Dordrecht (2002)
16. Vanderdonckt, J., Bereikdar, A.: Automated Web Evaluation by Guideline Review. Journal of Web Engineering. Rinton Press 4(2), 102–117 (2005)
17. Leporini, B., Paternò, F., Scorcia, A.: Flexible tool support for accessibility evaluation. Interacting with Computers 18(5), 869–890 (2006)
18. Abascal, J., Arrue, M., Fajardo, I., Garay, N., Tomás, J.: Use of Guidelines to automatically verify web accessibility. Universal Access in the Information Society 3(1), 71–79 (2004)
19. Arrue, M., Vigo, M., Abascal, J.: Including Heterogeneous Web Accessibililty Guidelines in the Development Process. In: Gulliksen, J., et al. (eds.) EIS 2007, vol. 4940, pp. 620–637. Springer, Heidelberg (2008)
20. Takata, Y., Nakamura, T., Seki, H.: Accessibility Verification of WWW Documents by an Automatic Guideline Verification Tool. In: Proceedings of the 37th Hawaii International Conference on System Sciences (2004)
21. Luque, V., Delgado, C., Gaedke, M., Nussbaumer, M.: WCAG Formalization with W3C Techniques. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 615–617. Springer, Heidelberg (2005)
22. Vanderdonckt, J.: Development milestones towards a tool for working with guidelines. Interacting with Computers 12, 81–118 (1999)
23. Abascal, J., Nicolle, C.: Why Inclusive Design Guidelines? In: Abascal, J., Nicolle, C. (eds.) Inclusive Design Guidelines for HCI, ch.1, pp. 3–13. Taylor & Francis, Abington (2001)
24. Mariage, C., Vanderdonckt, J., Pribeanu, C.: State of the Art of Web Usability Guidelines. In: Proctor, R., Vu, K. (eds.) The Handbook of Human Factors in Web Design, ch. 8, pp. 688–700. Lawrence Erlbaum, Mahwah (2005)