Including heterogeneous web accessibility guidelines in the development process

Myriam Arrue, Markel Vigo, and Julio Abascal

University of the Basque Country, Informatika Fakultatea, Manuel Lardizabal 1, E-20018, Donostia, Spain
{myriam, markel, julio}@si.ehu.es

Abstract. The use of web applications has extremely increased in the last few years. However, some groups of users may experience difficulties when accessing them. Many different sets of accessibility guidelines have been developed in order to improve the quality of web interfaces. Some of them are of general purpose whereas others are specific for user, application or access device characteristics. The existing amount of heterogeneous accessibility guidelines makes it difficult to find, select and handle them in the development process. This paper proposes a flexible framework which facilitates and promotes the web accessibility awareness during all the development process. The basis of this framework is the Unified Guidelines Language (UGL), a uniform guidelines specification language developed as a result of a comprehensive study of different sets of guidelines. The main components of the framework are the guidelines management tool and the flexible evaluation module. Therefore, sharing, extending and searching for adequate accessibility guidelines as well as evaluating web accessibility according to different sets of guidelines become simpler tasks.

1 Introduction

In recent years, the usage of web applications has considerably extended since their usefulness has been proved in a vast variety of contexts meeting diverse needs. Companies show a growing tendency to introduce web applications in their management processes [1]. The previous business standalone applications are evolving into light web applications which have proven to be more manageable and easier to centralize. The former simple static websites have turned into unmanageable large sites which can be used for performing diverse activities. Therefore, web applications have become more complex and nowadays they integrate different technologies. According to Murugesan and Ginige [2] currently web applications can be classified in different categories depending on their functionality: informational, interactive, transactional, workflow oriented, collaborative work environments and online communities or marketplaces.

Consequently, web applications development has changed from merely being a hypertext based interface design process to a much more complex task which involves different activities such as planning, system architecture design, evaluation, quality

assessment, system performance evaluation, maintenance, updates management, etc. The development of high quality web applications requires knowledge from a wide range of disciplines such as information engineering, indexing systems, information recovery, user interface design, human-computer interaction, graphical design, etc.

Designing an appropriate user interface for these applications is probably one of the most demanding task since end-users' abilities and specific characteristics are often unknown. Under some circumstances, web applications should be designed based on “Universal Access” paradigm. This concept is turning into something extremely significant for the current Information Society as it ensures access to the information in the World Wide Web by anyone, anywhere, and at any time [3] and fosters no discrimination. Consequently, Universal Access should be an essential quality target [4] in web applications development process.

A number of initiatives have been taken in order to support the Universal Access paradigm including the promulgation, in some countries, of laws against electronic exclusion. One of the most proactive initiatives is the Web Accessibility Initiative (WAI) [http://www.w3.org/WAI/] that was set up by the World Wide Web Consortium [http://www.w3.org/]. It has published the well-known Web Content Accessibility Guidelines (WCAG) [5] which is the most universally accepted and established set of guidelines for developing and evaluating web content accessibility. It is considered that the fulfilment of these guidelines ensures that the developed web application is accessible to some extent by all people.

Even though all these efforts are extremely useful for developing accessible web applications and have extended the awareness of accessibility among web developers community, they have proven not to be sufficient in order to achieve the Universal Access. Therefore, some groups of users are still experiencing accessibility problems when interacting with the majority of existing web applications.

This situation has lead to the development of large amount of web accessibility guidelines in recent years. These guidelines aim to improve users' experience when using services in the World Wide Web. Nowadays, in addition to general purpose guidelines such as WCAG, other sets of guidelines related to specific application type (e-learning, e-commerce, etc.), specific users' characteristics (elderly, children, deaf, etc.) and accessing devices (mobile devices, etc.) can be found. Some sets of guidelines can be built combining the mentioned guidelines, e.g.: guidelines for e-learning applications for children.

According to Mariage et al. [6], current accessibility sets of guidelines are defined based on different formats, they may include different contents and are defined in different level of detail. Guidelines range form specific rules to common sense statements. Thus, existing accessibility guidelines can be classified in different groups depending on their level of detail. In this sense, Figure 1 depicts the different types of existing web accessibility sets of guidelines.

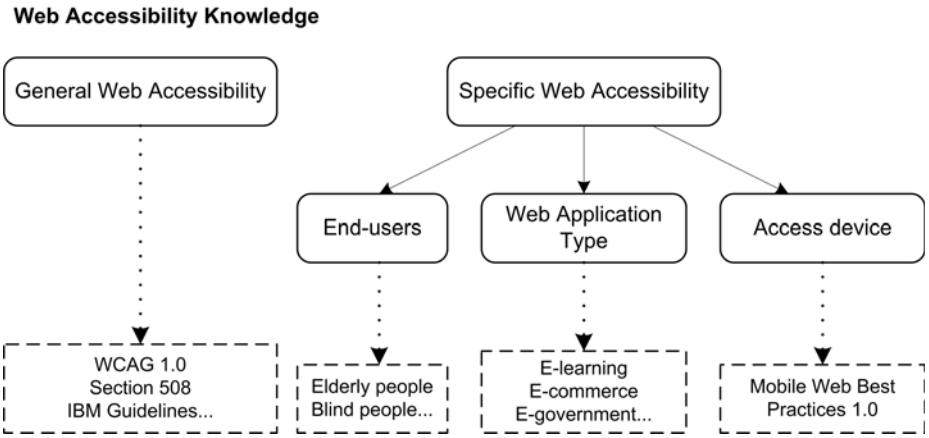

Fig. 1. A taxonomy for web accessibility sets of guidelines

Consequently, web developers should analyse the existing accessibility knowledge in order to select the most adequate guidelines, techniques and methods for their developments. In this sense, web developers usually have to deal with diverse complex tasks [7]:

- Search for the sets of guidelines which are significant for the current development.
- Select the most adequate sets of guidelines.
- Verify the coherence of the selected sets of guidelines.
- Analyse the applicability of the selected guidelines in the current development.
- Develop directly applicable design rules from the selected guidelines.
- Plan and perform frequent accessibility evaluations based on the selected sets of guidelines during the development process.

Due to the diversity of formats and structures used for defining accessibility guidelines, finding, selecting, applying and evaluating these guidelines are tedious tasks for practitioners. There are several automatic tools which assist developers evaluating the accessibility of web pages but most of them are based on general purpose sets of guidelines. Therefore, they are not flexible enough to evaluate guidelines for specific application type, user type or access device.

This paper proposes a framework for flexible web accessibility development. It will assist web developers to evaluate web interfaces according to the selected sets of guidelines. In addition, it will be useful during all the development process since it will provide several functionalities for guidelines definition, edition, searching and sharing. The basis for the development of such a framework is to define a unified definition language for accessibility guidelines so different formats and contents can be accommodated. In this sense, a comprehensive analysis of diverse sets of guidelines has been carried out and the results are outlined in Section 3. The rest of the paper is structured as follows: Section 2 is dedicated to present the related work and Section 4 describes the implementation of the evaluation logic and reporting

process of the developed framework. Finally, the reached conclusions are discussed in Section 5.

1.2 The role of accessibility evaluation

Evaluating accessibility is an essential stage in the development of accessible web applications. This process will confirm if the selected guidelines have been fulfilled. Diverse accessibility evaluations have to be performed in order to detect any possible barrier and repair it. In this sense, two different scenarios are considered: proactive and reactive evaluation. The former, concerns to the accessibility aware design and the later relates to the final application accessibility checking. Both scenarios require evaluations, including the proactive one as suggested in [8]. Performing comprehensive evaluations implies combining diverse kind of evaluations:

- Automatic evaluation with tools: this is a preliminary test stage aiming to remove the first and most "evident" obstacles. "Evident" means those errors automatically testable with the help of tools. According to Lang [9], this evaluation method presents diverse advantages in terms of costs and efficiency as the automatic evaluation tools report detected errors in a short period of time. Ivory provides a comprehensive description of different automatic evaluation methods and tools in [10]. The aim of this evaluation is to clear up the content so that forthcoming evaluations with experts and users take less time in order to focus on other complex issues. An effective evaluation tool should be able to validate the fulfilment of most of the guidelines. Yet, nowadays it is a far objective since there is not enough research done to evaluate some checkpoints such as WCAG 1.0 14.1 checkpoint: "Use the clearest and simplest language appropriate for a site's content". In addition, most of automatic accessibility evaluation tools only check the conformance with general purpose guidelines such as WCAG 1.0, Section 508 [11], etc. They are not flexible enough to evaluate other sets of guidelines or new versions as the evaluated guidelines are built-in within the source code. Consequently, incorporating new guidelines implies modifying the code of the tool. In this sense, the separation between guidelines and evaluation engine ensures the required flexibility.
- Expert driven manual evaluations: as previously mentioned the evaluation of some guidelines requires human judgement. Web accessibility experts perform evaluations based on heuristics in order to evaluate this kind of guidelines. Main tasks have to be defined and walkthroughs with different browsers, assistive technologies, devices, etc. are carried out. These evaluation methods will allow detecting accessibility barriers when the web application is used under different conditions as explained in [12].
- Evaluations with users: this evaluation type is essential since it allows detecting real accessibility barriers for users with specific characteristics. Selected users should cover the broader range of disabilities if a comprehensive evaluation is required. Users are asked for performing tasks coinciding commonly with the main functionalities of the web application. Evaluations are usually carried out in controlled environments such as specific laboratories where the experts can observe the actions of the users and gather information about the interaction

following accepted usability evaluation techniques such as the ones described by Nielsen and Mack [13] and Rubin [14]. However, results obtained from remote evaluations carried out in users' common browsing environment can be also useful as mentioned in [15].

All these evaluations are complementary and necessary. If only automatic evaluation is carried out the fulfilment of several guidelines will not be checked and the required minimum accessibility level is seldom reached. On the other hand, evaluations with users also help finding out usability barriers which accessibility guidelines and therefore automatic accessibility evaluation tools do not consider. The final objective of these evaluations is to repair the detected errors. As justified above, automatic accessibility evaluation is a necessary task indeed.

2 Related work

As previously mentioned, the basis for the development of a framework for flexible web accessibility evaluation is to separate the definition of guidelines and the evaluation logic. This objective is achieved by defining a language for guidelines specification independent of the evaluation engine. Thus, the defined grammar should be flexible enough to define forthcoming versions of existing guidelines, updates and new guidelines sets. In this sense, several approaches can be found in the literature.

In 2004, Abascal et al. [16] proposed the novel approach for automatic accessibility evaluation: separation of guidelines from the evaluation engine. The usefulness of this approach relies on its flexibility and updating efficiency. Adaptation to new guideline versions does not imply re-designing the evaluation engine but guidelines editing. The guidelines specification language is based on XML.

Following this first approaches, in 2005, Vanderdonckt and Bereikdar proposed the Guidelines Definition Language, GDL [17] and recently Leporini et al. the Guidelines Abstraction Language, GAL [18]. All these guideline specification languages make possible adapting quite straightforwardly to new guideline versions or novel guidelines.

However, these guidelines specification languages are mostly based on general purpose accessibility sets of guidelines. Consequently some specific purpose guidelines may not be defined since previous study of specific accessibility sets of guidelines and their definition in those languages is not provided. In addition, the developed definition languages are quite complex and appropriate tools for defining, editing, sharing and searching for accessibility information are needed. A new framework should be developed in the basis of a comprehensive study of different sets of guidelines and with the aim of assisting web developers during all the development process.

As far as evaluation logic is concerned, there is a growing tendency towards using XML querying languages. These languages are very powerful due to their expressiveness and flexibility. Takata et al. [19] proposed a pseudo-XQuery language for accessibility evaluation purposes and XPath/XQuery sentences are defined to check WCAG guidelines in [20]. We have adopted this technology in our new

approach since it allows us to design complex queries. As a result, lots of source code lines are saved.

3 Uniform Accessibility Guidelines Definition

We did not predict in 2004 the new amount of guidelines sets appeared, referring to specific user groups, environments or accessing devices. Therefore, a study of existing sets of guidelines has been carried out and a process for guidelines format standardization has been performed. As a result, it has been defined a new guidelines definition language: Unified Guidelines Language, UGL. The strength of our approach relies on the flexibility of the grammar since it has been defined after studying different sets of guidelines. It is flexible enough to define the guideline sets analysed in this paper and it also allows validating documents according to other criteria. The following table, Table 1, shows some sets of guidelines analysed and their classification regarding the taxonomy presented previously.

Table 1. Information about the analysed sets of web accessibility guidelines.

Name	Type	Description
WCAG 1.0 [5]	General Web Accessibility	Web Content Accessibility Guidelines 1.0
Section 508 [11]	General Web Accessibility	Section 508 of the Rehabilitation Act
IBM [21]	General Web Accessibility	IBM Accessibility Center: Developer Guidelines for Web Accessibility
CPB/WGBH [22]	Specific Application Type	Making Educational Software and Web Sites Accessible
WDGOP [23]	Specific Users' Characteristics	Research-Derived Web Design Guidelines for Older People
MWBP [24]	Specific Access Device	Mobile Web Best Practices

The developed language should be comprehensive enough to specify different information type: general information about the sets of guidelines, guidelines and methods or techniques and specific information for evaluation purposes such as evaluation procedures or test cases. In addition, the objective is to design a language which could be easily understood by web developers and accessibility experts, so that they are encouraged to specify new guidelines or new interpretations, incorporate them into the framework and share them with other users. The following sections present the fields included in the structure for each type of information.

3.1 General information

This information type refers to general information about the set of guidelines and methods or techniques which will not be processed by the accessibility evaluation tool.

- Guideline set information: this type of information is necessary for defining the general information about the set of guidelines. For instance, the classification of the set of guidelines according to the previously presented taxonomy.

- General guideline information: the necessary information for specifying each design guideline is specified, such as title, description and so on.
- Methods or techniques information: this information is necessary for training purposes so any web designer could find methods, techniques or examples of how to conform to the accessibility guidelines. This information is useful through all the web applications development phase.

General information about guidelines and sets of guidelines can be easily obtained from guidelines documents whereas specification of methods, techniques or examples requires some interpretation depending on the level of detail of guidelines. For instance, among the selected sets of guidelines the WDGOP are not defined in low level of detail. Therefore they require more effort to be interpreted and to define the methods or techniques.

3.2 Information for evaluation purposes

This information type refers to the necessary evaluation procedures for each guideline. Incorporating this information into the language schema will ensure that automatic accessibility evaluations will be possible for guidelines defined in this format.

However, not all web accessibility guidelines can be automatically evaluated. Therefore, they can be specified only with general information. For instance, the following guideline: “Use the clearest and simplest language appropriate for a site’s content” can not be validated by automatic tools since it requires human judgment. There is another type of guidelines that can not be automatically evaluated but can be triggered by tools. For instance, one of these guidelines is: “Organize documents so they may be read without style sheets. For example, when an HTML document is rendered without associated style sheets, it must still be possible to read the document”. An automatic tool can detect that a web page is associated with a style sheet but up to date it is not possible to automatically validate if the web page is well organized. Since this type of issues can be triggered by the content, they are known as semi-automatic test cases. An automatic evaluation tool will produce a *warning* if a semi-automatic test case is detected. On the other hand, an *error* will be produced if an automatic test case (a test case which can be evaluated automatically) is not fulfilled.

These automatic and semi-automatic test cases have to be defined in the language in order to ensure that the automatic evaluation process will be effectively performed. For this reason, different fields and values for defining test cases have to be incorporated into the language. The evaluation procedures for the guidelines contained in the different sets of guidelines have been analysed. This process has detected all the different semi-automatic and automatic test cases. Some of these test cases are simply validated analysing one HTML element such as IMG, TABLE, FRAME etc. whereas other type of test cases require analysing HTML elements and their attributes such as TYPE attribute of INPUT element, ALT attribute of IMG element, TITLE attribute of A element, etc. In addition, there are some complex test cases that require analysing one HTML element, its attributes and other associated HTML elements, for instance, a INPUT element with a value in its ID attribute requires the existence of a LABEL element with the same value in its FOR attribute.

All the different types of automatic and semi-automatic test cases defined in the analysed sets of guidelines have been compiled and are described in the following tables, Table 2, Table 3 and Table 4.

Table 2. This table shows the automatic (A) and semi-automatic (SA) test cases requiring only the analysis of HTML elements.

No.	Test Case Name	Description	Example	Type
1	Deprecated	The HTML element is deprecated.	*WCAG 1.0 Checkpoint 11.2 FONT*	A
2	Compulsory	The element is compulsory.	*WCAG 1.0 Checkpoint 3.2 DOCTYPE*	A
3	Text Required	A string is required between the open and close tags of the element.	*Section 508 Checkpoint (a) <APPLET>Text</APPLET>*	A
4	Avoid	It is recommended to avoid using the HTML element.	*WDGOP Checkpoint 9.1 MARQUEE*	A
5	Warning Produced	Using the HTML element may cause accessibility problems and have to be tested manually.	*Section 508 Checkpoint (m) OBJECT*	SA
6	Element Needed	Another HTML element is required.	*IBM Checkpoint 9 FRAMESET NOFRAMES*	A

Table 3. This table shows the automatic (A) and semi-automatic (SA) test cases requiring the analysis of HTML elements as well as their attributes.

No.	Test Case Name	Description	Example	Type
7	Compulsory	The attribute is compulsory.	*WCAG 1.0 Checkpoint 1.1 IMG ALT*	A
8	Compulsory Not Empty	The attribute is compulsory and it must have some value.	*IBM Checkpoint 9 FRAME TITLE*	A
9	Recommended	This attribute is recommended.	*CPB/WGBH Checkpoint 1.1 IMG LONGDESC*	A
10	Warning Produced	Using the attribute may cause accessibility problems and have to be tested manually.	*IBM Checkpoint 5 TABLE ONCLICK*	SA
11	Attribute Needed	Another attribute is required.	*IBM Checkpoint 5 SELECT ONBLUR ONFOCUS*	A
12	Error Produced	Use of this attribute must be avoided.	*WDGOP Checkpoint 1.3 INPUT ONDBLCLICK*	A
13	Determined Value	The value of the attribute has to be one of some specifically defined.	*WCAG 1.0 Checkpoint 4.3 HTML LANG= en, es, fr…*	A
14	Determined Part of Value	The value of the attribute must contain a determined value.	*WCAG 1.0 Checkpoint 3.4 TABLE WIDTH =%, em*	A
15	Avoid Value	Avoid a specified value for an attribute.	*WCAG 1.0 Checkpoint 7.4 META HTTP-EQUIV=refresh*	A
16	Value Warning	A value of an attribute may cause accessibility problems	*CPB/WGBH Checkpoint 2.2 A HREF=.wav*	SA

			and have to be tested manually.		
17	Value Attribute Not Empty	Requires	A specific value of an attribute requires another attribute which must have some value.	*IBM Checkpoint 1* *INPUT TYPE=img ALT*	A

Table 4. This table shows the automatic (A) and semi-automatic (SA) test cases requiring the analysis of associated HTML elements and their attributes.

No.	Test Case Name	Description	Example	Type
18	Attribute requires an Element with Determined Value	Element which contains an specific attribute requires the existence of another element with determined value.	*CPB/WGBH Checkpoint 1.1* *IMG LONGDESC* *<A…>D</A>*	A
19	Nested Element Not Empty Attribute	An element nested inside another HTML element requires an attribute which must have some value.	*IBM Checkpoint 1* *<A…>* *<IMG TITLE=value>* *</A>*	A
20	Elements Needed for Specific Attribute	An attribute requires the existence of a minimum number of occurrences of elements.	*IBM Checkpoint 2* *IMG ISMAP* *A element occurrences ≥ 2*	A
21	Attribute Value requires Element with Attribute Value	An attribute value requires the existence of an element with determined attribute value.	*Section 508 Checkpoint (n)* *INPUT id=value* *LABEL for=value*	A

3.3 Unified Guidelines Language, UGL

We have considered all test cases' characteristics in order to develop a common language to frame them. As a result, Unified Guidelines Language (UGL) is the resultant language which is defined according to a grammar defined in a XML-Schema. This language provides the necessary mechanisms for defining test cases for any mark-up language since it allows performing the following operations with the content within the opening and closing of a determined tag and with attribute values:

- Boolean operations
- Logical operations
- Dictionary queries for comparisons with large sets of words. E.g. checking the validity of the document language: en-us, en-gb, fr, eu, es…
- Counting

It is necessary to specify the relationships between different elements (labels and attributes) in the (X)HTML document. In addition, evaluation scope within the document can be set.

- Analyse HTML elements
- Analyse attributes within HTML elements
- Analyse associated elements of attributes and labels. There are infinite combinations since our schema is defined recursively. Therefore, it is possible to specify the following relationship: one label with a determined attribute requires a determined label with a determined attribute; one attribute requires a label (which is not its parent) with a determined attribute which at the same time requires another label and so on. Some relationships are unnecessary and useless but can be

used in some contexts. However they are useful to demonstrate the flexibility of the language and future versions of guidelines could take advantage of them.

Both XML-Schema and its graphical representation are rather large and it is out of the objective of this paper to explain thoroughly all the features of UGL. If further information is required in this sense, both schema and its picture can be found in our project's website[1].

However, relationships between different guidelines sets and its evaluation procedures can be modelled in a static diagram so that the readership could get a general idea. Figure 2 models the XML-Schema of UGL.

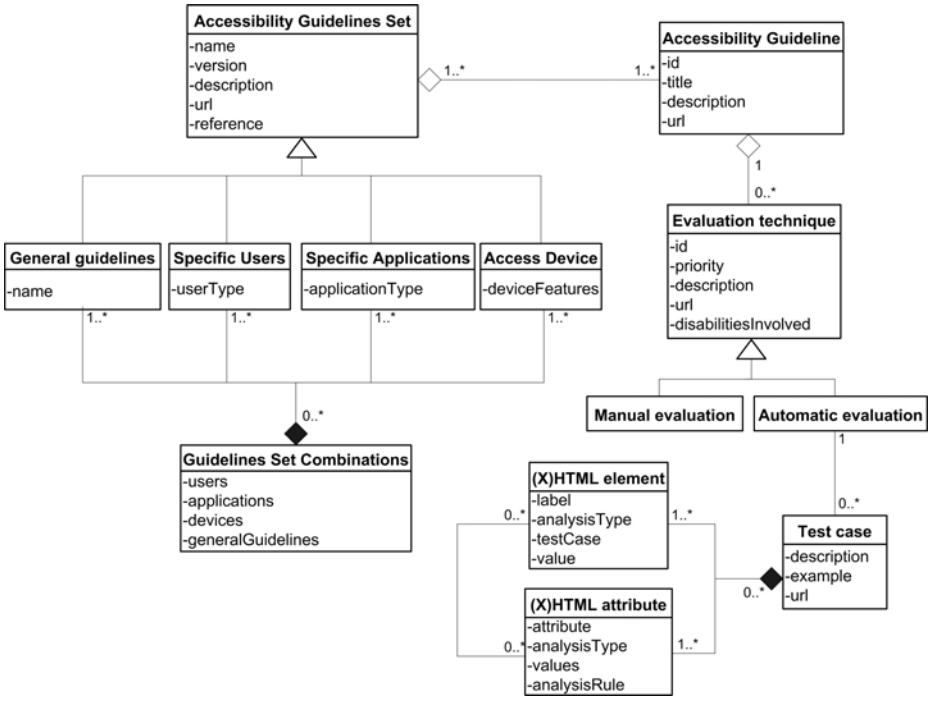

Fig. 2. A model of the relationships among entities in XML-Schema of UGL

3.4 Web Interface for guidelines management

Expert users may prefer to directly specify guidelines in UGL and upload them to the framework but novel users may get confused due to the complexity of the definition language. Therefore, a web application which guides the user specifying guidelines has been developed. Since it is accessible from the browser it has some advantages over other approaches such as the ones proposed by Mariage et al. [25] and Leporini

[1] XML-Schema of UGL: http://sipt07.si.ehu.es/evalaccess3/ugl.xsd. Its representation: http://sipt07.si.ehu.es/evalaccess3/ugl.png

et al. [26]. Both aim at abstracting the interaction with accessibility guidelines with graphical interfaces. Unfortunately, both are standalone applications which have some drawbacks compared with a web application.

Managing guidelines with a web application makes possible to have a centralized repository of guidelines. Hence, all users that sign up in the system are able to access and make evaluations with them, as well as search for specific guidelines. In addition, guidelines creators can set permissions to guideline sets such as *shared* and *shared but not editable*. The interaction is via XHTML forms and the browser is the interface between the user and the system which is accessible for everybody. Consequently, no plug-ins or software installations are required. As a result, this guidelines management interface leads to bridge the gap between developers and researchers since it is useful for knowledge sharing in this area.

The guidelines management interface is integrated in the evaluation framework proposed in this paper. Users are capable to search for guidelines and creating personal sets in order to perform automatic evaluations with them. Figure 3 shows a screenshot of the guidelines management application.

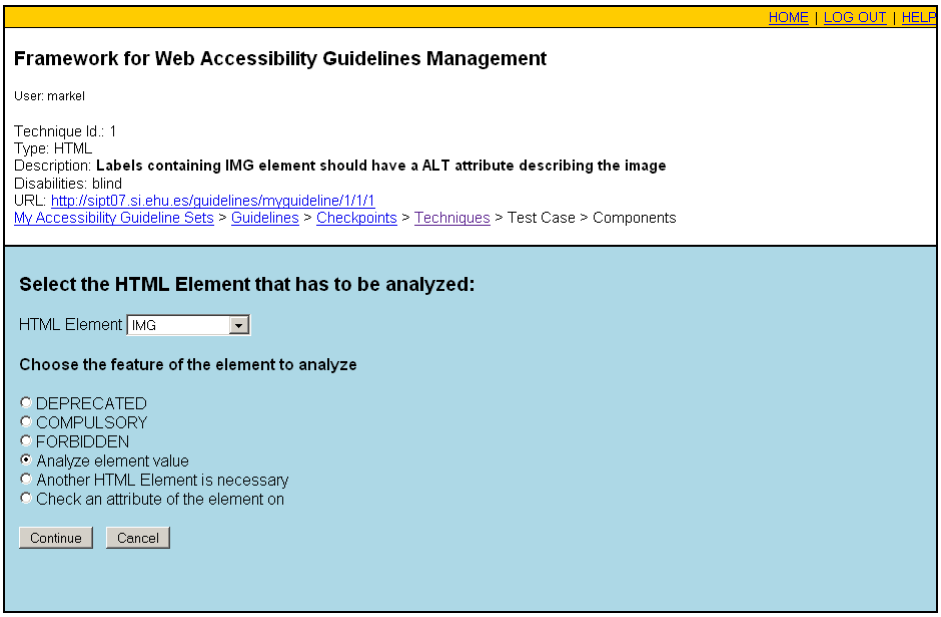

Fig. 3. Interface for guidelines management

Guidelines are stored in a relational data base. As soon as the guideline creation/edition process is concluded they are transformed into UGL. This transformation is automatically performed and the resulting UGL document is stored in a XML native data base afterwards.

4 Evaluating and reporting

The final objective of the framework is to evaluate web pages against the guideline sets stored in the guidelines repository. Thus, the management interface integrates into the whole guidelines evaluation framework and makes possible evaluating desired guidelines sets according to the requirements for a given development. However, in order to avoid searching, selecting repeatedly guidelines every time the user logs in the system, preferences regarding guideline sets will be saved in user's profile and there will be no need to repeat the process again. Therefore, unless new guidelines are required or existing ones changed, user's preferences are stored for forthcoming accesses. In order to explain the definition, evaluation and reporting stage, the evaluation of two test cases is going to be described step by step in the following subsections.

4.1 Test cases definition

Test case number 17 states: "a specific value of an attribute requires another attribute which must have some value". This test case includes examples defined in IBM Checkpoint 1 and their corresponding specification in UGL.

– Example 1: *INPUT type="img"* → *ALT*. If value of type attribute in element input is "img" an alternative description is required. The processing information for this test case is specified in UGL as follows:

```
<label>INPUT</label>
<analysis_type>check attribute</analysis_type>
<related_attribute>
        <atb>TYPE</atb>
        <analysis_type>value</analysis_type>
        <analysis_type>check attribute</analysis_type>
        <content test = "=">img</content>
        <related_attribute>
                <atb>ALT</atb>
                <analysis_type>compulsory</analysis_type>
        </related_attribute>
</related_attribute>
```

– Example 2: *INPUT name="go"* → *ALT*. If value of name attribute in element input is "go" an alternative description is required. The processing information for this test case is specified in UGL as follows:

```
<label>INPUT</label>
<analysis_type>check attribute</analysis_type>
<related_attribute>
        <atb>NAME</atb>
        <analysis_type>value</analysis_type>
        <analysis_type>check attribute</analysis_type>
        <content test = "=">go</content>
```

```
        <related_attribute>
            <atb>ALT</atb>
            <analysis_type>compulsory</analysis_type>
        </related_attribute>
    </related_attribute>
```

Test case 19 states that "An element nested inside another HTML element requires an attribute which must have some value". Its UGL representation:

```
<label>A</label>
<analysis_type>check element</analysis_type>
<related_element scope="inside">
        <label>IMG</label>
        <analysis_type>check attribute</analysis_type>
        <related_attribute>
            <atb>TITLE</atb>
            <analysis_type>compulsory</analysis_type>
            <analysis_type>value</analysis_type>
            <content test="not empty"></content>
        </related_attribute>
</related_element>
```

Fields in bold are the ones *editable* in each test case. In other words, they are the unique fields that when changing their value, the previously stated description still maintains its meaning. They are the fields that would play the role of variables in each test case as explained in the next section.

4.2 Evaluation

As mentioned in Section 2, existing novel approaches for Web documents evaluation published by Takata et al. [18] and Luque et al. [19] are the basis for our research. XQuery is a powerful query language for gathering information from XML documents quite straightforwardly. In our previous work [15], DOM and SAX technologies were used to navigate through the XML tree and the implementation required a big amount of source code compared with XQuery. Therefore, we have implemented a XQuery sentence for each test case.

Obviously, it is necessary to transform the original HTML document into XML when it comes to the evaluation of non XHTML files. JTidy[2] and Neko[3] parsers are commonly used for this task in Java environments.

All types of test cases defined in Table 2, Table 3 and Table 4 are linked to a XQuery template. This relationship is implicitly declared in a field of every test case in the UGL document. The templates contain gaps such as element name, attribute name, attribute value etc. which are filled out in a mapping process from UGL to XQuery sentences. These gaps are the previously mentioned *editable* fields and are mapped as soon as UGL guidelines have been built. Once XQuery sentences are ready, evaluation of web pages is performed by applying XQuery sentences to the

[2] JTidy HTML parser. Available at http://jtidy.sourceforge.net/

[3] CyberNeko HTML Parser 0.9.5. Available at http://people.apache.org/~andyc/neko/doc/html/

web page in (X)HTML. Figure 4 depicts the template for test case 17 and shows how values in UGL test cases are mapped there. Figure 5 shows a more complex query.

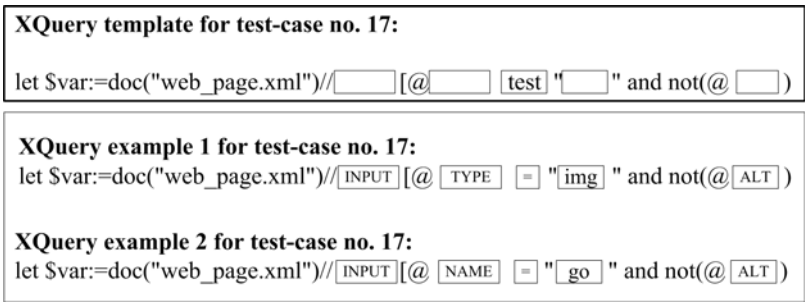

Fig. 4. XQuery template and sentences derived from test case no. 19 in UGL

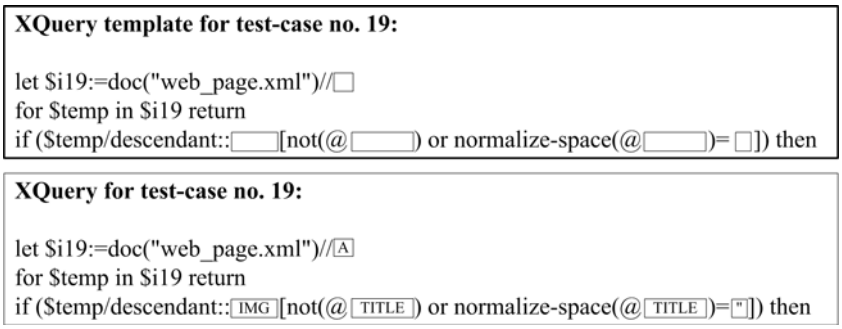

Fig. 5. XQuery template and sentence derived from test case no. 19 in UGL

Guidelines in UGL are useful for guidelines definition by experts. In this case, the expert can directly access and edit the UGL document without using the web interface. It is faster but it requires knowledge of the UGL language. Guidelines in UGL are also necessary in order to show their content in the Web interface while guidelines editing or extending. It takes less effort transforming a mark-up language such as UGL than XQuery for web publishing. In addition, since the guidelines management interface allows the user searching for guidelines, we take advantage of the facilities of the XML data base as data in relational data base data are spread in different tables and requires complex queries. Therefore, XQuery is used for evaluation purposes and UGL for guidelines definition, web publishing and guidelines search.

4.3 Reporting

The developed XQuery sentences also include useful information for detected errors reporting and reparation purposes such as the line in the (X)HTML document where the error has occurred and which element and attribute have provoked it. This

information and general information stored in UGL guidelines are put together in XML reports. This information is highlighted in the following example.

XQuery sentence

```
let $var:=doc("web_page.xml")//INPUT[@type='img' and not(@alt)]
for $temp in $var
return
<test_case no="17" type="error">
<label>{$temp/@line, $temp/name()}</label>
<attribute>type</attribute>
</test_case>
```

UGL guideline

```
<checkpoints id="1">
<priority>1</priority>
<evaluation_type>auto</evaluation_type>
<description>Provide alternative content for visual content</description>
<url>http://www-306.ibm.com/able/guidelines/web/webimages.html</url>
    <techniques id="1">
    <code>HTML</code>
    <description>Provide alternative content to images</description>
    <disabilities>blind</disabilities>
    <url>http://www-306.ibm.com/able/guidelines/web/webimages.html#techniques</url>
```

Final report

```
<checkpoint id="1">
<test_case no="17" type="error">
    <description>Provide alternative content for visual content</description>
    <url>http://www-306.ibm.com/able/guidelines/web/webimages.html</url>
    <techniques id="1">
        <description>Provide alternative content to images</description>
        <url>http://www-306.ibm.com/able/guidelines/web/webimages.html#techniques</url>
        <priority>1</priority>
        <label line="35">INPUT</label>
        <attribute>alt</attribute>
    </tecniques>
</test_case>
</checkpoint>
```

Nowadays accessibility evaluation tools reports do not have a uniform reporting format. EARL [27] is a RDF-based language supported by the W3C which aims at being the standard language for general reporting. Standardization of the reporting

format in web accessibility evaluation area is really useful since it will make possible automatically comparing the same evaluation made by different tools, keeping track of web accessibility evolution, etc. When a stable version of EARL is finally released the transformation of our evaluation report will be quite straightforward as it is XML-based.

5 Conclusions

The proposed framework assists web developers in developing accessible web applications. It is useful and reliable throughout the development process as different functionalities have been included. In this sense, web developers can edit, update, search for guidelines, include new accessibility guidelines as well as select guidelines for performing automatic accessibility evaluations. Consequently, it is flexible enough to facilitate the development of web applications according to diverse sets of guidelines.

In addition, all the functionalities included in the framework would allow creating a comprehensive repository of accessibility guidelines which could be shared among developers community. A web interface has been also developed for facilitating the access to the functionalities developed in order to assist developers with diverse level of experience.

The basis of the proposed framework is the UGL, Unified Guidelines Language. This guidelines specification language has been developed based on a comprehensive study of different types of accessibility guidelines. As a result, it integrates the necessary elements for defining a wide range of test cases. Moreover, the components integrated in this language will make possible to specify most of future versions of the existing sets of guidelines.

As far as the evaluation task is concerned, novel approaches based on XML querying technology such as XPath/XQuery are presented as well as the transformation mechanism from UGL to XQuery sentences. The use of these technologies provides a flexible evaluation module which can be easily extended in order to incorporate new features. The flexible reporting of detected errors has been also considered and will be easily updated for accommodating future standard reporting languages such as EARL.

Acknowledgements

Work of Markel Vigo is funded by the Department of Education, Universities and Research of Basque Government.

References

1. Hoffman, D., Grivel, E., and Battle, L. (2005). Designing software architectures to facilitate accessible Web applications. IBM Systems Journal. Vol. 44, No. 3, pp. 467-483.
2. Murugesan, S. and Ginige, A. (2005). Web Engineering: Introduction and Perspectives. In Woojong Suh (Ed.). Web Engineering: Perspectives and Techniques, Idea Group.
3. Stephanidis C. and Savidis A. (2001). Universal Access in the Information Society: Methods, Tools, and Interaction Technologies. Universal Access in the Information Society. Vol. 1, no. 1, pp. 40-55.
4. Savidis A. and Stephanidis C. (2004). Unified user interface development: the software engineering of universally accessible interactions. Universal Access in the Information Society. Vol. 3, no. 3-4, pp. 165-193.
5. Chisholm, W., Vanderheiden, G., and Jacobs, I. (Eds.). (1999, May 5). Web Content Accessibility Guidelines 1.0. Available at http://www.w3.org/TR/WAI-WEBCONTENT/
6. Mariage, C., Vanderdonckt, J., and Pribeanu, C. (2005). State of the Art of Web Usability Guidelines (chapter 41). The Handbook of Human Factors in Web Design. Lawrence Erlbaum.
7. Abascal, J. and Nicolle, C. (2001) Why Inclusive Design Guidelines? (chapter 1). Inclusive Design Guidelines for HCI, In Abascal, J. and Nicolle C. (Eds). Taylor & Francis.
8. Luque, V., Delgado, C., Gaedke, M., and Nussbaumer, M. (2005). Web Composition with WCAG in mind, Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A), pp. 38-45.
9. Lang, T. (2003). Comparing website accessibility evaluation methods and learnings from usability evaluation methods. Available at http://www.peakusability.com.au/pdf/website_accessibility.pdf
10. Ivory, M.Y. (2003). Automated Web Site Evaluation: Researchers' and Practitioners' Perspectives. Kluwer Academic Publishers. Dordrecht, The Netherlands.
11. Center for IT Accommodation (CITA) U.S. Section 508 Guidelines. Available at www.section508.gov
12. Brajnik, G. (2006). Web Accessibility Testing: When the Method Is the Culprit. Computers Helping People with Special Needs. In Miesenberger et al. (Eds.). Computers Helping People with Special Needs. Lecture Notes in Computer Science 4061. Springer-Verlag Berlin, Heidelberg, pp. 234-241.
13. Nielsen, J. and Mack, R. (1994). Usability Inspection Methods. John Wiley & Sons. New York.
14. Rubin, J. (1994). Handbook of Usability Testing. John Wiley & Sons. New York.
15. Petrie, H., Hamilton, F., King, N., and Pavan, P. (2006). Remote usability evaluations with disabled people. Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI 2006), pp. 1133-1141.
16. Abascal, J., Arrue, M., Fajardo, I., Garay, N., and Tomás, J. (2004). The use of guidelines to automatically verify Web accessibility. Universal Access in the Information Society. Springer Berlin, Heidelberg. Vol. 3, No. 1, pp. 71-79.
17. Vanderdonckt, J. and Bereikdar, A. (2005). Automated Web Evaluation by Guideline Review. Journal of Web Engineering. Rinton Press. Vol. 4, No. 2, pp. 102-117.
18. Leporini, B., Paternò, F., and Scorcia, A. (2006). Flexible tool support for accessibility evaluation. Interacting with Computers. Elsevier. Vol. 18, No. 5, pp. 869-890.
19. Takata, Y., Nakamura, T., and Seki, H. (2004). Accessibility Verification of WWW Documents by an Automatic Guideline Verification Tool. Proceedings of the 37th Hawaii International Conference on System Sciences.
20. Luque, V., Delgado, C., Gaedke, M., and Nussbaumer, M. (2005). Proceedings of the 14th international conference on World Wide Web, WWW 2005, pp 1146-1147.

21. IBM Accessibility Center: Developer guidelines for Web Accessibility. Available at http://www-306.ibm.com/able/guidelines/web/accessweb.html
22. Freed, G., Rothberg, M. and Wlodkowski, T. (2003) Making Educational Software and Web Sites Accessible. Available at http://ncam.wgbh.org/cdrom/guideline/
23. Kurniawan, S. and Zaphiris, P. (2005) Research-derived web design guidelines for older people. Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility (ASSETS 2005), pp. 129-135.
24. Rabin, J. and McCathieNevile, C. (Eds.). (2006, June 27). Mobile Web Best Practtices (W3C Candidate Recommendation). http://www.w3.org/TR/mobile-bp/
25. Mariage, C. and Vanderdonckt .(2004). Creating Contextualised Usability Guides for Web Sites Design and Evaluation. In R. Jacob et al. (Eds). Proceedings of the 5th International Conference on Computer-Aided Design of User Interfaces, CADUI 2004, pp. 147-158.
26. Leporini, B., Paternò, F., and Scorcia, A. (2006). An Environment for Defining and Handling Guidelines for the Web. In K. Miesenberger et al. (Eds.) Computers Helping People with Special Needs. Lecture Notes in Computer Science 4061. Springer-Verlag Berlin, Heidelberg, pp. 176-183.
27. Abou-Zahra, S. and McCathieNevile, C. (Eds.). (2006, September 27). Evaluation and Report Language (EARL) 1.0 Schema (Working draft). Available at http://www.w3.org/TR/EARL10/

Questions

Fabio Paterno:
Question: How did you calculate the line number where the error occurred?
Answer: This is done by the parser.