An Architecture for Multiple Web Accessibility Evaluation Environments

Nádia Fernandes, Rui Lopes, and Luís Carriço

LaSIGE, University of Lisbon, Edifício C6 Piso 3
Campo Grande, 1749 - 016 Lisboa, Portugal
{nadia.fernandes,rlopes,lmc}@di.fc.ul.pt

Abstract. Modern Web sites leverage several techniques that allow for the injection of new content into their Web pages (e.g., AJAX), as well as manipulation of the HTML DOM tree. This has the consequence that the Web pages that are presented to users (i.e., browser environment) are different from the original structure and content that is transmitted through HTTP communication (i.e., command line environment). This poses a series of challenges for Web accessibility evaluation, especially on automated evaluation software.In this paper, we present an evaluation framework for performing Web accessibility evaluations in different environments, with the goal of understanding how similar or distinct these environments can be, in terms of their web accessibility quality.

Keywords: Web Accessibility, Web Accessibility Evaluation Environments.

1 Introduction

The Web, as an open platform for information production and consumption, is being used by all types of people, with miscellaneous capabilities, including those with special needs. Consequently, Web sites should be designed so that information can be perceived by everyone in the same way, i.e., should be accessible. To analyse if a given Web page is accessible it is necessary to inspect its front-end technologies (e.g. HTML, CSS, Javascript) according to specific evaluation rules. From the different ways this inspection can be done, an interesting evaluation procedure concerns the usage of accessibility assessment software tools that algorithmically inspect a Web page's structure and content in an automated way.

Automatic accessibility evaluation can be performed in original or transformed HTML, resulting in different environments on which assessment takes place. One of the environments concerns the original HTML which is the HTML document derived from the HTTP. The other environment concerns the transformed HTML which is the resulting application of front-end technologies into the original HTML, as processed by CSS and AJAX/Javascript. This can substantially change the content structure, presentation, and interaction capabilities provided by a given Web page. This distinction between the original and transformed versions of a Web page's HTML is critical, since it is the latter that is presented and interacted by all users within a Web browser. Usually, the existent automatic evaluation procedures, such as those presented in [5, 10, 11], occur in the original HTML.

C. Stephanidis (Ed.): Universal Access in HCI, Part I, HCII 2011, LNCS 6765, pp. 206–214, 2011.

This paper presents an evaluation framework for perform Web accessibly evaluations in different environments. Taking into account that usually the existents automatic evaluation procedures occur in the original HTML conclusions over the accessibility quality of a Web page can be incomplete, or, in extreme erroneous. It is therefore important to access the transformed HTML documents and understand how deep the differences toward the original document are.

2 Related Work

To help create accessible Web pages, the Web Accessibility Initiative (WAI) developed a set of accessibility guidelines, the Web Content Accessibility Guidelines (WCAG) [9], that encourage creators (e.g., designers, developers) in constructing Web pages according to a set of best practices. If this happens, a good level of accessibility can be guaranteed [1, 2]. Although these guidelines exist and are supposed to be followed by the creators, most Web sites still have accessibility barriers that make very difficult or even impossible many people to use them [1]. Thus, WCAG can also be used as a benchmark for analysing the accessibility quality of a given Web page.

Web Accessibility Evaluation is an assessment procedure to analyse how well the Web can be used by people with different levels of disabilities, as detailed in [1]. Optimal results are achieved with combinations of the different approaches of Web accessibility evaluation, taking advantage of the specific benefits of each of them [1]. Therefore, conformance checking [3], e.g., with the aid of automated Web accessibility evaluation tools is an important step for the accessibility evaluation.

Automated evaluation is performed by software, i.e., it is carried out without the need of human intervention, which has the benefit of objectivity [2]. To verify where and why a Web page is not accessible it is important to analyse the different resources that compose the Web page. Two examples of automatic accessibility evaluators are: EvalAcess [6] that produces a quantitative accessibility metrics from its reports and the automatic tests of UWEM [5].

In the past, the predominant technologies in the Web were HTML and CSS, which resulted in static Web pages. Today, on top of these technologies, newer technologies appear (e.g., Javascript), and, consequently, the Web is becoming more and more dynamic. Nowadays, user actions and/or automatically triggered events can alter a Web page's content. Because of that, the presented content can be different from the initially received by the Web browser. To solve this problem, the accessibility evaluation should be applied to new environments, i.e., in the Web browser context. However, automatic evaluations do not consider these changes in the HTML document and, because of that, results can be wrong and/or incomplete. Expert and user evaluation are performed in the browser, they do not suffer with these changes.

The importance of the Web browser context in the evaluation results is starting to be considered and is already used in three tools named Foxability, Mozilla/Firefox Accessibility Extension, and WAVE Firefox toolbar [7]. However, these tools focus only evaluating Web pages according to WCAG 1.0. Furthermore, since their evaluation procedures are embedded as extensions, they become more limited in terms of their application.

Also, since these tools focus on providing developer-aid on fixing accessibility problems, the resulting outcomes from evaluations are user-friendly, thus less machine-friendly. Moreover, this "browser paradigm" - like is called in [7] - is very preliminary. Until now, to the best of our knowledge, differences between results in different evaluation environments are not clear. To perform correct comparisons, it must be guaranteed that tests are implemented in different environments in the same way, by reducing implementation bias.

3 Web Accessibility Evaluation Environments

Our study is emphasized in two main environments: Command Line and Browser. The Command Line environment represents the typical environment for automated evaluation, which includes existing evaluators that can be accessed online and the evaluation is performed into the original HTML document. In Browser environment users interact with the Web evaluation, performed into the transformed version of the HTML document.

Consequently, to better grasp the differences between the environments, we defined an architecture that allows for leveraging the same evaluation procedures in any environment, as detailed below. Afterwards, we explain how we implemented the ideas from this architecture, as well as how it was validated.

3.1 Architecture

The architecture of our evaluation framework is composed by five components, as depict in Figure 1: the QualWeb Evaluator, the Environments, the Techniques, the Formatters and the Web Server.

The *QualWeb Evaluator* is responsible for performing the accessibility evaluation in Web pages using the capabilities provided by the *Techniques* component; it uses the *Formatter* component to tailor the results into specific serialisation formats, such as EARL reporting [8]. Finally, *QualWeb Evaluator* can also be used in different *Environments*.

The *Techniques* component contains the individual front-end inspection code that is intended to be used in evaluation. In our case we chose the WCAG 2.0 [9], because it is one of most important accessibility standards. The *Techniques* component is built so that other techniques could be added, at any time, to be used in the evaluator.

The *Browser* is the environment where the transformed HTML is used and the evaluation is performed in a browser. In *Browser* could be consider two mechanisms to deliver the evaluation results, the *Server* and the *Embedded*. In the *Server* the HTML document is evaluated and the result is sent to the Web Server for subsequent analysis. In the *Embedded* the evaluation results are injected into the HTML document and shown to the developers/designers directly within the Web page.

Furthermore, other environments can be added to Environments component, in order to supply different HTML representations.

3.2 Implementation

In order to compare the proposed evaluation environments, we must use the same accessibility evaluation implementation. Given that one of the environments is the Web browser, we have a restriction on using Javascript as the implementation language. Thus, to develop the Command Line version of the evaluation process, we leveraged Node.js[1] an event I/O framework based on the V8 Javascript engine[2]. In addition to standard Node.js modules, we used several other ancillary modules[3], including:

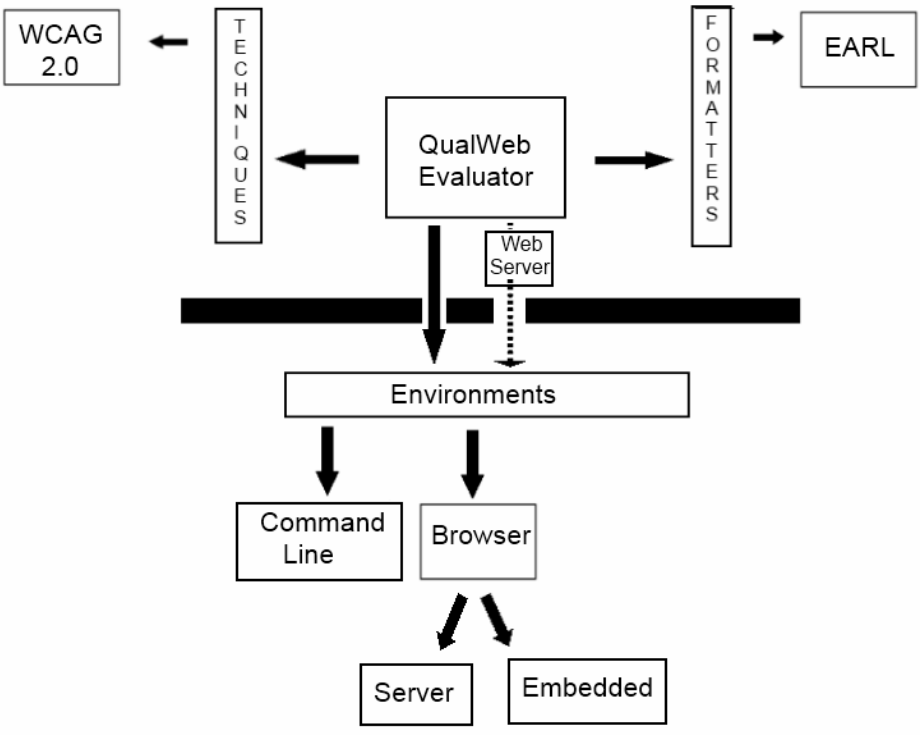

Fig. 1. Architecture of the Evaluation Framework

- *Node-Static*, which allowed for serving static files into the browser environment;
- *Node-Router*, a module that supports the development of dynamic behaviours, which we used to implement the retrieval and processing of evaluation results, and
- *HTML-Parser*, which provides support for building HTML DOM trees in any environment.

[1] Node.js: http://nodejs.org/
[2] V8 Javascript engine: http://code.google.com/p/v8/
[3] GitHub modules: https://github.com/ry/node/wiki/modules/

Besides these standard modules, we also implemented a set of modules for our evaluation framework, including:

- *EARL* module, which allows for the creation of EARL documents with the defined templates and parse EARL files using the *Libxmljs* library, and
- *Evaluator* module, which performs the accessibility evaluation with the implemented techniques.

Next it is presented an excerpt from WCAG 2.0 H64 technique.

```
function inspect(DOMList)
{
    if (typeof DOMList == "undefined" || DOMList.length == 0)
        return;

    for (var i = 0; i < DOMList.length; i++)
{
      position++;
      if (DOMList[i]["type"] == "tag" && (DOML
     ist[i]["name"] == "frame" || DOMList[i]["name"] ==
     "iframe"))
{
          if(DOMList[i]["attribs"]["title"] != "" && DOML
     ist[i]["attribs"]["title"] != "undefined" &&
     DOMList[i]["attribs"]["title"] != """ )
          {
             addElement(position,'cannotTell: title could not
     describe frame or frame','');
          }
          else
                addElement(position,'failed','');
           }
           inspect(DOMList[i]["children"]);
    }
}
exports.startEvaluation=startEvaluation;
```

Next, we present additional details on how we implemented both evaluation environments, as well as report generation and processing capabilities.

Command Line Environment

This environment obtains the HTML document from a URL using an HTTP request, executes the QualWeb evaluator on the HTML DOM tree, and serialises its outcome into EARL. All of these processes are implemented with a combination of the HTMLParser, EARL, and Evaluator modules, executed from a command line.

Browser Environment

This environment uses a bookmarklet (Figure 2) to trigger the execution of the evaluation within the browser. Bookmarklets are a kind of browser bookmark that has

the particularity of point to a URI that starts with the javascript: protocol. In front of this, pure Javascript commands follow. Thus, when a user activates the bookmarklet, these commands are executed.

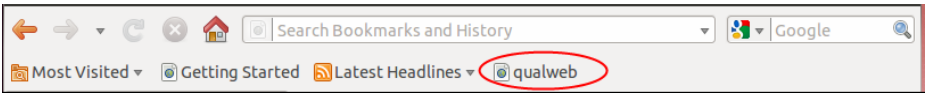

Fig. 2. Evaluation execution example on Browser

In the case of our evaluator, this bookmarklet injects the necessary functions to obtain the HTML DOM tree of the current Web page, executes the QualWeb evaluator, and sends the evaluation results to a server component. These results are transformed in the EARL serialisation format, and subsequently stored. To implement this browser-server execution and communication mechanism, we used the following modules:

- *Bootstrap*, to import the required base modules, and
- *LAB.js*, to inject all of the evaluation modules into the browser's DOM context.

Report Generation and Processing

Finally, to generate the evaluation reports containing the accessibility quality results, we used the following modules:

- *Node-Template*, to define EARL reporting templates,
- *Libxmljs*, to parse EARL reports, and
- *CSV* module, to recreate a comma-separated-values (CSV) counterpart from a given EARL report. This module allowed for a better inspection and statistical analysis with off-the-shelf spreadsheet software. Besides, to the best of our knowledge, there was nothing that performs the EARL parsing giving results in CSV.

While the EARL format allows for the specification of evaluation results, we had to extend EARL with a small set of elements that could allow for the analysis of the resulting outcomes from our experiment. Hence, we defined a Metadata field that supports the specification of HTML element count, as well as a Timestamp to state the specific time when the evaluation was performed.

The EARL reports served as the basis for generating CSV reports. Due to the extensiveness of EARL reports generated by our evaluator, especially in what respects to parsing and consequent memory consumption provided by generic DOM parsers, we implemented the EARL-CSV transformation procedures with SAX events.

Next, an EARL document example in RDF/N3[4] format.

```
<#QualWeb> dct:description ""@en;
    dct:hasVersion "0.1";
    dct:location "http://qualweb.di.fc.ul.pt/";
    dct:title "The QualWeb WCAG 2.0 evaluator"@en;
    a earl:Software.
```

[4] RDF/N3: http://www.w3.org/DesignIssues/Notation3

```
<assertion1> dc:date "1291630729208";
   a earl:Assertion;
   earl:assertedBy <assertor>;
   earl:mode earl:automatic;
   earl:result <result1>;
   earl:subject <http://ameblo.jp/>;
   earl:test <http://www.w3.org/TR/WCAG20-TECHS/H25#H25>.
<http://ameblo.jp/> dct:description ""@en;
   dct:title "The QualWeb WCAG 2.0 evaluator"@en;
   qw:elementCount "381";
   a qw:metadata,
   earl:TestSubject.
<http://www.w3.org/TR/WCAG20-TECHS/H25> dct:hasPart
<http://www.w3.org/TR/WCAG20-TECHS/H25#H25-tests/>;
    dct:isPartOf <http://www.w3.org/TR/WCAG20-TECHS/>;
   dct:title "H25"@en;
   a earl:TestCase.
<QualWeb> dct:description ""@en;
   dct:hasVersion "0.1";
   dct:title "The QualWeb WCAG 2.0 evalua-tor"@en;
    a earl:Software;
    foaf:homepage qw:.
<result1> dct:description "description"^^rdf:XMLLiteral;
   dct:title "Markup Valid"@en;
    a earl:TestResult;
    earl:info "info"^^rdf:XMLLiteral;
    earl:outcome earl:passed;
    earl:pointer <1>.
```

3.3 Testability and Validation

We developed a test bed comprising a total of 102 HTML documents, in order to verify if all the WCAG 2.0 implemented techniques provide the expected results. Each HTML document was carefully hand crafted and peer-reviewed within our research team, in order to guarantee a high level of confidence on the truthfulness of our implementation. For each technique success or failure cases were performed to test all the possible techniques outcomes. To get a better perspective on the implementation of our tests, we leveraged the examples of success or failure cases described for each WCAG 2.0 technique. The graph depicted in Figure 3 shows the number of HTML test documents defined for each technique that was implemented in the QualWeb evaluator.

To test the proper application of the implemented techniques in the two evaluation environments, we defined a small meta-evaluation of our tool. This meta-evaluation consisted on triggering the evaluation on the command line with a small automation script, as well as opening each of the HTML test documents in the browser, and triggering the evaluation through the supplied bookmarklet.

Afterwards, we compared the evaluation outcome for all HTML test documents and compared their results with the previously defined expected results. Since all of

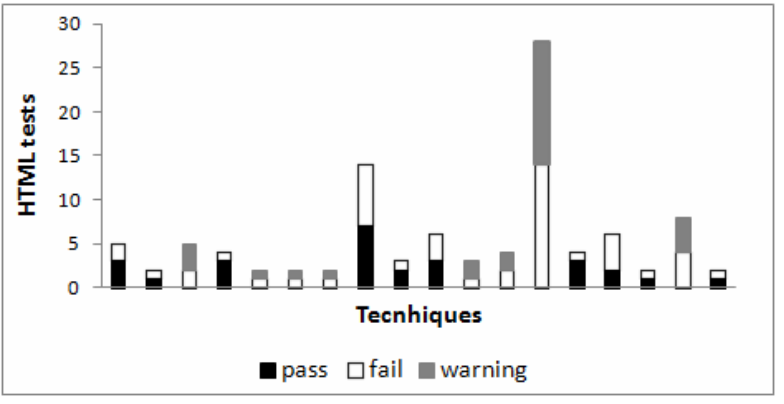

Fig. 3. Number of Test Documents per Technique

these HTML tests do not include Javascript-based dynamics that transform their respective HTML DOM tree, we postulated that the implementation returns the same evaluation results in both evaluation environments.

4 Conclusions and Future Work

The presented architecture for Multiple Web Accessibility Evaluation Environments that was implemented for: Command Line and Browser environments. The architecture was used in accessibility evaluation tests successfully. In this work were implemented new modules to facilitate this type of evaluations. These modules will be available online.

Some limitations of this work are: the evaluations do not occur exactly at the same time in both environments, so we could not guarantee 100% that Web page generation artefacts were not introduced between requests for each of the evaluated Web pages, and injection of accessibility evaluation scripts could be blocked with cross-site scripting (XSS) dismissal techniques.

Ongoing work is being conducted in the following directions: 1) an in-depth implementation of WCAG 2.0 techniques for different front-end technologies, as well as its application in different settings and scenarios; 2) implementation of more WCAG 2.0 tests; 3) continuous monitoring of changes in the HTML DOM thus opening the way for detection of more complex accessibility issues, such as WAI ARIA live regions [12]; 4) detecting the differences in DOM manipulation, in order to understand the typical actions performed by scripting in the browser context, and 5) the implementation of additional evaluation environments, such as developer extensions for Web browsers (e.g., Firebug[5]), as well as supporting an interactive analysis of evaluation results embedded on the Web pages themselves.

[5] Firebug: http://getfirebug.com/

Acknowledgements. This work was funded by Fundação para a Ciência e Tecnologia (FCT) through the *QualWeb* national research project PTDC/EIA-EIA/105079/2008, the Multiannual Funding Programme, and POSC/EU.

References

1. Harper, S., Yesilada, Y.: Web Accessibility. Springer, London (2008)
2. Lopes, R., Gomes, D., Carriço, L.: Web not for all: A large scale study of web accessibility. In: W4A: 7th ACM International Cross-Disciplinary Conference on Web Accessibility, ACM, Raleigh (2010)
3. Abou-Zahra, S.: Wai: Strategies, guidelines, resource to make the web accessible to people with disabilities conformance evaluation of web sites for accessibility (2010), http://www.w3.org/WAI/eval/conformance.html (last accessed on November 11 2010)
4. Sullivan, T., Matson, R.: Barriers to use: usability and content accessibility on the web's most popular sites. In: CUU 2000: Proceedings on the 2000 conference on Universal Usability, pp. 139–144. ACM, New York (2000)
5. Velleman, E., Meerveld, C., Strobbe, C., Koch, J., Velasco, C.A., Snaprud, M., Nietzio, A.: Unified Web Evaluation Methodology, UWEM 1.2 (2007)
6. Vigo, M., Arrue, M., Brajnik, G., Lomuscio, R., Abascal, J.: Quantitative metrics for measuring web accessibility. In: W4A 2007: Proceedings of the 2007 International cross-disciplinary Conference on Web Accessibility (W4A), pp. 99–107. ACM, New York (2007)
7. Fuertes, J.L., González, R., Gutiérrez, E., Martínez, L.: Hera-ffx: a firefox add-on for semi-automatic web accessibility evaluation. In: W4A 2009: Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibililty (W4A), pp. 26–34. ACM, New York (2009)
8. Abou-Zahra, S., Squillace, M.: Evaluation and report language (EARL) 1.0 schema. Last call WD, W3C (October 2009), http://www.w3.org/TR/2009/WD-EARL10-Schema-20091029/
9. Caldwell, B., Cooper, M., Chisholm, W., Reid, L., Vanderheiden, G.: Web Content Accessibility Guidelines 2.0. W3C Recommendation, World Wide Web Consortium, W3C (2008), http://www.w3.org/TR/WCAG20/
10. Sullivan, T., Matson, R.: Barriers to use: usability and content accessibility on the Web's most popular sites. In: CUU 2000: Proceedings of the Conference on Universal Usability, pp. 139–144. ACM, New York (2000)
11. Vigo, M., Arrue, M., Brajnik, G., Lomuscio, R., Abascal, J.: Quantitative metrics for measuring web accessibility. In: W4A 2007: Proceedings of the 2007 International Cross-disciplinary Conference on Web Accessibility (W4A), pp. 99–107. ACM, New York (2007)
12. Craig, J., Cooper, M.: Accessible rich internet applications (wai-aria) 1.0. W3C working draft, W3C (September 2010), http://www.w3.org/TR/wai-aria/