Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 14 (2012) 28 – 35

Procedia Computer Science

Proceedings of the 4th International Conference on Software Development for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2012)

Evaluating the Accessibility of Web Applications

Nádia Fernandes[*], Daniel Costa, Carlos Duarte, Luís Carriço

LaSIGE/University of Lisbon, Campo Grande, Edifício C6, 1749-016 Lisbon, Portugal

Abstract

The Web has been growing in size and complexity and is used for the most diverse activities in our everyday life, becoming almost indispensable. Besides, Web applications are becoming more popular, and consequently used by a wide range of people. Thus, it is important to evaluate the accessibility of those applications to guarantee that everyone can access the information.
Currently, there are some tools to evaluate the accessibility of classical Web pages, which use WCAG guidelines. However, Web applications impose different challenges, so it is mandatory to find a way to automatically obtain the dynamically introduced HTML code, in order to evaluate what users really experience.
This paper details a new process of accessibility evaluation of Web applications, which evaluates the content by triggering possible events that partially change the Web page. It also presents an experimental study with several Web applications, demonstrating the potential of this framework in evaluating Web applications.

Keywords: Web accessibility; Web science; Web browser processing; Automated evaluation; Web application.

1. Introduction

The constantly evolving Internet, in both size and complexity, has become a nearly indispensable tool in our everyday life. As a consequence of this evolution, sharing information and content has become an easy and common practice for every user. More than consumers, users are now also producers of information.

With the Web development, dynamic Web applications are becoming a new trend. These Web pages are no longer static content display documents but rather complex applications that replicate some common desktop

[*] nadiaf@di.fc.ul.pt

applications and their functionalities. The use of dynamic scripting, which changes the content of the Web page on real time, is converting static HTML Web pages into rich and interactive applications.

Although Web applications are indeed a new advance for Web development and Human Computer Interaction, they can cause real issues in terms of accessibility of a Web page. Accessibility is a big concern for the Web, given that everyone should be able to access and interact with a Web page.

As most of the automated evaluators are not capable of assessing dynamic content injected due to some triggered event (e.g. mouse click, key press), some accessibility problems can pass unnoticed. This means that some evaluators could classify a Web page accessible when in fact it is not. This problem was addressed before in *Fernandes et al* [1], where it was concluded that the most appropriate procedure is to perform accessibility evaluations after browser processing because a big percentage of the content of a Web page is not visible until loaded by the Web browser.

This paper follows the enhancement of the *QualWeb Evaluator* [1] to perform accessibility evaluations of Web applications and try to understand the differences when evaluating these kind of applications over the classical Web pages. New features of *QualWeb evaluator 3.0* include 1) implementation of more WCAG 2.0 tests; 2) integration of a new module which allows automated navigation through the Web page interacting with it in order to trigger dynamic changes; 3) continuous monitoring of changes in the HTML DOM which are sent to the *QualWeb 3.0* for further evaluation.

2. Dynamic Web

Web applications are able to retrieve new content using *Javascript/AJAX*, without refreshing or loading a new page. This ability can be seen, for instance, with *Google Gmai*[1]*l*, where there is no need to refresh the page to see when a new mail arrives in the mailbox. Web applications can also provide alternative interaction to the common mouse click, such as keyboard commands (similar to regular desktop applications) which enable a quicker navigation and interaction, triggering eventually some dynamic content change.

Classical Web sites are based on several pages integrated or linked together. With the introduction of the dynamic Web, new features were added and new types of services are made possible by the ability of Web applications to share or aggregate data. The content of the pages now is constantly updated without the need to reload it. Such behaviour can be seen on flight trackers, stock tickers or e-mail services. Although these applications are becoming more desktop-like programs, they still use Web technologies (transmission of content, encoding, presentation). However, these technologies are used and combined in new ways that threatens accessibility [2].

The increase of video usage on the Web is, for instance, becoming a problem when considering deaf or blind people. Text alternatives should be made available. Dynamically created content and *AJAX* based Web pages are growing and most of them are not considered accessible as screen readers, such as *JAWS*, do not seem to work satisfactorily[3].

Dynamic Web enables the change of the Web page's content and structure, usually by displaying or hiding information and HTML elements, injecting new HTML code or even removing it [1].

As can be seen, guaranteeing that the advantages of this new *"Web of applications"* are available to everyone, demands for a complete and rigorous accessibility evaluation process capable of handling the characteristics of these type of Web pages.

[1] Google Gmail: https://mail.google.com/

2.1. Accessibility Guidelines

An accessible Web means that the Web can be used by all, regardless of the impairments users may have. It means it does not have barriers that make the interaction impossible or the content not reachable. A Web page that excludes a user from its service cannot be classified as accessible.

To help create accessible Web pages, the Web Content Accessibility Guidelines (WCAG) [4] define guidelines that encourage designers and developers in constructing and evaluating Web pages according to a set of best practices.

WCAG 2.0 contains 12 guidelines written as testable sentences and chosen to address specific problems of people with disabilities. These guidelines provide the goals that should be used to make Web content more accessible. Each guideline has a testable success criteria [5]. Each criterion is a statement that can be either true or false, when a specific Web content is tested against it [6]. Some examples of WCAG 2.0 are [4]: provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language; make all functionalities available from a keyboard; provide means to help users navigate, find content, and determine where they are; help users avoid and correct mistakes of input, etc.

However, while some tests can be automated others require human testers. In this work, we focus on automatic evaluation of Web applications, because it allows us to objectively evaluate a bigger number of Web applications [7].

2.2. How the dynamic works

The use of Javascript and Cascading Style Sheets (CSS) to make Web sites more dynamic has been growing steadily in the past years. CSS enables richly styled elements which can be explicitly placed on the page. Scripting provides, besides logic and calculations, a way to dynamically update the page via *XmlHttpRequest* [8].Scripting and data transfer via *XmlHttpRequest* are the original key technologies of the *AJAX* concept (Asynchronous JavaScript and XML) [8].

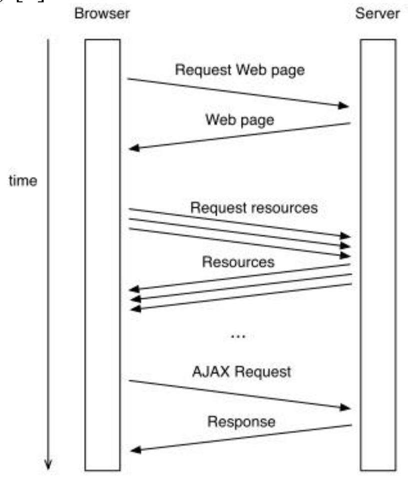

Fig. 1. Web Browsing Resource Interaction

This technology enabled the update of partial content without the need of reloading the entire page. In the general process of communication between the Web browser and the server [1], we can find three main phases: loading the Web page itself, the main resource that defines the content's skeleton; loading the remaining resources such as style sheets (CSS), scripts (Javascript, JQuery), images or media; then the *AJAX*, which is transmitted after the browser triggers the loading events (the sequence is showed on Figure 1).

After all resources are successfully delivered to the Web browser, four steps are sequentially executed before users are able to interact with the Web page, as depicted in Figure 2:

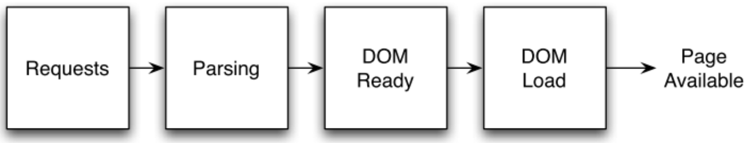

Fig. 2. Web page loading process

The first step in the Web page loading process, *Requests*, concerns with getting all resources that compose the Web page. After that, the Web browser *parses* these resources, building the HTML DOM tree, the ``visual layout" using CSS, and constructing the execution plan based on the existing scripted behaviors. Afterwards, the Web browser triggers two events in sequence: *DOM Ready* and *DOM Load*. The former is triggered when the HTML DOM tree is ready, whereas the second is triggered after all resources are ready (e.g. CSS, images, etc).

Web pages typically attach a set of behaviors to these events. This way, scripts are executed before the user gets the chance to start interacting. Since the HTML DOM tree is available for manipulation by these scripts, they can enable the addition/removal/transformation of this tree. Consequently, the Web page a user is presented with might be from slightly to heavily different from the URI's resource representation that is initially transmitted to the browser from the Web server.

Additionally, user actions can also trigger events that perform the same effect (e.g. clicking a button).

2.3. Triggering and detecting changes

An interesting tool to perform automated tests on Web applications is presented by Mesbah and van Deursen [9]. It takes into account the dynamics of these applications and common challenges that we face now, when evaluating accessibility.

The event-driven nature of *AJAX* presents the first serious test difficulty. A strategy to access those event changes is adopting a web crawler, capable of triggering the events on, for example, the *clickable* elements of the user interface.

Besides triggering, it is important to detect if they changed the DOM in any way. Thus, they assume that a new state is a DOM tree alteration inside the same page independently of the source (client or server).

2.4. In Browser evaluation

As mentioned before, the predominant technologies in the Web were HTML and CSS, which resulted in static Web pages. However, current Web pages, by means of user or automated events, can change their content. Thus, the final presented content can be different from the initially loaded by the Web browser [1].

Unfortunately, most of the current automated evaluators [10] are not capable of detecting those changes which results in incomplete and erroneous evaluations of the Web pages. In order to solve that problem, our evaluator should evaluate the Web applications in the Web browser context. Additionally, the other evaluators that already work in the browser do not use WCAG 2.0 (i.e. *Foxability*, *Mozilla/Firefox Accessibility Extension* and *WAVE Firefox toolbar* [11]) or are semi-automatic (i.e. *Hera-FFX* 2.0 [12]).

3. Accessibility evaluation process for Web applications

This section describes the approach and architecture of a framework that performs automated accessibility evaluation of Web applications.

The architecture of our accessibility evaluation framework is composed of four major components: Interaction Simulator, QualWeb Evaluator 3.0, Techniques and Formatters. The Interaction Simulator component was added to the previous version of the evaluator [1], and replaces the Environments module.

3.1. Interaction Simulation

This component is responsible for simulating actions and activating interactive elements of the interface in order to trigger changes in the HTML document. This is required as the goal is to evaluate if dynamic changes affect the accessibility of the application.

First, we simulate the processing of the Web browser using *CasperJs*[2]. *Casper.js* is a navigation scripting & testing utility for PhantomJS, written in Javascript. *PhantomJS*[3] is a command-line tool that uses WebKit(e.g. WebKit is the rendering engine used in web browsers to render pages [13]), it works like a WebKit-based web browser, with no need to display everything on the screen. Besides, it could be controlled using Javascript, being consistent with our implementation. This way, we can run a lot of evaluations sequentially. Besides, we can obtain the HTML document after the ``Web browser processing'' -- *onLoadFinished* -- simulated by the *Phantom.js*.

To perform the simulation of the several stages of a Web page, we used *Crawlers* (similarly to *Mesbah and van Deursen* [9]) which are attached to each element that is *clickable* and has an *onclick* function assigned. These *Crawlers* periodically perform the click action on these interactive elements. Every time a new version of the same Web page is detected (i.e. a new *state*), a scan is performed in order to find eventual new interactive elements.

Simultaneously, we have an *Observer* which has the responsibility to detect changes in the DOM tree. In case that happens, we consider it as a new state and the DOM is sent to the evaluation module.

It is important to refer that the Interaction Simulator keeps navigating the Web application until all the interactive elements have been activated. For that, it keeps a list of the found elements and which of those have been already activated. Besides, to avoid sending duplicated DOM threes to evaluation, a list of already evaluated documents (i.e. states) is also kept.

[2] http://casperjs.org/
[3] http://phamtomjs.org/

3.2. QualWeb Evaluator 3.0

The QualWeb Evaluator 3.0 is an upgrade of its previous version, developed by *Fernandes et all* [1], which already performs accessibility evaluations in browser context. In this new version, QualWeb performs the Web accessibility evaluations for each HTML DOM tree obtained by the Interaction Simulator.

To perform the evaluation, QualWeb uses the features provided by the Techniques component. It is now capable to evaluate a total of 26 techniques (20 HTML WCAG 2.0 and 6 CSS WCAG 2.0). Besides, QualWeb 3.0 tailors the results into specific serialization formats using the Formatters component.

4. Experimental study

We devised an initial experimental study on Web sites classified as Web applications from the 50 inspiring Web application[4].

This study centred on analysing the Web accessibility evaluation results from Web applications. We observed that some Web pages use *buttons* to perform redirection to other pages. We decided to follow only 5 of these redirections per page, to minimize the complexity that can be created with these redirections.

In the following section, we detail the results of this experiment.

4.1. Results

Our evaluation observed a total of 50 Web applications, with an average of 1010 elements per page.

We focused our study in the differences of evaluation outcomes (*fail*, *pass*, *warning*) and also as means to discover the average number of states that a Web application page may have.

Regarding the outcomes, there are significant differences in the number of HTML elements detected by the Web accessibility evaluation per type of outcome, which can be seen in Figure 3. With these results, we may conclude that for this experience *pass* represents an average of 28% of the accessibility results, *fail* represents an average of 12%, and *warning* represents an average of 70%. It is important to mention that an average of 43% of the *warnings* are detected by the CSS techniques implemented.

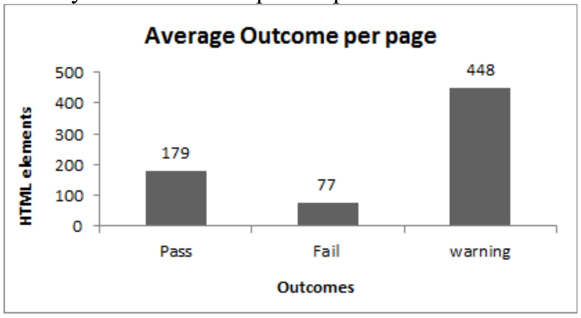

Fig. 3. Average of outcomes per page

[4] http://speckyboy.com/2009/09/02/50-inspiring-web-application-and-service-web-site-designs/

Considering the number of states, we detected on average 5 possible states per page. This means that, on average, each page can have 5 difference scenarios inside the same page, which are triggered by users interaction with the page itself. It is important to mention that with the inclusion of new elements on the page, resulting from the trigger of a particular process the number of *warnings* increased an average of 10%, and the number of *fails* increased an average of 4% per page, comparing with its initial values for the page.

These 5 states would not be triggered in a traditional automatic Web accessibility evaluation, and consequently would not be evaluated, which means an important number of accessibility problems would go unnoticed.

5. Discussion

One of the interesting aspects of this experiment is the higher percentage of passes compared with the percentage of fails, on average per page, relative to previous results[1]. This can probably be explained by the usage of tools to generate the code, which nowadays can have less accessibility problems. Other explanation would be that the majority of the problems or possible problems can be on the CSS, widely used in dynamic applications, which usually have more warnings results. Because of that the number of problems that can be completely "diagnosed`` decreased.

Additionally, there is an increase of problems or possible problems when a new state of the page is considered, showing that if these states were not considered some accessibility problems would not be identified and corrected.

5.1. Impact on Web accessibility research and perception

The results obtained show that it is possible for accessibility experts and researchers to evaluate automatically all the states that can be triggered in a page, without the need of human intervention to trigger them.

Assessment methods should be modified in order to stop considering Web pages as unique states, given all the dynamics Web applications can exhibit; we can find a complete application inside only a single page. However, previous evaluators only would consider a simple page without all the other content that can appear if triggered.

5.2. Impact on Designing Web pages

Using an evaluation procedure that considers all the possible states of the Web application, designers can develop more accessible content. This happens because they have access to a more complete page/applications analysis, which they may use to improve the accessibility quality of the page/application.

Some new evaluation mechanisms directed to developers/designers could integrate code and editing tools, which would help developers/designers in different states/scenarios. Hence, they could see the evaluations' results in different perspectives.

5.3. Constraints of the experiment

We had to impose some constraints to our experiment, due to the requirements that a large scale study would demand. The constraints were:

- the allowed number of redirections per page, which could have reduced the average number of states per page;

- the number of applications on the Web application sample, and
- the coverage of WCAG 2.0.

Despite these constraints the evaluation framework proved its added value. It demonstrated that the dynamics of Web pages can introduce accessibility problems, which are not detectable otherwise.

6. Conclusions and Future Work

Web pages have become more complex and evolved from simple information display into applications, meaning that content is not any longer static but dynamic instead. In order to automatically evaluate the accessibility in these cases, we suggest an evaluator capable of assessing HTML documents dynamically changed.

We can conclude that there are, in fact, differences between this procedure of evaluation and the regular evaluations used on the Web pages. Besides, we identified a few different states where each page can be in a Web application, which can influence the accessibility if their results are not accounted for.

Ongoing work is being conducted in the following directions:

- implementation of more techniques;
- testing with more applications;
- remove the limitation of redirections implemented.

References

[1] N. Fernandes, R. Lopes, L. Carric¸o, On web accessibility evaluation environments, in: Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, W4A '11, ACM, New York, NY, USA, 2011, pp. 4:1–4:10. doi:http://doi.acm.org/10.1145/1969289.1969295. URL http://doi.acm.org/10.1145/1969289.1969295

[2] M. Cooper, Accessibility of emerging rich web technologies: web 2.0 and the semantic web, in: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A), W4A '07, ACM, New York, NY, USA, 2007, pp. 93–98. doi:http://doi.acm.org/10.1145/1243441.1243463.

[3] M. Zajicek, Web 2.0: hype or happiness?, in: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility(W4A), W4A '07, ACM, New York, NY, USA, 2007, pp. 35–39. doi:http://doi.acm.org/10.1145/1243441.1243453.

[4] M. Cooper, L. G. Reid, G. Vanderheiden, B. Caldwell, Techniques for WCAG 2.0 - Techniques and Failures for Web Content Accessibility Guidelines 2.0, W3C Note, World Wide Web Consortium (W3C), http://www.w3.org/TR/WCAG-TECHS/ (October 2010).

[5] M. Cooper, L. G. Reid, G. Vanderheiden, B. Caldwell, Understanding WCAG 2.0, W3C Note, World Wide Web Consortium (W3C), from http://www.w3.org/TR/UNDERSTANDING-WCAG20/Overview.html (October 2010).

[6] F. Alonso, J. L. Fuertes, A. L. Gonz´alez, L. Mart´ınez, On the testability of wcag 2.0 for beginners, in: Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A), W4A '10, ACM, New York, NY, USA, 2010, pp. 9:1–9:9. doi:http://doi.acm.org/10.1145/1805986.1806000.

[7] R. Lopes, L. Carric¸o, Macroscopic characterisations of Web accessibility, Found. Trends Web Sci. 16 (3) (2010) 1–130.

[8] B. Gibson, Enabling an accessible web 2.0, in: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility(W4A), W4A '07, ACM, New York, NY, USA, 2007, pp. 1–6.

[9] A. Mesbah, A. van Deursen, Invariant-based automatic testing of ajax user interfaces, in: Proceedings of the 31st International Conference on Software Engineering, ICSE '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 210–220.doi:http://dx.doi.org/10.1109/ICSE.2009.5070522. URL http://dx.doi.org/10.1109/ICSE.2009.5070522

[10] S. Abou-Zahra, Complete List of Web Accessibility Evaluation Tools, available from: http://www.w3.org/WAI/ER/tools/complete (march 2006).

[11] J. L. Fuertes, R. Gonzlez, E. Gutirrez, L. Martnez, Hera-ffx: a firefox add-on for semi-automatic web accessibility evaluation, in: W4A '09: Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibililty (W4A), ACM, New York, NY, USA, 2009, pp.26–34.

[12] J. L. Fuertes, R. Gonzlez, E. Gutirrez, L. Martnez, Developing hera-ffx for wcag 2.0, in: W4A '11: Proceedings of the 2011 International Cross-Disciplinary Conference on Web Accessibililty (W4A), ACM, New York, NY, USA, 2011.

[13] Webkit, The webkit open source project, available from: http://www.webkit.org/ (2011).