Computer and Information Science; Vol. 5, No. 6; 2012
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

WCAG 2.0 Semi-automatic Accessibility Evaluation System: Design and Implementation

Hend S. Al-Khalifa[1]

[1] Information Technology Department, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia

Correspondence: Hend S. Al-Khalifa, Information Technology Department, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. E-mail: hendk@ksu.edu.sa

Received: September 18, 2012 Accepted: October 11, 2012 Online Published: October 21, 2012

doi:10.5539/cis.v5n6p73 URL: http://dx.doi.org/10.5539/cis.v5n6p73

Abstract

The current state of web accessibility evaluation systems is encouraging, yet not sufficient. Many evaluation systems were developed for evaluating websites based on WCAG 2.0 recommendations, however, their effectiveness is somewhat incomplete. Specifically, web accessibility evaluation systems, not being able to handle a website language poses a series of challenges for web accessibility evaluation. This paper details the design and implementation of Level A WCAG 2.0 semi-automatic accessibility evaluation system capable of processing Arabic websites. The system builds on previous work in this area and overcomes the problem encountered while dealing with Arabic websites. Our system evaluation shows that, in fact, there are considerable differences between our system and other accessibility evaluation systems, in terms of having distinct evaluation results.

Keywords: web accessibility, semi-automated evaluation, WCAG 2.0, Arabic language

1. Introduction

The World Wide Web (WWW) is evolutionary changing in terms of its technologies, recommendations and guidelines (Harper & Chen, 2012). With this evolution, access to information on the web must be granted to all people regardless of their disability. This assumption derived the field of web accessibility to flourish and become a requirement when developing websites.

Web accessibility is defined as enabling people regardless of their disabilities to access interact and use the web without any difficulties. Many guidelines and techniques were created to ensure that the web is equally accessible to all people. Among these guidelines and techniques is a dominant policy document recommended by the World Wide Web Consortium (W3C), the governing body of the web and its standards, which include the Web Content Accessibility Guidelines (WCAG) (W3C, 2008b).

WCAG 2.0 is a technology-independent standard that provides general criteria and techniques for evaluating accessibility across technologies. Beyond these techniques, developers need to conduct additional research to ensure that the content or applications they create meet WCAG 2.0 success criteria. Conducting this evaluation manually is time consuming, because of that different automated tools were developed.

In this paper, we present a semi-automatic WCAG 2.0 web accessibility evaluation system. The system currently supports the processing of Arabic websites as well as providing evaluation reports in Arabic language. Our experience in building this system gave us some interesting perspectives with regard to the weaknesses of current evaluation tools.

The rest of the paper is organized as follows: Section 2 gives a brief background about WCAG and presents related work in the area of automated web accessibility evaluation. Section 3 outlines our system analysis and design this includes: the system architectural design and the steps followed to distill WCAG 2.0 criteria for automatic evaluation. Section 4 provides the implementation details of our system. Finally, the paper concludes with evaluating the system and summarizing the main contribution of this work.

2. Background and Related Work

2.1 Web Content Accessibility Guidelines

When talking about how to achieve web accessibility, we must first refer to the World Wide Web Consortium (W3C), which aims to recommend standards for web accessibility. The Web Content Accessibility Guidelines (WCAG) is among the standards proposed by W3C.

WCAG consist of a set of guidelines for making content accessible, especially for disabled users. There are two versions of WCAG: WCAG 1.0 (1999) and WCAG 2.0 (2008). WCAG 2.0 succeeds WCAG 1.0, it is easier to use, understand and properly tested with automated testing tools and human evaluation (Shawn, 2009). It is also backward compatible with WCAG 1.0 and can be applied widely in the field of advanced technologies, thus it is recommended that accessibility practitioners reference WCAG 2.0.

WCAG 2.0 consists of four principle (Perceivable, Operable, Understandable, Robust), 12 guidelines, 61 success criteria and over 571 WCAG 2.0 techniques distributed among 12 categories as follows: General Techniques, HTML and XHTML Techniques, CSS Techniques, Client-side Script Techniques, Server-side Script Techniques, SMIL Techniques, Plain Text Techniques, WAI-ARIA Techniques, Flash Techniques, Silverlight Techniques, PDF Techniques and Common Failures. Basically, techniques are "specific authoring practices that may be used in support of the WCAG 2.0 success criteria" (W3C 2008b). The techniques are the focus of this paper.

Figure 1 shows the relationship between principles, guidelines, success criteria and techniques in WCAG 2.0. Each principle has a set of guidelines. Each guideline has a set of success criteria and user category. The user might be limited by Hardware, Software or has physical limitations. Finally each success criteria may employ different techniques.

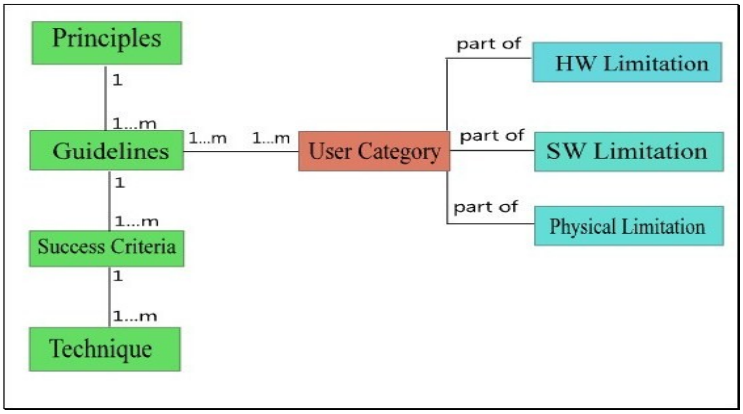

Figure 1. The relationship between principles, guidelines and success criteria in WCAG 2.0

Table 1 gives an example of a principle, a guideline, a success criteria and a technique that tackles non-text content in WCAG 2.0.

Table 1. Example of a principle, guideline, success criteria and a technique from (W3C 2008b)

Principle 1	Perceivable	
Guideline 1.1	Text alternatives	Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, speech, symbols or simpler language.
Success criteria 1.1.1	Non-text content	All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below.
Technique	Controls, input	If non-text content is a control or accepts user input, then it has a name that describes its purpose.

WCAG 2.0 has three levels of conformance (W3C 2008a), divided into: Single-A (minimum level of conformance with minimum level of accessibility), Double-A (intermediate level of conformance with enhanced level of accessibility) and Triple-A (high level of conformance with additional accessibility enhancements). Each level of conformance has many testable techniques that are used to evaluate websites accessibility (Al-Khalifa, 2010).

2.2 Accessibility Evaluation Systems

Web accessibility evaluation tools are software programs or online services that help determine if a website meets accessibility guidelines. We can categorize these tools into two types: (1) general tools that evaluate most WCAG guidelines and (2) special tools that evaluate specific topics covered by the guidelines e.g. color checker. Evaluation tools can be further categorized into (EOWG, 2005):

- Fully automated: can examine the whole website with all guidelines.
- Semi-automated: attempts to evaluate web pages with little or no user intervention. Usually these tools produce reports with the results of checks they evaluate.
- In-page feedback evaluation tools: display the results of automated accessibility checks on the respective locations of the web page by inserting icons and markup into the code of the page.
- Wizard interface: guides web developers through a series of checks in a defined sequence in order to determine the conformance of the web content to accessibility guidelines.
- Page transformations: assist web developers in evaluating checkpoints which need to be manually evaluated by modifying the appearance of the web pages.

There are several methods for evaluating the accessibility of a website:

1. Evaluate a remote file: This method evaluates a single web page.
2. Evaluate entire website: This method evaluates all pages in the website.
3. Evaluate HTML source: This method evaluate the accessibility of the HTML code.

There are different evaluation systems that can be used to test the accessibility of websites. However, the majority of these systems follow WCAG 1.0 (the previous version of the accessibility guidelines), which has outdated. Also, the list of Web Accessibility Evaluation Tools in the W3C website was not updated since 2006 (Shadi, 2006), which made us search for WCAG 2.0 evaluation tools. So as of the date of this project we have found only five systems providing support for WCAG 2.0: (1) TAW (Web Accessibility Test) (TAW, 2012); (2) Worldspace FireEyes (Deque, 2012); (3) Total Validator (Total Validator, 2012); (4) Web Accessibility Assessment Tool (WaaT) (Oikonomou et al., 2011) and (5) aChecker (aChecker, 2012).

TAW is an online accessibility tool. It analyzes websites according to W3C Web Accessibility guidelines (WCAG 1.0 and WCAG 2.0) with recommendations for fixes. Accessibility violations in TAW are presented in three categories (problems, warnings, and not reviewed).

Worldspace FireEyes is a free web accessibility evaluation tool which is introduced by Deque Systems, Inc. In this tool, the accessibility evaluation of a website can be performed according to WCAG 1.0, WCAG 2.0 as well as Section 508. FireEyes is designed as an add-on that works with Firefox browser through the Firebug tool.

Total Validator is a stand-alone application that validates (X) HTML, web accessibility as well as it works as a spell checker and a broken links checker. It validates against WCAG (1.0 and 2.0) and Section 508. It also comes as a browser extension for Chrome and Firefox.

Web Accessibility Assessment Tool (WaaT) is a stand-alone application that provides web accessibility evaluation according to WCAG 2.0. This application allows a user to perform specific evaluation process, by selecting different constraints (e.g. different types of impairments and disabilities, different sets of guidelines, personas, etc.) It also has three possible outputs: “Errors”, “warnings” or general information about evaluated page “info” e.g. number of forms in pages.

aChecker is an online web accessibility evaluation tool. It supports the evaluation of BITV 1.0 (Level 2), Section 508, Stanca Act as well as WCAG 1.0 and 2.0. The tool implementation is based on TinyMCE HTML editor, an open source editor, used in many open source Web Applications such as ATutor learning management system. This tool categorizes accessibility problems into three types: Known problems, Likely problems and Potential problems.

It is also worth mentioning HERA-FFX for WCAG 2.0 (Fuertes et al., 2011). This Firefox add-on supports the manual evaluation of web accessibility based on WCAG 2.0. It represents evaluation results in six-element array

as follows: fail, N/A, verify, ok, unknown, and partial. Similarly, (Fernandes et al., 2011) developed an evaluation framework that was implemented for: Command Line and Browser environments. The evaluation results are represented in three categories: pass, fail and warning. The environment is still under development to include more WCAG 2.0 techniques.

Although the above examined systems have powerful evaluation engines, some of them faced problems while processing Arabic websites. As our focus is on Arabic websites, we discovered that some accessibility errors have emerged only from using Arabic language. These errors are generated under the 3.1 guideline: readable, make text content readable and understandable (level A). Some systems also displayed Arabic content as gibberish (Figure 2). In such case, if we evaluated Arabic websites, the number of errors resulted by such systems will be always more than they actually are. It is indeed not correct for reliable accessibility evaluation systems to report Arabic language content as an accessibility error.

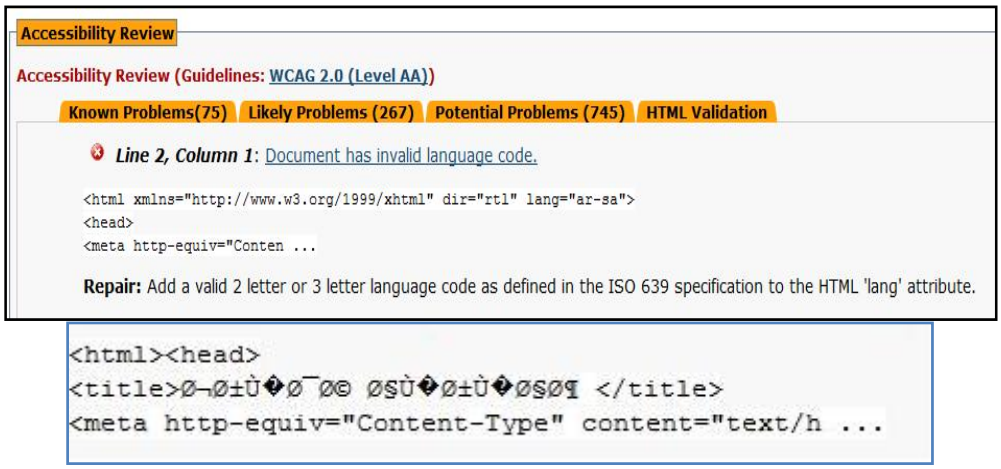

Figure 2. Arabic language problems with aChecker, top shows the results with invalid language, bottom is the tested page source code in Gibberish

So to overcome this problem, the aim of this project is to design and develop a WCAG 2.0 web accessibility evaluation system that is capable of handling Arabic websites and generating accessibility reports in Arabic for the sake of Arabic web developers and practitioners.

3. System Analysis and Design

Before designing our system, we first need to study and understand thoroughly WCAG 2.0 techniques for two reasons: (1) distill those applicable for automatic evaluation and (2) classify the techniques in a way that makes processing them programmatically more efficient. So, as a first step in our system we focused only on level A techniques. Moreover, we studied similar automated accessibility evaluation systems interfaces and benefited from their best practices in designing our system interface.

3.1 Analyzing WCAG 2.0 Techniques for Automatic Evaluation

After studying 138 WCAG 2.0 level A techniques (48 were HTML techniques (H), 9 were CSS techniques (C), 81 were general techniques (G)), we realized that the techniques can be implemented either fully automated or semi-automated (i.e. needs human intervention or more complex techniques). Therefore, in our project we have implemented 13 fully automated and 15 semi-automated techniques. Other techniques were not considered, because most of them require complex processing procedures, which were beyond the objective of this project.

We also found that the current categorization of techniques (i.e. HTML techniques, CSS techniques, general techniques, etc.) has some overlaps between them in their procedures, for instance, there were common steps in more than one category to check for images. For this reason we created a new categorization to minimize the overlap between techniques. The new categorization is based on media types (images, forms, pages, text, links and tables). After classifying the techniques based on their media type we created a table to show the relationship between the techniques. This helped in reducing the number of common (overlapped) evaluation procedures in more than one technique; in addition this resulted in having the number of errors more valid and accurate as we will see in section 5.

Table 2 shows the dependent techniques of the fully automated techniques for each media type.

Table 2. Fully automated dependent techniques, where G means general techniques and H means HTML techniques

Media Type	Technique	Dependent technique
Image	G95	H37
Image	H2	-
Forms	G162, H71	-
Forms	G80	H32
Page	H74	G134
Page	H46, H35	-
Table	H43, H51	-

Table 3 shows the fully implemented techniques in our system.

Table 3. Implemented fully automated techniques

Technique	Description	Corresponding success criterion
H2	Combining adjacent image and text links for the same resource.	1.1.1
H35	Providing text alternatives on applet elements.	
H46	Using noembed with embed.	
G95	Providing short text alternatives that provide a brief description of the non-text content.	
H37	Using alt attributes on img elements.	
H51	Using table markup to present tabular information.	1.3.1
H43	Using id and headers attributes to associate data cells with header cells in data tables.	
H71	Providing a description for groups of form controls using fieldset and legend elements.	
H32	Providing submit buttons.	3.3.2
G80	Providing a submit button to initiate a change of context.	
G162	Positioning labels to maximize predictability of relationships.	
G134	Validating Web pages.	4.1.1
H74	Ensuring that opening and closing tags are used according to specification.	

For each technique, an evaluation algorithm was produced by converting W3C Test procedures written in plain English into Pseudocode. Table 4 demonstrates the Pseudocode for technique number 51 that evaluates the table element.

Table 4. Pseudocode for html technique of table element

Technique H51: Using table markup to present tabular information.

Test procedure from W3C:

1. Check for the presence of tabular information.
2. For each occurrence of tabular information: Check that table markup with at least the elements table, tr and td is used.

Expected results: PASS if #1 and #2 are true.

Pseudocode:

```
Check for <TABLE> elements
  IF <TR> elements exist
          NO error
    ELSE
          INCREMENT number of errors
          INCREMENT number guideline 1.3 errors
UPDATE report with page title, line of code, line code number, hints and Success Criterion 1.3.1
END IF
    IF <TD> elements exist
          NO error
    ELSE
          INCREMENT number of errors
          INCREMENT number of guideline 1.3 errors
          UPDATE report with page title, line of code, line code number, hints and Success Criterion 1.3.
      END IF
  END IF
```

Table 5 shows the semi-automated implemented techniques in our system. Fifteen semi-automated techniques were partially checked by our system, this was performed by checking the steps that do not require human intervention to check for no errors then issuing a warning to advise the user to perform manual check.

Table 5. Implemented semi automated techniques

Technique	Description	Corresponding success criterion
H30	Providing link text that describes the purpose of a link for anchor elements.	1.1.1
G73	Providing a long description in another location with a link to it that is immediately adjacent to the non-text content.	
H86	Providing text alternatives for ASCII art, emoticons, and leetspeak.	
H36	Using alt attributes on images used as submit buttons.	
H65	Using the title attribute to identify form controls when the label element cannot be used.	1.1.1 3.3.2 1.3.1

H44	Using label elements to associate text labels with form controls.	1.1.1 3.3.2 4.2.1
H39	Using caption elements to associate data table captions with data table.	1.3.1
H63	Using the scope attribute to associate header cells and data cells in data tables	
H56	Using the dir attribute on an inline element to resolve problems with nested directional runs.	1.3.2
G88	Providing descriptive titles for Web pages.	2.4.2
H25	Providing a title using the title element.	
H4	Creating a logical tab order through links, form controls, and objects.	2.4.3
H24	Providing text alternatives for the area elements of image maps.	2.4.4
H57	Using language attributes on the html element.	3.1.1
H88	Using the title attribute to provide context-sensitive help.	4.1.1

Table 6 shows one semi- automated technique, notice that it was not fully automated at step 2.

Table 6. Pseudocode for html technique of area element

Technique H24: Providing text alternatives for the area elements of image maps.

Test procedure from W3C:

For each area element in an image map:

1. Check that the area element has an alt attribute.
2. Check that the text alternative specified by the alt attribute serves the same purpose as the part of image map image referenced by the area element of the image map.

Expected results: Pass if #1 and #2 are true.

Pseudocode:

```
IF alt attribute of <AREA> elements include in <MAP> elements
    NO error
  ELSE
    INCREMENT number of errors
    INCREMENT number guideline 2.4 errors
    UPDATE report with page title, line of code, line code number, hints and Success Criterion 2.4.4
   END IF
   IF "alt" attribute serves the same purpose as the part of image map referenced by the area element
   of the image map.
        NO warning
  ELSE
    INCREMENT number of warnings
    INCREMENT number guideline 2.4 warnings
    UPDATE report with page title, line of code, line code number, hints and Success Criterion 2.4.4
   END IF
```

3.2 Design Patterns

Designing a usable and accessible user interface is very important for the success of our system. So, based on our previous research on similar systems and examining their interfaces we distilled the following best design patterns:

3.2.1 Results Types

All examined systems divided their accessibility problems into different sections; in our system we divided our accessibility results into two types:

- Errors: automatically checked and mean that there is an accessibility violation.
- Warnings: need human intervention and mean that there might be an accessibility violation.

This classification is similar to what **WaaT** is using.

3.2.2 Report Generation

Most systems provide two kinds of reports: (1) summary report and (2) detailed report. In our system, these two kinds are implemented as follows:

1)- Summary Report:

This report type gives a high-level overview of the results. It displays the number of errors and warnings of each guideline in each page.

Figure 3 shows a screenshot of our system summary report with the following elements: (1) number of errors, (2) number of warnings, (3) accessibility score bar (this was computed using the following equation: 100 - ((number of errors/number of evaluated page lines)*100), and (4) success criteria table.

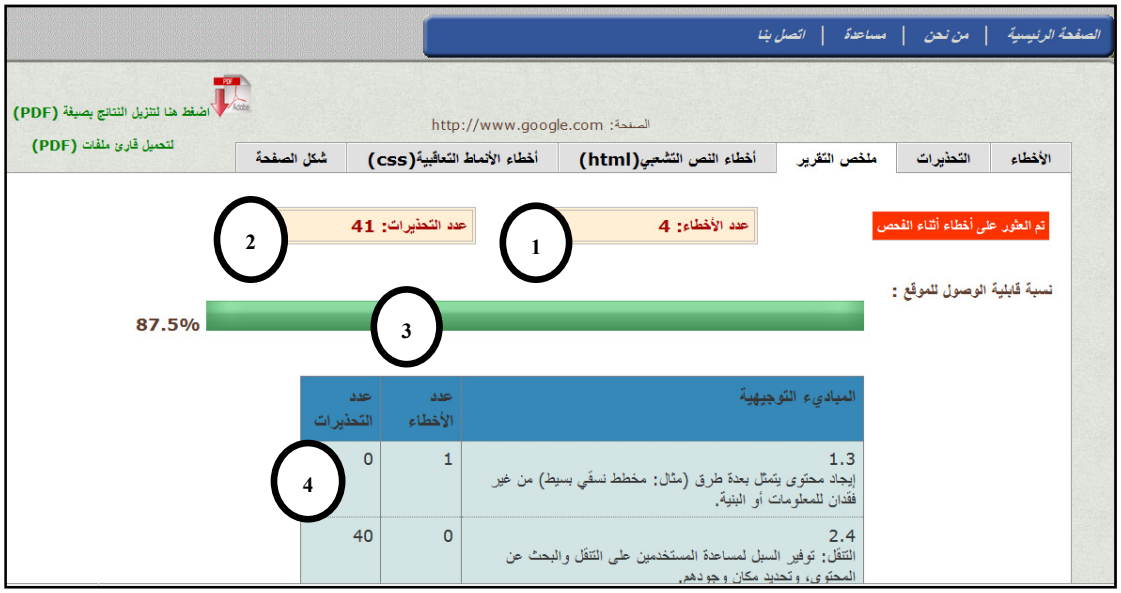

Figure 3. Summary report showing number of errors and warnings for each success criteria

2)- Detailed Report:

It consists of a more detailed information about (errors and warnings) which display the errors and warnings reporting for each page according to media type (text, link, image, table, page, form) and display: page title, line of code, success criteria, code line number and hint for the error or warning. Also it displays detailed HTML report in a table view which contains code line number, column number, error type and source code. Finally it displays detailed CSS report that shows a table containing code line number, type of error and source code. Figure 4 shows a screenshot of the detailed report.

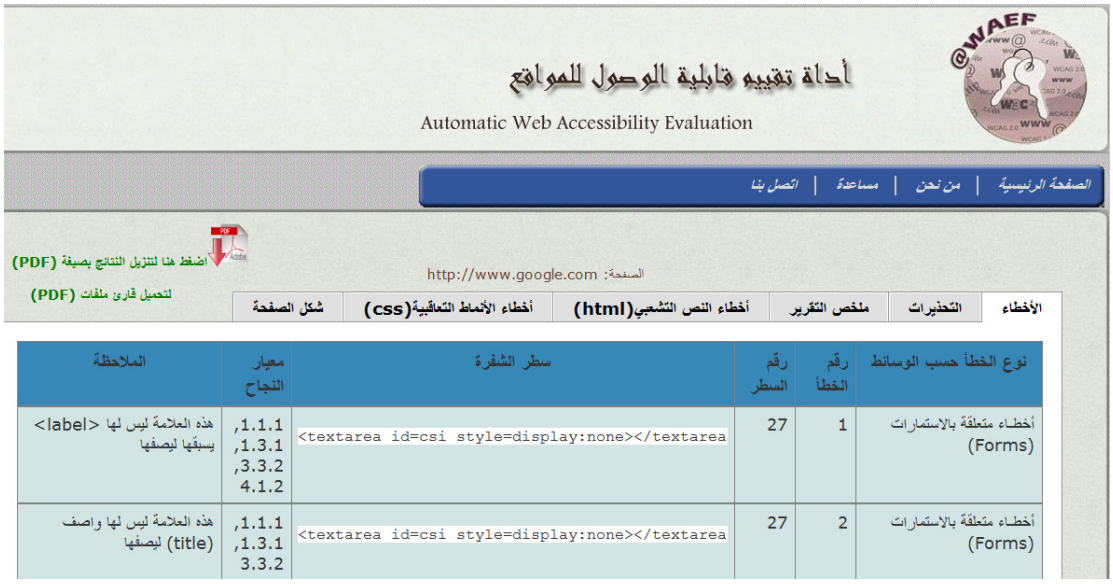

Figure 4. The detailed report with different tabs for errors and warnings

3.2.3 Viewing Line of Code

Viewing a line of code that caused an error or a warning is an important thing for developers to capture the violating code (i.e. for tracing purposes). During our research we found that other systems follow diversified ways to view line of code, basically, we took advantage of viewing line of code with sufficient hint as **aChecker** did and combine it with **Worldspace & FireEyes** data table used for organizing violations.

3.2.4 Annotated Page View

Viewing visually the places of violations in a page might be more helpful for the novice evaluators than viewing line of code. Figure 5 demonstrates a web page with annotated icons showing the places of errors (with x) and warnings (with exclamation mark).

Figure 5. Page view with annotation violations

3.2.5 Providing Hints

Telling the user 'where' the errors are is very effective when used alone. However, supporting the error place with 'how' to repair is more effective. Providing hints create two challenges:

- Writing in an easy and understandable way.
- Writing clear Arabic terms that reflect the gist of the error.

We provided detailed information about each success criteria, including normative text and techniques to be

applied for assessment as well as W3C descriptive link for each success criteria violation. This is similar to what **Web Assessment Tool** does to support the user with more information.

3.2.6 Website Depth

All accessibility evaluation systems evaluate only one page except for **Web Assessment Tool**, which evaluates more than one page; however it is not clear what criteria the tool is using for page selection.

In our system we evaluated a website by tracing its hierarchy (i.e. its depth). Our evaluation depends on two levels of a website hierarchy (level 0 and level 1). Level 0 means evaluating one page (the target). Level 1 means evaluating pages under the target (max=5 sibling pages).

3.2.7 Downloading Report

Our system allows the user to download the result as a PDF file. The PDF file describes the accessibility results of the evaluated web page. This PDF file contains an overview of the result as well as details for all detected errors, warnings, HTML and CSS validation.

4. System Implementation

Our system is considered a function oriented pipelining model where the system is decomposed into functional modules that accept input data and transform it into output data. Figure 6 shows the Data Flow Diagram (DFD) of our system.

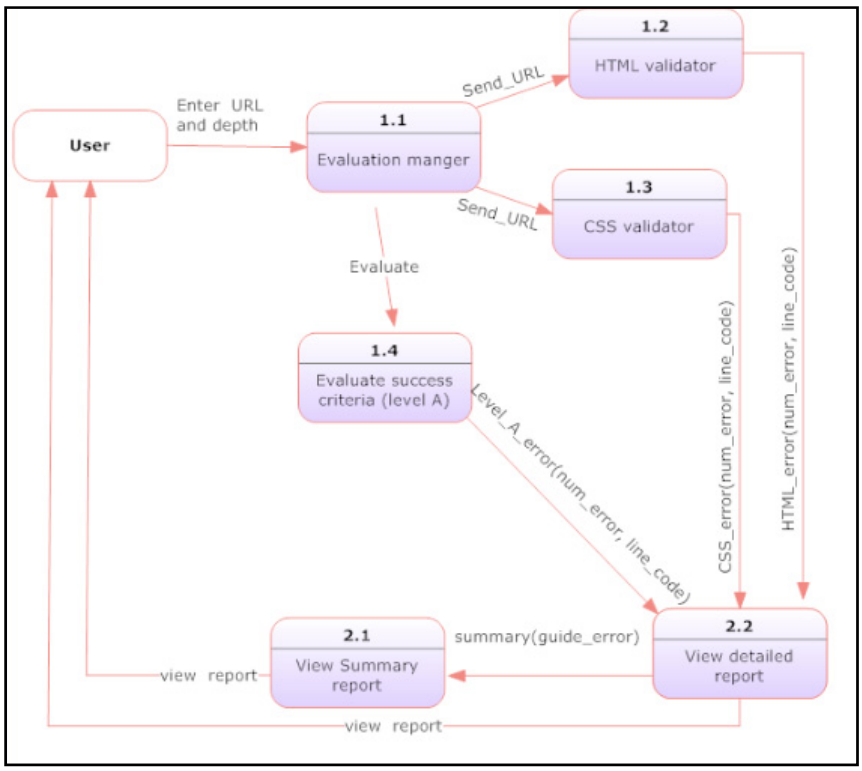

Figure 6. System DFD

4.1 System Processes

The system consists of six major processes grouped into two main groups as follows:

Process 1.1: Evaluation manger

This process receives the URL of a website and the evaluation depth from the user. Then, it finds the URL's of all pages under the received URL according to the given depth using depth first searching technique. After collecting the URL's, they are sent to: HTML validator, CSS validator, and Evaluate success criteria processes.

Process 1.2: HTML Validator

This process receives the URL of each page that needs to be evaluated, and then send them to W3C HTML

validator API. The results of the validation are sent back to the View detailed report process.

Process 1.3: CSS Validator

Similar to HTML Validator process, this process receives the URL of each page that needs to be evaluated, and then send them to W3C CSS validator API. The results of the validation are sent back to the View detailed report process.

Process 1.4: Evaluate Success Criteria (level A)

This process receives the URL of each page that needs to be evaluated, and then performs WCAG 2.0 level A evaluation techniques on each page. The results of the validation are sent to the View detailed report process.

Process 2.1: View Detailed Report

This process consolidates the evaluation results returned back from the Evaluate success criteria process and HTML and CSS validators. It generates two reports, the first report contains detailed information of errors, and the second one contains detailed information of code warnings. Each report views code line number that contains the error or warning, page title, line code that contains an error or warning, success criteria from WCAG 2.0 which is supposed to be applied, and hints that help to understand and repair the error or warning.

Process 2.2: View Summary Report

This process receives detailed information from the View detailed report process, and then generates a summary report which contains each guideline from WCAG 2.0 and number of errors and warnings in each guideline.

4.2 Implemented Modules and Used Tools

A set of modules were implemented which include:

- **URL Checker**: this module checks the validity of the entered URL in terms of syntax and existence.
- **Depth function**: this module is used to get the URLs under the entered URL.
- **Image techniques**: this module implements WCAG 2.0 image techniques.
- **Table techniques**: this module implements WCAG 2.0 table techniques.
- **Form techniques**: this module implements WCAG 2.0 form techniques.
- **Link techniques**: this module implements WCAG 2.0 link techniques.
- **Page techniques**: this module implements WCAG 2.0 page techniques.
- **HTML validation**: this module receives the URL of each page that needs to be evaluated. It validates the HTML of each page according to W3C HTML criteria.
- **CSS validation**: this module receives the URL of each page that needs to be evaluated. It validates the CSS of each page according to W3C CSS criteria.

Besides, our system was implemented using PHP 5.0, JavaScript, HTML and CSS. It used open source libraries to perform certain functions. For HTML DOM parsing, we used (Simplehtmldom V1.11) (Note 1), for URL manipulation, we used (URL-to-Absolute) (Note 2) to combine the base URL to some relative URL to produce absolute URL. For the user interface narratives we consulted the glossary of information technology terms in Arabic (Alhafez, 2007).

5. System Testing and Evaluation

In our system we conducted two types of testing: user acceptance via usability testing and performance testing via comparing our system performance against other accessibility systems.

5.1 Usability Testing

To measure our system usability and user satisfaction, users must have at least the basic knowledge about web terminologies as well as being familiar with the domain of web development. Our target users were Arabic speaking web developers.

They were asked to try our system first then answer the questionnaire. So, five students who already studied a web course and accessibility standards and developed a website have tried our system. After that we asked them to grade the usability of our system using System Usability Scale (SUS) (Brooke 1996). SUS is a five scale Likert questionnaire, (ranging from strongly disagree to strongly agree), which consist of the following items:

(1) I think that I would like to use this system frequently.

(2) I found the system unnecessarily complex.

(3) I thought the system was easy to use.

(4) I think that I would need the support of a technical person to be able to use this system.

(5) I found the various functions in this system were well integrated.

(6) I thought there was too much inconsistency in this system.

(7) I would imagine that most people would learn to use this system very quickly.

(8) I found the system very cumbersome to use.

(9) I felt very confident using the system.

(10) I needed to learn a lot of things before I could get going with this system.

Table 7 shows SUS results, and the score for each user. From the table we can see that our system in general performed well in terms of ease of use. However, the answers to question 1, 2 and 7 were not very encouraging, basically because the users pointed out the importance of having English technologies enclosed after their Arabic counterparts, because they are not familiar to their translations.

Table 7. SUS Results

	User1	User2	User3	User4	User5
Result from 100	95	92.5	87.5	85	92.5

This result encouraged us to conduct another deep evaluation using remote testing to further discover any problems, if exists, in our system and understand any difficulties that might limit our system functionality.

5.2 Remote Testing

We conducted another survey for remote users; we know that we can get more responses online more than conducting live usability tests. We wrote a survey directly related to our system tasks. We published this survey on Facebook, a social networking site, and asked interested people to evaluate the website and fill out the survey.

The survey consisted of twelve closed questions listed in Table 8 with a corresponding comments area for each question. Every question has three options “agree”, “partially agree” and “disagree”.

The feedback gathered from the remote testing was more than the feedback gained by the usability testing. We got 16 responses, fourteen were females and two were males, ages ranged between 15 and 25 years old. All have experiences in building websites except for one user. Thirteen respondents have some background in WCAG and web accessibility standards.

Table 8. Remote testing questions

1. Was the evaluation process easy?
2. Are errors content clear and understandable?
3. Are the hints provided for the errors clear and understandable?
4. Are warnings content clear and understandable?
5. Are the hints provided for the warning clear and understandable?
6. Is the summary report understandable?
7. Is the content of the (success criteria**)** in the summary report clear and understandable?
8. In “HTML Violations” tab did you find the errors and warnings understandable?
9. Is the content of the “Error Type” clear and understandable?
10. In “CSS Violations” tab did you find the errors and warnings understandable?
11. Could you distinguish easily between errors and warnings in the “page capture” tab?
12. Is our system easy to use?

Figure 7 summarizes the results of the remote testing. It shows that Arab users in general accepted the system output with Arabic language; however, question number 5 and 11 shows the heights number of disagreement compared to the rest of the questions. Both questions focused on the understandability of the provided warnings either alone or when they come with errors, which seems that warnings' narratives have confused the users. On the other hand, questions 2, 4 and 7 demonstrate high scores for partially agree answers. This observation also contributes to the previous observation about the clarity of narratives for warning and errors.

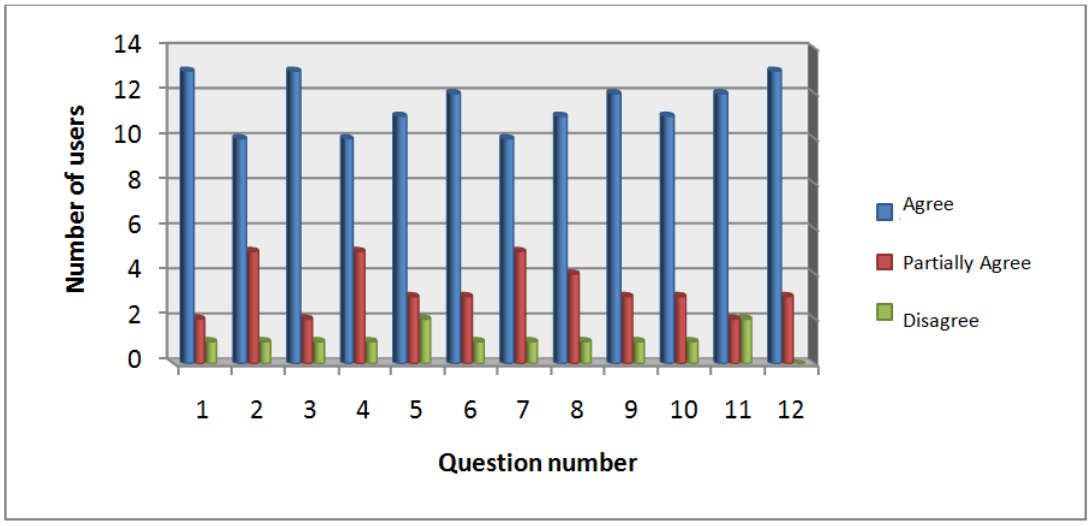

Figure 7. Summary of the remote testing results

5.3 Performance Comparison

In this section, we compared the performance of our system against the performance of previous studied systems. We choose a sample of three Arabic websites that have different content (government, education and newspaper websites).

Figure 8 shows the number of Level A errors each system has generated.

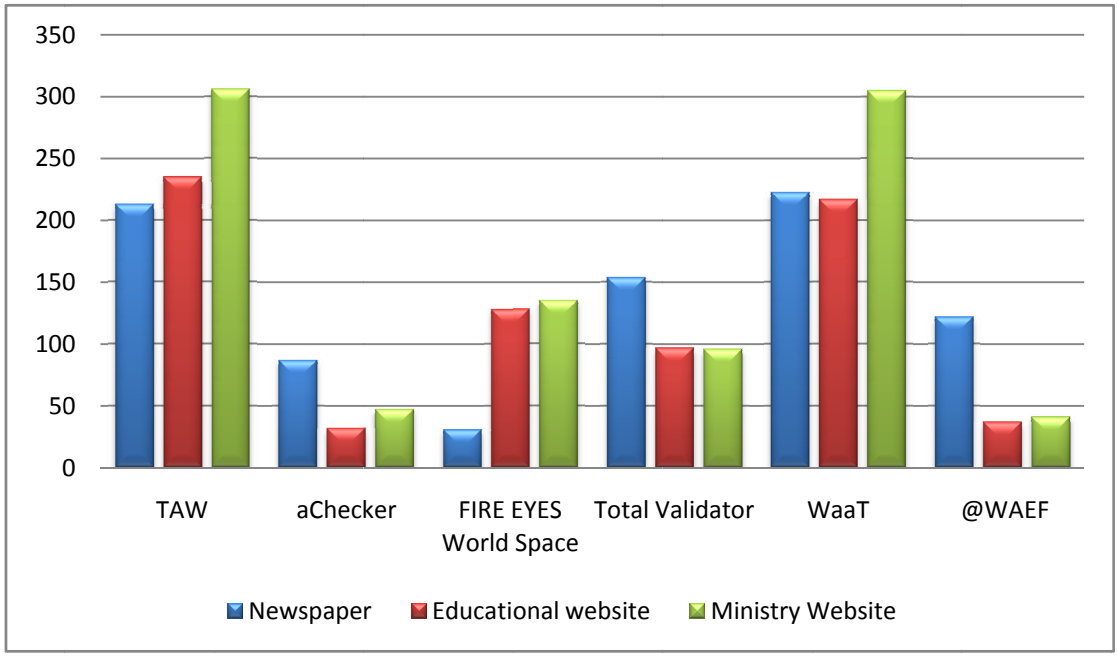

Figure 8. The accessibilty violation results for each system

To further investigate why these tools generated different errors, we performed detailed accessibility test for the Newspaper website taking into account (the number of errors only). We observed that there is a significant discrepancy between the number of errors generated by these tools. We next justify why our system sometimes generated fewer errors than other tools like (TAW and WaaT). We can summarize the reasons in the following points:

- If there are two errors or warnings with the same content in different techniques, our system does not repeat it. It only shows one of them. For example: In techniques (H44 and G162), we have the same error message in the test procedure: (Check that the label element is visible). So in this case we did not repeat the message error. We report it only once.
- Our system does not consider HTML markup errors as accessibility errors.
- Since our system did not implement all Level A techniques there are some techniques that are not applied such as audio, video and flash techniques.

Finally, as we have mentioned previously, although the above evaluated tools have powerful evaluation engines, some of them faced problems. One of the major problems was the inability to process Arabic pages; however, our system solved this problem regarding the Arabic language encoding.

6. Conclusion and Future Research

In this paper we detailed the design and implementation of an online Arabic web accessibility system designed to evaluate Arabic websites accessibility based on W3C WCAG 2.0 level A success criteria. Our system recognizes two types of accessibility problems: errors and warnings and produces two types of reports: summary report and detailed report.

Our system was evaluated using two types of testing: user acceptance via usability testing and performance testing via performance comparison. The results of the evaluation highlighted the strength and weaknesses of our system and provided recommendations for future improvements.

While working on our system we faced many difficulties that limit our work, among them are:

(1) Finding proper Arabic translation for many technical terminologies.

(2) Translating messages in a readable way. Some of the translated technical terms were not understandable by Arab web developers.

(3) Our depth method used to extract pages' URLs for a given website cannot deal with some websites that use dynamic URL generation.

Albeit the limitations in our system, the novelty of our system is present besides the adoption of the latest WCAG 2.0 accessibility standard, in overcoming of the evaluation of Arabic websites and providing a complete system with Arabic user interface. Moreover, our system proposed a new method for evaluating websites accessibility.

There are many local and global impacts of our system. From a local perspective, our system can help people interested in web accessibility to learn more about the guidelines and success criteria and how a website will be accessible in their native Arabic language. From a global perspective, our system has an impact on the field of automated web accessibility tools. As our project is considered the first Arabic Web Accessibility evaluation system that is based on WCAG 2.0, this contributes to the limited number of systems dedicated for evaluating web accessibility using WCAG 2.0.

The future work of our project is to extend our evaluation algorithms to cover level AA and AAA success criteria, as well as complete the remaining level A techniques.

Also, because the number of people accessing the web using their mobile phones is increasing rapidly, we plan to provide an engine for mobile websites accessibility.

Acknowledgment

The author is thankful to her project students (Maryam Al-Kanhal, Hailah Al-Nafisah, Noura Al-soukaih, Elaf Al-hussain and Moneerah Al-onzi) for implementing and evaluating the system. She is also thankful to the Motah grant (http://motah.org.sa) provided by King Abdul Aziz City for Science and Technology.

References

aChecker. (2012). IDI Web Accessibility Checker: Web Accessibility Checker. Retrieved August 17, 2012, from http://achecker.ca/checker/index.php

Al-Khalifa, H. S. (2010). The accessibility of Saudi Arabia government Web sites: an exploratory study. *Universal Access in the Information Society, 11*(2), 201-210. http://dx.doi.org/10.1007/s10209-010-0215-7

Brooke, J. (1996). SUS: A quick and dirty usability scale. In *Usability evaluation in industry*. Taylor and Francis.

Deque Systems. (2012). Worldspace FireEyes. Retrieved August 17, 212, from http://www.deque.com/products/worldspace-fireeyes

EOWG. (2005). Selecting Web Accessibility Evaluation Tools. Retrieved August 15, 2012, from http://www.w3.org/WAI/eval/selectingtools.html

Fernandes, N., Lopes, R., & Carriço, L. (2011). An architecture for multiple web accessibility evaluation environments. In *Proceedings of the 6th international conference on Universal access in human-computer interaction: design for all and eInclusion - Volume Part I*. UAHCI'11. Berlin, Heidelberg: Springer-Verlag, pp. 206-214. Retrieved from http://dl.acm.org/citation.cfm?id=2022591.2022616

Fuertes, J. L., Gutiérrez, E., & Martínez, L. (2011). Developing Hera-FFX for WCAG 2.0. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*. W4A '11. New York, NY, USA: ACM, pp. 3:1-3:9. Retrieved August 17, 2012, from http://doi.acm.org/10.1145/1969289.1969294

Harper, S., & Chen, A. (2012). Web accessibility guidelines. *World Wide Web, 15*(1), 61-88. http://dx.doi.org/10.1007/s11280-011-0130-8

Oikonomou, T., Kaklanis, N., Votis, K., & Tzovaras, D. (2011). An accessibility assessment framework for improving designers experience in web applications. In *Proceedings of the 6th international conference on Universal access in human-computer interaction: design for all and eInclusion - Volume Part I*. UAHCI'11. Berlin, Heidelberg: Springer-Verlag, pp. 258-266. Retrieved August 17, 2012, from http://dl.acm.org/citation.cfm?id=2022591.2022622

Shadi, A. Z. (2006). Complete List of Web Accessibility Evaluation Tools. Retrieved August 17, 2012, from http://www.w3.org/WAI/ER/tools/complete

Shawn, H. (2009). How WCAG 2.0 Differs from WCAG 1.0. Retrieved August 15, 2012, from http://www.w3.org/WAI/WCAG20/from10/diff.php

TAW. (2012). TAW (Web Accessibility Test). Retrieved August 17, 2012 from http://www.tawdis.net/ingles.html?lang=en

Total Validator. (2012). Total Validator. Retrieved August 17, 2012, from http://www.totalvalidator.com/

W3C, (2008a). WCAG 2.0 Conformance Level. Retrieved August 15, 2012, from http://www.w3.org/TR/WCAG20/#conformance

W3C. (2008b). Web Content Accessibility Guidelines (WCAG) 2.0. Retrieved from http://www.w3.org/TR/WCAG20/

Notes

Note 1. http://simplehtmldom.sourceforge.net/

Note 2. http://sourceforge.net/projects/absoluteurl/