

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/309073778>

Improving web accessibility

Article · January 2016

CITATIONS

7

READS

590

4 authors:



Islam Elkabani
Alexandria University

54 PUBLICATIONS 363 CITATIONS

SEE PROFILE



Rached Zantout
Rafik Hariri University

79 PUBLICATIONS 600 CITATIONS

SEE PROFILE



Lama Hamandi
Northeastern University

24 PUBLICATIONS 297 CITATIONS

SEE PROFILE



Simar Mansi
Beirut Arab University

1 PUBLICATION 7 CITATIONS

SEE PROFILE

Toward Better Web Accessibility

Islam Elkabani^{#&1}, Lama Hamandi^{^2}, Rached Zantout^{*3}, Simar Mansi^{#4}

[#]Mathematics and Computer Science Department, Beirut Arab University
Beirut, Lebanon

¹islam.kabani@bau.edu.lb

⁴simarmansi@gmail.com

[^]Electrical and Computer Engineering Department, American University of Beirut,
Beirut, Lebanon

²lh13@aub.edu.lb

^{*}Electrical and Computer Engineering Department, Rafik Hariri University
Beirut, Lebanon

³zantoutrn@rhu.edu.lb

Abstract— Web page evaluation systems are needed to evaluate the accessibility of websites. Such evaluation is important so that website administrators can change their websites so people with disabilities would be able to use them. Currently, such systems produce varying results that are not always useful for website administrators. Moreover it is difficult to ascertain for sure to what degree the web pages conform to accessibility guidelines. In this paper, the current state of open-source web accessibility evaluation tools is reviewed. As a result, AChecker, a system that evaluates web page compliance with WCAG 2.0, is identified as the best available. Three types of deficiencies in AChecker are identified, unclear comments, redundancy of errors reporting and lack of automatic or semi-automatic repair. An interactive evaluation tool (IWAET) is developed based on AChecker. The usability of the new system is studied. The study involved a group of web developers with different levels of programming experience and no accessibility knowledge. Both quantitative and qualitative approaches were adopted in usability evaluation. The developed system is compared to AChecker. The results of the evaluation are presented which show clearly the superiority of the new system compared to the currently available systems.

Keywords— Web Accessibility, Evaluation, Error Checking, Usability

I. INTRODUCTION

Because our world is moving more and more towards online content, this content should be accessible to everyone, including people with disabilities [1]. HTML has been designed to make web pages more accessible to people with disabilities [2]. However, website developers may not use HTML functionalities which make the website accessible to the disabled. For that purpose web accessibility has emerged as a hot topic. The goal of web accessibility is to let web developers and designers make their products accessible to all people regardless of their disability.

Web Accessibility Evaluation is an assessment procedure to verify the conformance of a website with the guidelines set by the standards organizations [3,4]. Recent studies showed that many websites are still not accessible to people with disabilities [5,6, and 7]. Experts, for example developers, perform manual or automatic evaluations. In

manual evaluation the expert manually inspects Web pages to find out problems. The expert might have potential bias while manually evaluating the accessibility quality of a Web page [6, 8]. Automatic evaluation verifies conformance with guidelines using software. Software usually minimizes the need for human intervention. Moreover, automatic testing is scalable and objective [8]. However, automatic evaluation has some limitations. Automatic evaluation cannot go to the same depth as user's evaluation nor can it be as complete. This is why both types of evaluations are used.

The results of accessibility evaluations are used by developers to correct the web pages. A considerable amount of effort is spent in such a correction. Such efforts cannot be reused among developers when they try to correct the same type of mistakes.

Adaptive technologies are devices or tools for use by people with disabilities to gain more independence [5]. Such tools allow the disabled persons to perform activities which they were not able to perform or improve the accessibility process so that they would be able to perform these activities much faster. For example visually impaired people who are not able to view text and images in a website can use screen readers or Braille terminals. Subtitles or sign language overlays help the hearing impaired access sound based content on the web. Although adaptive tools are of great help to the disabled, such tools need to be organized and structured in a specific way. A Web developer has to conform to the guidelines so that adaptive tools would be useful to the disabled while surfing the internet. This means that a system to verify that a website is indeed accessible is indispensable to developers as well as evaluators of web sites.

As a response to the increasing demand for inclusive web [9], several Web accessibility guidelines were developed. Examples of such guidelines are the American Section 508, the Italian Stance Act, the German BITV and the Web Accessibility Initiative (WAI). The WAI [10] was established by the World Wide Web Consortium (W3C). WAI also develops accessibility for mobile devices. Mobile accessibility verifies the accessibility of websites when

accessed using mobile phones. The same guidelines are used for mobile accessibility as the webpage accessibility guidelines in W3C [11]. The objective of WAI is to develop strategies, guidelines, and resources to help making the web accessible. WAI published several guidelines, some of them are known as the Web Content Accessibility Guidelines (WCAG) 1.0 and 2.0 [3].

In this paper, only the WCAG 2.0 is considered. This is because it was developed to be a single standard for Web accessibility internationally. WCAG 2.0 is technology-independent and composed of twelve guidelines. Each guideline is associated with a defined success criterion. A success criterion is marked passed or failed depending on the results of its associated techniques.

The techniques are applicable to different technologies such as HTML, CSS, Flash, or PDF. In total there are eleven different types of techniques. These are General, HTML and XHTML, CSS, Client-side Scripting, Server-side Scripting, SMIL, Plain Text, ARIA, Flash, Silverlight, and Puff techniques.

The WCAG guidelines were initially created for manual verification. Soon they were coded as part of evaluation tools also known as accessibility evaluators [12]. Such software is now able to pinpoint areas in a website/webpage where accessibility guidelines might not be respected. Such software exists nowadays as commercial software as well as open-source. Developers can use these tools to inspect their websites. This is much better than manual verification which is incomplete, error prone, time consuming and costly. The automatic evaluation tools cover only a certain part of the guidelines. Several evaluation tools are available; each tool implements parts of a specific set of guidelines. In general, the tool receives a file or an active URL as input. It then verifies the guidelines they implement and presents the results. What distinguish tools from each other are the guidelines they implement as well as the way they display their results.

In this paper, the state of accessibility verification systems is evaluated. A general observation is that most of the open-source systems were designed to be used by website developers who should have a solid background in web accessibility and its guidelines. The best open-source accessibility evaluation software (AChecker) [13] is then studied and its problems and issues are identified. Three categories of deficiencies with AChecker's results are identified, namely, redundant errors, unclear errors and errors with no repair suggestions. A system (IWAET) is then described which solves the problems and issues identified before. Redundant errors are consolidated into one item. Unclear errors are described more clearly. Suggestions for repairing the third types of errors are added so that the user would be able to repair such errors. Solving the three types of errors transformed IWAET to a system which can be used by developers independent of their background in accessibility and guidelines issues. The design, implementation and verification of this system are detailed. Moreover, description of how IWAET was used by people with different backgrounds in IT and web

accessibility is presented. Usability tests and comparison with AChecker are also presented.

In section 2, literature is reviewed regarding web accessibility. The basic technologies used in web accessibility as well as web accessibility definition, guidelines, and tools are discussed. In section 3, the accessibility evaluation systems are analyzed and compared. AChecker is identified as the best open-source tool currently available. Section 4 includes a detailed study of AChecker. AChecker is experimentally evaluated to identify its problems; suggestions to solve these problems in our new system IWAET are provided. In section 5, the system design and implementation of IWAET are presented. Section 6 contains the system evaluation of IWAET. Section 7 concludes this paper with a summary of the achievements and suggested future work.

2. LITERATURE REVIEW

AChecker [14, 15] is an online tool, which evaluates HTML pages according to the BITV 1.0, Section 508, the Stance Act and the WCAG 1.0 and WCAG 2.0. The results are distributed in levels A, AA and AAA. AChecker is a semi-automated evaluator as it cannot verify all guidelines. Three types of errors are identified namely Known, Likely and Potential. Likely and Potential problems include the problems that AChecker is not conclusive about and require human intervention to make a decision. Errors are presented by Success Criteria, identifying what is wrong and how the problems can be solved.

WAVE [16] is an evaluation tool developed by Web Aim. WAVE gives a visual representation of the page being evaluated with yellow notifications indicating errors. Notifications on the right side indicate accessibility problems. While those on the left side give a summary of the problems found. WAVE also produces a lateral report which contains the number of errors and possible errors along with some explanation. WAVE does not have a repair tool and does not show the implemented guidelines.

TAW is a free accessibility tool developed by the Spanish Foundation Centre for the Development of Information and Communication Technologies in Asturias (CTIC). It has an English version and a Spanish one. TAW tests the accessibility according to WCAG 1.0 and 2.0. The accessibility problems produced by TAW are of three categories: problems, warnings, and "not reviewed." The "not reviewed" errors have no repair options. TAW does not show the implemented guidelines [17].

Some repair tools are also available which not only identify problems but also repair them automatically. These tools are not as common as evaluation tools due to the fact that the repair process is more complex than evaluation [18].

HTML Tidy is a free tool which is capable of automatically fixing simple HTML problems and identifying some potential accessibility problems [19]. HTML Tidy allows input HTML in three forms: as a URL, as direct HTML code written or pasted into the text area, or as an uploaded file. This tool allows the user to select several options related to the repairing process. Example of such minor repair is the Break before BR which outputs a

line break before each
 element. After uploading the HTML code and selecting the “Tidy” option, the repairs are done.

Social Accessibility Project [20] has a Web page repair tool. A user who has a difficulty on a webpage communicates with the developer who is expected to correct the situation. Communication is very slow and overburdens developers with requests from many users. A solution was set forth which is to enable problems to be solved by a community of members rather than a single developer.

Each of the above systems follows a different approach to web evaluation and repair. AChecker produces the accessibility problems in a text based fashion; it asks the developer for confirmations and lacks the ability to automatically repair errors. WAVE displays tiny icons all over the page; one needs to hover over them to see their meaning. HTML Tidy is a valuable tool for simple HTML / XHTML repairs; its main concern is simple html structures and does not implement any specific accessibility guidelines. Tidy presents a list of warnings which signal errors that could not be repaired. It presents the results in a simple manner and allows downloading of repaired files. The social accessibility project uses crowdsourcing to help improve the web. However it does not implement any specific guidelines and needs constant availability of volunteers.

The system presented in this paper (IWAET) is a repair tool that implements the WCAG 2.0 guidelines. IWAET repairs as many errors as possible, suggests repairs which are known but not currently automated and displays warnings for errors which do not have known repairs.

III. ANALYSIS

The classification and usage of accessibility evaluation are studied and analyzed to find the tool that suits our study. The following features are used in the classification of web accessibility evaluation tools: The standards and guidelines used by the tool, its platform, its scope, its report style, its price (free or commercial) and its functionality (evaluation only or evaluation and repair).

The World Wide Web Consortium (W3C) published a complete list of the web accessibility evaluation tools in 2006 [11]. This list was then replaced by an updated version with the useful links of accessibility tools.

Our study will be limited to the evaluation of accessibility tools that are free, stand-alone, and implementing WCAG 2.0. Some of these tools are under open source license that can be edited and updated. The compared tools are AChecker, TAW, and WAVE. The accessibility of various websites using the three tools was tested. Then the numbers of accessibility errors produced by those tools are compared in Table I. The tested websites have Arabic content with different functionalities such as governmental and educational. Besides, the websites are issued by different countries such as Qatar, Lebanon, and Jordan. AChecker generates three categories of errors, namely known, likely, and potential errors. The errors generated by WAVE are of type errors, alerts, features, structural elements, HTML5 and ARIA, and contrast errors.

The produced accessibility errors in TAW are classified into errors, warnings, and not reviewed. The unique tool that shows the implemented guidelines is AChecker, using its database, whereas the TAW and WAVE hide the processes and guidelines of their web accessibility evaluation.

TABLE I

ACCESSIBILITY OF WEBSITES FROM QATAR, LEBANON, AND JORDAN

Website	AChecker	TAW	WAVE
www.gov.qa	540	375	171
www.qu.edu.qa	461	203	99
www.qatarairways.com	25	29	3
www.al-watan.com	1227	958	353
www.moim.gov.lb/	598	812	137
www.mea.com.lb/	167	69	23
www.lebanongovernment.org/	257	160	126
www.ul.edu.lb	1426	414	663
www.aub.edu.lb	1025	740	210
www.bau.edu.lb	1191	449	230
www.jordan.gov.jo	500	761	168
www.rj.com	89	206	64
www.ju.edu.jo	872	509	270

Analyzing the evaluation results, we remark that the number of errors reported by different web accessibility tools differs significantly; in almost all websites, AChecker produces a larger number of accessibility problems than TAW and WAVE. This means that AChecker has better performance in detecting website violations. This is why we focused our research on AChecker and ways to improve its performance and interactivity.

WCAG 2.0 contains guidelines that are related to the language of a web page; one of these guidelines specifies that the language of a web page should be specified by the programmer, so the screen reader will be able to read the content of the website. An experiment was conducted to test the accessibility of various websites containing different languages using AChecker and WAVE tools. TAW gives the user limited privilege to test the accessibility of websites, permission is needed from the development company to the full access of accessibility testing. The accessibility of the same webpage is tested for different languages, namely Arabic, English, and French and the total numbers of errors are found. Most tested websites have Arabic and English content, and have no French content. The number of errors produced by AChecker for the Arabic version of a website is found to be higher than the number of errors found in the non-Arabic version of that website. The results of WAVE vary among Arabic, English, and French versions of websites.

Another test was created to check the accessibility problem of the Arabic language. We built a sample webpage and generated three versions having English, Arabic and French content respectively. Then we tested their accessibility using AChecker and compared the number of errors for each version. The results show that the Arabic page has one more error due to the existence of the Arabic content in it; this is due to the direction of the Arabic text that is right to left. Therefore the direction should be

specified in the website. AChecker does not produce a violating error for Latin languages because the default direction is that of Latin languages which is left to right.

IV. SUGGESTIONS FOR IMPROVING ACHECKER

The database of AChecker contains all the errors that are related to different guidelines. We concentrate in this study on WCAG 2.0 guidelines.

The problems of AChecker are detected by conducting experiments to test various websites using AChecker and by using its database to study the implemented techniques. The revealed problems of AChecker are categorized into three groups which are unclear, redundant, and no-fix errors. The unclear errors occur when the description of the error is not appropriate or hard to understand. Our approach to improve AChecker is to replace this description by a simple and clear one that is understood by any developer. The redundant errors are similar errors produced more than one time. Our approach to correct this flaw is to report all redundant errors into one notification. The no-fix errors occur when AChecker has no suggested repair for the errors. Our system suggests the error correction, when the user confirms the existence of the error.

The problems detected by AChecker are categorized according to the elements that have accessibility problems. For example the problems are produced on an image, link, frame, table, header, input element, area code, strong paragraph, associated label, keyboard, embedded element, form, auto refresh, legend, and color contrast. A few of these elements will be discussed below regarding the problems with the errors generated by AChecker and our suggested solutions.

An image can cause an accessibility problem in case its "alt" attribute is missing or the "alt" attribute text is too long. In the first case, AChecker produces a text that shows how the errors are fixed without the option to correct and edit the violating code. As a better alternative, our system directly adds an "alt" element with a descriptive text entered by user. In case of a long text in the image's "alt" attribute, AChecker does not mention the limit of the text; It just produces an error "Image Alt text may be too long", and there is no suggestion or opportunity to correct the error. Our system first indicates to the user that the maximum number of characters of the text alternative of an image is 100, then it allows the user to edit the "alt" attribute of the image and reduce its length. Or as a second alternative, our system produces suggestions related to fixing the error. When the user confirms the solution, our system automatically keeps the first hundred characters of the "alt".

A link is a connection from one Web resource to another.. The text of the link should be descriptive and meaningful; AChecker produces two errors in case of an unclear link. To correct this redundancy, our system merges these errors into one that specifies that the link should describe the destination in a meaningful way.

A frame should have a title attribute that describes the purpose of the frame. In case of the absence of the frame's title, AChecker produces an error in the form of a vague text, unclear to the unexperienced developer. As a solution,

our system adds the title attribute of the frame automatically while allowing the text to be specified by the user.

The web page content should be accessible by keyboard and mouse. In case it is not, AChecker just issues errors with no suggested solutions. Our system achieves keyboard accessibility by adding automatically the equivalent of "onmousedown" event which is "onkeydown". Similarly the "onmouseout", "onmouseover" and "onmouseup" have equivalent keyboard events "onblur", "onfocus" and "onkeyup" respectively.

In conclusion, the deficiencies of AChecker can be classified into three categories redundancy, unclearness, and unrepaired errors. Table II shows the types of errors linked to a webpage element. The majority of the detected problems are related to the no repair issue.

TABLE II
TYPES OF ERRORS

Webpage Element	Redundant	Unclear	No repair
Images	X		X
Link			X
Frame			X
Header		X	
Title			X
Input Element	X		
Table	X		X
Area Code			
Strong Paragraph		X	X
Keyboard Accessible	X		
Embedded Element			
Form	X	X	
Auto Refresh			X
Color Contrast	X		X
Legend Element			X

V. SYSTEM DESIGN AND IMPLEMENTATION

This section contains the specification of our suggested accessibility evaluation system IWAET. It presents the functional and non-functional requirements of the system and its architecture. It describes the components needed, explains all the necessary design decisions, and details the implementation of the IWAET tool.

A. System Requirement

The main functions of IWAET are the categorization and repairing of accessibility errors. For this purpose, a new database is constructed that contains the types of errors with the solution for fixing them. The types of studied accessibility problems are categorized into unclear, redundant, and repair module to enhance the accessibility results. The new tool is introduced in an automated interactive interface that facilitates the accessibility testing with repairing option. When the accessibility errors are solved, the obtained results can be exported to file or printed.

The non-functional requirements of IWAET are related to performance, operational, reliability, maintainability and interoperability, and usability [21]. IWAET should be hosted on an Apache web server and MySQL database to retrieve its information. Moreover multiple users can access

the system simultaneously without installing any special software. The system supports various web browsers. More non-functional requirements are considered in the design and development of IWAET as listed below.

- Performance requirements: Average system response time shall be in proportion with the complexity of the objects that are tested.
- Operational requirements
 - Appropriate system components can generate automated error messages in case of system malfunctions and/or users mistakes.
 - All system components are easily accessible by people with different knowledge and capabilities (developers, designers, testers, etc.)
- Reliability requirements: Most modules should provide diagnostic messages in case of unsuccessful or uncertain operations.
- Maintainability and Interoperability requirements
 - All software modules developed in the project will be released under an open source license.
 - All system components will be implemented in modular, open source system architecture.
- Usability requirement: The usability of the system is calculated according to efficiency, effectiveness, and satisfaction quantitatively and qualitatively.

B. Architecture

Our tool IWAET is an extension of AChecker and is composed of five modules.

- Parse module
- Unclear module
- Redundant module
- Repair module
- Results module

Each one of the first three modules has a specific role in the evaluation process. While the Repair and Results modules are related to the repair objective.

In the parse module, IWAET communicates with AChecker and gets the accessibility results in HTML format. The parsing algorithm accesses the HTML files and gets the content of the HTML tags and saves them into an array of results. The results contain three properties which are code, description and error. The code identifies the accessibility problem by a unique identifier which is the error number. The description is a text that shows the accessibility violation according the WCAG 2.0. The error is the html code of the website where the accessibility problem exists.

The Unclear Module works on the unclear errors which are the accessibility errors with no clear description. It replaces their description with simple, understandable, and clear words. The Redundant module merges redundant errors so only one error is produced. Both unclear and redundant results are based on IWAET database.

The Repair module of IWAET is a new approach for fixing of errors. Whereas AChecker is text based and describes the error only and suggests how to fix it, IWAET corrects the errors in an interactive way, and gives the user the ability to confirm the correction. Moreover the

procedure of repairing is semi-automated to help developers who have little or no accessibility knowledge. This module requires an evaluation to run beforehand, as it must receive the results structure as input. A string is created containing the possible steps the user can follow in order to repair the problem. The repair suggestions are written in simple language so that they are understood by developers, regardless of their level of experience.

The Results module stores the accessibility problems encountered, each described by its id, description, and fixed code. These results can be reviewed in an interactive interface. They can also be exported into a report in excel format.

C Implementation

HTML is used to display pages to the user. This HTML is dynamically generated using server side scripting. The processes are executed by two levels which are the server and client side. The repair of errors is a tool that is performed by a JavaScript code. IWAET is designed to have a simple and attractive graphical user interface. After processing the results in few seconds, IWAET displays them in tabular format containing the error number, the error description and the violating code.

VI. SYSTEM EVALUATION

System evaluation was performed on IWAET and AChecker. The usability metrics of satisfaction, efficiency, and effectiveness are calculated quantitatively and qualitatively. This comparative usability study has been conducted on various websites with Arabic content using both accessibility tools. The participants in this study are developers with no previous knowledge about the web accessibility tools and the web accessibility guidelines. The aim of this experiment is to understand and correct the accessibility problems using AChecker and IWAET. Table III shows the number of redundant, unclear, and fixed accessibility errors solved by IWAET.

TABLE III
NUMBER OF REDUNDANT, UNCLEAR, AND FIXED ERRORS

	Error			Warning		
	Re	Un	fix	Re	Un	fix
www.gov.qa	1	1	2	1	2	1
www.qu.edu.qa	1	1	3	2	6	5
www.qatarairways.com	1	1	2	0	1	1
www.moim.gov.lb/	1	2	9	1	5	4
www.mea.com.lb/	1	1	2	1	3	1
www.ul.edu.lb	1	3	7	1	4	4
www.aub.edu.lb	1	3	9	2	4	6
www.bau.edu.lb	1	3	9	2	6	9
www.jordan.gov.jo	1	2	7	2	6	5
www.rj.com	1	1	2	1	4	3
www.ju.edu.jo	1	3	7	2	6	6

The user testing was conducted with the help of ten participant developers who were selected from the private industry employees as well as Beirut Arab University employees and students. All have no accessibility knowledge, although they have various levels of

programming experience. All the participants used both IWAET and AChecker as an evaluation tool for accessibility testing of the same website. Tests were all conducted on the same computer with Windows 8 operating system platform, and Google Chrome web browser.

After conducting the experiments, the participants completed a survey in order to measure their satisfaction using Single Usability Scale metric [21]. The users' satisfaction results about IWAET showed that the system brings higher satisfaction than AChecker. Satisfaction results are shown in Figure 1.

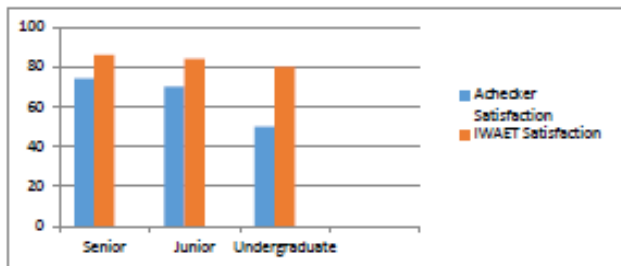


Fig. 1 User satisfaction of IWAET and AChecker

VII. CONCLUSIONS AND FUTURE WORK

This research aims at improving the existing Web accessibility evaluation tools. For this purpose we studied the available web accessibility tools and guidelines. Tools studied were free, online tools in conformance with the WCAG 2.0 guidelines and covering the highest number of guidelines. An analytical study for the AChecker evaluation Tool, which was identified as the best among the studied systems, was conducted to identify its main deficiencies. These deficiencies were identified as due to unclear comments, redundancy of errors reporting and lack of automatic or semi-automatic repair.

In this paper, the design and implementation of a new interactive web accessibility tool, IWEAT, was described. Although IWAET is based on AChecker, it overcomes the deficiencies identified in AChecker. Finally, a usability evaluation study for the IWEAT system was conducted. IWEAT was able to improve the AChecker web accessibility evaluation tool in many ways. It increased the simplicity of the tool by reorganizing the evaluation process. Also by changing the approach to the evaluation procedure, IWAET was able to display the results in an organized interactive way. The results of the usability study showed that developers with no previous accessibility knowledge are able to test the accessibility of websites using our evaluation tool (IWAET).

As future work, updating the corpus of IWAET to handle most common errors is suggested. This can be done by analyzing the accessibility errors in top rated websites and enabling IWAET to correct such errors. Moreover only the WCAG 2.0 guidelines are supported currently; the work can be extended to other guidelines such as BITV, Stanca ACT, and Section 508. Future work should also concentrate on developing a purely automatic repair tool, where developers would be freed to do more meaningful work than applying known repairs manually. This idea seems challenging since

only few techniques are straightforward enough to be repaired automatically without developers' intervention.

REFERENCES

- [1] Hackett S, Parmanto B, Zeng X, Accessibility of Internet Websites through Time, proceeding of the 6th international ACM SIGACCESS conference on Computers and accessibility, issue 32-39, p 32-39, 2004.
- [2] Al-Khalifa H. WCAG 2.0 Semi-automatic Accessibility Evaluation System: Design and Implementation, Computer and Information Science, Vol. 5, 2012.
- [3] Guarino Reid L, Snow-Weaver A, WCAG 2.0: A Web Accessibility Standard for the Evolving Web, 08 Proceeding of the 2008 International cross-disciplinary conference of web accessibility (W4A '08), p.109-115, 2008.
- [4] Harper S, Yesilada Y, Web Accessibility, Springer, London, United Kingdom, 2008.
- [5] Burgstahler S, Jirikowic T, Kolko B, Eliot M, Software Accessibility, Usability Testing and Individuals with Disabilities. Information Technology and disabilities Journal, Vol. 10(2), 2004.
- [6] Efficiency /effectiveness, Retrieved 12-12-2014 <http://www.businessdictionary.com/>
- [7] Kern, WEB 2.0 – End of accessibility analysis of most common problems with WEB 2.0, International Journal of Public Information Systems, Vol. 4, 2008.
- [8] Lopes R, Van Isacker K, Carriço L, Redefining assumptions: Accessibility and its stakeholders. In The 12th International Conference on Computers Helping People with Special Needs (ICCHP). Vol. 6179, p. 561-568, 2010.
- [9] Trewin S, Cragun B, Swart C, Brezin J, Richards J, Accessibility Challenges and Tool Features: An IBM Web Developer Perspective , Proceeding of the 2008 International cross-disciplinary conference of web accessibility (W4A '10), article No. 32, 2008.
- [10] Web Accessibility Initiatives, www.w3.org/WAI/, Retrieved 2-12-2014
- [11] HTML, www.w3c.com, Retrieved 11-12-2014
- [12] Web Accessibility Evaluation tools, <http://www.w3.org/WAI/ER/tools/>, Retrieved 25-12-2014
- [13] AChecker: IDI Accessibility, <http://www.atutor.ca/achecker/>, Retrieved 1-1-2015
- [14] Fukuda K, Saito S, Takagi H, Asakawa C, Proposing New Metrics to Evaluate Web Usability for the Blind, Processing CHI '05 Extended Abstracts on Human Factors in Computing System, p. 1387-1390, 2005.
- [15] Gay G, Qi Li C, AChecker: Open, Interactive, Customizable, Web Accessibility Checking, W4A '10 In Proceedings of the 10th International Cross Disciplinary Conference on Web Accessibility, (W4A), New York, NY, USA, Article 23, 2010.
- [16] WAVE - web accessibility evaluation tool, www.wave.webaim.org , Retrieved 1-10- 2014.
- [17] Inicio, www.tawdis.net ,Retrieved 1-11-2014
- [18] Fernandes N, Carriço L, Assessing the effort of repairing the accessibility of web sites, Proceedings of the 13th international conference on Computers Helping People with Special Needs, Vol. part 1, p. 369-403, 2012.
- [19] Tidy, <http://tidy.sourceforge.net/docs/quickref.html>, Retrieved 12-12-2014
- [20] The Social Accessibility Project, http://www.research.ibm.com/social/projects_sap.shtml, Retrieved 25-10-2015
- [21] Brooke J. ,SUS: A Retrospective, Journal of Usability Studies, United Kingdom ,Vol. 8, p. 29-40, 2013.