Inspection-based, Automatic Support for Obtaining Usable Web Sites for Vision Impaired Users

Francesco Correani, Barbara Leporini, Fabio Paternò
ISTI - C.N.R.
56124 - Pisa, Italy
{ francesco.correani, barbara.leporini, fabio.paterno}@isti.cnr.it

Abstract. The aim of this work is to provide designers and developers of Web applications with support to obtain systems that are usable for vision-impaired users. To this end, we have defined a number of design criteria to improve Web site navigation through screen readers or other similar devices. A navigation test by blind and vision-impaired subjects showed that our criteria improved Web site usability both qualitatively and quantitatively. An inspection-based tool has been developed to ease application of such criteria. Its main features are presented along with a discussion of some application results.

1 Introduction

In recent years interest in accessibility and usability issues has been increasing, since it is more and more important that the information on the Internet be easily reached by all categories of users. However, these issues are often addressed by two separate communities with two different focuses, one on usability and the other on accessibility. Indeed, the W3C consortium has developed guidelines only for accessibility, whereas in the human-computer interaction area many methods aim to evaluate only usability aspects. Vision-impaired users need to have both accessible and usable applications. Recently, designers and developers are becoming aware that there is a need for integrating these two aspects in order to obtain Web sites for a wide variety of users, including those with disabilities. Indeed, if such integration is lacking then it is possible to obtain usable sites with low accessibility (sites easy to use but not accessible for users with disabilities) or vice versa accessible sites but with low usability (where even users with disabilities can perform their tasks but with difficulty or at least not easily as it could be).

When only accessibility guidelines are applied a number of navigational problems can be found if the user interaction is performed through screen readers or magnifiers:

- *Lack of page context* – reading through the screen reader or a magnifier the user may lose the overall context of the current page and read only small portions of texts. For example, when skipping from link to link with the tab key, a blind user reads the link text on the braille display or hears it from the synthesizer (e.g. ".pdf", "more details", etc.), but not what is written before and after.

- *Information overloading* – The portions of the page (links, frames with banners, etc.), which are fixed across multiple pages, may overload the reading through a screen reader, because the user has to read every thing almost every time, thus slowing down the navigation.
- *Excessive sequencing in reading the information* – the command for navigating and reading can constrain the user to follow the page content sequentially. Thus, it is important to introduce mechanisms to ease the identification of precise parts in the page. An example is the result page generated by a search engine. Usually, at the beginning of such pages, there are several links, advertisements, the search fields and buttons, and so on, and only after such elements the actual search results begin.

To overcome these problems, we have developed a number of criteria [5] aiming at identifying the meaning of usability when Web sites are accessed by users with disabilities through screen readers. In this paper we present a tool that provides support for designers and developers interested in applying such criteria.

In the following sections, after discussing related work we present the proposed criteria, then we introduce the tool developed to support designers and developers interested in applying such criteria. Lastly, some concluding remarks are drawn along with indications for future work.

2. Background

2.1 Related Work

Well-defined criteria and guidelines must be provided to guide designers in the development process of more usable and accessible Web sites. Up to now, usability and accessibility guidelines have usually been proposed separately, whereas we propose an integrated approach. Many detailed usability guidelines have been formulated for both general user interfaces and Web page design. Most accessibility issues are taken into account especially by W3C (World Wide Web Consortium) in the Web Accessibility Initiative (WAI), in which a set of specific guidelines and recommendations has been defined: "Web Content Accessibility Guidelines 1.0" (WCAG 1.0) [14]. Currently, a new version 2.0 of Web Content Accessibility Guidelines as a Recommendation is in progress [15]. A number of tools (for example, BOBBY [3], LIFT [12]) have been proposed to identify accessibility problems mostly following the guidelines of Section 508 and W3C.

LIFT also supports usability criteria for users without disabilities but do not support specific usability criteria for users accessing through screen readers.

There are various international projects involving accessibility and usability of user interfaces for people with special needs [7]. Stephanidis' group has long been working on user interfaces for all, elaborating methods and tools allowing the development of unified user interfaces [9]. In the AVANTI project, a "Unified Web Browser" has been developed: it employs adaptability and adaptivity techniques, in order to provide accessibility and high-quality interaction to users with different abilities and needs (e.g., blind users or those with other disabilities). In particular, for vision-impaired people, it incorporates techniques for the generation of a list of large push buttons containing the links of a page. However, apart from this feature, which is similar to some checkpoints of the criteria proposed in our work, the AVANTI browser focuses on accessibility issues, but does not specifically address navigation usability through screen readers. The analysis of Web site accessibility and usability by means of guidelines, similarly to other inspection methods used in usability/accessibility assessment, requires observing, analysing and interpreting the Web site characteristics themselves. Since those activities involve high costs in terms of time and effort, there is a great interest in developing tools that automate the process of registration, analysis and interpretation of these accessibility data. Ivory & Hearts [4] distinguish between automatic capture, analysis and critique tools. Automatic capture tools assist the process of collecting relevant user and system information. Examples of such tools are Web server logging tools and client-side logging tools (e.g. WebRemUsine [8]), and so on.

Many automatic evaluation tools were developed to assist evaluators with guidelines review by automatically detecting and reporting violations and in some cases making suggestions for fixing them. EvalIris [1] is an example of tool that allows designers and evaluators to easily incorporate new additional accessibility guidelines. The tool proposed herein aims at working on the basis of the checkpoints associated with the criteria, in order to evaluate and repair Web sites through an interactive process with the evaluator/developer.

Regarding usability of Web site navigation from the perspective of users with disabilities, Barnicle [2] reports on some first usability testing of GUI applications for blind and vision-impaired users interacting through screen readers. However, despite progress in screen reader development, blind users of GUI applications still face many obstacles when using these applications. In [12] a study about usability of Web site navigation through screen readers is discussed. In particular, this work addresses accessibility supported in the 508 standard [11]. Indeed, through a user testing conducted with 16 blind users, they showed the lack of support of usability criteria according to the 508 standard guidelines. From the empirical evaluation, they suggested 32 guidelines aimed at improving usability for blind users. Some of those guidelines are furnished for Web site developers and others for screen reader developers. As a result, an unstructured and unorganised list of guidelines was proposed. Such list appears difficult to use as reference by developers because of the lack of a clear structure and organization of the guidelines. In contrast to their approach, we have sought to formulate general principles according to the three main properties of usability of its standard definition. In addition, we have classified the criteria on the basis of their impact on the Web user interface. Furthermore, although further investigations are in progress, at the moment the guidelines proposed in [12] refer only to blind users and do not consider low vision users. Besides, some important aspects for blind users are not considered, such as “messages and dynamic data management”, “sound usage”, “appropriate names for frames or tables”, etc. Considering how their guidelines are expressed, it is likely to be difficult to perform automatic evaluation of them. Moreover, no indication is given about the development of a tool for their automatic support.

2.2 The Proposed criteria

The usability of a Web site depends on many aspects. In order to improve the navigability through a screen reader, we propose 19 criteria [5] that we have divided into three subsets according to different aspects of usability indicated by the standard usability definition (ISO 9241): "effectiveness" criteria (5) ensure the accomplishment of the task, for example using a logical partition of interface elements or ensuring a proper link content, "efficiency" criteria (10) shorten the time required to complete that task, for example using proper names for frames, tables and images or providing importance levels for the elements or identifying the main page content; "satisfaction" criteria (4) provide Web pages with minor additional characteristics (addressed to improve the navigation) without the need to use specific commands.

The other parameter that we have used to classify the criteria is the user interface aspect involved: presentation criteria are indicated by an “a” and those related to the user interface dialogue by a “b”. Table 1 shows the list of the proposed 19 criteria. To identify each criterion we used the format I.J.L where: I denotes the usability aspect addressed, that is 1 for effectiveness, 2 for efficiency, or 3 for satisfaction; J is a progressive index number to enumerate the criteria ($j=1..N_i=5|10|4$); L can be a (presentation) or b (dialogue) to indicate the aspect type on which the criterion has an effect.

Effectiveness	1.1.b Logical partition of interface elements
	1.2.a Proper link content
	1.3.b Messages and dynamic data management
	1.4.a Proper style sheets
	1.5.b Layout and Terminological Consistency

Efficiency	2.1.b Number of links and frames 2.2.b Proper name for frames, tables and images 2.3.a Location of the navigation bar 2.4.b Importance levels of elements 2.5.b Keyboard shortcuts 2.6.a Proper form use 2.7.b Specific sections 2.8.b Indexing of contents 2.9.b Navigation links 2.10.b Identifying the main page content
Satisfaction	3.1.b Addition of short sounds 3.2.a Colour of text and background 3.3.a Magnifying at passing by mouse 3.4.a Page information

Table 1 - List of the proposed criteria

The 19 formulated criteria address usability issues of Web interfaces when a screen reader is used. The criteria intend to be general principles that should be considered by Web designers/developers during the development phases of a Web site. Such principles are aimed at structuring user interface elements and content of the page, as well as providing additional features, which can help users to move about better in the Web site through a screen reader. Some possible examples of criteria application are: appropriately marking the navigation bar and side-menu; logically spreading out the content in the page; using meaningful names and labels for textual/graphical links and buttons; keeping consistency among pages. Many criteria affect visually the Web interface (e.g. coloured areas or element magnifications), whereas others are detected only by the screen reader (e.g. hidden labels or names of frames).

To facilitate their application by Web site developers, we have suggested 54 technical checkpoints. A checkpoint is a specific construct in a language for Web page development that, when provided, it supports the application of a given criterion. For each criterion, we provide a number of different technical solutions to support it, taking into account developers' choices in building the Web site (e.g., frames, JavaScripts, etc.). Thus, usability aspects are addressed in terms of associated criteria, while the technical solutions in terms of checkpoints. For instance, the criterion "proper form usage" has four checkpoints related to: (1) button labels, (2) groups of control elements, (3) Onchange event (to be avoided), (4) matching labels and input elements; whereas, the criterion "Loading suitable style sheets" has three checkpoints referring to different devices: (1) voice synthesizer, (2) display and (3) printer Braille device.

In spite of the effort of providing an objective classification of the criteria, the inclusion of some of them in one group rather than in another may be somewhat open to personal interpretation although this is rarely significant.

2.3 Empirical testing of the criteria

In order to estimate the impact of our proposed criteria on the user interface for visually-impaired users, a user testing was conducted [6]. Usability testing provides an evaluator with direct information regarding the way people use applications and the problems they encounter when they use the tested interface. In our case, the test was conducted with two groups of users: twenty blind and visual impaired people were recruited for the testing. All the participants had been using Windows 98/ME and Jaws (as screen reading application) for at least one year at the moment of the testing. Thus, it could safely be assumed that they were adept at using the combination of a screen reader and Windows with the Internet Explorer browser.

Half of the participants were blind and the other half had a partial vision deficit: in any case, no-one could spot elements on the screen without an auxiliary support. The experience with the screen reader was extremely different within the group of participants, their level ranging from beginner to expert. The testing procedure adopted was based on two remote evaluation techniques (remote logging complemented with a remote questionnaire) and was performed by using two Web site prototypes and two tests, each one composed of 7 assigned tasks. The remote evaluation was used for capturing objective data: participants used the system to complete a pre-determined set of tasks while the system recorded (via log files) the results of participants' interaction (i.e. time spent), whereas through the questionnaire subjective information on navigation quality were collected from users (e.g. opinions about sound usage, colour contrast, shortcut preference, tasks more difficult, and other personal considerations), and other qualitative data not obtainable by the logging tool were collected.

For our testing, we considered a Web site containing specific information about the “The Tuscan Association for the Blind” (Unione Italiana Ciechi – Regione Toscana). This testing site was chosen with the intent of putting blind people in a comfortable situation by providing them with familiar information, thus reducing navigation difficulties.

Two versions of the same Web site prototype were considered: a "control version" implemented without applying our criteria (used as control in our testing protocol) and an "implemented version" created according to our 19 criteria. Practically, in the “implemented version” we applied all the proposed criteria analysing how to apply one checkpoint instead of another (e.g. heading levels rather than frames for logical partition). However, we tried to cover most checkpoints by applying various solutions for different pages (remember that one criterion can be applied through several checkpoints). Half of the participants were asked to start with the “control version” and the other half with the “implemented version”. The testing procedure was conducted through two sessions driven automatically: “*test0*” (control version) and “*test1*” (implemented version). A Wizard was implemented with the purpose of indicating the assigned sequence of tasks to perform (necessary for the subsequent evaluation) to the users without constraining participants’ behaviour. Each participant was asked to carry out a set of seven tasks per test (from easier to more difficult), in the two Web sites. The time required for performing the tasks was recorded in both cases. The tasks included common navigation operations, such as page opening, content reading, and information search. *Test0* and *test1* included the same types of tasks, which differed only in some minor aspects (e.g. the file to download, the information to find, etc.).

During both testing procedures, the main interaction activities performed by each user were captured and logged. The log files contained a wide variety of user actions (such as mouse clicks, text typing, link selections…) as well as browsing behaviour, such as page loading start and end. In particular, the tool logged the time when each specific interaction was performed. Consequently, it was possible to compute the time spent for carrying out each task as the difference between the times corresponding to the beginning and the end of the task. Thus, all data gathered through the testing procedure were analysed in order to evaluate the overall improvement of the Web site after the application of our criteria. Such improvement was measured in terms of navigation time saved by users in accomplishing given tasks.

At the end of the testing procedures, two log files per users were available. Each log file contained a set of couples <event type / performing time>, which were used to compute the time spent per task. The difference between the time spent performing each task in *test0* and *test1* (“control site” and “implemented site” respectively) was used to verify if and to what extent the application of our criteria had improved navigability. So, the time saved by users was taken as an indicator of Web site improvement.

In order to assess the statistical significance of the data analysed, non-parametric statistic tests were applied to raw data: α was fixed at 0.05 (significance) and 0.01 (high significance). We found a significant difference between the total time spent by all users performing each task in *test0* and *test1* (Wilcoxon matched pairs test). For each task, the total time was calculated by summing the time spent by each user (twenty users). We also found a highly significant difference between the total time spent by each user performing all the given tasks in *test0* and *test1* (Wilcoxon matched pairs test). For each user, the total time required in *test0* and *test1* was computed by summing the time required for each task. The time analysis revealed a wide range of differences for tasks and users, possibly due to the different ability of users/difficulty of tasks. However, on average, application of our criteria to the Web site has led to a significant time saving for all users and tasks.

Among the seven tasks, the "looking for information in a long page" task turned out to be the most influenced by our criteria. This is likely due the fact that low vision users could considerably reduce their navigation time by using the side submenus (e.g. local links or list boxes) to move quickly to a specific section of the page, while blind users cut navigation time thanks to the list of heading levels generated by the screen reader commands.

In conclusion, our results showed that both blind and low vision users benefited from application of our criteria, which saved them around 40% in terms of navigation time.

3. NAUTICUS: A Tool Supporting Usability Criteria

Basing on the encouraging feedback from the testing, we decided to create automatic support for our criteria, especially addressed to developers who want to make their Web sites usable for vision-impaired users. Current tools only support accessibility (e.g. Bobby) or usability evaluation (e.g. WebRemUSINE) but not both of them when users access through a screen reader.

3.1 The Tool Goals

The NAUTICUS tool (New Accessibility and Usability Tool for Interactive Control in Universal Sites) has been developed with the intent of checking whether a Web site is usable for users interacting through screen readers. To this end, the tool checks how satisfactorily our criteria are applied to the code of Web pages. This is obtained through automatic identification of the checkpoints associated with each criterion and analysis of the associated constructs and attributes to check whether they provide the necessary information. Through a list of checkboxes spread into tree panels, the evaluator can decide which criteria have to be checked.

The tool is not limited to checking whether the criteria are supported but, in case of failure, it also provides support in modifying the code in order to make the resulting Web site more usable and accessible. Thus, it points out what parts of the code are problematic and provides support for corrections indicating what elements have to be modified or added. The process is not completely automatic because in some cases the tool requires designers to provide some information that cannot be generated automatically. Examples of criteria that require the designer's intervention are:

- Proper link content, in this case the tools asks the designer to provide meaningful text for the link;

- Proper style sheets, in this case the tool requires an indication of the file containing the external style sheets;
- Proper names for frames, tables and images; here the designer may have to provide the value for the summary attribute for tables or for the alt attribute associated with images. This can also happen for frame titles and names. In the event the two values are different, then the tool makes them consistent and provides the designer with the possibility of modifying the resulting value.

A useful feature of NAUTICUS is that the tool can be applied to Web sites written in different languages. In fact, NAUTICUS uses external dictionary files where some typical and common terms necessary for the evaluation of some specific criteria (e.g. appropriate names for frames or for link texts) are stored. Therefore, by simply loading a different dictionary file containing terms in a different language, the evaluation can be applied to that language. So, it is not necessary recompiling again the application, it is sufficient writing just the files of the external dictionaries.

3.2 The Tool User Interface

The main layout of the tool user interface is structured into three main areas:

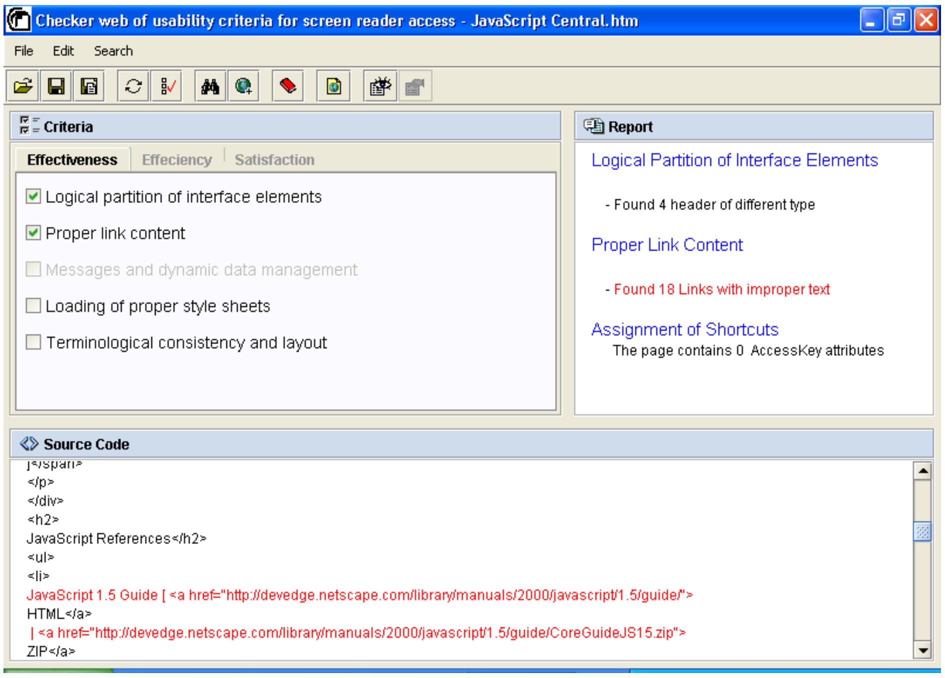
(1) *Criteria*, which provides access to the supported criteria selections;

(2) *Report*, with the results of the selected criteria analysis relating to the loaded page;

(3) *Source Code*, which shows the source code of the loaded page;

Supported criteria are grouped depending on the main usability aspect to which they refer and they are visualized using a tabbed pane providing access to the various groups. The designer can select the application of all or only part of them through check-boxes. This feature can be useful for better focusing on specific criteria by highlighting the involved parts limited to the selected criteria.

In the report, blue labels are used to indicate the criteria analysed, while the elements that do not satisfy the criteria are highlighted in red, and the black parts are considered to satisfy the criteria. In the case of Figure 1, the criteria regarding “Logical partition of interface elements“, “proper link content“ and “Assignment of shortcuts” have been selected and the report with the corresponding list of issues is displayed in the “report” panel. For each issue, the number of occurrences is indicated as well. The evaluation is obtained by selecting the relative checkboxes of the 3 criteria selected, and checking the loaded page (Figure 2 shows it).


Checker web of usability criteria for screen reader access - JavaScript Central.htm
File
Edit
Search
Criteria
Effectiveness
Effeciency
Satisfaction
Report
Source Code

- [✓] Logical partition of interface elements
- [✓] Proper link content
- [ ] Messages and dynamic data management
- [ ] Loading of proper style sheets
- [ ] Terminological consistency and layout

Logical Partition of Interface Elements

- Found 4 header of different type

Proper Link Content

- Found 18 Links with improper text

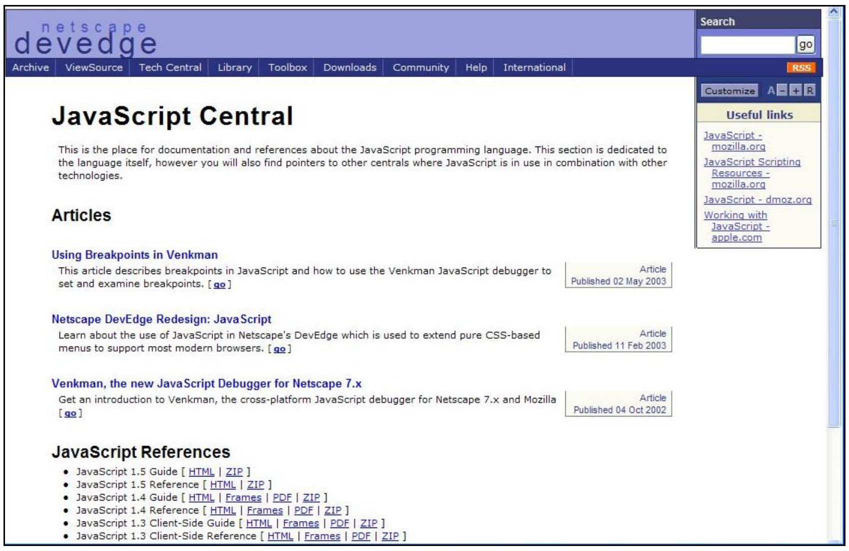
Assignment of Shortcuts
The page contains 0 AccessKey attributes

```
]</span>
</p>
</div>
<h2>
JavaScript References</h2>
<ul>
<li>
JavaScript 1.5 Guide [ <a href="http://devedge.netscape.com/library/manuals/2000/javascript/1.5/guide/">
HTML</a>
 | <a href="http://devedge.netscape.com/library/manuals/2000/javascript/1.5/guide/CoreGuideJS15.zip">
ZIP</a>
```

Figure 1: Tool output related to a criteria-based evaluation

As the page is structured in several sections by using heading levels (i.e. <h1>…<h4>), the tool reports (in black) that four heading levels have correctly been used. Whereas, regarding link text and title, NAUTICUS points out (in red) that 18 links could not have appropriate texts or titles.

Noting that in order to perform an evaluation of the frame names, appropriate links, adequate summaries for tables, and so on, a complete objective evaluation can not be done. So, for this purpose we defined a set of dictionary files in which a list of "wrong terms" or "appropriate potential terms" are stored. For example, terms such as "click here", "here", "pdf", "more information", and so on, are stored as inappropriate text for links; or names such as "left", "central", "sx", etc., are listed as frame names to be avoided. All these files can be updated and modified, so that evaluators can customize them. In addition, the use of such dictionaries allows designers to change languages. Therefore, changing a language implies changing the dictionary used for evaluating / repairing.



JavaScript Central

This is the place for documentation and references about the JavaScript programming language. This section is dedicated to the language itself, however you will also find pointers to other centrals where JavaScript is in use in combination with other technologies.

Articles

Using Breakpoints in Venkman
This article describes breakpoints in JavaScript and how to use the Venkman JavaScript debugger to set and examine breakpoints. [go]
Article Published 02 May 2003

Netscape DevEdge Redesign: JavaScript
Learn about the use of JavaScript in Netscape's DevEdge which is used to extend pure CSS-based menus to support most modern browsers. [go]
Article Published 11 Feb 2003

Venkman, the new JavaScript Debugger for Netscape 7.x
Get an introduction to Venkman, the cross-platform JavaScript debugger for Netscape 7.x and Mozilla [go]
Article Published 04 Oct 2002

JavaScript References
- JavaScript 1.5 Guide [HTML | ZIP]
- JavaScript 1.5 Reference [HTML | ZIP]
- JavaScript 1.4 Guide [HTML | Frames | PDF | ZIP]
- JavaScript 1.4 Reference [HTML | Frames | PDF | ZIP]
- JavaScript 1.3 Client-Side Guide [HTML | Frames | PDF | ZIP]
- JavaScript 1.3 Client-Side Reference [HTML | Frames | PDF | ZIP]

Useful links
JavaScript - mozilla.org
JavaScript Scripting Resources - mozilla.org
JavaScript - dmoz.org
Working with JavaScript - apple.com

Figure 2: The Example Considered.

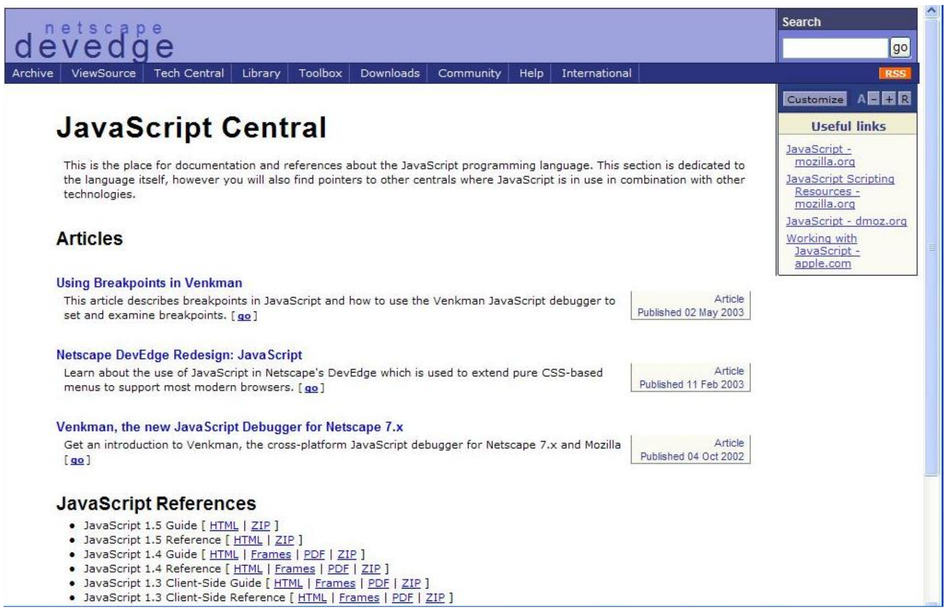
3.3 Some examples of criteria evaluation

In order to show how the NAUTICUS tool evaluates Web pages according to our proposed criteria, we report on some examples regarding the "Logical partition of interface elements", "Proper link contents" and "Proper name for frames, tables and images" criteria.

Proper link content

Let us consider the page of the Informatics Ph.D. Web site of the University of Pisa. This page contains several links pointing to different formats of application forms which can be downloaded: rtf, pst, and latex. As each link has a short text like “pdf” or “latex” and no longer titles has been used, NAUTICUS found 8 non-appropriate links.

A similar result was obtained for the page shown in Figure 3, which is an page index of a java script reference manual. Each topic can be read in html, pdf or zipped format. Therefore, several links such as “pdf”, “html”, or “zip” are used for linking the corresponding resource. By evaluating that page NAUTICUS points out 18 non-appropriate link.



JavaScript Central

This is the place for documentation and references about the JavaScript programming language. This section is dedicated to the language itself, however you will also find pointers to other centrals where JavaScript is in use in combination with other technologies.

Useful links

JavaScript - mozilla.org
JavaScript Scripting Resources - mozilla.org
JavaScript - dmoz.org
Working with JavaScript - apple.com

Articles

Using Breakpoints in Venkman
This article describes breakpoints in JavaScript and how to use the Venkman JavaScript debugger to set and examine breakpoints. [go]
Article
Published 02 May 2003

Netscape DevEdge Redesign: JavaScript
Learn about the use of JavaScript in Netscape's DevEdge which is used to extend pure CSS-based menus to support most modern browsers. [go]
Article
Published 11 Feb 2003

Venkman, the new JavaScript Debugger for Netscape 7.x
Get an introduction to Venkman, the cross-platform JavaScript debugger for Netscape 7.x and Mozilla [go]
Article
Published 04 Oct 2002

JavaScript References

- JavaScript 1.5 Guide [HTML | ZIP]
- JavaScript 1.5 Reference [HTML | ZIP]
- JavaScript 1.4 Guide [HTML | Frames | PDF | ZIP]
- JavaScript 1.4 Reference [HTML | Frames | PDF | ZIP]
- JavaScript 1.3 Client-Side Guide [HTML | Frames | PDF | ZIP]
- JavaScript 1.3 Client-Side Reference [HTML | Frames | PDF | ZIP]

Figure 3: Another example of problematic Web page.

Proper name for frames, tables and images: tables

Let us consider the main page of the well-known search engine Google (see Figure 4). For rendering the page content one layout table has been used for the navigation links – i.e. Web, Images, Groups, News, and so on – which is a table with one row and one column. It should be useful applying for this table a summary attribute like “navigation bar” or “navigation links” so that when the screen reader recognises that table, it reads the summary content and the user can understand more promptly its purpose. A second table is used for rendering the search field and buttons: it is not recognised as layout table because it has one row and three columns; in any case also this table has not summary attribute. A summary like “search form” could be used in order to facilitate its identification.

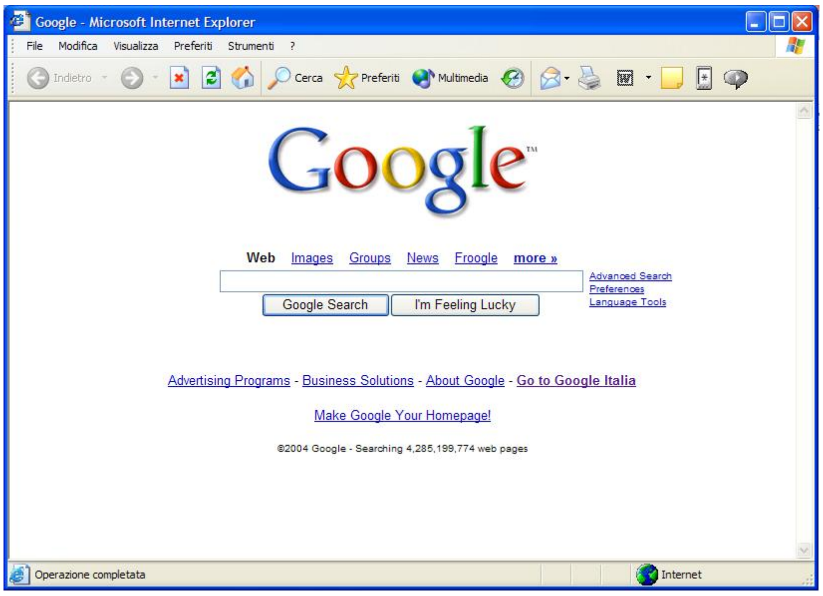
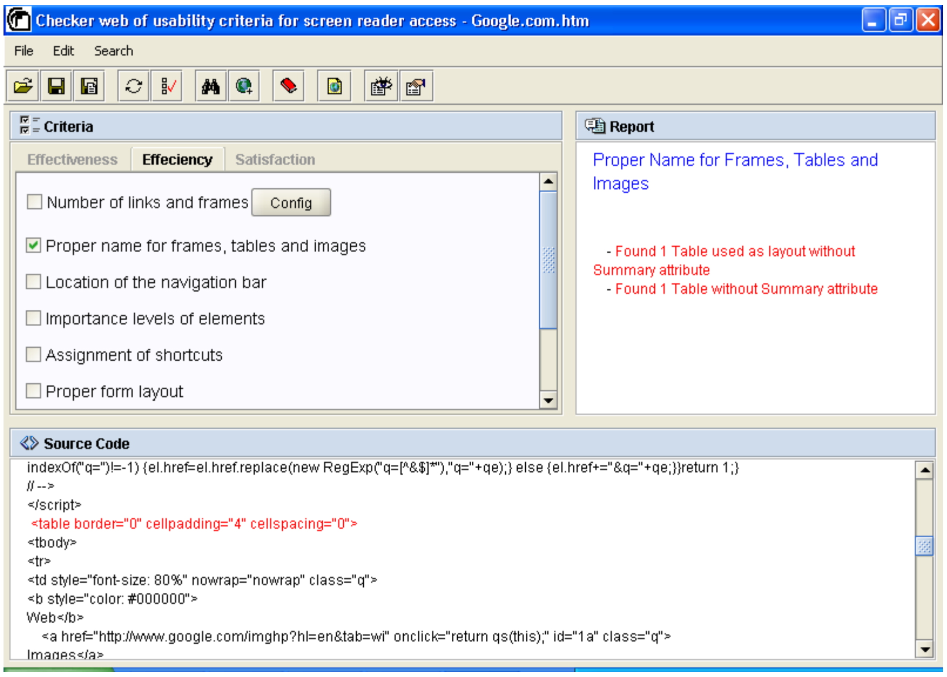

Google - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro Cerca Preferiti Multimedia
Web Images Groups News Froogle more »
Google Search I'm Feeling Lucky
Advanced Search
Preferences
Language Tools
Advertising Programs - Business Solutions - About Google - Go to Google Italia
Make Google Your Homepage!
©2004 Google - Searching 4,285,199,774 web pages
Operazione completata
Internet

Figure 4: The Google Interface.

Next figure shows the result found out when evaluating the criteria “proper names for tables” for the Google page: in “report” panel the two tables without summary are notified.



- [ ] Number of links and frames
- [x] Proper name for frames, tables and images
- [ ] Location of the navigation bar
- [ ] Importance levels of elements
- [ ] Assignment of shortcuts
- [ ] Proper form layout

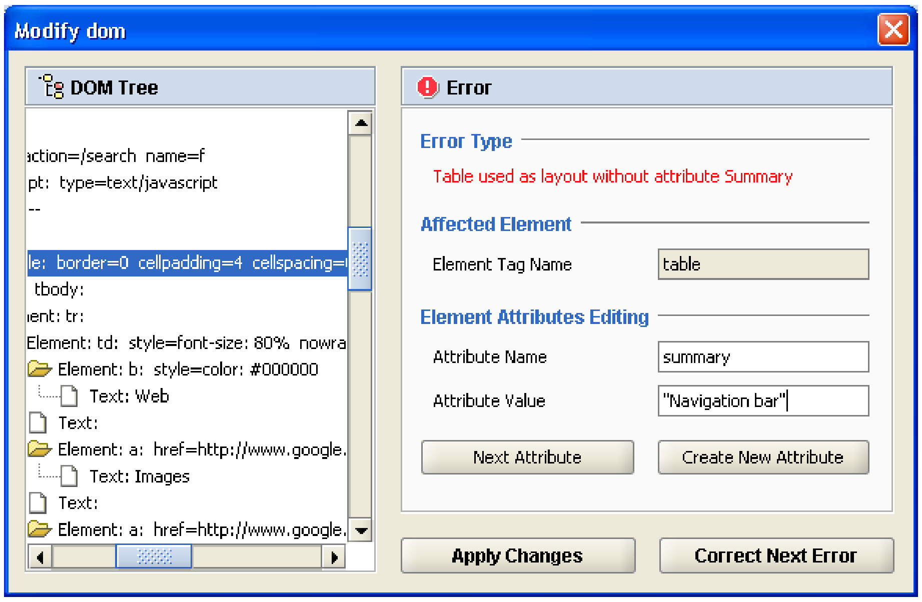
Proper Name for Frames, Tables and Images

- Found 1 Table used as layout without Summary attribute
- Found 1 Table without Summary attribute

```
indexOf("q=")!=-1) {el.href=el.href.replace(new RegExp("q=[^&$]*"),"q="+qe);} else {el.href+="&q="+qe;}}return 1;}
// -->
</script>
 <table border="0" cellpadding="4" cellspacing="0">
<tbody>
<tr>
<td style="font-size: 80%" nowrap="nowrap" class="q">
<b style="color: #000000">
Web</b>
   <a href="http://www.google.com/imghp?hl=en&tab=wi" onclick="return qs(this);" id="1a" class="q">
Images</a>
```

Figure 5: Evaluation of the Google Interface with NAUTICUS.

The code can be corrected either through the DOM (Document Object Model) [13] or by editing the page. The Document Object Model is a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of Web documents. The document can be further processed and the results of that processing can be incorporated back into the presented page. Our tool supports direct access to the DOM: the designer has to select a criterion and then activate the analysis and ask for correction. At this point, the tool shows the corresponding interface (see an example in Figure 6). In the left part there is a tree representing the DOM elements of the current page, where it is possible to distinguish the elements from the attributes and texts.



```
ıction=/search  name=f
pt:  type=text/javascript
--
le:  border=0  cellpadding=4  cellspacing=
 tbody:
ıent: tr:
Element: td:  style=font-size: 80%  nowra
  Element: b:  style=color: #000000
     Text: Web
  Text:
  Element: a:  href=http://www.google.
     Text: Images
  Text:
  Element: a:  href=http://www.google.
```

Error Type
Table used as layout without attribute Summary

Affected Element

Element Tag Name	table

Element Attributes Editing

Attribute Name	summary
Attribute Value	"Navigation bar"

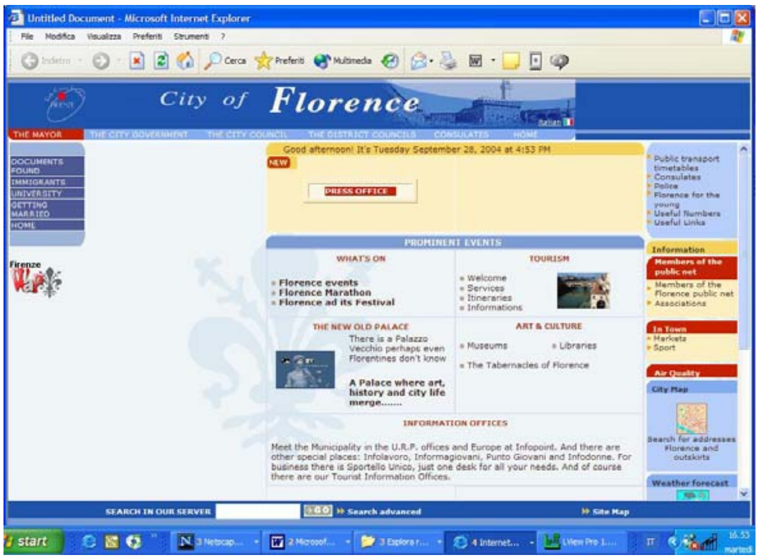
Figure 6: Tool-support identification and repair of problematic parts through the DOM

The right part displays useful information to identify and repair the problematic parts of the tags currently under analysis. It shows the error type, the affected element and its associated attributes and values.

Through the "*Correct Next Error*" button it is possible to access the next tag that raises an error according to the current criterion. The left part displays the hierarchical structure of the DOM with the possibility of folding/unfolding elements. In addition, through the controls, it is possible to scroll and modify the attributes of the selected element or create new ones. The modifications made can be saved in order to immediately apply them to the DOM. It is also possible to automatically search for the next error. In Figure 6 we can see how the tool immediately identifies the first element that does not satisfy the selected criterion (proper use of frames, tables and images). In the example it is a table. Then, the designer can edit it, for example by adding a summary attribute (i.e. summary="navigation bar").

Proper name for frames, tables and images: frames

Next figure shows what the tool reports when a page built with a frameset where each frame has generic names like “frame1”, “frame2” and “frame3”. A sample page used for evaluating this criteria belongs to the Florence City council Web site (see Figure 7).



Good afternoon! It's Tuesday September 28, 2004 at 4:53 PM

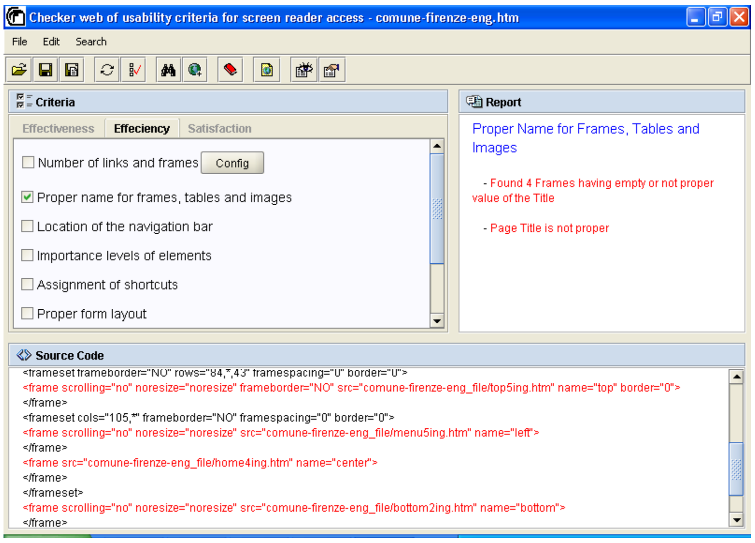
There is a Palazzo Vecchio perhaps even Florentines don't know

A Palace where art, history and city life merge.......

Meet the Municipality in the U.R.P. offices and Europe at Infopoint. And there are other special places: Infolavoro, Informagiovani, Punto Giovani and Infodonne. For business there is Sportello Unico, just one desk for all your needs. And of course there are our Tourist Information Offices.

Figure 7: Another example considered.

This page has four non-appropriate names for frames: top, left, center and bottom. Therefore, the tool detects that there are four non correct names or titles for frames.


Checker web of usability criteria for screen reader access - comune-firenze-eng.htm
File
Edit
Search
Criteria
Effectiveness
Effeciency
Satisfaction
Config
Report
Source Code

- [ ] Number of links and frames
- [x] Proper name for frames, tables and images
- [ ] Location of the navigation bar
- [ ] Importance levels of elements
- [ ] Assignment of shortcuts
- [ ] Proper form layout

Proper Name for Frames, Tables and Images

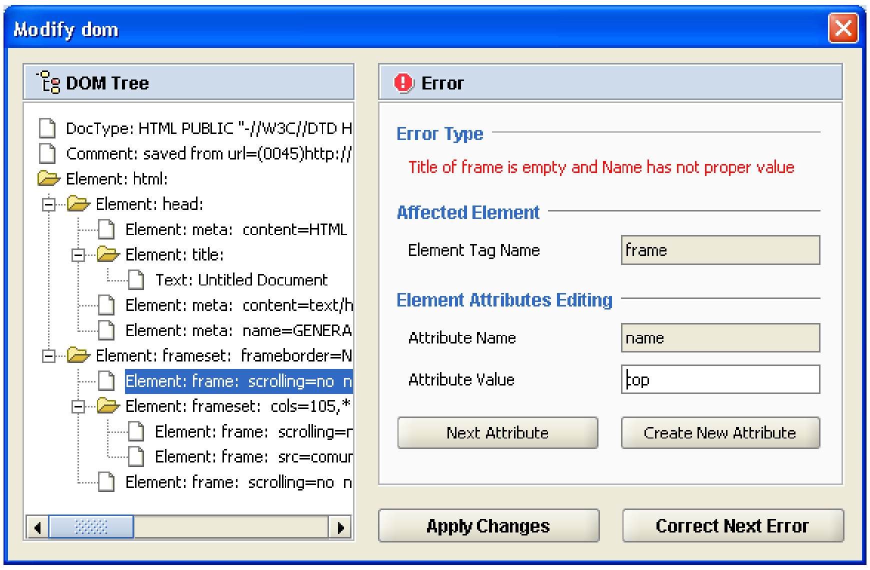
- Found 4 Frames having empty or not proper value of the Title

- Page Title is not proper

```
<frameset frameborder="NO" rows="84,*,43" framespacing="0" border="0">
<frame scrolling="no" noresize="noresize" frameborder="NO" src="comune-firenze-eng_file/top5ing.htm" name="top" border="0">
</frame>
<frameset cols="105,*" frameborder="NO" framespacing="0" border="0">
<frame scrolling="no" noresize="noresize" src="comune-firenze-eng_file/menu5ing.htm" name="left">
</frame>
<frame src="comune-firenze-eng_file/home4ing.htm" name="center">
</frame>
</frameset>
<frame scrolling="no" noresize="noresize" src="comune-firenze-eng_file/bottom2ing.htm" name="bottom">
</frame>
```

Figure 8: Evaluation of the example.

Next figure shows how the name and title attributes for each frame can be fixed through the DOM structure. The NAUTICUS tool focuses the frame tag and the attributes to be fixed; then the developer/evaluator has to write an appropriate value for the attribute. For example, in this case the name for frames could be fixed by changing "top" into "navigation links"; "left" into "submenu"; "center" into "main content"; and "bottom" into "search in the Web site". By the buttons “next attribute” or “correct next error” the evaluator can navigates through the error tree and thus fixing the found problems.


Modify dom

DOM Tree

```
DocType: HTML PUBLIC "-//W3C//DTD H
Comment: saved from url=(0045)http://
Element: html:
  Element: head:
    Element: meta:  content=HTML
    Element: title:
      Text: Untitled Document
    Element: meta:  content=text/h
    Element: meta:  name=GENERA
  Element: frameset:  frameborder=N
    Element: frame:  scrolling=no  n
    Element: frameset:  cols=105,*
      Element: frame:  scrolling=r
      Element: frame:  src=comur
    Element: frame:  scrolling=no  n
```

Error

Error Type

Title of frame is empty and Name has not proper value

Affected Element

Element Tag Name: frame

Element Attributes Editing

Attribute Name: name

Attribute Value: top

Figure 9: Error correction in the example considered.

3.3 The tool architecture

The tool has been implemented in Java. It first checks through the Tidy library whether the page is well-formed and then corrects any syntactical errors. Then, for each evaluation criterion there is a class implementing the associated algorithm to check its application. It mainly analyses the DOM to see whether the associated constructs are provided along with the necessary attributes.

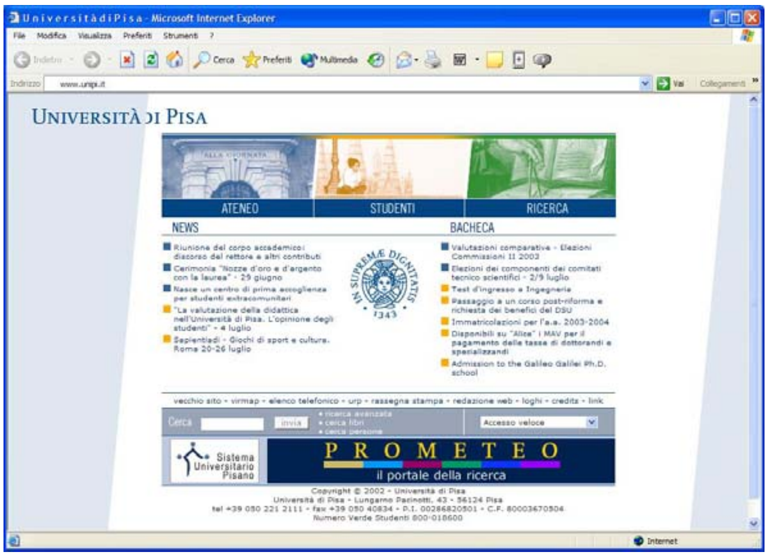
The architecture is structured in a number of modules implemented through the Java packages:

- Effectiveness: this package contains all the classes implementing the effectiveness criteria;
- Efficiency: this package contains all the classes implementing the efficiency criteria;
- Satisfaction: this package contains all the classes implementing the satisfaction criteria;

- Gui: this package contains all the classes implementing the graphical user interface of the tool;
- Utility: this package contains frequently used classes such as text analyzers, DOM manipulation, …
- Configuration: this package contains classes that handle the files that are loaded at the beginning of the application, such as dictionaries and images.
- Exception: this package contains the classes handling the exceptions of Tidy, the HTML parser.
- Org: this package contains the DOM and Tidy classes.

4. Example of Tool Application

The tool has been applied to the University of Pisa Web site (see Figure 10) and a number of problems were immediately detected: no style sheets specific for vocal synthesizers, lack of alt attributes for images used as background and in the layout, lack of summary attributes to comment the many tables used in the document. There was no use of tabindex and accessKey, which are very useful for blind users to quickly go through the Web pages.


UniversitàdiPisa - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro Cerca Preferiti Multimedia
Indirizzo www.unipi.it
Vai Collegamenti
Internet

UNIVERSITÀ DI PISA

ATENEO | STUDENTI | RICERCA

NEWS

- Riunione del corpo accademico: discorso del rettore e altri contributi
- Cerimonia "Nozze d'oro e d'argento con la laurea" - 29 giugno
- Nasce un centro di prima accoglienza per studenti extracomunitari
- "La valutazione della didattica nell'Università di Pisa. L'opinione degli studenti" - 4 luglio
- Sapientiadi - Giochi di sport e cultura. Roma 20-26 luglio

BACHECA

- Valutazioni comparative - Elezioni Commissioni II 2002
- Elezioni dei componenti dei comitati tecnico scientifici - 2/9 luglio
- Test d'ingresso a Ingegneria
- Passaggio a un corso post-riforma e richiesta dei benefici del DSU
- Immatricolazioni per l'a.a. 2003-2004
- Disponibili su "Alice" i MAV per il pagamento delle tasse di dottorandi e specializzandi
- Admission to the Galileo Galilei Ph.D. school

vecchio sito - virmap - elenco telefonico - urp - rassegna stampa - redazione web - loghi - credits - link

Cerca invia • ricerca avanzata • cerca libri • cerca persone Accesso veloce

Sistema Universitario Pisano

PROMETEO il portale della ricerca

Copyright © 2002 - Università di Pisa
Università di Pisa - Lungarno Pacinotti, 43 - 56124 Pisa
tel +39 050 221 2111 - fax +39 050 40834 - P.I. 00286820501 - C.F. 80003670504
Numero Verde Studenti 800-018600

Figure 10: The Web site considered for the tool application**.**

One of the most serious problems was that the access to some university services can be achieved only through the use of a pull-down menu (the "Accesso Veloce" element on the bottom part of the page), which was implemented by a <select> tag with a Javascript associated with the OnChange attribute. If the link were accessible through a text or an image, there would have been no problem, but the OnChange attribute creates many difficulties. Blind users often use the keyboard for navigation and the TAB key to move from one interface element to the next. When they reach the <select> element, since an onChange attribute has been defined, then the first associated link is automatically selected, even if the user is not interested in it. In order to avoid this problem it would have been sufficient to use the OnClick event instead of the OnChange, because in this case the link would have been selected only after an explicit link selection from the user. This aspect is checked through the criterion 2.6.a (proper form use).

5. Conclusions

In this paper, we have discussed a set of usability criteria to improve Web navigation for vision impaired people. Then, we have presented an automatic tool supporting such criteria and report on its application to a case study.

The tool provides interactive support for checking the application of our criteria and help designers to improve their Web sites in case it detects problems. We are extending the tool in order to support evaluation of Web sites obtained through dynamic pages.

Future work will be dedicated to further extending the evaluation tool in order to integrate it with assessment performed through other methods (such as automatic log analysis) and to support designers even in the development phase.

Acknowledgments

We thank Francesco Conversano for his help in the development of the tool and Domenico Natale (SOGEI) for useful discussions on the topics of this paper.

References

1. Abascal J., Arrue M., Fajardo I., Garay N., Tomás J., *Use of Guidelines to automatically verify Web accessibility*. Universal Access in the Information Society, special Issue on "Guidelines, standards, methods and processes for software accessibility", Springer Verlag, Vol.3, N.1, 2004, pp. 71-79.
2. Barnicle, K. *Usability Testing with Screen Reading Technology in a Windows Environment*. Proceedings of the 2000 Conference on Universal Usability (CUU-00), pp. 102-109, ACM Press, November 16-17 2000.
3. Clarck D., Dardailler D. (1999) *Accessibility on the Web: Evaluation and repair tools to make it possible*. In proceedings of the CSUN Technology and Persons with disabilities Conferences, Los Angeles, CA. Available at http://www.cast.org/bobby.
4. Ivory, M. and Hearts, M. (2001) The State of the Art in Automating Usability Evaluation of User Interfaces. ACM Computing Surveys, Vol, 33, No 4: 470-516.
5. Leporini, B., Paternò, F. (2004). *Increasing Usability when Interacting through Screen Readers*, International Journal Universal Access in the Information Society (UAIS), special Issue on "Guidelines, standards, methods and processes for software accessibility", Springer Verlag, Vol.3, N.1, pp. 57-70.
6. Leporini, B., Paternò, F. Testing the effects of web usability criteria for vision impaired users.
   ISTI-CNR Technical report, January 2004, Submitted paper.
7. Nicolle, C., Abascal J. *Inclusive design guidelines for HCI*, p. 285, Taylor & Francis, 2001.

8. Paternò, F., Paganelli, L., 2001. *Remote evaluation of Web sites based on task models and browser Monitoring*. Proceedings of CHI'01, Extended Abstracts, (Seattle, WA, USA, April 2001), pp 283-284
9. Stephanidis, C., Paramythis, A., Karagiannidis, C., Savidis, A. *Supporting Interface Adaptation: the AVANTI Web-Browser*. 3rd ERCIM Workshop on "User Interfaces for All", Strasbourg, France, November 3-4, 1997.
10. Section 508 standards. http://www.section508.gov
11. Theofanos, M.F., Redish, J (2003). Bridging the gap: between accessibility and usability. ACM Interactions magazine, New York: ACM Press, Nov.-Dec. 2003 issue, pp.36-51
12. USABLE NET (2000) LIFT ON LINE. Available at http://www.usablenet.com/
13. W3C Document Object Model (DOM) http://www.w3.org/DOM/
14. Web Accessibility Guidelines 1.0. Web Accessibility Initiative, W3C Recommendation 5-May-1999. Accessible at http://www.w3.org/WAI/GL/WCAG10/
15. Web Content Accessibility Guidelines 2.0, W3C Working Draft 1 March 2004available at http://www.w3.org/WAI/GL/WCAG20/