Automatic Inspection-based 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

CATEGORY: LONG PAPER

KEYWORDS: USABILITY AND ACCESSIBILITY, AUTOMATIC INSPECTION, VISION-IMPAIRED USERS, GUIDELINES, TOOLS

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, a number of design criteria to improve Web site navigation through screen readers or other similar devices have been defined. A user test by blind and vision-impaired subjects showed that such criteria improve Web site usability both qualitatively and quantitatively. An inspection-based tool has been developed to ease the application of the defined criteria. Its main features are presented in this paper along with some example applications and related discussion.

1. Introduction

In recent years, interest in accessibility and usability issues has been increasing, since it is more and more important that 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 (HCI) area many methods aim to evaluate only usability aspects. Vision-impaired users need to have applications that are both accessible and usable. 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 arise if 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 does not read what is written before and after the link.
- *Excessive sequencing in reading* –commands 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 a page. An example is the result page generated by a search engine. Usually, at the beginning of such pages, there are several links, advertisements, search fields and buttons, etc., and only following such elements the actual search results begin.

Such usability problems have been identified by interviewing and observing vision-impaired people accessing Web sites complying with accessibility standards. To overcome these problems, a number of criteria have been developed [5], aiming at identifying the meaning of usability when Web sites are accessed by users with disabilities through screen readers. However, just providing a set of design criteria may not be enough for the wide variety of Web designers and developers who often have to develop their applications in a short time. Thus, it becomes important to provide them also with tools able to automatically check at least part of such criteria and generate consistent evaluation results. Applying all the proposed criteria, as well as evaluating if they were used by developers, requires considerable efforts and experience. Assisting developers in editing/repairing their Web pages is the main goal of the tool presented in this paper. Currently available tools focus on accessibility guidelines, limiting

themselves to check the presence or absence of attribute values, such as verifying if ALT attributes have been defined, but without checking how they are applied. The ALT attribute is used for providing an alternative description to information not accessible to disabled users, such as images, objects or graphical links. However, no control of the appropriateness of such attributes values is usually considered, whereas the tool described in this paper is designed for evaluating automatically also their values (as much as possible). In addition, since the adopted criteria consider usability aspects, which are not addressed in other guidelines (e.g., usage of sounds), they extend the support provided by current accessibility tools. Thus, the developed tool is particularly useful for those developers who are not expert in handling page code when the particular requirements of vision-impaired users are considered. The tool provides support for designers and developers interested in applying the set of proposed criteria considering both usability and accessibility aspects.

In the following sections, after discussing related work and introducing the proposed criteria, we present the tool developed to support designers and developers interested in applying such criteria. Example tool applications are reported as well. Lastly, some concluding remarks are provided along with indications for future work.

2. 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.

2.1 Accessibility Guidelines

Many detailed usability guidelines have been formulated for both general user interfaces and Web page design. Professionals in the field of HCI are becoming increasingly more aware of the needs of the elderly and people with disabilities. In [7], possible inclusive design guidelines are discussed.

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]. Also a number of initiatives at governmental level such as the Section 508 [10] has been undertaken to stimulate awareness of such issues by Web interface developers and service providers, in order to ensure that public Web sites are accessible by all.

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 Web pages through screen readers.

Both W3C guidelines and section 508 standards consider mainly accessibility issues in order to define the general principles to be followed for removing technical barriers which might prevent access to information. Thus, the main underlying goal is access to information, whereas navigation easiness or simplicity of use is not specifically addressed. Besides, those guidelines concentrate on various categories of people with disabilities. The approach proposed in this paper concentrates especially on the difficulties encountered by visually impaired people when navigating through screen readers or magnifier programs. For example, W3C guidelines state the importance of colour contrast between background and foreground. The criteria listed in section 3.1 below suggest how to use different contrasts; how different coloured areas can be used for differentiating various sections in the page (e.g. navigation bar, side-menu, etc.); how highlighting the objects pointed by mouse, and so on. Also, accessibility guidelines suggest assigning summary attributes only to data tables, but they do not clarify their appropriate use. On the other hand, the proposed criteria suggest using summary values also for layout tables by assigning names, which can be useful when skipping from one table to the next through special screen reader commands (e.g., navigation bar, search section, submenu, etc.). In fact, developers usually apply this guideline by assigning the summary attribute only to data tables and by providing a value such as “this table contains…” or “in this table several values are reported…”, which is not particularly useful for users who depend on hearing to understand the content. A shorter summary content is more efficient when it is read aloud by the voice synthesizer. The same applies to

link contents and frame names. The elaborated criteria also advise using different sounds for providing additional information, which is an aspect not taken into account by W3C guidelines. Therefore, in brief, the main difference between our criteria and W3C guidelines, as well as section 508, regards the usability aspects addressed (i.e., the proposed approach focuses on the screen reader interaction with Web pages in order to improve the navigation and make it easier as well as simpler).

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. In [11] a study about the usability of Web site navigation through screen readers is discussed. In particular, this work addresses accessibility supported in the 508 standard [10]. A user testing experiment conducted with 16 blind users showed the lack of support of usability criteria according to the 508 standard guidelines. Based on the empirical evaluation, 32 guidelines were suggested, aimed at improving usability for blind users. Some of those guidelines concern Web site development, while others concern screen reader developers. As a result, an unstructured and unorganised list of guidelines was proposed. Such list appears difficult to use as a reference by developers because of the lack of a clear structure and organization of the guidelines. In contrast to such an approach, in the work presented in this paper general principles have been formulated according to the three main properties of usability in its standard definition. In addition, the criteria have been classified 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 [11] refer only to blind users and do not consider low vision users. Besides, some aspects important for blind users, such as "messages and dynamic data management", "sound usage", "appropriate names for frames or tables", etc., are not considered. Considering how the guidelines in [10] are expressed, it is likely to be difficult to perform automatic evaluation of them. Moreover, no indication is provided about the development of a tool for their automatic support.

2.2 Tools

Various international projects have addressed accessibility and usability of user interfaces for people with disabilities. For example, in the AVANTI project, a "Unified Web Browser" has been developed [9] following the concept of User interfaces for All and the method of Unified User Interface development. The AVANTI browser 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 this paper, 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 these activities involve high costs in terms of time and effort, there is a great interest in developing tools that automate gathering, analysis and interpretation of such 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]).

Many automatic analysis 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.

3. Criteria for Web accessibility and usability by blind and low vision users

3.1 The Proposed criteria

The usability of a Web site depends on many aspects. In order to improve interaction and navigation through a screen reader, a number of criteria have been proposed in [5]. Such criteria have been identified in a process during which first vision-impaired users have been interviewed regarding their problems with interaction devices, such as screen readers. Then, possible solutions have been identified, and small tests have been performed in order to understand their impact and refine them. The final criteria have been categorised into three subsets according to different aspects of usability indicated by the standard usability definition (ISO 9241): "effectiveness" criteria (5) are targeted to ensuring the accomplishment of the task, for example using a logical partition of interface elements or ensuring a proper link content, "efficiency" criteria (10) are targeted to shortening 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) are targeted to providing Web pages with 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”, whereas criteria related to the user interface dialogue are indicated by a “b”. Table 1 shows the list of the proposed 19 criteria. Each criterion is identified using 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; 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 and developers during the development phases of a Web site. Such principles are aimed at structuring user interface elements and the content of a page, as well as providing additional features that can help users to better navigate 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 frame names).

To facilitate their application by Web site developers, 54 technical checkpoints have been elaborated. A checkpoint is a specific construct in a language for Web page development that supports the application of a given criterion. The

checkpoints have been identified through a thorough analysis of the implementation constructs provided by the X/HTML and JavaScript languages. For each criterion, a number of different technical solutions are provided, 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 are expressed 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; on the other hand, 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.

3.2 Empirical testing of the criteria

In order to estimate the impact of the 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 the specific 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 a screen reading application for at least one year at the time 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 testing scenarios, 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, task difficulty, and other personal considerations), along with other qualitative data not obtainable through the logging tool.

A Web site containing specific information about the “The Tuscan Association for the Blind” (Unione Italiana Ciechi – Regione Toscana) was used for the experiment. 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. Regarding the tasks accomplished during user testing, common activities which are usually performed on a Web site were chosen: searching for a specific piece of information, downloading a certain file/document, exploring a Web page, etc.

Two versions of the same Web site prototype were considered: a "control version" implemented without applying the proposed criteria (used as control in the testing protocol) and an "implemented version" created according to the 19 criteria. In practice, in the “implemented version” all the proposed criteria were applied analysing which checkpoints were more suitable for supporting the criteria. In this work, we aimed to use most checkpoints by applying different 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 to the users when they could start the next task performance (according to the predefined sequence) without constraining participants’ behaviour. In practice, when the user selected the "next task" button, the wizard indicated what task to perform through a specific dialog window. Once users read the assigned task, they could select the "continue" button in order to freely navigate in the Web site. When they considered to have accomplished the task, they could select again the "next task" button in order to proceed.

Each participant was asked to carry out a set of seven tasks per test (varying from easier to more difficult) in the two Web sites. The time required to perform each

task was automatically 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 [8]. The log files contained a wide variety of user actions (such as mouse clicks, text typing, and link selections) as well as browsing behaviour, such as page loading start and end. 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* (corresponding to "control site" and "implemented site" respectively) was used to verify if and to what extent the application of the proposed criteria had improved navigability. 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). It was decided to use a non parametric test, instead of a parametric one, mainly because the data distribution does not fit with a normal distribution (basing on the Kolmogorov-Smirnov test). Besides, in order to determine if the application of the criteria could really improve navigation with a screen reader, it was very important to use a more conservative test before deciding to accept or reject the null hypothesis. A significant difference was found 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. However, on average, the application of the criteria to the Web site has led to a significant time saving for all users and tasks. Figure 1 shows the average task performance. For each of the seven tasks, the left bar indicates the average

performance with the traditional Web site and the right one provides the same information but obtained with the Web site applying the criteria.

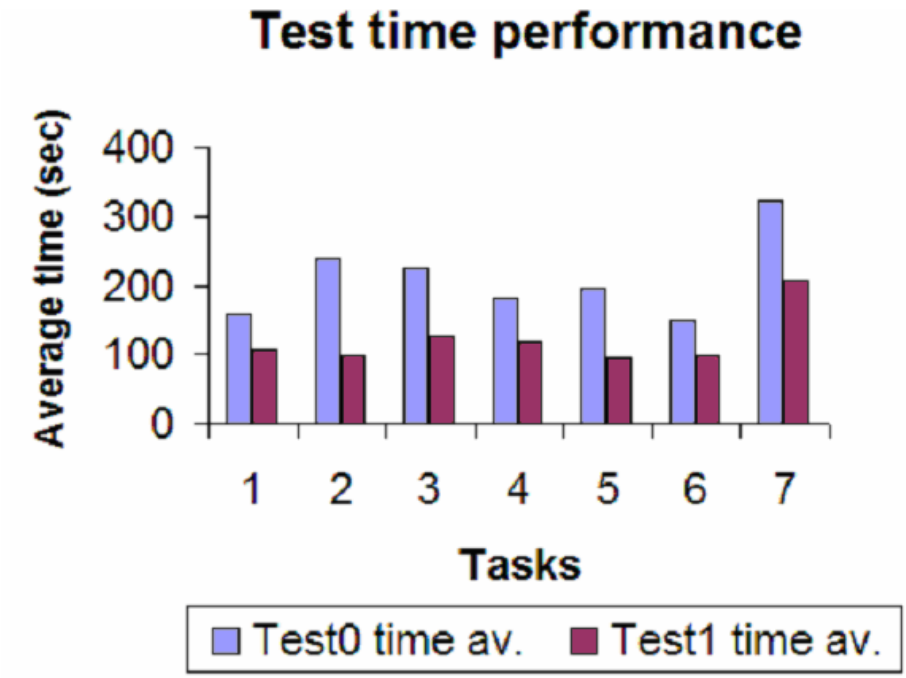

Figure 1: Bar-chart indicating average task performance for each task in both Web sites.

Among the seven tasks, the "looking for information in a long page" tasks turned out to be the most influenced by the criteria (e.g., Task 2 and 3). The reason for this is 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 through the screen reader using the list of heading levels.

In conclusion, the obtained results showed that both blind and low vision users benefited from the application of the criteria, which saved them around 40% in terms of task performance.

4. NAUTICUS: A Tool Supporting Usability Criteria

After the encouraging feedback from the testing, it was decided to create automatic support for the 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 in combination with the use of a screen reader.

4.1 The Tool Goals

The NAUTICUS tool (New Accessibility and Usability Tool for Interactive Control in Universal Sites) has been developed with the intent of automatically checking whether a Web site is usable for users interacting through screen readers. To this end, the tool checks how satisfactorily the criteria for Web accessibility and usability by blind and low vision users reported in Section 3 are applied in the code of Web pages. This is obtained through the automatic identification of the checkpoints associated with each criterion, and the analysis of the associated constructs and attributes to check whether they provide the necessary information. Through a list of checkboxes spread into three 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 for 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 tool 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 alternative text 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.

4.2 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 related to the loaded page;
(3) *Source Code*, which shows the source code of the loaded page;
Supported criteria are grouped according to the main usability aspect that they refer to, and 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 and identifying parts involved to the application of 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 2, 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 performed by selecting the checkboxes associated with the three criteria, and loading the page to check (Figure 3).

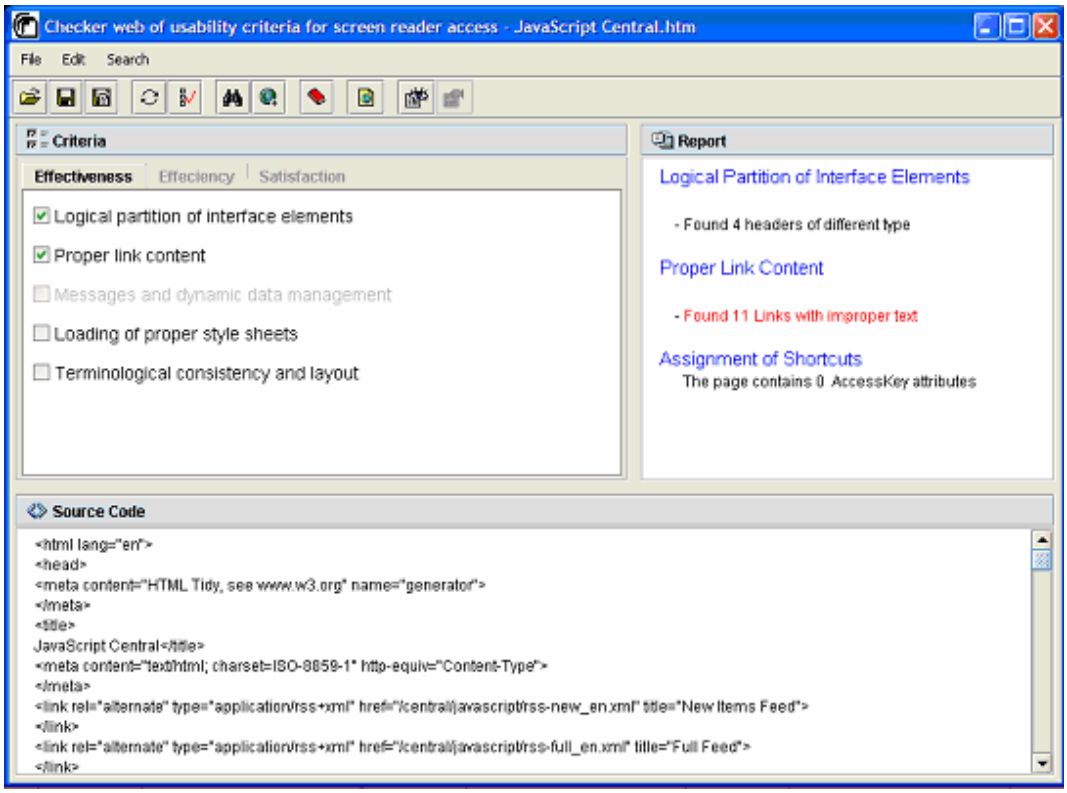

Figure 2: Tool output related to a criteria-based evaluation

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, while regarding link text and title, NAUTICUS points out (in red) that 11 links could not have appropriate texts or titles.

It is worth to note that a complete automatic evaluation of frame names, appropriate links, adequate summaries for tables, etc, is not possible. Therefore, for this purpose a set of dictionary files was defined 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, and 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 language implies changing the dictionary used for evaluating and repairing Web pages.

4.3 Support for Dynamic Pages

Nowadays, the general trend in Web site development is to design more and more pages containing server-side scripting elements which are used to make the pages ‘dynamic’. When using a server-side scripting languages such as PHP (recursive acronym for "PHP: Hypertext Preprocessor"), JSP (JavaServer Pages) or ASP (Active Server Page), programming statements are embedded inside mark-up tags. The corresponding code elements are made recognisable by enclosing them in some kind of brackets (e.g., <?…?> for PHP).
All scripting languages include a statement, often called echo or print, which is used to generate extra tags. The code of a server-side Web page generated dynamically is composed of X/HTML tags and parts of script statements related to the chosen scripting language. When the page is required by the client-browser, the server processes the scripted page and sends a X/HTML page. Thus, when the Web browser reads the page, no script statement is contained within the code. Therefore, the issue arises of how the criteria can be checked in these cases. If the Web page is checked on the client side, its code is simply X/HTML pages generated by the server after processing script statements. This method allows checking the page code without high-level script constructs, but requires an explicit access to the server. Additionally, it is problematic to check all the possible pages that can be generated. In fact, as conditional statements could be embedded within the scripting code, a certain script in a given branch may never be executed (e.g,. because it is carried out only in particular situations). Consequently, a Web page which is correctly verified at a certain point in time, may not be accessible or usable at other times. Thus, another kind of evaluation should be considered.

To better understand this issue, one can consider a user accessing the home page of a Web site providing a welcome greeting such as “Good morning to you”. Furthermore, for having a nicer welcome greeting, the message is shown through a label stored in a gif image. It can also be assumed that the greeting message differs on the basis of the current time (i.e., “Good morning”, “Good afternoon” and “Good evening”). Therefore, three image files are necessary for differentiating the three greeting messages. So, according to accessibility and usability principles, an appropriate alternative text has to be assigned to each

greeting image. Such a result can be obtained by using a script, such as that in Table 2. In order to show the right message, a conditional statement based on the current time can be used: each branch performs the echo() command, which inserts the appropriate message in the HTML page at runtime (i.e., the <img> tag necessary to show the corresponding gif image).

```
<html>
<head>
<title> Dynamic page example </title>
</head>
<body>
<p>
The php script writes appropriate greetings according to current time. <br>
The greetings "Good morning", "Good afternoon" and "Good evening" are written through an image to which an alt attribute has to be assigned.
</p>

<p>
<?php
$theDate = date("H");
if($theDate < = 12)
echo ('<img src="goodmorning.gif" alt="Good morning to you">');
else if($theDate > 12  and < 18)
echo ('<img src="goodafternoon.gif" alt="Good afternoon to you">');
else
echo ('<img src="goodevening.gif">');
?>
</p>
</body>
</html>
```

Table 2: An example of php scripting code for generating a dynamic page

The three instances of the php echo() command in bold are the script statements used to generate the dynamic message from the server when the page is required by the client. Depending on the current time, only one echo() command is executed. Therefore, just one <img> tag is added to the HTML page. So, what happens when the page is checked for accessibility and usability? Table 3 shows the code of two pages generated dynamically at two different times. The code of page (A) is correct, as the <img> tag has the appropriate alt attribute, whereas page (B) has an accessibility error: the <img> has no alt attribute.

<html> <head> <title> Dynamic page example </title> </head> <body> The php script writes appropriate greetings according to current time. The greetings "Good morning", "Good afternoon" and "Good evening" are written through an image to which an alt attribute has to be assigned. **<img src="goodmorning.gif" alt="Good morning to you">** </body> </html>	<html> <head> <title> Dynamic page example </title> </head> <body> The php script writes appropriate greetings according to current time. The greetings "Good morning", "Good afternoon" and "Good evening" are written through an image to which an alt attribute has to be assigned. **<img src="goodevening.gif">** </body> </html>

Table 3: Examples of generated code when the page is required. On the left (A) the generated page code is correct; on the right (B) the generated page code contains an error.

If a client-side evaluation is performed, the code that is checked is that obtained after generation by the server (i.e., the page code shown in Table 3). Moreover, if the checking is done in the morning or in the afternoon then the code generated is correct (i.e., accessible) as the <img> tag has the right alt attribute. However, the code generated in the evening is not correct, though the error might be never found because it occurs only under specific circumstances.
One solution to avoid this undesirable situation is to carry out the evaluation directly at the server-side on the static code containing the original script (i.e., the page code shown in Table 2). Specifically, it is necessary to check the three php echo() commands (see the parts in bold) aimed at writing the <img> tag in the HTML page. In this way, all the possibilities that can occur at runtime are verified and the missing alt attribute for the third tag <img> can be detected.

To this end, NAUTICUS includes a method for checking script-based, server-side Web pages. Briefly, the tool analyses the page's X/HTML code including the parts with script statements. The method checks those script statements aimed at producing the HTML tags to be sent to the client when the pages are dynamically generated. Specifically, statements like "echo()" or "print()" (when PHP scripting language is used) are taken into account. Thus, in general, through this kind of method it is possible to perform both a local evaluation, without the need of remote access, and a more thorough static inspection at the server-side by checking the original page code so that all dynamically generated pages are considered.

4.4 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. For each evaluation criterion there is a class implementing the associated algorithm to check its application. It mainly analyses the DOM (Document Object Model) [13] to see whether the associated constructs are provided along with the necessary attributes. 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 architecture is structured in a number of modules implemented through the Java packages:

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

5. Some Examples of Tool Application

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

5.1 Proper link content

The proper link content criterion is exemplified on the page index of a java script reference manual Web site (see Figure 3). Each topic can be read in html, pdf or zipped format. For this purpose, 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 links (those in the bottom left area) because each link has a short text and no longer titles have been used.

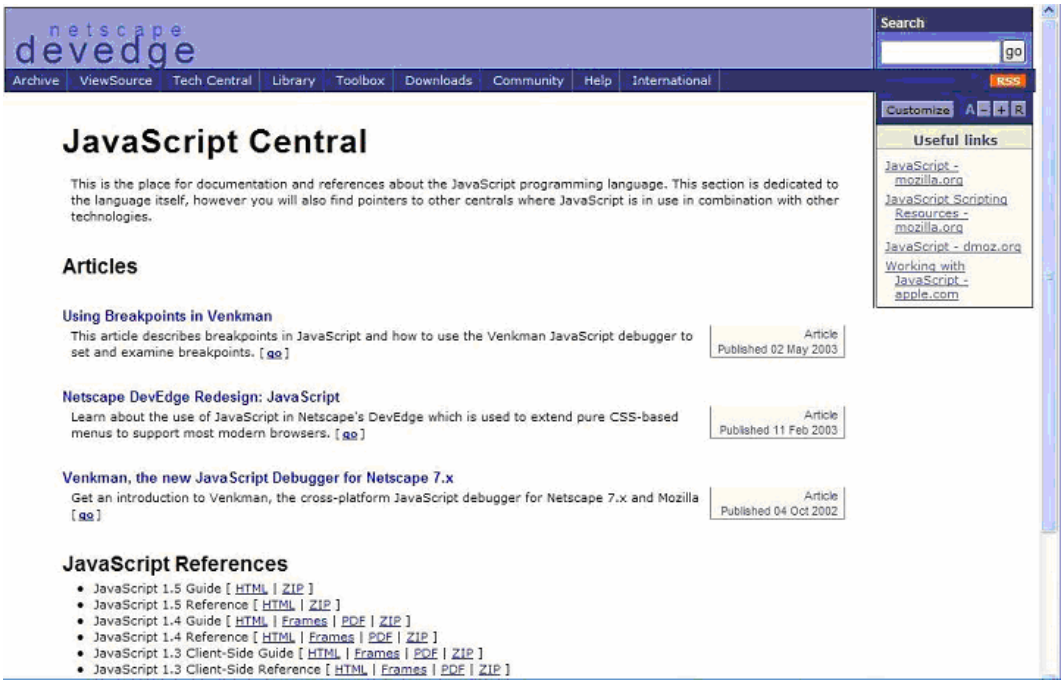

Figure 3: Another example of problematic Web page

5.2 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, etc. – which is a table with one row and one column. It should be useful applying a summary attribute such as “navigation bar” or “navigation links” for this table, 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, which not recognised as a layout table because it has one row and three columns; in any case, also this table has not summary attribute. A summary such as “search form” could be used in order to facilitate its identification.

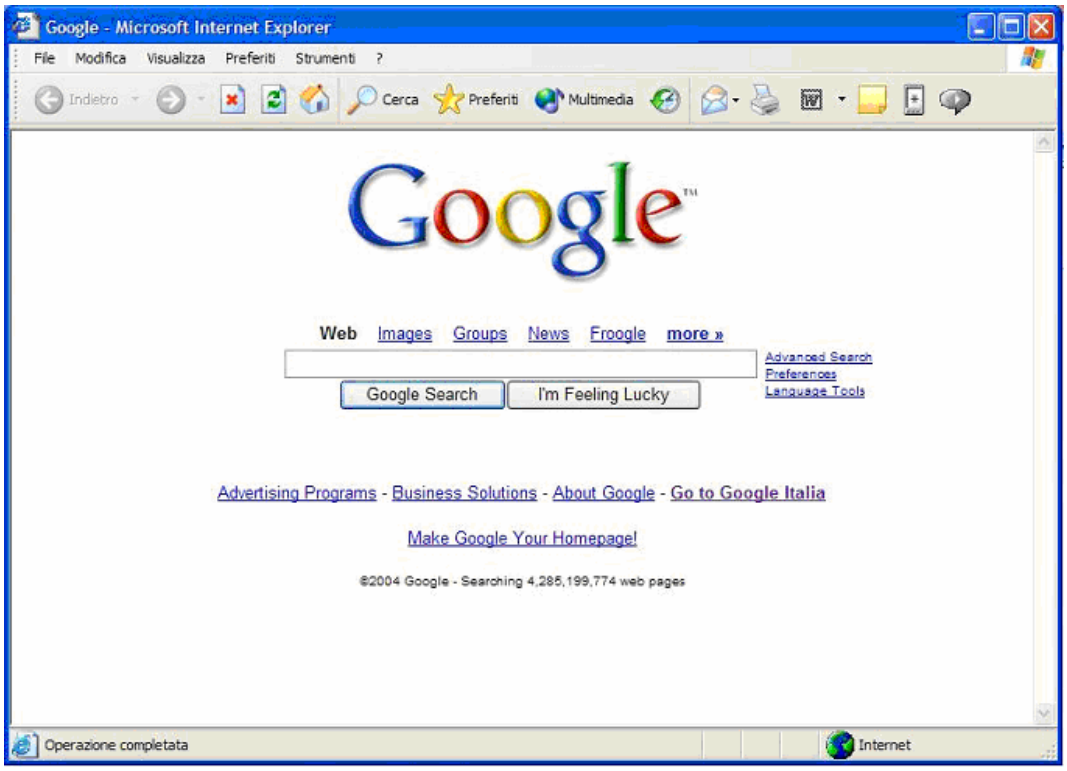

Figure 4: The Google Interface

Figure 5 shows the result found out when evaluating the criteria “proper names for tables” for the Google page: in the “report” panel the two tables without summary are indicated.

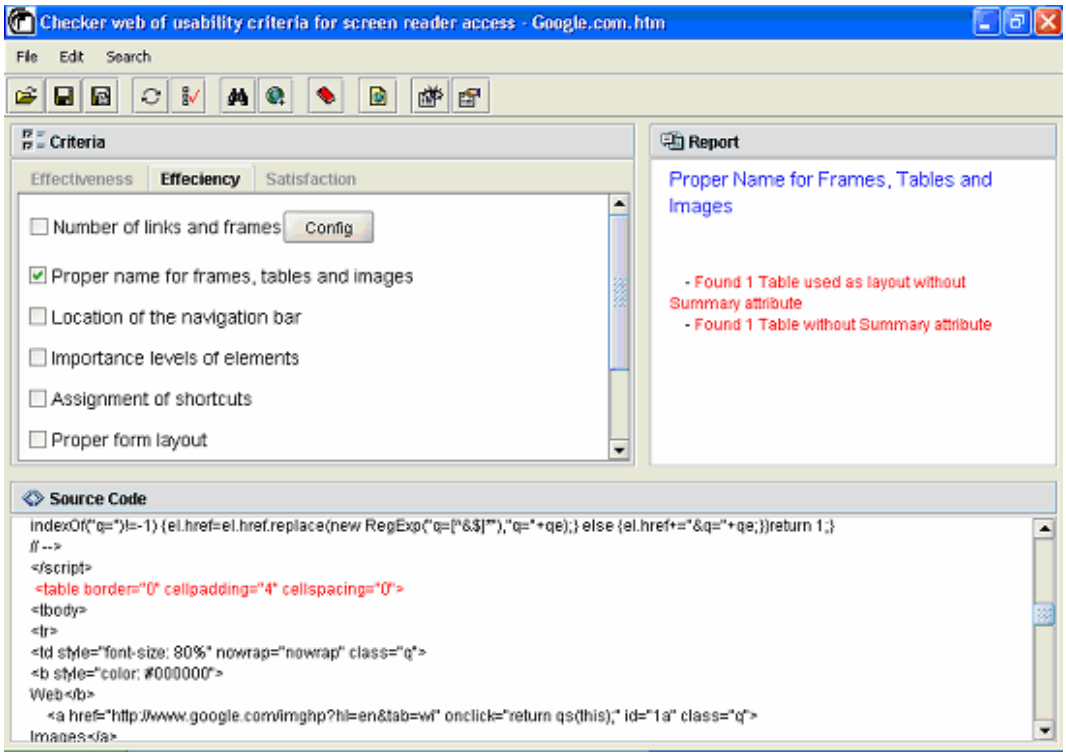

Figure 5: Evaluation of the Google Interface with NAUTICUS

5.3 Proper name for frames, tables and images: frames

This example concerns the analysis of a page with a frameset where each frame has generic names such as “frame1”, “frame2” and “frame3”. The sample page used for evaluating these criteria belongs to the Florence City council Web site (see Figure 6).

Figure 6: Another example considered

This page has four non-appropriate names for frames: top, left, centre and bottom. Therefore, the tool detects that there are four non correct names or titles for frames (see Figure 7).

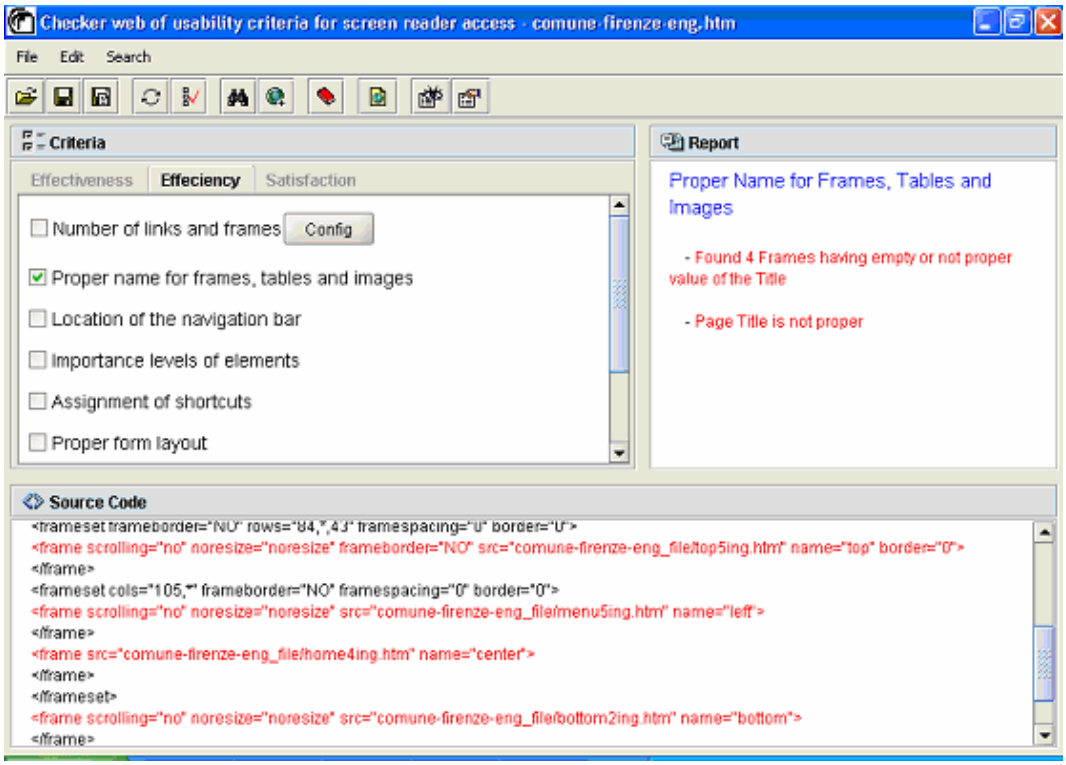

Figure 7: Evaluation of the example

5.4 NAUTICUS support for Web page correction

The code can be corrected either through the DOM (Document Object Model) [13] or by editing the page. The document can be further processed and the results of that processing can be incorporated back into the presented page. NAUTICUS 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 interface supporting it (see an example in Figure 8). 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.

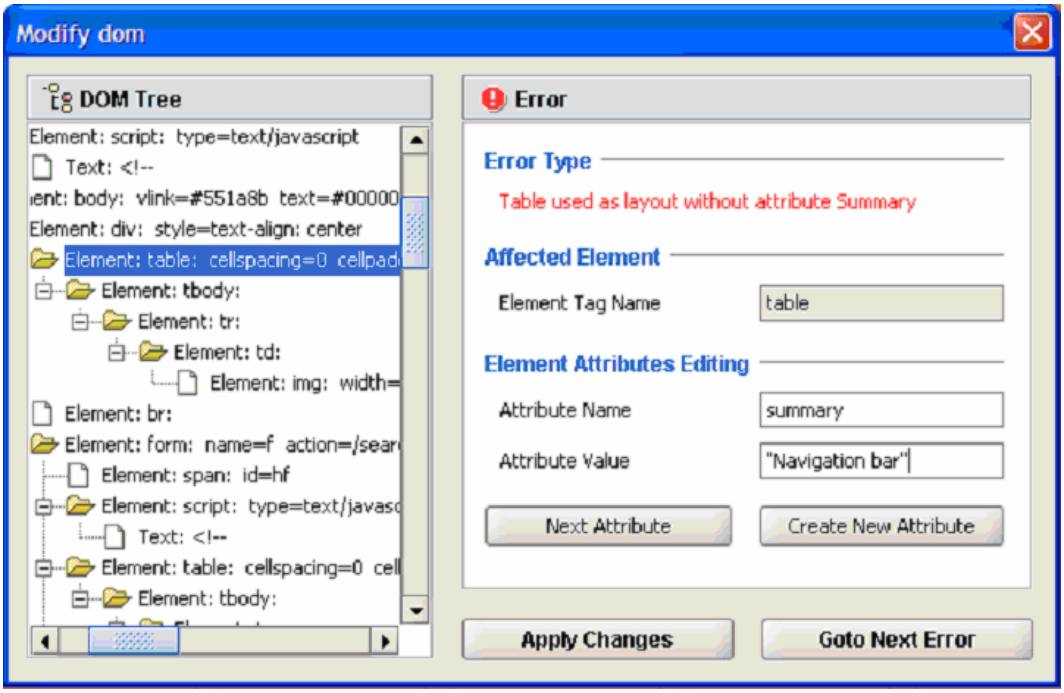

Figure 8: Tool support for the identification and repair of problematic parts through the DOM

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 "*Goto 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 and 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. Figure 8 illustrates 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").

Figure 9 shows how the name and title attributes for each frame can be fixed through the DOM structure considering the example in Figure 6. NAUTICUS identifies 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 frame names could be fixed by changing "top" into "navigation links", "left" into "submenu", "center" into "main content", and "bottom" into "search in the Web site". Using the buttons “next attribute” or “correct next error” the evaluator can navigates through the error tree and fix the identified problems.

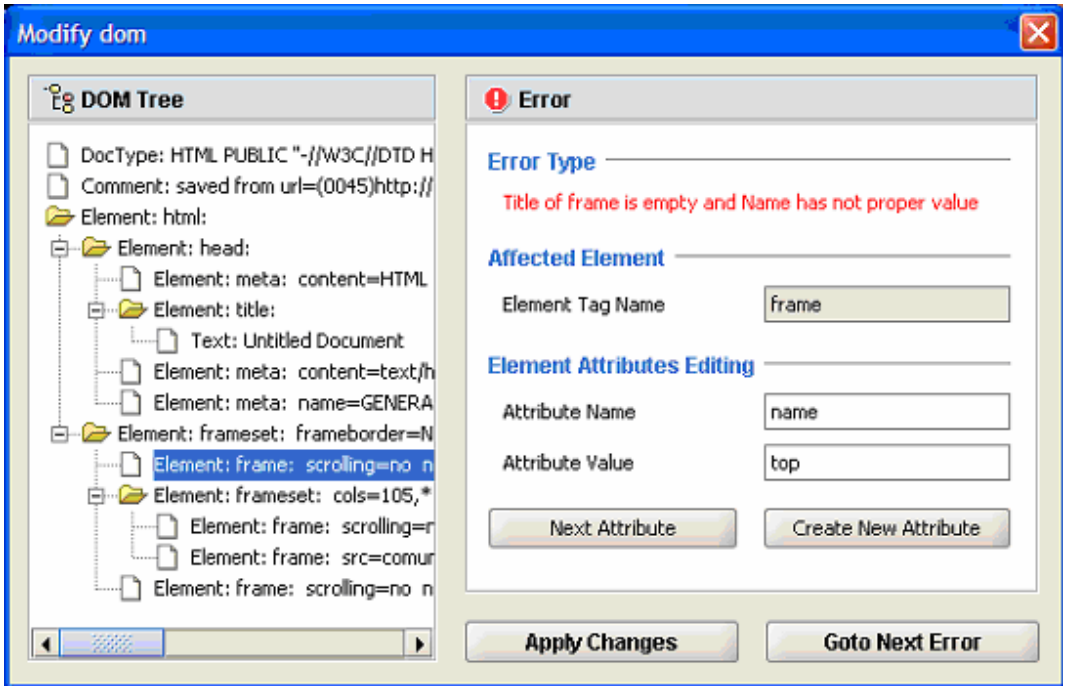

Figure 9: Another Example of Error Correction with NAUTICUS.

5.5 Example Web Site Analysed through NAUTICUS

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.

Figure 10: The Web site used for the tool application

One of the most serious problems was that 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).

6. Conclusions

This paper has introduced a set of usability criteria to improve Web navigation for vision-impaired users, and has presented an automatic tool supporting such

criteria, showing some application examples and reporting on its application to a case study.

The tool provides interactive support for checking the application of the elaborated criteria and helping designers to improve their Web site implementations when it detects potential problems. This is an important contribution in reducing the cost of the evaluation according to the proposed criteria, thus making such evaluation affordable for a larger number of Web designers and developers. It is also useful in order to obtain a consistent application of the proposed criteria with respect to the results that can be obtained through manual approaches.

Future work will be dedicated to further extending the evaluation environment in two directions: integration of the current method based on automatic code inspection with assessments performed through other methods (such as automatic log analysis) and integration of such support also in the development phase in such a way that designers are guided to implement their Web sites in accordance with the proposed usability and accessibility design criteria. In particular, the accessibility module will be based on W3C guidelines and other accessibility criteria (i.e., those indicated by the Italian law on accessibility for Web sites in public administrations). Besides, a restructuring of the tool interface for supporting several languages will be added.

Acknowledgments

The authors 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, 2004-TR-45, Submitted paper.
7. Nicolle, C., Abascal J. *Inclusive design guidelines for HCI*, p. 285, Taylor & Francis, 2001.
8. Paganelli, L., Paternò, F., Tools for Remote Usability Evaluation of Web Applications through Browser Logs and Task Models, Behavior Research Methods, Instruments, and Computers, The Psychonomic Society Publications, 2003, 35 (3), pp.369-378, August 2003.
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 2004, available at http://www.w3.org/WAI/GL/WCAG20/