# Transcoding for Web Accessibility for the Blind: Semantics from Structure

ANTÓNIO R. FERNANDES; ALEXANDRE CARVALHO; J. JOÃO ALMEIDA; ALBERTO SIMÕES

Departamento de Informática
Universidade do Minho
Campus de Gualtar
4710-057 Braga, PORTUGAL
e-mails: arf@di.uminho.pt, alexandre@quintal.dyndns.org,
jj@di.uminho.pt, ambs@di.uminho.pt

True accessibility requires minimizing the scanning time to find a particular piece of information. Sequentially reading web pages do not provide this type of accessibility, for instance before the user gets to the actual text content of the page it has to go through a lot of menus and headers. However if the user could navigate a web page based through semantically classified blocks then the user could jump faster to the actual content of the page, skipping all the menus and other parts of the page. We propose a transcoding engine that tackles accessibility at two distinct, yet complementary, levels: for specific known sites and general unknown sites. We present a tool for building customized scripts for known sites that turns this process in an extremely simple task, which can be performed by anyone, without any expertise. For general unknown sites, our approach relies on statistical analysis of the structural blocks that define a web page to infer a semantic for the block.

**Keywords:** Web accessibility, visually impairments, transcoding, assistive technologies

## 1. INTRODUCTION

Web accessibility for visually impaired users is an issue that is far from being over. Although this problem is usually tackled from the web page design/construction side, it is not straightforward to determine whether the problem lies on the pages being read, or if the problem lies on the systems that are used to read the pages.

Testing with blind users is typically a hard task making this distinction harder. A study by Petrie et al. (2004), including an accessibility test of 100 web sites with 51 disabled users, concludes that blind users had significantly more difficulty in using websites, even when compared to other disabled users.

Tests deal with small number of blind subjects (Petrie et al. 2004) (Fernandes et al. 2004), and/or use regular sighted users but deprive them of the screen (Yesilada et al. 2004) (Campos and Fernandes, 2005).

Another problem regarding these tests is related to the variability in the test subject and target audience population for tools and also websites. As pointed out by Seeman (2004): besides the fact that the users are all visually impaired, there is not much more that can be said to characterize the population.

Due to this lack of information, it is not straightforward to pin point where the problem lies: are the accessibility problems found related to the way the web pages are designed; or are the problems related to the tools used (e.g. web browsers and screen readers)?

A mixed approach to this issue may be more reasonable, i.e. while it is true that some web pages are built focusing only on the visual impact without any thought regarding accessibility for blind users, it is also true that most web surfing tools were not designed to facilitate surfing to blind users.

Following the W3C guidance (Brewer 2004) regarding accessibility is a good start to improve accessibility, but it does not solve the problem of fast access to a page's content. Some researchers have worked on the area of verifying the correctness of a page (Stone and Dhiensa, 2004). We argue that the correctness of the page, although useful, is not enough to help the blind user to minimize the scanning time when searching for a particular piece of information.

In FIGURE 7 several web pages from a well known web site are presented with blocks superimposed on the content. As can be seen, for instance "Titles" appears in several places on the pages presented. Furthermore, when using a sequential page reader, a blind user who wants to get to the actual content of the page, must go through a set of sections that are not immediately relevant to the user's purposes. To complicate things further, the blind user will have to listen to the headers, search form and menus three times if visiting all the pages presented in the figure.

Based on the example on FIGURE 7, we can raise two issues: first, how to get the user to the section that the user is looking for; and second how the user is supposed to know that a particular section is being read?

This may be harder than expected for a blind user. In here we propose a solution that deals with both issues. We propose a transcoding system that automatically provides semantic classification of the web page's structural blocks. The system applies a set of heuristics, based on statistical analysis, to determine where are the actual text content, menus, and other section of the web page.

The paper starts by reviewing the related work. It proceeds to the description of our new approach. Afterwards it details some parts of the implementation to illustrate the strategy adopted, and shows how it can be applied both to general and specific sites. Conclusions are then presented together with some suggestions for future work.

## 2. RELATED WORK

Some solutions have been proposed for the visually impaired that attempt to increase the degree of accessibility of the web. General purpose talking browsers, such as Home Page Reader (IBM), Brookes Talk (Zajicek et al. 2000) or the AudioBrowser (Fernandes et al. 2001), are examples of such tools. Other researchers have approached the problem of specific web page elements and proposed tools such as EVITA, a table reader (Yesilada et al. 2004). Parente (2003) proposes the usage of enriched links, as a preview of the links's destination page.

While it is true that web pages have become progressively more complex through the years (Hackett et al. 2004), this does not necessarily imply less accessible pages and some researchers (Bohman and Anderson, 2004) have pointed some avenues to build more accessible web pages.

Another research direction that shows promising results is based on the concept of middleware (Harper et al. 2004), or HTTP proxies. These are systems that act as an interface between the web server where the page is requested from, and the client browsing tool that requested it. These systems may perform transcoding on the web page to simplify, add annotations, and/or extract parts of the page. Transcoding web pages has been explored in several previous works

When using HTTP server proxies introduces some problems as related by Hanson and Richards (2004). Nevertheless what we're discussing in here is the concept of transcoding, and not the proxy per se. As Hanson and Richards pointed out, a proxy based service can be transformed into a local service, eliminating most of the proxy related problems.

. Transcoding can be performed considering only the source of the page, or relating the page to its neighbors on the site and perform a similarity/difference analysis (Takagi and Asakawa 2001).

Some solutions create annotations for links in a web page to improve mobility (Harper et al. 2004). Transcoding to provide semantic information is presented in several articles, for instance (Takagi and Asakawa, 2002), (Takagi and Asakawa, 2000), (Pontelli et al. 2002), and (Mukherjee et al. 2004). However, in those approaches there is a significant human intervention required, as the process is only partially automatic.

Lee (2004) has presented a work in which the web page is divided based on spatial information and visual properties of blocks. This is a closely related work in the sense that the author is also trying to classify blocks, although the classification is not semantic, and is performed based on the visual and spatial information, as opposed to the approach in here where the goal is to infer a semantic classification from structure.

Aurora (Huang and Sundaresan 2000) focuses on the general services provided by Web sites rather than focusing on the details of Web pages themselves. Aurora uses the semantic model proposed paradigms developed to address these challenges reinforce in this paper to implement a rule-based transcoding system for specific sites.

## 3. SEMANTICS FROM STRUCTURE

In here we propose a solution that explores web page transcoding to present the user with more accessible contents and also faster access to information.

The essence of our approach is to automatically determine semantic information based on the page's HTML structure. Although HTML is often used as a graphical layout specification, it is possible to extract important semantic information.

We extract the semantics based on statistical analysis and heuristics. Our notion of semantic is a broader notion than the previously cited works, as we're not looking for items that refer to a particular subject, for instance "Politics" or "Science", but instead we're classifying the blocks of a web page, for instance as menus, content areas, titles.

The page is then reorganized by groups of semantically similar blocks, and headers and anchors added automatically to identify and navigate through these semantic groups. The transcoding engine is built upon a HTML processing library that simplifies the writing of new transcoding scripts.

The presentation of the transcoded pages can be performed efficiently on any standard browser with the assistance of a screen reader. The user will be presented at the top of a page with a set of added anchors that will take the user right to the section it wants. For instance a user looking for the main text content of the page can select the respective anchor to jump directly to that section.

Our system tackles accessibility at two different levels: for a set of known web sites, it provides customized transcoding scripts hence providing accurate non-graphical versions of the web pages; for other sites it uses scripts based on heuristics that provide a semantic classification to blocks of a web page.

We present a tool that allows this customization in an extremely easy process. Custom scripts for known sites can be contributed by the community, or by the site designers themselves, thereby creating a repository of scripts. Note that custom scripts go beyond the notion of accessibility applied to web pages. Considering Netscape's portal, a script could provide the user with a single section from the page, for instance "Today on Netscape", or "Need to Know".

Each script creates a view of the page, and several scripts can be used on the same page, thereby providing a set of complementary views of the page. A view can also be simply

the result of a cleaning operation of the HTML of the page, thereby providing all the content of the page, but in a hopefully more accessible manner.

## 4. HTML PROCESSING LIBRARY

In this section we present a brief description of the structure of the library and some strategies to transcode HTML. Functions on the HTML processing library can be split into different classes:

**Segmentation and Arborization:** Segmentation provides the framework to analyze an HTML document as a hierarchy of segments. By segmenting a document and applying analysis methods to each segment we may increase accuracy rates when, for instance, labeling these blocks with semantic information. This process may produce different segments according to different techniques. These reveal different benefits whether we are classifying or extracting semantic blocks. Arborization methods consist of building trees representing an HTML extract or document structure. These trees may focus on every HTML element or specific ones. A common and helpful representation tree may focus only on tables and/or other structural elements commonly used to layout the web pages, such as ‘P’ or ‘DIV’.

**Semantic classification:** Semantically classifying segments of a HTML block is based on heuristics that rely on computed statistical information about the block. Such methods are applied on HTML blocks returned by the segmentation process. Typical semantic labels are menu, section menu, table of contents, banner, ground text (text of a news article, blog), titles menu (headlines on a news web-site), etc.

**Completion and Simplification:** Functions belonging to this class will be responsible for actually transforming an HTML document or extract, whether they just fill missing attributes of a certain element, change the order of appearance of semantically classified HTML blocks or simplifying style related aspects of a document. Many of these functions are applied after segmenting, and classifying the document blocks, and their results do, hopefully, increase the accessibility of a document.

### EXAMPLES OF METHODS

A short introduction to some methods of each of these classes is now presented in order to show some of the adopted mechanisms as well as the heuristic’s nature.

### SEGMENTATION AND ARBORIZATION – Class::segment

Given an HTML document, this function will produce segments of the original document. The markup elements by which the segmentation will occur are specified in a list, for instance common values are 'table' or 'div'.

There are several options to segment an HTML document. In FIGURE 1 we introduce two possibilities. An option is to extract blocks of a given element removing any block of the same element contained in it, see FIGURE 1 (left). Another possibility is to extract blocks of a given element without removing the inner blocks, see FIGURE 1(right).

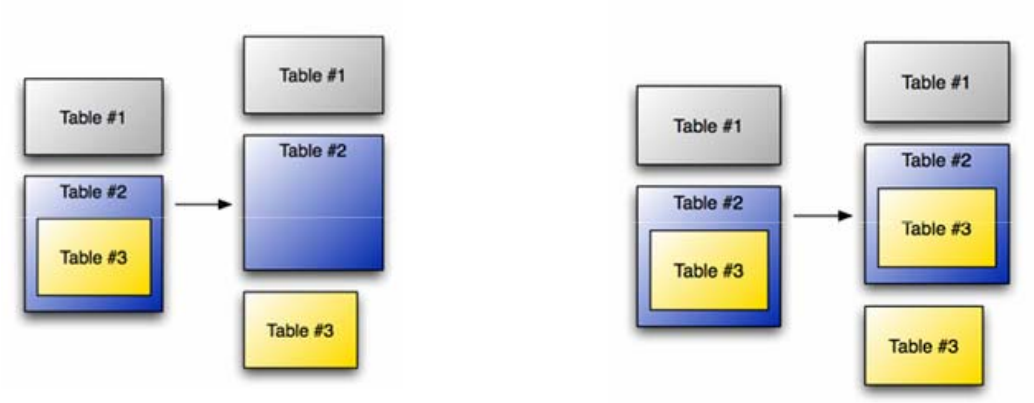


**FIGURE 1 - SEGMENTATION EXAMPLES**

SEMANTIC CLASSIFICATION - Class::is_menu

Given a block, this function tries to detect if it represents a menu. This function measures the length of text present in the block and compares it with the length of the text contained in block's links. If this ratio and the number of links, are above threshold values the function concludes that the block is a menu.

```
<table>
<tr><th>Sports</th></tr>
<tr><td><a href="soccer/">Soccer</a></td></tr>
<tr><td><a href="cycle/">Cycling</a></td></tr>
<tr><td><a href="tennis">Tennis</a></td></tr>
<tr><td><a href="bball">Baseball</a></td></tr>
<tr><td><a href="rugby">Rugby</a></td></tr>
<tr><td><a href="bask">Basketball</a></td></tr>
<tr><td><a href="curling">Curling</a></td></tr>
<tr><td><a href="motors">Motor</a></td></tr>
</table>
```

Applying function is_menu to the HTML block above:

```
is_menu($html, {ratio=>0.75,links=>4} )
```

classifies this block as a menu based on the following values:

Number of links = 8
Links text length = 61
Full text length = 67
Ratio = 0.910

Further distinction can be made in here by looking at the links themselves. For instance a menu with only external links could be classified as a collection of links and a menu with only internal links could be classified as an index of the web site.

SEMANTIC CLASSIFICATION - Class::is_titles_menu

This function is similar to is_menu, but it takes into account the length of text of the links. FIGURE 2 shows an example of the difference between the two types of menus: a) is classified as a titles_menu, whereas b) is classified as menu.

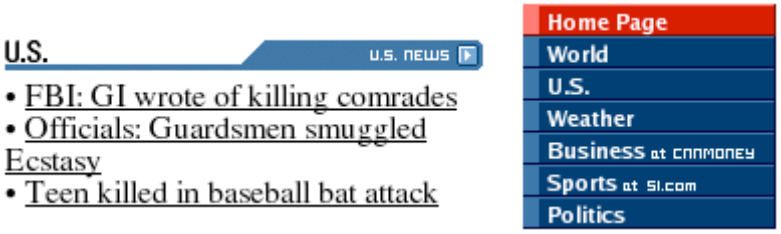


**FIGURE 2 TWO SEMANTICALLY DIFFERENT MENUS. LEFT(A) ) A SET OF TITLES; RIGHT(B) ) NAVIGATION OPTIONS.**

SEMANTIC CLASSIFICATION - Class::is_text_content

Given a block, the amount of text, not including the text in the tags, is used to determine if the block is actual content. A block may have blocks inside it, so the full web page may be classified as text content. Therefore, in order to detect the actual content, the largest block that satisfies this function in which all sub-blocks are classified as text content is selected.

SIMPLIFICATION Class::clean_block

After a block is classified a cleaning operation is performed to eliminate/replace structural tags such as TABLE or DIV. For instance, in a menu, each option may be inside a table cell in the original document. Cleaning such a menu is the equivalent of merging all this cells and adding list tags (UL,LI) for the menu options. If the table has only this cell then the table itself is removed, eliminating unnecessary clutter in the transcoded page.

COMPLETION Class::fill_alts

In order to read HTML for blind persons a common task is to substitute images by their alternate text (if it exists). But a significant number of web designers forget to provide the alternate attribute and, worst than that, use those images as links for other pages.

So, these images become critical and can not be ignored. We must find a suitable way to maintain the link and make it usable. To do that, we can get the linked page title and use it as alternate text.

Given an HTML document or an extract of it, this function will cycle through all the 'img' elements, check if the 'alt' attribute is set and in case it isn't it will infer a value to fill in this attribute. Different heuristics may be adopted to infer this value. Perhaps the simplest consists of extracting the image file name from the 'src' attribute. In case the image is surrounded by an element 'a' and if this element only contains the 'img' element, another reasonable heuristic may be determining the 'alt' attribute value by extracting the page title to which the 'a' element refers to on the 'href' attribute. See (Harper et al. 2004) for alternative approaches.

The process is structural: for each image tag found, we replace it by the alternate attribute. If it does not exist we check if its father is a link and, if it is, we fetch the linked page and extract the title.

## 5. SITE SPECIFIC TRANSCODING SCRIPTS

As stated previously, we saw the utility of building specific transformation scripts to deal with a set of very important sites that were considered crucial to the community. This set of crucial sites includes:

- the most important newspapers;
- some institutional pages;
- the pages of some popular encyclopedias, dictionaries, and other reference resources;
- some archives and information portals.

The example that is presented next is a simple script to help writing specific table-based site transformation tools. As mentioned before, tables are often used as geometric building-blocks in complex (and often polluted) pages. Hence we built a simple tool to help build this customized scripts in a straightforward manner. FIGURE 3 shows a partial view of the CNN home page with the blocks outlined.

When called without arguments, the script outputs the complete hierarchy of blocks, labeled according to their position in the hierarchy. FIGURE 4 shows a small subset of these blocks.


**FIGURE 3** CNN HOME PAGE WITH BLOCKS OUTLINED


**FIGURE 4** EXCERPT OF BLOCK EXTRACTION SCRIPT


**FIGURE 5** EXCERPT OF SCRIPT OUTPUT FOR CNN.COM

The next step is the selection of the elements that correspond to relevant blocks of information in the page. For instance, if we select Table 6, visible in FIGURE 4, and Table 26 we get the result shown in FIGURE 5.

In order to show how simple such a script is with the help of our library, we now present it:

```
1       use HTML::REng qw(:all);
2       use LWP::Simple;
3       my $url = shift;
4       my $html = get($url);
5       if(@ARGV==0) {
6               print buildIndex($html,{by=>'table'}) }
7       else {
8               print extract_blocks($html,{by=>'table'},\@ARGV)}
```

A few notes about this script (see (Simões 2004) for more details):

- line 1 - Import the HTML Reverse Engineering module (library).
- line 2 - Import the LWP Simple module for fetching remote documents.
- lines 3, 4 - Get the URL specified at the command line and fetch the document using LWP Simple get function.
- line 5 - Check if there is any other command line argument
- line 6 - In case there are no arguments build an HTML document with the tables present in the document fetched.
- line 8 - In case there is a list of arguments (a list of tables id's), call function extract_blocks with this list as one if it's arguments.

This procedure will build an HTML document only with the tables specified. In both cases the resulting HTML will be printed to standard output. Using the process described above, anybody who wishes to help this project to improve accessibility may help in determining relevant blocks for each URL.

Under this approach more than one script may be defined for a particular page. A possible setup is to use scripts as filters that fetch only a particular piece of information, for instance the weather, or the stock exchange information.

## 6. TRANSCODING FOR GENERAL SITES

Building a large number of site specific scripts may bring very good results in terms of simplification and increased accessibility. Although, maintaining these scripts may become a hard task due to frequent structural changes of the website's HTML. Another drawback of relying exclusively on this kind of solution is the lack of results when visiting uncharted territory, i.e. a website without any script developed yet. In order to overcome these issues, simple scripts using the heuristics discussed previously do generally produce accessible HTML documents, semantically organized, and with an added semantic index.

The process of transcoding a general page consists of a sequence of steps. First segmentation is applied. For each block semantical classification is performed. Then, for each block, scripts for cleansing and completion, for instance inferring alt text for images, are executed. The last step is the rearrangement of blocks. Blocks are grouped by semantic category and headers/anchors are added automatically according to the semantic classification of the group. Blocks that are empty from a content point of view, for instance tables that only serve layout purposes are removed.

The script explained next outputs HTML documents with increased accessibility for a large number of websites. The argument for this script is the URL of the document to process.

```
1   use HTML::REng qw(:all);
2   use LWP::Simple;
3   my $url = shift or die("usage: siteProc url");
4   my $html = get($url);
5   my $head = h($html);
6   my @tables = segment_str($html,{by=>'table'});
7   for(@tables) {
8           my $type = get_type($_);
9           if($type eq 'm') { push(@menus,$_) }
10          elsif($type eq 'tm'){push(@titles_menu,$_)}
11          elsif($type eq 'c'){push(@contents,$_)}}
12  print "<html>$head<body>";
13  print "<h1>Contents Text Areas:</h1><br/>";
14  for (@contents) {print "$_<hr/>"};
15          print "<h1>Menus Areas</h1><br/>";
16  for (@menus) {print "$_<hr/>"};
17          print "<h1>Titles Menus Areas</h1><br/>";
18  for (@titles_menu) {print "$_<hr/>"};
19  print "</body></html>";
```

Notes about this program:

line 1 - Import the HTML Reverse Engineering module (library).
line 2 - Import the LWP Simple module for fetching remote documents.
lines 3, 4 - Get the URL specified at the command line and fetch the document using LWP Simple get function.

line 5 - Extract head element in order to keep important information about the document (title, encoding).
line 6 - Segment the HTML document by tables. This can be easily changed. Save the results to array tables.
line 7 - Cycle through the elements of the array (tables)
line 8 - Get semantic type of the current table, we will be interested only in three types: menus, titles (headlines menus at most news websites) and content blocks (text of a news article).
lines 9, 10, 11 - Add the current menu, titles menu or contents tables to arrays menus, titles, menus and contents respectively.
line 12 - Print the beginning of the document with the original head element.
lines 13, 14 - Print contents title. Cycle through the elements of the array contents and print each one of them separated by an horizontal rule.
lines 15, 16 - Similar procedures as lines 13 and 14, this time for the menus array.
lines 17, 18 - Similar procedures as lines 13 and 14, this time for the titles menus array.
lines 19 - Print the end of the HTML document.

This script provides results as shown in FIGURE 6 for the CNN home page. In the figure it can be seen that the blocks of the page have been rearranged. Furthermore, the contents section has been put on top of the page, reducing drastically the amount of time a screen reader takes to get to the main content of the referred web page.

With this script, good results can be obtained for cbsnews.com, gnu.org, slashdot.org, abola.pt, and other top web sites.

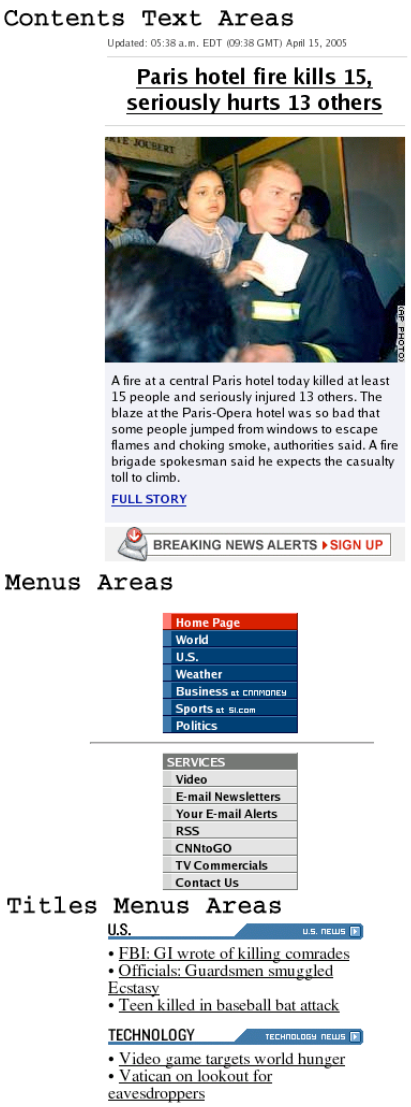


**FIGURE 6** EXTRACT OF GENERIC SCRIPT APPLIED TO CNN.COM

## 7. CONCLUSIONS

Accessibility is not solved only by writing proper, clean HTML. Accessibility also has to do with fast access to information; hence it requires that the user has direct access to the sections of the page. This is particularly relevant for blind users.

Accessibility was tackled at two different levels: for specific sites and for general sites. Using the tools provided it is extremely simple to customize a script to present views of web pages that provide direct access to pieces of information on a known web page. For general sites a solution was also provided based on the semantic classification of blocks. Fast access is available with the insertion of anchors, and reorganization of the page's text.

Automatic semantic classification of web page's blocks based on a set of heuristics was presented. This allows for a fully automatic annotation process without human intervention.

Regarding future work, emphasis will be placed on testing and refining our heuristics for the automatic semantic classifier. User testing will also be performed to determine how to transcode the page once semantic classification of its blocks is performed. Our current approach involves a reorganization of the page, but other solutions may be found.

Another important issue relates to the presentation of the processed pages. We are currently exploring two avenues of research on this issue: adapting a talking browser to

enable fast navigation between the semantically classified blocks; building a plugin for a standard browser that takes advantage of the classification performed.

## ACKNOWLEDGEMENTS

The research reported in here was supported by FCT, Fundação para a Ciência e Tecnologia, the Portuguese national science board, and POSI/2001 (Operational Program for the Information Society) with funds partly awarded by FEDER.

## REFERENCES

**BOHMAN, P.R., AND ANDERSON, S. 2004**. An Accessible Method of Hiding HTML Content, in Proceedings of International Cross-Disciplinary Workshop on Web Accessibility.

**BREWER, J. 2004.** Web accessibility highlights and trends. In Proceedings of the 2004 international Cross-Disciplinary Workshop on Web Accessibility.

**CAMPOS, J.C, AND FERNANDES, A.R. 2005.** Testing AudioBrowser, (to be published) in Universal Access in H.C.I., Proceedings of HCI International. Lawrence Erlbaum Associates.

**FERNANDES, A. R., MARTINS, F. M., PAREDES, H., AND PEREIRA, J. 2001.** A different approach to real web accessibility In Stephanidis, C., editor, Universal Access in H.C.I.,Proceedings of HCI International 2001, volume 3, pages 723–727. Lawrence Erlbaum Associates.

**FERNANDES, A. R., PEREIRA, J. & CAMPOS, J. C. 2004.** Accessibility and Visually Impaired Users. In I. Seruca & J. Filipe & S. Hammoudi & J. Cordeiro, editor(s), ICEIS 2004: Proceedings of the 6th International Conference on Enterprise Information Systems (vol. 5.

**HACKETT, S., PARMANTO, B., AND ZENG, X. 2004.** Accessibility of Internet websites through time. In Proceedings of the 6th Int. ACM SIGACCESS Conf. on Computers and Accessibility.

**HANSON, V. L. AND RICHARDS, J. T. 2004.** A web accessibility service: update and findings. In Proceedings of the 6th Int. ACM SIGACCESS Conference on Computers and Accessibility

**HARPER, S., GOBLE, C., STEVENS, R. AND YESILADA, Y. 2004.** Middleware to Expand Context and Preview in Hypertext, in Proceedings of ASSETS' 04, ACM Press, 63-70, October, USA.

**HUANG, A. W. AND SUNDARESAN, N. 2000.** A semantic transcoding system to adapt Web services for users with disabilities. In Proceedings of the Fourth international ACM Conference on Assistive Technologies.

**IBM CORPORATION**, Home Page Reader 3.0; available at http://www.ibm.com/able/solution_offerings/hpr.html

**LEE, A. 2004.** Scaffolding visually cluttered web pages to facilitate accessibility. In Proceedings of the Working Conference on Advanced Visual interfaces

**MUKHERJEE, S., RAMAKRISHNAN, I., MICHAEL, K. 2004.** Semantic Bookmarking for Non-Visual Web Access, ASSETS 2004, 185-192, October, USA.

**PARENTE, P. 2004.** Audio enriched links: web page previews for blind users. In Proceedings of the 6th international ACM SIGACCESS Conference on Computers and Accessibility

**PETRIE, H, HAMILTON, F. AND KING, N. 2004.** Tension, what tension? Website Accessibility and Visual Design, In Proceedings of Int. Cross-Disciplinary Workshop on Web Accessibility.

**PONTELLI, E., GILLAN, D., XIONG, W., SAAD, E., GUPTA, G., AND KARSHMER, A. I. 2002**. Navigation of HTML tables, frames, and XML fragments. In Proceedings of the Fifth international ACM Conference on Assistive Technologies

**SEEMAN, L. 2004.** The Semantic Web, Web Accessibility, and Device Independence, In Proceedings of International Cross-Disciplinary Workshop on Web.

**SIMÕES, A.M. 2004.,** Down Translating XML with XML::DT, The Perl Review, Vol.1 i1 Winter 2004, http://www.theperlreview.com/
**STEVENS, R. D. AND EDWARDS, A. D. N. 1996.** An approach to the evaluation of assistive technology. In Proceedings of ASSETS '96. ACM Press. 64–71.
**STONE, R. G. AND DHIENSA, J. 2004.** Proving the validity and accessibility of dynamic web-pages. In Proceedings of the 2004 Int. Cross-Disciplinary Workshop on Web Accessibility.
**TAKAGI, H. AND ASAKAWA, C. 2000.** Transcoding Proxy for Non-Visual Web Access, ASSETS, 164-171, November, USA.
**TAKAGI, H. AND ASAKAWA, C. 2001.** Transcoding System for the Non-Visual Web Access., In CSUN's Sixteenth Annual International.
**TAKAGI, H., ASAKAWA, C., FUKUDA, K., MAEDA, J. 2002**. Site-wide Annotation: Constructing Existing Pages to be accessible, ASSETS 2002, 81-88, July, Scotland
**YESILADA, Y., STEVENS, R., GOBLE, C., HUSSEIN, S. 2004.** Rendering Tables in Audio: The Interaction of Structure and Reading Styles, In Proceedings of ASSETS, ACM Press, 16-23.
**ZAJICEK, M., VENETSANOPOULOS, I., AND MORRISSEY,W. 2000.** Web access for visually impaired people using active accessibility. In Proc Int. Ergonomics Association 2000/HFES.

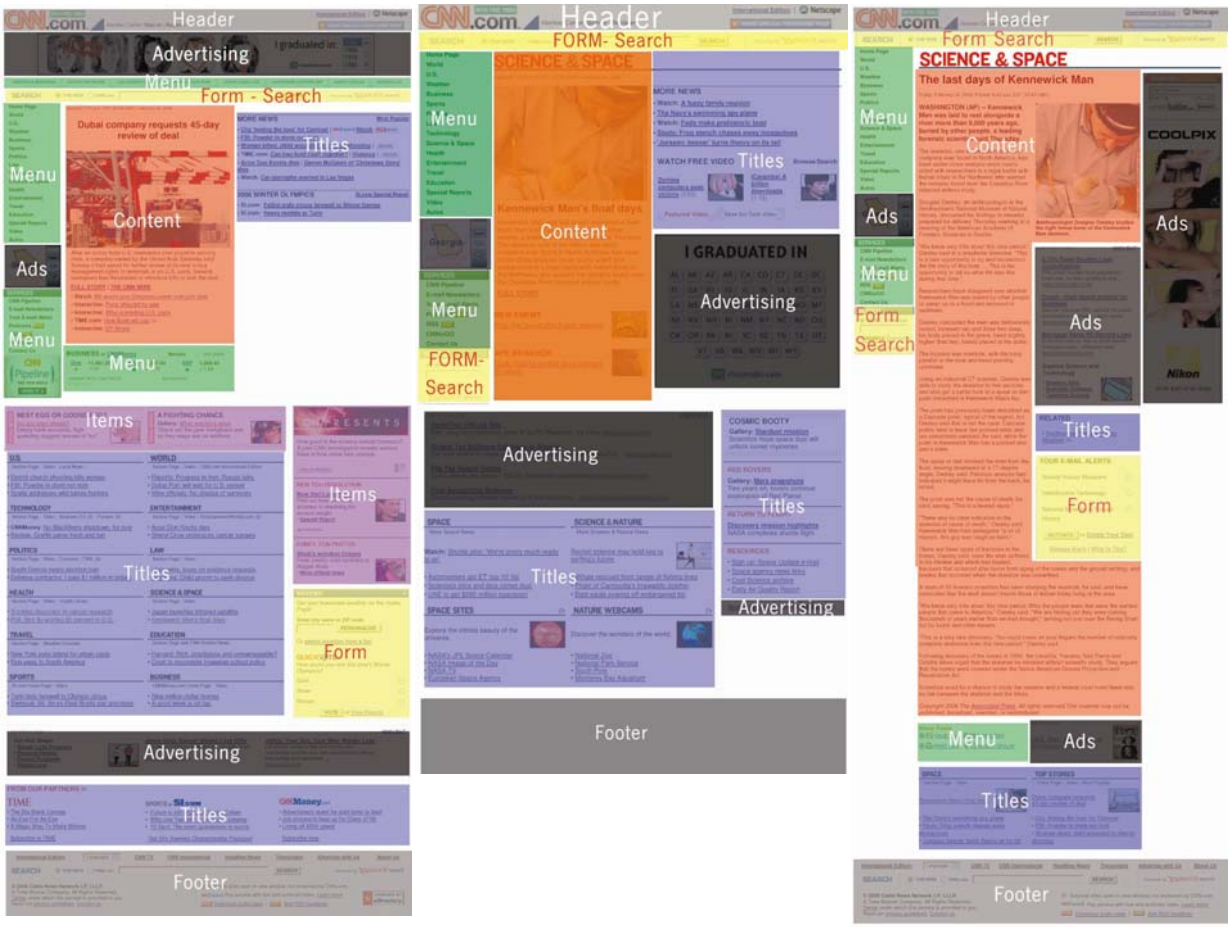


**FIGURE 7 A WELL KNOWN WEB SITE WITH SEMANTICALLY CLASSIFIED BLOCKS HIGHLIGHTED IN DIFFERENT COLOURS. LEFT: HOME PAGE; CENTER: SECTION PAGE; RIGHT: NEWS ITEM PAGE**