Personalizable Edge Services for Web Accessibility

Gennaro Iaccarino
ISISLab
Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"
Università di Salerno, Italy
iaccar@dia.unisa.it

Delfina Malandrino
ISISLab
Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"
Università di Salerno, Italy
delmal@dia.unisa.it

Vittorio Scarano
ISISLab
Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"
Università di Salerno, Italy
vitsca@dia.unisa.it

ABSTRACT

Web Content Accessibility guidelines by W3C [29] provide several suggestions for Web designers on how to author Web pages in order to make them accessible to everyone. In this context, we are proposing the use of edge services as an efficient and general solution to promote accessibility and breaking down the digital barriers that inhibit users with disabilities to actively participate to any aspect of our society.

To this aim, we present in this paper PAN: Personalizable Accessible Navigation, that is a set of edge services designed to improve Web pages accessibility, developed and deployed on top of a programmable intermediary framework [8].

The characteristics and the location of the services, i.e. provided by intermediaries, as well as the personalization and the opportunities to select multiple profiles make PAN a platform that is especially suitable in accessing the Web seamlessly also from mobile terminals.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]; H.5 [**Information Interfaces and Presentation**]; K.4 [**Computers and Society**]: Social issues—*Handicapped persons/special needs*

General Terms

Human Factors

Keywords

Web Accessibility, Programmable edge servers, disability, universal access.

1. INTRODUCTION

The World Wide Web, with its ability to is a "24x7 source of information", and the services it provides, is a remarkable reality in our society. Many different explanations can motivate this astonishing success, from technological to sociological to economical ones. Among the former, very important is the emphasis on the capabilities of the standards to accommodate a wide range of services, therefore, pushing the World Wide Web as a universal access portal to the information, wherever located and however accessed.

In spite of this tremendous and strengthened success, an important challenge affects the structure of the pages that the Web increasingly provide every day: most of them are published without any consideration about an important question: *"is this Web page accessible for everyone*?".

More specifically, Web accessibility means that people with disabilities can easily navigate and interact with the Web. Conversely, currently most Web sites have accessibility barriers that make difficult or impossible for many people with disabilities to use and access the Web. Using keyboards and mouse, hearing video and audio multimedia files, browsing through some intrusive Web pages (i.e. Web pages with pop-ups, advertisements, etc.) could appear as a normal activity for "non-disabled" people, but, on the other hand, it appears as a not simple task for users with some type of disability.

Currently there are no universally accepted categorizations of disability, but as described in [10] they can be classified into the following main categories: *Visual disabilities*, that include blindness (uncorrectable loss of vision) low vision and color blindness (a lack of sensitivity to certain colors); *Hearing Impairments* that include deafness (uncorrectable impairment of hearing) or a mild hearing impairment; *Physical disabilities* that include motor disabilities (weakness, limitations of muscular control, such as lack of coordination, limitations of sensation in hands and arms); *Speech disabilities* that include difficulty in producing speech that is recognizable by software for voice recognition; *Cognitive disabilities* that include, for example, dyslexia, attention deficit disorder, intellectual and memory impairments.

In particular, in the above classification, visual impairments produce the most common source of difficulty for users accessing the Web. Small fonts, some text and background color combinations, background images, and blinking text represent a problem for Web users. Moreover, blind people notice several problems if Web pages are composed of a lot of images, tables, and not only of linear text. In addition, for users with hearing impairments, the most critical difficulty comes from the increasing popularity of audio and video multimedia applications that, on the other hand, represent an increasingly important part of many Web sites today.

Of course, the accessibility problems are greatly enhanced by the navigation by mobile terminals, since the "traditional" transcoding operates only taking into account the limitations of the devices and not taking into account the potential disabilities of the user. In literature, we can find several examples of intermediary adaptation systems for mobile terminals such as iMobile by AT&T [25], the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
W4A at WWW2006, 23-26 May 2006, Edinburgh, UK.
Copyright 2006 ACM 1-59593-281-x/06/05 ...$5.00.

TACC infrastructure [11], RabbIT[1] and Privoxy[2].

Our goal. In this paper we present PAN: Personalizable Accessible Navigation, that is, a set of edge services that aims to make the navigation on the Web more accessible for any user that access it. In particular, our main goal is to provide efficient adaptation services, that is, services that are able to apply different type of *on-the-fly* transformations on Web pages, in order to meet different users preferences/needs/abilities.

The rationale that lead our project is the *effectiveness* of intermediary frameworks to *efficiently* provide services in a transparent way for end users that, therefore, could benefit of complex functionalities anytime, anywhere and by using any client device. Users with disabilities can take advantage of the provided edge services by simply choosing a profile, among the provided ones (or to define a new profile from scratch), that mainly suit their needs. Finally, it must be emphasized that improvements on Web pages could positively affect all users, for example, support for speech output could not only benefits blind users, but also Web users whose eyes are busy with other tasks.

Organization of the paper. In the next section, we present the important issue of Web accessibility and, therefore, strongly motivate our research. Moreover, we present some examples of accessibility systems that exist in literature, and some example of intermediary frameworks that could be programmed to provide accessibility services. In particular, we first describe how programmable proxies can be employed to promote accessibility, and in general, tackle a relevant part of the problems that are generated by the dynamic nature of the Web, then, we briefly we introduce the platforms for programmable HTTP proxies that are available nowadays. In Section 3 we provide some details about PAN and its configuration and the multiple profiles options. In Section 4 we present PAN's services that can be used for efficiently providing accessibility to people for disabilities, in several contexts, included the mobile Web. Finally, in Section 5, we conclude with some final comments.

2. WEB ACCESSIBILITY

The W3C Consortium is employing a lot of efforts and is leading a lot of activities to make the most famous information space accessible for anyone, and then allow people with disabilities to actively perceive, understand, navigate, and interact with the Web.

Within this area our interest is mainly devoted to the provision of personalized applications, that is, applications that are able to customize Web content according to users' preferences and needs. In particular we have implemented some examples of services that make Web pages more accessible for users with visual and motor disabilities.

Some disabilities, such as hearing impairments, do not require any transformation on HTML pages accessed by users. Visual and motor impairments could be addressed by using screen readers or *assistive* technologies, respectively (modified keyboards or software for keyboard and mouse emulation). But our aim is how to address these challenges with an instrument that is able to personalize the Web without any modification both on client and server systems.

Some surveys[3] on accessibility in Web design estimate that there are 180 million people worldwide who are visually impaired, 45 million are blind and 135 million are partially sighted. In the US, 16.5 million people suffer of vision impairments. Finally, the efforts in making the Web a more accessible place is very important since the the rapid aging of the population and the estimated increase of disabled users (a reasonable estimate is a doubling by 2020 worldwide and a total of 20 million US people by 2010).

2.1 Examples of Web accessibility systems

Assistive devices, like screen readers and audio browsers, are being used by blind users to access the Web. However, these systems show limitations since they are not able to filter Web pages for removing useless information involving users, bothered by more and more intrusive elements, to quickly leave their frustrating navigation.

In order to address this issue, several systems have been developed with the aim at providing Web content accessible to everyone. These systems, that can be classified as filter and transformation tools, include for example, systems that build an only-text version of Web documents (BETSIE and textualise), systems that apply transformation according to users' preferences (Web Access Gateway and Tablin), systems that apply transformations according to specific rules and heuristics (WAB).

BETSIE, acronym of BBC Education Text to Speech Internet Enhancer [6], is a simple CGI Perl script whose main goal is to tackle the difficulties experienced by people using text-to-speech systems for Web browsing. BETSIE produces on-the-fly text-only version of every Web page navigated by the end user and, in particular, it modifies Web pages by handling frames, by linearizing tables, by removing images, Java and Javascript code, etc. Finally, it allows any user to choose among some color-theme, specific sizes and font for text. An important limitation of BETSIE is that it has not been possible for the BBC to offer a full text-only version of the entire Internet, this means that BETSIE only works with BBC sites.

The {textualise;} system [27], transforms a Web site from a graphics-heavy and inaccessible version, to a text-only version that is easily accessible for visually impaired users. By following the W3C's Web Content Accessibility Guidelines, it removes elements that screen readers cannot handle, allows user customization of fonts and background colors, it removes Java applets, Javascript code, graphics and fixes ALT attributes, replaces Shockwave, Flash, and other plug-in applications with a link to them. etc. A server component, within the textualise; system, performs the required transformations by using a set of transformation rules.

The Access Gateway system [7], specifically designed for people with low vision or dyslexia, is able to handle frames (by serializing them), tables (by linearizing them), to remove images (by substituting them with missing ALT attributes), JavaScript codes, Flash, cookies. Another goal is how to allow speakers of other languages to view Web pages written in these languages, when the encodings are not supported by their browsers. It is implemented as CGI script in C++.

WebAdapter [19] is WWW browser that provides accessibility functionalities for blind, visual and physically impaired people. Their idea is to include these functionalities within a browser without affecting the UI for non disabled users. In particular the adaptations provided by WebAdapter include adaptation for physically impaired users (customization of images sizes), adaptations for visually impaired users (turning-off of background images), and finally, adaptations for blind users (sequential presentation of tables). Finally, WebAdapter use an integrated speech synthesizer to read HTML documents for people with visual disabilities.

The Accessibility Transformation Gateway [23] is a transcod-

[1] http://rabbit-proxy.sourceforge.net
[2] http://www.privoxy.org
[3] http://www.lighthouse.org/

ing gateway designed to apply on-the-fly transformations of Web pages in order to adapt them for users with disabilities. The Web pages, filtered by the gateway, can be easily handled and rendered by screen readers or assistive technologies. The access gateway intercepts requests/responses by applying transformations according the accessibility and the usability rules [29]. An important step include the construction of the Document Object Model (DOM) tree representation of the requested HTML page. Finally, some chosen rules include: *(a)* providing alternate text for images, applet and for each Object, *(b)* linearizing tables, *(c)* avoiding any URL redirect and automatic page refresh, etc. Other functionalities, implemented as part of the accessibility transcoding gateway project, are *simplification* (i.e. deletion of clutter information on Web pages) and *summarization* (building of a preview of Web pages) [22].

The IBM system described in [17] enables universal access to the Internet content by allowing different types of devices, and people with different abilities, to receive content that is suitable for their needs. It removes comments and JavaScripts, handles images (by modifying colors or removing them at all), linearizes tables, allows text summarization. The evolution of this system led to a most famous IBM WebSphere Transcoding Publisher technology[4] that dynamically translates Web content and applications to meet different client devices capabilities and users preferences.

Crunch [15, 16] is a Web proxy that uses a set of heuristics to extract content from HTML pages in order to make it accessible according to W3C Guidelines. It employs a set of heuristics, that is filters operating on a DOM representation of a Web page, to remove all extraneous of useless information (not recognizable, for example, by screen readers). In particular, the content extractor navigates the DOM tree recursively and remove and adjust specific nodes by leaving only the content behind. Crunch provides filters for removing images, links, scripts, and more complex filters to remove advertisements, link lists, empty tables. Once entirely parsed and adapted, the page can be rendered both in HTML and plain text (for example for text to speech and summarization). Finally, the Crunch is not able to handle Flash and dynamic generated codes and to distinguish between different accessing users.

Other examples include WAB, a not customizable HTTP proxy (based on CERN httpd) that modifies Web pages to assist blind users. The page is transformed in a text-only version to make it easily readable by screen readers. Tablin [26] is a filtering system, developed by the WAI Evaluation & Repair (ER) group, that can be used to linearize HTML tables and render them in a form suitable for their reading from screen readers.

2.2 Intermediary Frameworks

Most Web sites, today, are designed to follow the *"one-size-fits-all"* philosophy, by providing content and services without taking care of the abilities of users that access them. In fact, because of its increasingly complex infrastructure, the Web does not really provide equal access and equal opportunity for users with disabilities. The documents available on the Web exhibit a growing complexity, especially for people with visual disabilities, that often are unable to access, summarize and distill information on a Web pages or groups of pages. For people with motor disabilities the WWW represents a very important source of information, a familiar and ubiquitous environment where to get information about any aspect of life: education, employment, shopping, business, government and more.

Hence, as the Internet continues to evolve with an increasing diversity and heterogeneity, there is a growing demands for technological solutions that are able to allow the universal access to the Web content, by breaking down the accessibility barriers that inhibit the access by users with disabilities. These technologies could be provided server-side, client-side or on intermediary systems. While server-side solution could involve several efforts for Web designers for writing an re-designing accessible Web pages, and client-side solution require the development of adaptive browsers or other assistive technologies, intermediary solutions could be provided without any intervention both on client and server systems, by allowing transcoding and Web personalization on-the-fly, transparently for users with disabilities [1, 20].

By summarizing, the main advantages of edge servers systems are that they can be transparently deployed without involving hardware and/or software changes both on client and server systems, they can reduce both complexity on servers (that will only take care of providing the requested resources) and system requirements on clients (independently from device's capabilities), add new services without stopping the servers, by ensuring their fault-tolerance, offer increased flexibility (where new components can be deployed) and scalability, and, finally, improve the quality of access to the resources available on the Web.

Increasingly often, intermediaries have been used to develop and deploy services for accessibility. Famous examples of intermediary systems that can be easily programmed to promote accessibility include Muffin [`http://muffin.doit.org`], a Web HTTP proxy that provides functionalities to remove cookies, kill GIF animations, remove advertisements, add, remove, or modify any HTML tag, remove Java applets and JavaScript code, etc.

RabbIT [`http://rabbit-proxy.sourceforge.net`]is a Web HTTP proxy that accelerates the delivery of Web contents to end users by compressing text pages and images, by removing unnecessary parts of HTML pages (background images, advertisements, banners, etc.) and, finally, by caching filtered documents before forwarding them to the clients.

Web Based Intermediaries (WBI) [3, 4, 5] is a dynamic and programmable framework, developed at IBM Almaden Research Center [`http://www.almaden.ibm.com/cs/wbi`], whose main goal is to personalize the Web by realizing an architecture that simplifies the development of intermediary applications. WBI defines a programming model that can be used to implement all form of intermediaries, from simple server functions to complex distributed applications.

On top of the WBI programmable proxy, as part of the more complex SEcS: Scalable Edge-computing Services system [12], has been developed the Test-To-Speech service [2], that allows the speaking of the text of HTML pages during their displaying towards end users. Moreover, the Text-To-Speech Service can help the comprehension of documents that are written in foreign languages, since a reader that is partially familiar with the spoken language and not with its written form can be supported in getting, at least, the meaning of the information contained in the document. Finally, the service can be used as a support toward kids that want to learn a language since provides a written/spoken presentation of material of their interest during a *natural* experience as navigating the Web.

The framework that has been use to develop PAN is Scalable Intermediary Software Infrastructure for edge services (SISI): a brief description of the framework and its functionalities are presented next.

2.3 Scalable Intermediary Software Infrastructure (SISI)

The Scalable Intermediary Software Infrastructure for edge services (SISI) [8] is a flexible, dynamic and *programmable* intermedi-

[4] http://www-306.ibm.com/software/pervasive/transcoding_publisher/

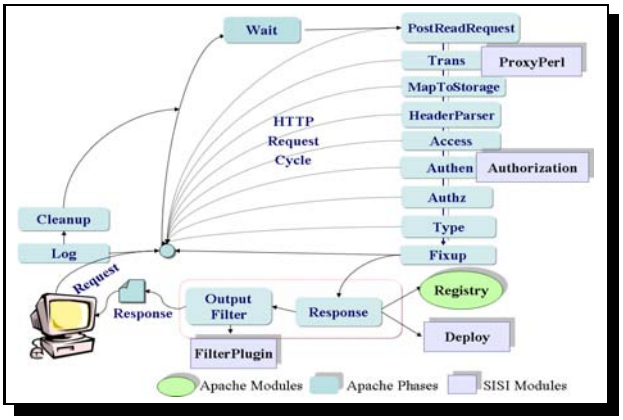

Figure 1: Placement of SISI modules into the Apache request Life-cycle.

ary infrastructure that enables universal access to the Web content. This framework has been designed with the goal of guaranteeing an efficient and scalable delivery of personalized services at intermediate edge server on the WWW.

SISI programmability is a crucial characteristics since it allows an easy implementation and assembling of adaptation services that enhance the quality of services perceived by users during their navigation. To allow programmability, the SISI framework provides a programming model and a set of APIs that can be used for a quick prototyping and a easy development of new services to improve the navigation on the Web for not-disabled users as well as for disabled ones. Services can be assembled and configured to enhance the set of pre-defined functionalities.

The architecture is innovative since it provides programmability without giving up efficiency, offers users profiles management primitives, deployment/un-deployment and authentication/authorization mechanisms. The work is placed on top of existing open-source applications such as Apache Web server and mod_perl because of their quality and their wide popularity.

The SISI framework is composed of different modules, entirely written in Perl language, each acting in a specific phase of the Apache HTTP Request life-cycle (See Fig. 1).

The *ProxyPerl Module* intercepts all clients requests and initializes the transaction process. Its most important task is to fetch the requested URLs manipulating, if necessary, HTTP headers. If no transformation has to be applied on the HTTP flow, the requested document will be sent back unchanged to the client. Otherwise, the transaction will be managed by the handlers invoked according to the user's profile.

The *Authorization Module* is useful both to restrict access to the proxy server as well as to distinguish between users, so that each user can have different SISI services applied to her requests.

The *FilterPlugin Module* acts as a dispatcher within the architecture by activating the services according to the users' preferences.

Finally, the *Deploy Module* is used by the programmer to add new services to the framework. It consists of an automatic modules generation process that implements intermediary services starting from simple Perl files.

The SISI framework also provides a Graphical User Interface implemented to simplify the management and the debugging of the SISI components. The goal is to relieve the system administrator and programmers from learning or remembering complex and tedious commands during the administration phase and the debugging of new deployed modules.

SISI user-friendly configuration of services is an important feature since in this phase the users can easily provide information about the required services and personalize their navigation on the Web. In fact, by allowing services configuration, SISI is able to affect the adaptation of a given delivery context, and to change accordingly the user experience.

3. Personalizable Accessible Navigation

Personalizable Accessible Navigation, developed on top of the SISI framework, offers to people with disabilities important functionalities described below.

Configurability: each user is authenticated and services can be applied according to the chosen configuration. Of course, it is possible to activate/deactivate accessibility services as preferred, by acting on a personal profile page.

Easy deployment: client-side configuration is quite simple, since it is enough to use the proxy setting on the browser and providing the authentication to PAN to use services. Moreover, it provides ubiquitous services from any node and does not require installation (and therefore, administrator privileges) which makes it particularly useful when accessing the Web from public terminals as well from mobile terminals. Our main goal is to avoid any complex and computationally onerous browser-dependent technology [18] and provide the same set of services to each user, in any context (be it home, workplace or public terminal) with different devices (such as fixed or mobile terminals).

Efficiency: the accessibility services are executed on the path from server to client, therefore amortizing the cost of implementing expensive transcoding techniques onto the "natural" delay that is considered physiologic.

3.1 Installation

To use the PAN's set of accessibility services, users have, firstly, to install the SISI framework, that is available as raw source code for Unix/Linux platforms and in a pre-compiled version for Windows. The installation of the SISI framework requires the installation of the Apache Web server and mod_perl, and some packages for image manipulations (PerlMagick library) and for developing Web client applications (LWP User Agent perl package).

The installation and the deployment of PAN is accomplished by simply using the deployment mechanism provided by the SISI framework (see [8] for more details).

3.2 Configuration and profiles

Once deployed, the PAN set of services is accessible through a specific configuration page (See Fig. 2).

In fact, our approach to manage user profiles is to explicit ask the user what s/he needs and use this information with a rule-based approach to personalize the content. In particular, users have to fill-out forms to create new profiles and to modify or delete the existing ones, as shown in Fig. 2.

Our user-friendly configuration of services is an important feature since in this phase the users can easily provide information about the required services and personalize their navigation on the Web. In fact, by allowing services' configuration, it is possible to affect the adaptation of a given delivery context, and to change accordingly the user experience.

When a new user is added to the system, a default profile is automatically generated and the user can modify it the first time s/he logs into the system to choose his/her preferences.

As an example, a user with a moderate low-vision disability may be able to navigate with a modest help of PAN when using his/her usual terminal (due to good indoor illumination and large-size screen) but may need more help when accessing the network by public terminals (i.e. ordinary-sized screens) or even more help is needed when browsing the Web by a mobile terminal. In each context, the user can select a different profile, previously set and configured on PAN, obtaining the right amount of support, right there, when he/she needs it.

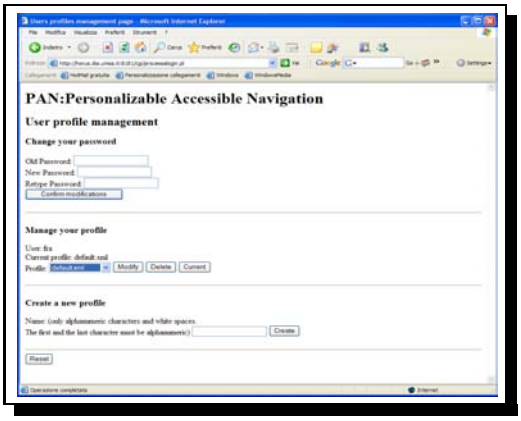

Figure 2: Users's Profile Management.

Finally, it must be mentioned that PAN is the only system that allows personal configuration of services as well as multiple profiles when compared to the systems described in section 2.1.

4. PAN SERVICES FOR ACCESSIBILITY

The services provided by PAN are grouped in four main categories depending whether they act on text, links, images or other objects on the HTML page (such as pop-up windows, etc.) according, also, to the classification implicitly provided in [29]. Of course, many of the guidelines provided by W3C for making accessible a Web site can be also found on the corresponding Best Practices for Mobile Web [14]: as a matter of fact, dealing with user limitations or terminal limitations can be seen as the two faces of the same goal: promote universal access to Web resources.

4.1 Text-based edge services

Here we describe two services that adapt Web pages by taking into account the rules suggested by W3C to improve accessibility [29] and to enhance, in general, the navigation of Web pages [28], and, more specifically, of CSS files [9].

4.1.1 The CSS-Restyling edge service

Cascading Style Sheets (CSS) files benefit accessibility since they separate document structure from its presentation. More specifically, they allow to define how different elements, such as headers, text, links, etc., should appear to end users. They establish a precise control over the structure of Web pages, by providing rules for characteristics such as spacing, alignment and positioning. They also allow to define a precise control over text style effects such as font size, color, and color-contrast, etc.. Finally, style sheets allow Web designers to simplify the structure of Web pages and clean up them by useless information and dynamic elements, by making them more slight and more accessible at the same time (i.e. more readable from screen readers).

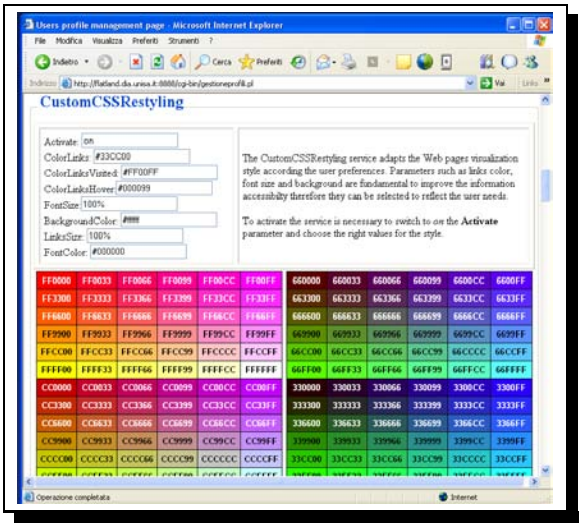

Figure 3: Form to for the CSS-Restyling edge service customization.

Our CSS-Restyling edge service accesses the Web page retrieved by the origin server, parses it and, for each HTML tag of the CSS file, modifies its attribute by replacing it with accessible values as suggested by [9, 14] . In particular, in order to deliver information to a greater number of users, we perform the following changes:

- Font size (text and links) is changed according to service parameters (Guideline 11 in [13]);
- Text and link colors, background and foreground color combinations are changed to provide sufficient contrast when viewed by users with color deficiency (Checkpoint 2.2 *Color contrast*, Guideline 2 *Text formatting and position* in [13]);
- Blinking content is turned off (Checkpoint 7.2 *Text style effects*, Guideline 7 in [13]);
- Textual cues instead of images (by always providing for each image the corresponding `alt` attribute with the content attribute of the img tag, if present, or the name of the image) (Checkpoint 3.3 *Text instead of images*, Guideline 3 in [13]).

The CSS-Restyling edge service applies transformations on Web pages by taking into account both internal and external CSS files. The values provided by the CSS-Restyling service can be customized by the user according to his/her needs and abilities by accessing a form on the proxy side (see Fig. 3).

An example of application of the CSS-Restyling edge service is shown in Fig. 4.

4.1.2 The TextColor-Restyling edge service

The TextColor-Restyling edge service ensures that foreground and background color combinations of Web pages provide sufficient contrast when displayed to users with color deficiencies as defined by the Guideline 2 in [29], and Section 5.3: *Page Content and Layout* in [14].

The TextColor-Restyling service accesses the HTML page retrieved by the origin server and, for each HTML tag, analyzes its corresponding attribute looking for background and/or foreground text information. More precisely, the following attributes will be taken into account: color, bgcolor, background, text, link, alink, vlink and style's attributes that specify images/backgrounds/colors.

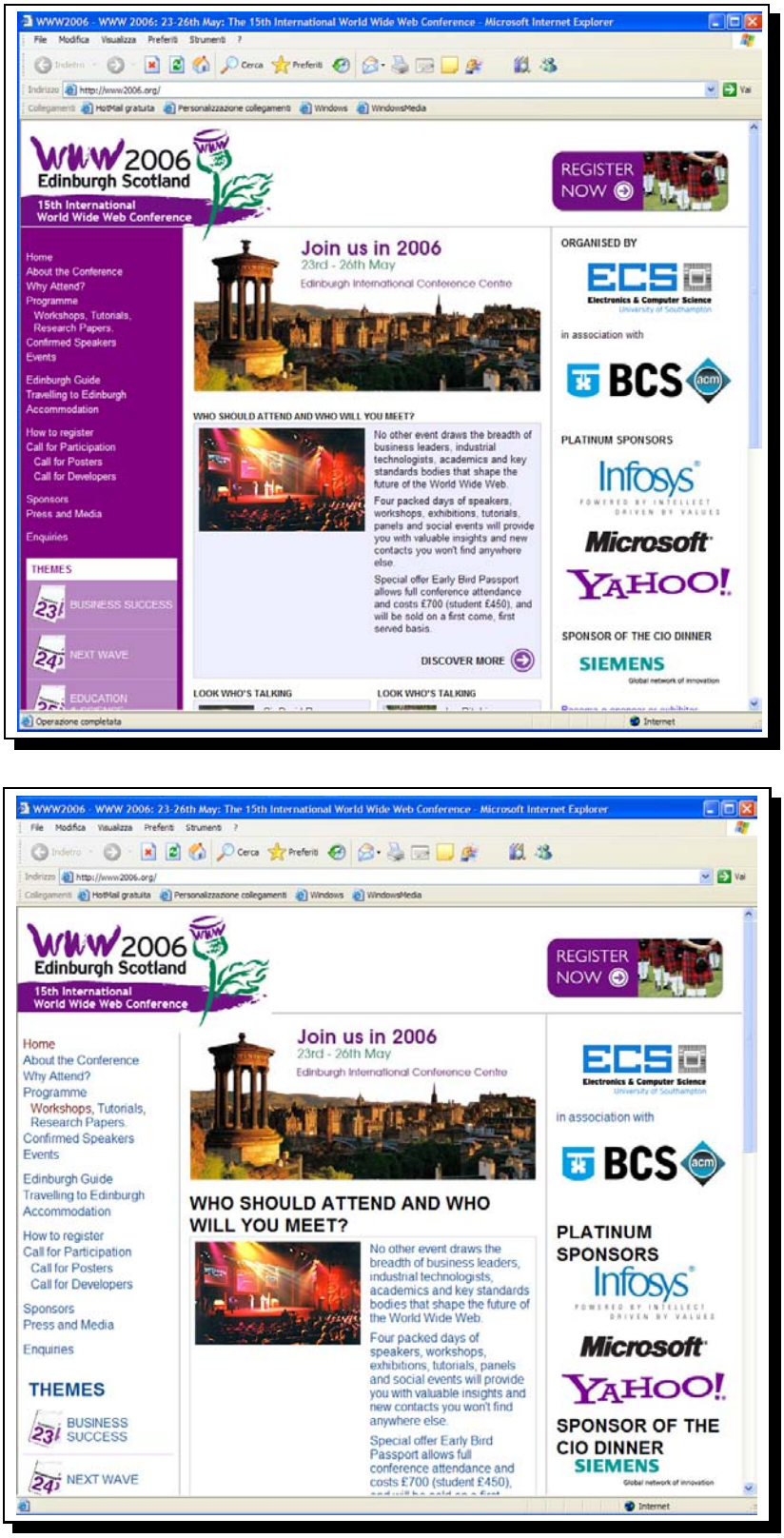

Figure 4: The CSS-Restyling edge service. The original WWW 2006 Website (top) and the same page with the application of the CSS-Restyling service by PAN (bottom).

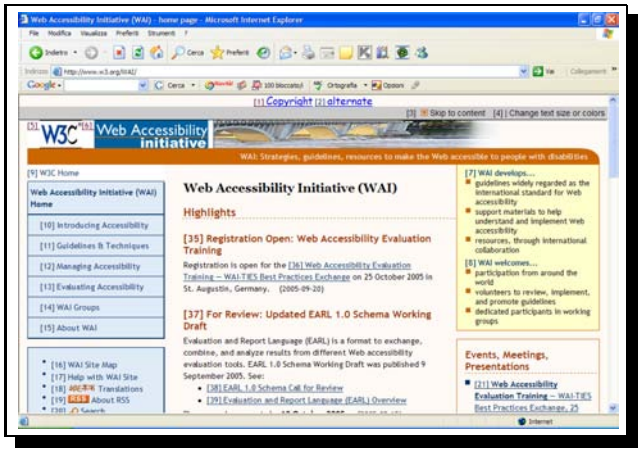

Figure 5: The LinkAccessKey and Linkrelationship edge services.

The main goal of this service is to adapt Web pages in order to make them more accessible for people with color deficiencies.

4.2 Link-based Services

This section include services that act on links of Web pages in order to make Web pages more readable when users use assistive technologies such as speech synthesizers, screen readers, etc.

4.2.1 The RemoveLink Service

By following the Guideline 1 defined in [29], the RemoveLink edge service brutally removes all anchors (the HTML `<A>` tag) from HTML pages, by replacing them with plain text (the text between the open and close of the anchor tag).

The main advantage of this service is that screen readers or other vocal browsers will avoid to read useless information in Web pages when users that access them are not able to see links and follow them. Examples of Web pages include Web pages news, meteorology, encyclopedias pages, etc., in which no links to other resources are required.

4.2.2 The LinkAccessKey Service

The W3C Guideline 9 *Design for device-independence* [29], in section Keyboard access, and the section 5.2 *Navigation and Links* (5.2.6 Access Keys) in [14] state the need to use access keys HTML elements to allow users with some disabilities to browse the Web.

Content developers should always ensure that users may interact with a page with input devices other than a mouse. Unfortunately, since this limitation is not often considered, we provided a solution by developing a service that allows both motor and visual disabilities users to browse the Web without limitations.

This service adds to any link embedded in a Web page a numeric *Access Key* in such a way to make it accessible through a simple combination of keyboard keys, ALT+*Access Key*+Return for example (See Figs. 5 and 6). Pressing the access key assigned to an element gives focus to the element, and the corresponding action will be executed. In particular by pressing the access key, the browser will follow the corresponding destination link. Moreover, the numeric value of the access key is also added into the HTML source of the Web page so it can be easily read by any screen reader. The LinkAccessKey edge service could be useful to improve the Web navigation on devices with limited display capabilities.

4.2.3 The LinkRelationship edge Service

As defined by the W3C guideline 13 *Provide clear navigation mechanisms* and by section 5.2 *Navigation and Links* (5.2.2 Navigation Bar) in [14] content developers should use the `LINK` element and link types to describe document navigation mechanisms and organization. Some user agents may synthesize navigation tools or allow ordered printing of a set of documents based on such markup. The `LINK` element may also be used to designate alternative documents. Browsers should load the alternative page automatically based on the user's browser type and preferences. For example, content developers can produce a different content for browsers that support "braille" rendering.

Our service adds a toolbar containing the `LINK` attributes on top of each HTML page, as shown in Figure 5. It can also be useful to make HTML pages more accessible and more readable by screen readers, and for improving the Web navigation trough devices with limited capabilities.

The W3C defines, other than the traditional set of relationship links (*Start*, *Preview*, *Next*, *Help*, *Bookmarks*), another useful set for Web accessibility. It includes, for example, the relationship link *Alternate*, designed to provide alternative versions of documents, the *Contents* link relationship, that refers to a summary or to as site map, the *Glossary*, that refers to a glossary of terms contained in the current, the *Copyright*, that it refers to a copyright of the current document, etc.

4.2.4 The DeleteTarget edge service

The `target` attribute of the HTML `A` tag specifies the frame where the document will be opened. In particular, the `blank` value allow to open a new browser window for displaying the corresponding page.

As defined by the W3C Guideline 10 *Use interim solutions* [29], new browser windows represent a critical problem for screen readers because of their impossibility to *jump* among different pages. Moreover, when the focus comes back on the original window, the screen reader starts again from the begin of the page to speech the corresponding content.

The main goal of the DeleteTarget edge service is, therefore, to avoid any target attribute in both link and anchor elements, by displaying all pages in the same browser window.

4.2.5 The LinkLinearizing edge service

The LinkLinearizing edge service allows to organize and to lexicographically order links embedded in Web documents in order to simplify their direct access by users with visual and motor disabilities, by also adding a numerical access key (See Fig. 6).

This service parses HTML pages and places all discovered links in a table on the top/bottom of them (as specified by the user during the configuration and customization of the service).

4.3 The FilterImages edge services

Web images represent a serious problem for blind users since they cannot be read by screen readers. Moreover, particular images formats or compositions, represent a barrier also for cognitive disabled people, as shown next.

4.3.1 The ImageRemoval edge Service

The ImageRemoval edge service removes any image embedded in a Web page by replacing it with a link to it, as suggested in section 5.4 *Page Definition* (5.4.5 Non Text Items) [14].

In particular, this service parses the Web page looking for HTML `img` tags, and replace them with `A` tags whose content attribute will contain the link information *(See image:)* followed by the original `img` content attribute, if it exists, and only the *(See image)* link

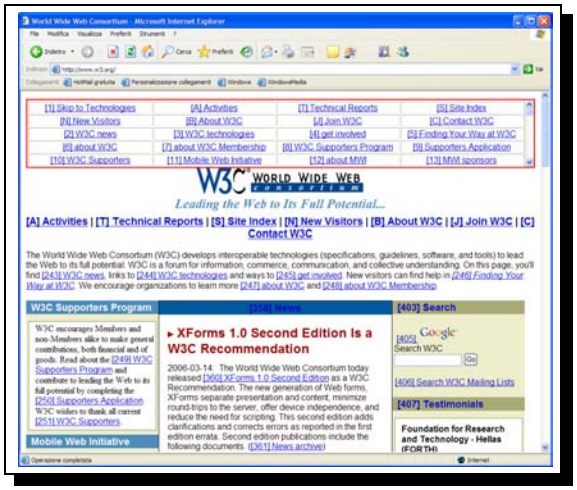

Figure 6: LinkLinearizing and AccessKey edge services.

message, conversely. In this way, only if requested, the image will be displayed to end users (See Fig. 7).

Screen readers will be able ro read all important information available in a Web page, by facilitating its comprehension by users with disabilities.

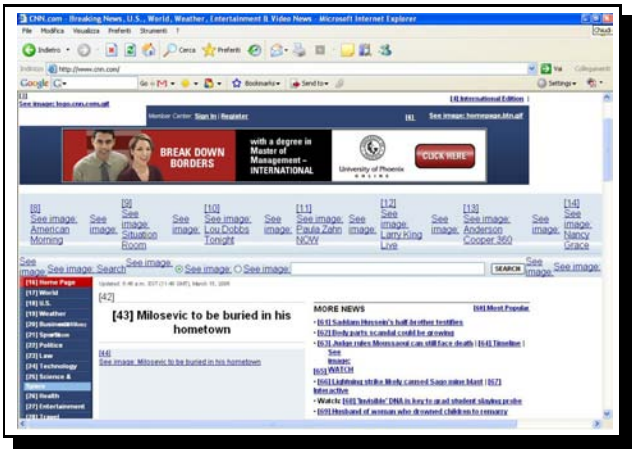

Figure 7: RemoveImages and LinkAccesskey edge services.

4.3.2 The GIF-Deanimate Service

GIF animated images represent a digital barrier for users with cognitive disabilities. In fact, these images reduce attention from the content, involving important problems to users that suffer of dyslexia or of general attention deficit pathologies.

By following the Guideline 7: *Ensure user control of time-sensitive content changes* [29] we have developed a GIF-Deanimate edge service that replaces each animated image embedded in a Web page with a static one, by showing only its first frame.

The GIF-Deanimate service has been implemented by using the well-known PerlMagick [24] library, that is an object-oriented Perl interface to ImageMagick [21], a free software that allows to create, edit, and compose images with different formats.

4.4 Easy and smooth navigation service

4.4.1 AnnojanceFilter edge service

As defined in the Guideline 10: *Use interim solutions* [29] navigating with several pop-up windows, banners, scripts etc., can be difficult for people with visual and cognitive disabilities.

The main goal of the AnnojanceFilter edge service is to get rid of particularly annoying abuse during the navigation on the Web. More precisely, the AnnojanceFilter service provides functionalities for removing advertisement, banners, pop-ups in JavaScript and HTML, JavaScript code, for disabling unsolicited pop-up windows, etc. During service's configuration users can choose the functionality to enable by providing parameters if required.

4.4.2 The HTMLClean edge service

The main goal of this service (based on the `HTML::Clean` Perl library) is to clean HTML pages from useless or redundant code. It is very useful for Web pages built with programs that silently add internal code. Other functionalities include removing white spaces (Checkpoint 3.3: *Text formatting and position* [13]), useless (i.e. non-standard) META elements, HTML comments, replace tags with equivalent shorter tags. By using the provided methods, the service is able to reduce the size of Web pages thus speeding up the download.

The HTMLClean service is able to ensure that documents are clear and simple so they may be more easily understood (Guideline 14: *Ensure that documents are clear and simple.* [29]).

5. CONCLUSIONS

Personalizable Accessible Navigation represents an efficient, effective, simple and personalizable tool to provide easier access to the Web resources to people with disabilities. Its nature (being an intermediary) and the characteristics of its services make it suitable for adapting the access to Web resources in different contexts, such as fixed/mobile terminals. Moreover, by implementing easy configuration and multiple profiles PAN enhancing the usability of the accessibility services in many different contexts.

We have realized preliminary tests in order to evaluate the effectiveness of Personalizable Accessible Navigation accessibility services. We selected a set of Web pages of international relevance, i.e., the US White House, CNN and New York Times Web portals, the Eclipse and Sun Web sites, and finally, the IEEE, and ACM Organization sites.

Their Web pages are tested using the Bobby Web Accessibility Tester (available at the URL: [`http://webxact.watchfire.com`]) that is a comprehensive Web accessibility tool designed to aid Web masters to test Web sites in order to improve their accessibility.

Moreover, Bobby tests Web pages by using the guidelines established in [28].

We compare the results obtained on the original Web pages by using Bobby with the results obtained on the Web pages adapted by PAN (by saving the adapted pages on a local server and accessing them with Bobby later on). The following PAN's services were tested: ImageRemoval, the CSS-Restyling, LinkAccessKey and AnnojanceFilter. Results are showed in Fig. 8 for the original Web pages and in Fig. 9 for the pages adapted by PAN.

In these figures we report on the Y-axis the number of errors or warnings obtained by Bobby on each page for the categories "Image Evaluation", "Insufficient Color Contrast", "Animated Objects", "Pop-ups and new browser windows", i.e, the lower the number, the more accessible is the page.

The Bobby Accessibility Tester checks if analyzed Web pages are fully compliant with all of the automatic and manual checkpoints of the W3C Web Content Accessibility Guidelines. PAN,

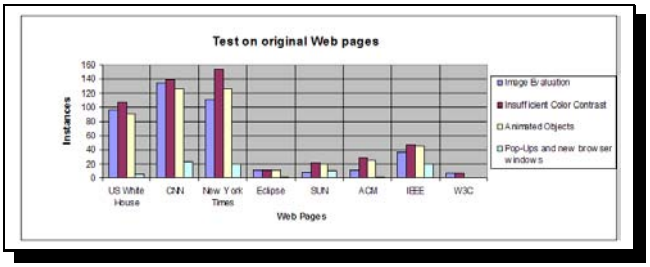

Figure 8: Bobby results on original pages.

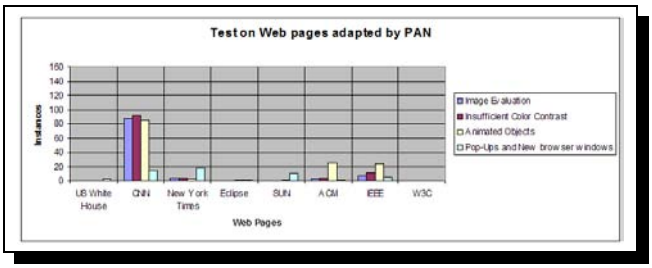

Figure 9: Bobby results on PAN's adapted pages.

instead, addresses at this time, a subset of them. This explain the reason whereby the accessibility degree is not highly improved in all the Web pages tested.

More precisely, the accuracy that comes out form tests is inconstant for Insufficient Color Contrast, Image Evaluation and Animated Objects categories since Bobby observes errors, closely related to image color contrast and Flash animations, that we do not still have provided as services in our system. On the other hand, we have already envisioned these functionalities, and we can quickly implemented them, since the programmability of the SISI's framework where PAN is leveraging on.

Moreover, since the accessibility degree is highly improved in almost the Web pages tested, our immediately important steps include further automatic tests and, in addition, detailed usability tests in order to also estimate the effective benefits that users can perceive during their navigation on the Web.

Finally, we must mention that the PAN location in the World Wide Web architecture makes it usable in three different ways: the first one is the traditional proxy setting and could be suitable for providing public services to communities. But, PAN can also act as an HTTP surrogate, that is as an intermediary that acts on behalf of an origin server to provide complex functionalities, as shown in Fig. 10 therefore adding to a non-accessible Web site the opportunity to become accessible.

On the other hand, PAN, can also be placed at the other end of the client-server path, by playing the role of an HTTP delegate on behalf of client systems, by providing personalized functionalities that can be also used for accessing off-line HTML repositories as well as intranet contents (see Fig. 11).

Acknowledgments: The authors gratefully acknowledge the support and the interesting discussions within the ISISLab. In particular, we thank Angelo Esposito for collaborating in the implementation of CSS restyling PAN service. Part of the research was supported by EU "Leonardo da Vinci" project Web for handicap integrated training environment (WHITE), I/03/B/F/PP-154143, 2003.

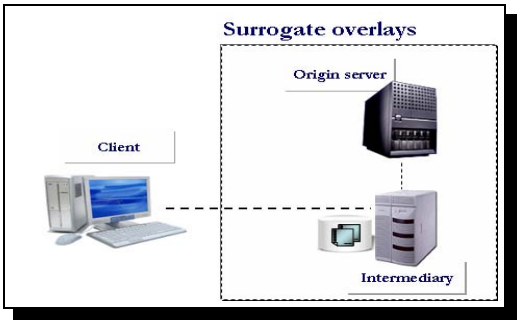

Figure 10: Intermediaries as logical extension of origin servers.

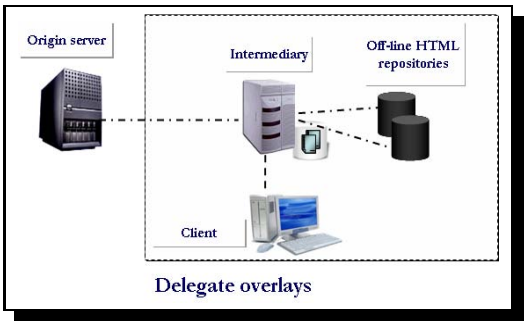

Figure 11: Intermediaries as logical extension of user agents.

6. REFERENCES

[1] C. Asakawa and H. Takagi. Annotation-based transcoding for nonvisual web access. In *Assets '00: Proceedings of the fourth international ACM conference on Assistive technologies*, pages 172–179. ACM Press, 2000.

[2] M. Barra, R. Grieco, D. Malandrino, A. Negro, and V. Scarano. TextToSpeech: an heavy-weight Edge computing Service. In *Poster Proc. of* 12^{th} *International World Wide Web Conference*. ACM Press, May 2003.

[3] R. Barrett and P. Maglio. "Adaptive Communities and Web Places". In *Proceedings of the* 2^{th} *Workshop on Adaptive Hypertext and Hypermedia, HYPERTEXT 98*, 1998.

[4] R. Barrett and P. P. Maglio. Intermediaries: An approach to manipulating information streams. *IBM Systems Journal*, 38(4):629–641, 1999.

[5] R. Barrett and P. P. Maglio. WebPlaces: Adding people to the Web. In *Proceedings of* 8^{th} *International World Wide Web Conference*, Toronto (Canada), 1999. ACM Press.

[6] BBC Education Text to Speech Internet Enhancer, 1999. http://www.bbc.co.uk/education/betsie/.

[7] S. S. Brown and P. Robinson. "A World Wide Web Mediator for Users with Low Vision". In *Proceedings of CHI 2001 Workshop No. 14*, 2001.

[8] M. Colajanni, R. Grieco, D. Malandrino, F. Mazzoni, and V. Scarano. A scalable framework for the support of advanced edge services. In *Proc. of HPCC 2005*, September 2005.

[9] CSS Techniques for Web Content Accessibility Guidelines 1.0, November 2000. http://www.w3.org/TR/WCAG10-CSS-TECHS/.

[10] How People with Disabilities Use the Web, W3C Working Draft, 10 December 2004. http://www.w3.org/WAI/EO/Drafts/PWD-Use-

Web/Overview.html.

[11] A. Fox, Y. Chawathe, and E. A. Brewer. Adapting to Network and Client variation using active proxies: Lessons and perspectives. *IEEE Personal Communications*, 5(4):10–19, 1998.

[12] R. Grieco, D. Malandrino, and V. Scarano. SEcS: scalable edge-computing services. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1709–1713, New York, NY, USA, 2005. ACM Press.

[13] W3C Note – Techniques for Web Content Accessibility Guidelines 1.0, November 2000. http://www.w3.org/TR/WCAG10-TECHS/.

[14] W3C Working Draft – Mobile Web Best Practices 1.0, January 2006. http://www.w3.org/TR/2006/WD-mobile-bp-20060113/.

[15] S. Gupta and G. Kaiser. Extracting content from accessible web pages. In *W4A '05: Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A)*, pages 26–30, New York, NY, USA, 2005. ACM Press.

[16] S. Gupta, G. E. Kaiser, P. Grimm, M. F. Chiang, and J. Starren. Automating Content Extraction of HTML Documents. *World Wide Web*, 8(2):179–224, 2005.

[17] R. Han, P. Bhagwat, R. Lamaire, T. Mummert, V. Perret, and J. Rubas. Dynamic Adaptation In an Image Transcoding Proxy For Mobile Web Browsing. *IEEE Personal Communications*, 5(6), December 1998.

[18] V. L. Hanson, J. P. Brezin, S. Crayne, S. Keates, R. Kjeldsen, J. T. Richards, C. Swart, and S. Trewin. Improving Web accessibility through an enhanced open-source browser. *IBM System Journal*, 44(3):573–588, 2005.

[19] D. Hermsdorf. WebAdapter: A Prototype of a WWW Browser with new Special Needs Adaptations, 1998.

[20] M. Hori, G. Kondoh, K. Ono, S. Hirose, and S. Singhal. Annotation-Based Web Content Transcoding. In *Proceedings of the* 9^{th} *International World Wide Web Conference*, Amsterdam (The Netherland), 2000. ACM Press.

[21] ImageMagick 6.2.6, 2006. http://www.imagemagick.org/script/index.php.

[22] B. Parmanto, R. Ferrydiansyah, A. Saptono, L. Song, I. W. Sugiantara, and S. Hackett. AcceSS: accessibility through simplification & summarization. In *W4A '05: Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A)*, pages 18–25, New York, NY, USA, 2005. ACM Press.

[23] B. Parmato, R. Ferydiansyah, X. Zeng, A. Saptono, and I. W. Sugiantara. Accessibility Transformation Gateway. In *Proc. of 38th Hawaii International Conference on System Sciences*, 2005.

[24] PerlMagick 6.22, 2005. http://www.imagemagick.org/script/perl-magick.php.

[25] C. Rao, Y. Chen, D.-F. Chang, and M.-F. Chen. iMobile: A Proxy-Based Platform for Mobile Services. In *Proceedings of the First ACM Workshop on Wireless Mobile Internet (WMI 2001)*. ACM Press, 2001.

[26] WAI Evaluation and Repair (ER) group."Tablin: an HTML Table linearizer". http://www.w3.org/WAI/References/Tablin/.

[27] Textualise; codix.net ltd., 2003. http://aquinas.venus.co.uk/.

[28] Web Accessibility Initiative (WAI), 2005. http://www.w3.org/WAI/.

[29] Web Content Accessibility Guidelines 1.0, W3C Recommendation, May 1999. http://www.w3.org/TR/WCAG10/.