

Personalizable Edge Services for Web Accessibility

UGO ERRA¹, GENNARO IACCARINO¹, DELFINA MALANDRINO¹,
VITTORIO SCARANO¹

*ISISLab,
Dipartimento di Informatica ed Applicazioni “R.M. Capocelli”
Università di Salerno, Italy*

Email: {ugoerr, iaccar, delmal, vitsca}@dia.unisa.it

Abstract: Web Content Accessibility guidelines by W3C [46] provide several suggestions for Web designers regarding how to author Web pages in order to make them accessible to everyone. In this context, this paper proposes 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 society.

The idea behind edge services mainly affect the advantages of a personalized navigation in which contents are tailored according to different issues, such as client's devices capabilities, communication systems and network conditions and, finally, preferences and/or abilities of the growing number of users that access the Web. To meet these requirements, Web designers have to efficiently provide content adaptation and personalization functionalities mechanisms in order to guarantee universal access to the Internet content.

The so far dominant paradigm of communication on the WWW, due to its simple request/response model, can not efficiently address such requirements. Therefore, it must be augmented with new components that attempt to enhance the scalability, the performances and the ubiquity of the Web. Edge servers, acting on the HTTP data flow exchanged between client and server, allow on-the-fly content adaptation as well as other complex functionalities beyond the traditional caching and content replication services .

These value-added services are called *edge services* and include personalization and customization, aggregation from multiple sources, geographical personalization of the navigation of pages (with insertion/emphasis of content that can be related to the user's geographical location), translation services, group navigation and awareness for social navigation, advanced services for bandwidth optimization such as adaptive compression and format transcoding, mobility, and ubiquitous access to Internet content.

This paper presents 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.

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 for accessing the Web seamlessly also from mobile terminals.

Key words Web accessibility, programmable edge servers, disability, universal access.

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

1 Introduction

The World Wide Web, with its ability to represent a “24x7 source of information”, and the services it provides, is a remarkable reality in modern society. Many different explanations can motivate this astonishing success, ranging 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 success, an important challenge affects both the structure of the pages that the Web increasingly provide every day, and their multimedia contents: most of them are published without any consideration about “accessibility”.

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, browsing through some intrusive Web pages (i.e., Web pages with pop-ups, advertisements, etc.) and hearing video and audio multimedia files (flash animations and audio/video contents), 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 [14], they can be classified into the following main categories: *Visual disabilities*, that include blindness (non correctable loss of vision) low vision and color blindness (a lack of sensitivity to certain colors); *Hearing Impairments* that include deafness (non correctable 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. These categories are not clearly distinguished, since there is no easy test to determine to which group a subject belongs.

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 constitute a problem for Web users. Moreover, blind people encounter several problems if Web pages are composed of a lot of images, tables, and not only of linear text. It is very important to address these issue, since a high percentage¹ of people world-wide have visual deficiencies (180 million people worldwide are visually impaired, 45 million are blind and 135 million are partially sighted. In the US, 16.5 million people have vision impairments). The efforts in making the Web a more accessible place is very important since 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).

¹ <http://www.lighthouse.org/>

In addition, for users with hearing impairments (almost 441 million worldwide), the most critical difficulty comes from the increasing popularity of audio and video multimedia applications that, on the other hand, also represent an increasingly important part of many Web sites today.

Finally, to make things even more complicated, users may have multiple disabilities (with various degrees), and that imposes several constraints on the configurability and personalization of services targeted to improve accessibility.

Accessibility problems are also worsened by the navigation on 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, several examples can be found of intermediary adaptation systems for mobile terminals, such as iMobile by AT&T [39], the TACC infrastructure [17], RabbIT² and Privoxy³.

However, although current generation (3G, UMTS, etc.) devices are increasingly able to present high resolution images of standard Web pages, ergonomic considerations of the used devices suggest that the problem is not going to disappear.

This paper presents 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. The main underlying 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 and abilities.

The rationale behind this project is the *effectiveness* of intermediary frameworks to *efficiently* provide services in a transparent way for end users, allowing them to access to the Internet, and benefit of specific functionalities anytime, anywhere and by using any client device [3, 22]. Users with disabilities can take advantage of the

² <http://rabbit-proxy.sourceforge.net>

³ <http://www.privoxy.org>

personalization of the edge services by simply choosing a profile, among the provided ones (or to define a new profile from scratch), that mainly suit their needs.

This paper is organised as follows. The next section strongly motivates the conducted research by presenting important issue of Web accessibility. Some examples of accessibility systems that exist in literature are also presented, along with some example of intermediary frameworks that could be programmed to provide accessibility services. In particular, the issue is addressed of 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, platforms for programmable HTTP proxies that are available nowadays are briefly introduced. Section 3 provides some details about PAN, its configuration and the multiple profiles options. Then, Section 4 presents PAN's services that can be used for efficiently providing accessibility to people with disabilities, in several contexts, including the mobile Web. Section 5 substantiates the claim that the services provided are, indeed, effective and efficient, by showing the results of experiments on the efficaciousness of several services (determined by Bobby Web Accessibility Tester, a comprehensive Web accessibility tool) and on the efficiency of the most computational heavy service introduced (the Colorblind filter) with an implementation on a graphical card. Finally, Section 6 concludes with some final comments.

2 Web accessibility

The W3C Consortium devotes considerable efforts and leads a wide variety of activities⁴ to make the most famous information space accessible for anyone, thus allowing people with disabilities to actively perceive, understand, navigate, and interact with the Web. Within this area, the interest is mainly devoted to the provision of personalized applications, that is, applications that are able to

⁴ <http://www.w3.org/Consortium/activities>

customize Web content according to users' preferences and needs. In particular, some examples of services that make Web pages more accessible for users with visual and motor disabilities have been implemented.

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 other assistive technologies (e.g., modified keyboards or software for keyboard and mouse emulation). However, the 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. It must also be emphasized that improvements on Web pages could positively affect all users, for example, support for speech output could not only benefit blind users, but also Web users whose eyes are busy during other tasks.

2.1 Examples of Web accessibility systems

Assistive devices, such as screen readers and audio browsers, are 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, thus leading 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 of 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 (e.g., BETSIE and {textualise;}), systems that apply transformation according to users' preferences (e.g., Web Access Gateway and Tablin), systems that apply transformations according to specific rules and heuristics (e.g., WAB). In the subsequent subsections, these systems are classified according to the role assumed on the Web and the network location where transformation functionalities are provided (server, client or proxy).

2.1.1 Server-side accessibility systems

BETSIE, acronym of BBC Education Text to Speech Internet Enhancer [8], 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, linearizing tables, removing images, Java and Javascript code, etc. Finally, it allows any user to choose among some color-themes, specific sizes and fonts for text. An important limitation of BETSIE is that it only works on BBC sites.

The {textualise;} system [43], 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 to customize fonts and background colors, 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 [10], specifically designed for people with low vision or dyslexia, is able to serialize frames, linearize tables, and remove images (by substituting them with missing ALT attributes), JavaScript codes, Flash, and cookies.

2.1.2 Client-side accessibility systems

WebAdapter [30] is a WWW browser that provides accessibility functionalities for blind, visual and physically impaired people. The 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 uses an integrated speech synthesizer to read HTML documents for people with visual disabilities.

2.1.3 Proxy-side accessibility systems

The Accessibility Transformation Gateway [36] is a transcoding gateway designed to apply on-the-fly transformations of Web pages in order to adapt them for users with disabilities. Web pages, filtered by the gateway, can be easily handled and rendered by screen readers or assistive technologies. The access gateway intercepts requests and responses by applying transformations according accessibility and the usability rules [46]. An important step includes the construction of the Document Object Model (DOM) tree representation of the requested HTML page. Finally, 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* (i.e., building a preview of Web pages) [35].

The IBM system described in [28] enables universal access to 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, and allows text summarization. The evolution of this system led to the famous IBM WebSphere Transcoding Publisher technology⁵ that dynamically translates Web content and applications to meet different client devices capabilities and users preferences.

Crunch [26, 27] is a Web proxy that uses a set of heuristics to extract content from HTML pages in order to make it accessible according to

⁵ http://www-306.ibm.com/software/pervasive/transcoding_publisher/

W3C Guidelines. It employs a set of heuristics, that is filters operating on a DOM representation of a Web page, to remove all extraneous or useless information (not recognizable, for example, by screen readers). In particular, the content extractor navigates the DOM tree recursively and removes and adjusts 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, and 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 purposes). 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 [42] 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 reading by screen readers.

2.2 Intermediary frameworks

2.2.1 Edge services

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, extract and summarize information on Web pages or groups of pages. For people with motor disabilities, the WWW represents a very important source of 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 demand for technological solutions that are able to allow universal access to the Web content, by breaking down accessibility barriers that inhibit the access by users with disabilities. These technologies could be provided server-side, client-side or through intermediary systems. While server-side solution require Web designers to write or re-design accessible Web pages, and client-side solution require the development of adaptive browsers or other assistive technologies, intermediary solutions can be provided without any intervention on client and server systems, by allowing transcoding and Web personalization on-the-fly, transparently for users with disabilities [1, 31].

Other advantages of edge servers systems are: (i) 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), (ii) new services can be added without stopping the servers, by ensuring fault-tolerance, (iii) increased flexibility (where new components can be deployed) and scalability are offered, and (iv), the quality of access to the resources available on the Web is improved.

Finally, the edge services role in the World Wide Web architecture makes them usable in three different ways. The first one is the traditional proxy setting, which is suitable for providing public services to communities. An edge service can also act as an HTTP surrogate, that is as an intermediary that acts on behalf of an origin server to provide complex functionalities. On the other hand, an edge service 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 (implemented on the client) that can be also used for accessing off-line HTML repositories as well as intranet contents.

Intermediaries are increasingly used to develop and deploy services for accessibility. This issue is discussed in the following subsection.

2.2.2 Edge servers that promote Web 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) [4–6] is a dynamic and programmable framework, developed by the 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 Scalable Edge-computing Services system (SEcS) [23], the Test-To-Speech service [2] has been developed, 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 obtaining, at least, the meaning of the information contained in the document.

The framework used to develop PAN is Scalable Intermediary Software Infrastructure for edge services (SISI). A brief description

of the framework and its functionalities are presented in the next section.

2.3 Scalable Intermediary Software Infrastructure (SISI)

2.3.1 SISI features

The Scalable Intermediary Software Infrastructure for edge services (SISI) [12] is an *efficient* and *programmable* intermediary 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 characteristic, 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 quick prototyping and easy development of new services to improve the navigation on the Web for non disabled users, as well as for disabled ones. Services can be assembled and configured to enhance the set of pre-defined functionalities.

This framework is, therefore, oriented both to programmers and users. First of all, SISI is able to offer a simple interface that allows the personalization of the content adaptation services, as well as their combination. 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.

Moreover, SISI provides programmers with a software platform that contains all basic functions that are necessary for an advanced intermediate server (i.e., user authentication and authorization, source content fetching, content delivery, management of users preferences, logging and auditing) and a set of already implemented content

adaptation services that may be customized and composed in several ways. None of the systems described before is oriented to both users and programmers, facilitates programmers to create new content adaptation services and integrate them, and provides programmers with a mechanism to transparently handle user preferences. Unlike other intermediary frameworks, SISI supports also multiple concurrent users, each with an individual profile, and offers a powerful mechanism to chain multiple content adaptation services. Besides programmability, SISI pursues also the efficiency of the provided solutions, so as to avoid typical performance degradation of most frameworks when they apply multiple content adaptation services. As shown in [12], while some of the other edge services infrastructures exhibit efficiency or programmability, SISI is able to offer both and, additionally, it also include personalization and remote configuration through easy and dynamic per-user per-profile management of the services.

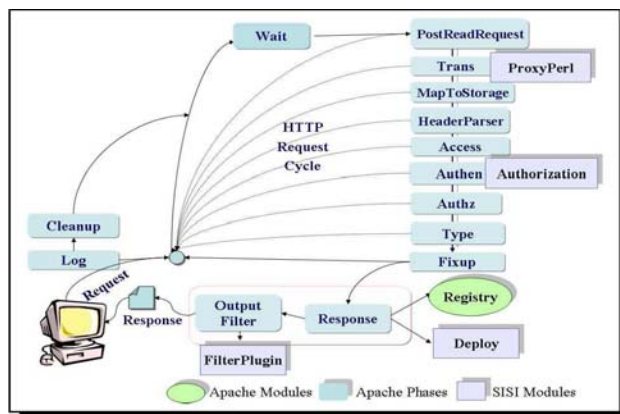


Fig. 1 Placement of SISI modules into the Apache request Life-cycle.

2.3.2 SISI architecture

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

The *ProxyPerl Module* intercepts all client requests and initializes the transaction process. Its most important task is to fetch the requested URLs by 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 individual requests.

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

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.

Finally, in order to simplify the management and the debugging of the SISI components, the framework provides a Graphical User Interface, which relieves system administrators and programmers from learning or remembering complex and tedious commands during the administration phase and the debugging of new deployed modules.

It should be emphasized that, since SISI relays on the Apache architecture and works as a set of Apache modules, it can also work in HTTPS, which, in general, is an increasingly interesting feature given the heterogeneity of the information that are transmitted on the Web.

3 Personalizable Accessible Navigation

Personalizable Accessible Navigation, developed on top of the SISI framework, offers to people with disabilities the following important functionalities:

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

Easy deployment: client-side configuration is quite simple, as it only requires using 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 as from mobile terminals. The main goal is to avoid any complex and computationally onerous browser-dependent technology [29] 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 [37]) and a set of Perl modules which provides a simple and consistent application programming interface for developing Web client applications (LWP User Agent perl package[34]).

The installation and the deployment of PAN is accomplished by simply using the deployment mechanism provided by the SISI framework (see [12] 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).

User profiles are managed by explicitly asking the user what s/he needs and using this information with a rule-based approach to personalize the content. In particular, users have to fill-in forms to create new profiles and to modify or delete the existing ones, as shown in Fig. 2.

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 the user experience accordingly.

When a new user is added to the system, a default profile is automatically generated and the user can modify it at the first log in to choose individual preferences.

As an example, a user with a moderate low-vision disability may be able to navigate with a modest help from 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 through public terminals (i.e., ordinary-sized screens), and especially when browsing the Web through 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 when needed.

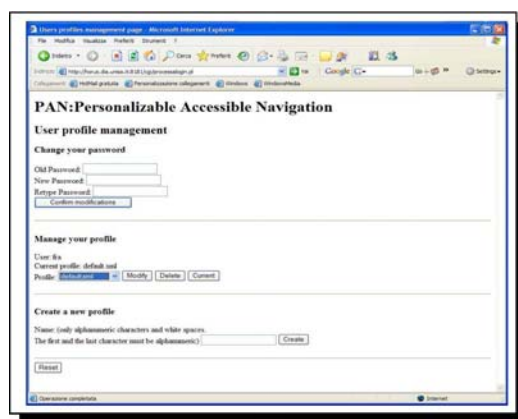


Fig. 2 Users's Profile Management.

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. In fact, SISI is the only intermediary system that allows *remote* users' services configuration by using the browser: each user can access profiles (as a set of services), create new profiles, switch among them and choose the one that is active (i.e., the set of services that are to be applied to the HTTP flow). In this way, different contexts can be easily accommodated by users, increasing the overall usability.

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 [46]. This section also describes a filter for color blind users that would fit in the image-based category but, because of its complexity, deserves its own subsection.

Of course, many of the guidelines provided by W3C for making accessible a Website can be also found in the corresponding Best Practices for Mobile Web [25]: 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

This subsection describes two services that adapt Web pages by taking into account the rules suggested by W3C to improve accessibility [46] and to enhance, in general, the navigation of Web pages [44], and, more specifically, of CSS files [13].

4.1.1 The CSS-Restyling service

Cascading Style Sheets (CSS) files benefit accessibility since they separate document structure from its presentation. More specifically, they allow defining 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 defining 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 accessible from screen readers.

The PAN CSS-Restyling 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 [13,25]. In particular, in order to deliver information to a greater number of users, the following changes are performed:

- Font size (text and links) is changed according to service parameters (Guideline 11 in [24]);
- 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 [24]);
- Blinking content is turned off (Checkpoint 7.2 *Text style effects*, Guideline 7 in [24]);
- Textual cues are provided 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 [24]).

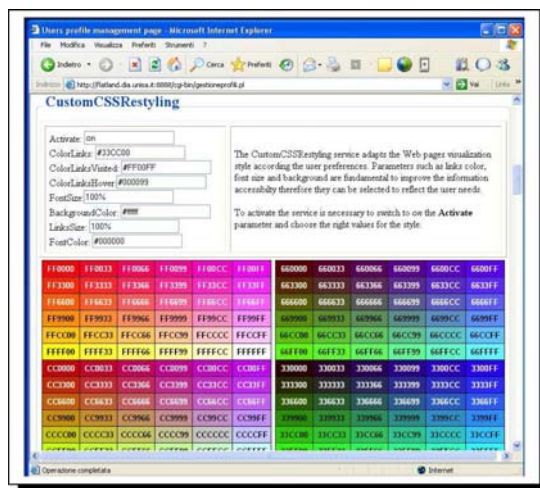


Fig. 3 Form to for the CSS-Restyling edge service customization.

The CSS-Restyling 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 individual 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 service

The TextColor-Restyling 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 [46], and Section 5.3: *Page Content and Layout* in [25].

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 foreground text information. More precisely, the following attributes are taken into account: color, bgcolor, background, text, link, alink, vlink and style's attributes that specify images/backgrounds/colors.

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

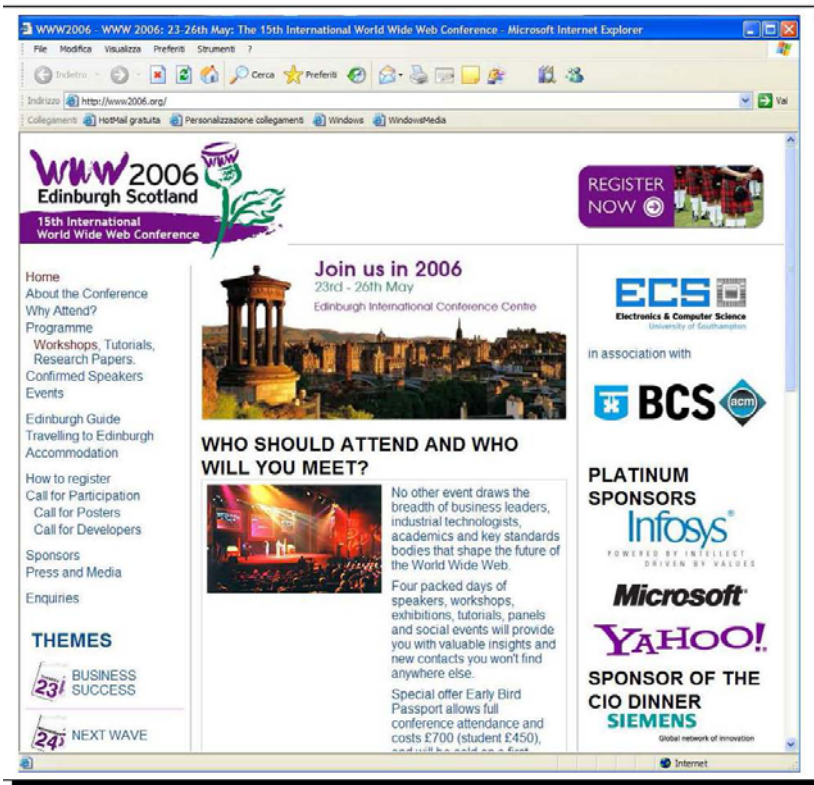


Fig. 4 The CSS-Restyling edge service. The original WWW 2006 Website (top) and the same page with the application of the CSS-Restyling service by PAN (bottom).

4.2 Link-based Services

This section discusses 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*

The RemoveLink edge service 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 not 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 *TheLinkAccessKey service*

The W3C Guideline 9 *Design for device-independence* [46], in section Keyboard access, and the section 5.2 *Navigation and Links* (5.2.6 Access Keys) in [25] state the need to use access keys HTML elements to allow users with disabilities to browse the Web.

Content developers should always ensure that users may interact with a page with input devices other than a mouse. Since this limitation is not often considered, PAN provides 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 also be useful to improve Web navigation on devices with limited display capabilities.

4.2.3 The LinkRelationship 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 [25], 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 different content for browsers that support Braille rendering.

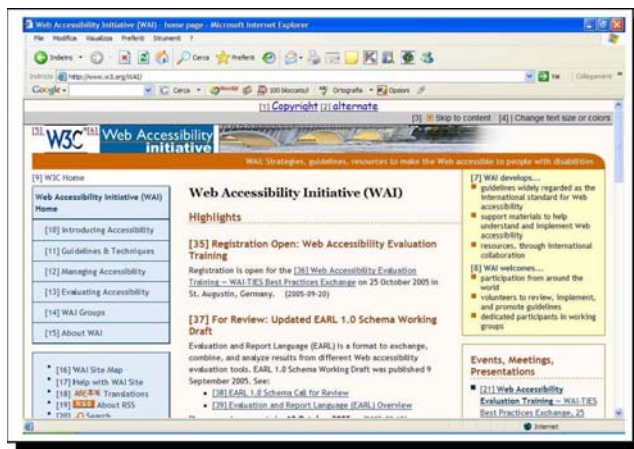


Fig. 5 The LinkAccessKey and Linkrelationship edge services.

The LinkRelationship 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 Web navigation through 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 a site map, the *Glossary*, that refers to a glossary of terms contained in the current page, the *Copyright*, that refers to a copyright of the current document, etc.

4.2.4 The DeleteTarget 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* [46], 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 to the original window, the screen reader starts again from the beginning of the page to speech the corresponding content.

The main goal of the DeleteTarget 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 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 or bottom of the page (as specified by the user during the configuration and customization of the service).

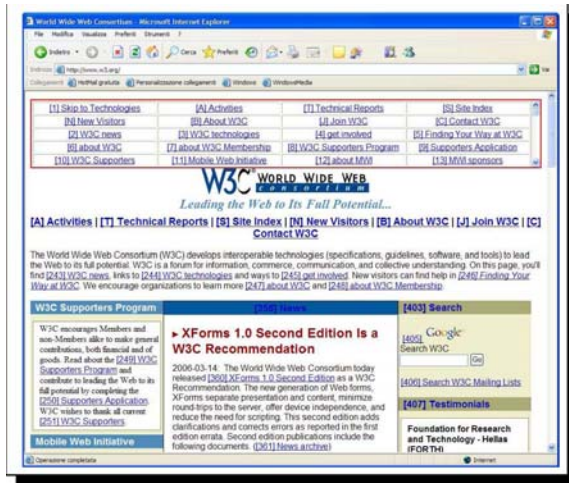


Fig. 6 LinkLinearizing and AccessKey services.

4.3 Image-based services

Web images represent a serious problem for blind users, since they can not be read by screen readers. Moreover, particular images formats or compositions represent a barrier also for cognitively disabled people, as discussed in subsection 4.3.2. At the end of this section, in subsection 4.5.2, we will also describe a filter for colorblind users that introduces also a heavy-load image filtering technique.

4.3.1 The ImageRemoval service

The ImageRemoval edge service removes any image embedded in a Web page by replacing it with a link, as suggested in section 5.4 *Page Definition* (5.4.5 Non Text Items) [25].

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

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

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, causing important problems to users that suffer of dyslexia or of general attention deficit.



Fig. 7 RemoveImages and LinkAccesskey services.

By following the Guideline 7: *Ensure user control of time-sensitive content changes* [46], the 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 [37] library, that is an object-oriented Perl interface to ImageMagick [33], a free software that allows to create, edit, and compose images with different formats.

4.4 Filtering of other objects

4.4.1 AnnojanceFilter service

As defined in the Guideline 10: *Use interim solutions* [46] 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 service is to eliminate particularly annoying elements during the Web navigation. 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.

This is a very critical service, because animated objects, banners or Popup browser windows could contain important information about the Webpage (not only as images). The customization of this service through SISI is, therefore, crucial for avoiding errors and subsequent *distorted* Web pages during their displaying on clients.

4.4.2 The HTMLClean 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* [24]), non standard META elements, and HTML comments, as well as 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.* [46]).

4.5 Support for Colorblind Users

This section describes separately an edge-service based solution to the problem of colorblind users [32]. The *Colorblind filter* service implements an algorithm that modifies any color in Web pages, by increasing contrast and lightness, in order to make them accessible for users with such a disability. The filter acts on text (see subsection 4.1.2) as well as on images, and is the most complex of the filters presented in this paper. This Section first introduces *colorblindness*, and then describes the developed algorithm and its implementation on top of PAN.

4.5.1 Colorblindness

The human eye perceives an electromagnetic radiation only in a particular segment of the enormous range of its frequencies, called “visible light spectrum”, that ranges approximately from 380 to 780 nm ($1 \text{ nm} = 10^{-9} \text{ m}$). Each individual wavelength within the spectrum of visible light wavelengths is representative of a particular color.

Moreover, the human eye contains three types of color sensitive receptors, called cones, which are responsible for the vision of a different portion of the spectrum, that is, *Long* (reddish), *Middle* (greenish), or *Short* wavelengths (bluish) [18]. A color deficiency is present when one or more of the three cones light sensitive pigments have a spectral sensitivity similar to the sensitivity of another cone type or when they are entirely missing.

The most common forms of color deficient vision, called protanopic and deuteranopic, are characterized by difficulties in distinguish between green and red. This type of deficiency, also known as color blindness, is mostly a genetic condition, and it is much more common in men than in women (roughly 8% of the male population against only 2% of female population).

It must be emphasized that red and green colors are absolutely unknown for color blind users. Therefore, all the analyzed approaches, both in computer graphics and mathematics, for preserving the reproduction of colors, have mainly addressed the problem of how to avoid, as much as possible, any “loss of information” due to a genetic condition.

Related research has been mainly conducted in the field of computer graphics and mathematics (see, as an example [9]). As a consequence, results are typically not suitable to be used in on-the-fly filtering, as in the case of WWW navigation by a color blind user [45]. Among previous efforts, the interesting work by Dougherty and Wade [15] also proposed a mechanism (on a website) for digital images correction. More recent work [19] proposed algorithms that transform color to gray scale by preserving image details. In [41] an extension

of this algorithm is shown to allow a re-coloring of images for color-deficient viewers. While being an interesting extension, their technique is far from having real-time performance [40], that is, instead, the main motivation that has guided the present work in this field.

4.5.2 Colorblind Filter service

The Colorblind Filter Service (CBFS) translates on-the-fly any HTTP response in order to obtain a page that is more accessible to users with color blindness. More details on the efficiency of the solution are provided in section 5.2.

The adopted approach is to tackle accessibility for dichromatic users by modifying on-the-fly both HTML and the images embedded. In fact, the main goal of the CBFS is to modify background and foreground colors in HTML pages, and to re-color embedded images, including animated GIF images, in order to make more recognizable the red/green contrast for dichromatic users.

The service parses on-the-fly each Web page and, for each HTML tag, analyzes the corresponding attributes to modify background and foreground text and images, if some correction is required, by also taking into account inline Internal and External Cascading Style Sheets.

The following attributes of HTML pages are considered: *color*, *bgcolor*, *background*, *img*, *text*, *link*, *alink*, *vlink* and style attributes that specify images/backgrounds/colors, following the rules suggested by “Techniques For Accessibility Evaluation And Repair Tools” of W3C (<http://www.w3.org/TR/AERT>).

The CBFS uses the HSL representation of colors, by specifying them in terms of Hue (H), Lightness (L) and Saturation (S). The Hue value describes the individual colors (the portion of the spectrum that contains the color), the Saturation value represents the intensity of a specific color, and the Lightness value determines the perceived intensity (light or dark color). The HSL color representation model was chosen instead of the RGB one due to its ability to manipulate

lightness, that represents the most important discrimination element for dichromatic people.

The goal of the algorithm is to reduce all stimuli along the so-called “confusion lines”, that are the lines of intersection between the plane of not visibility for dichromatic people and the 3D color space of normal users [9]. In fact, by changing proportionally hue, saturation and lightness values, it is possible that all stimuli fall in two different half-planes (of the 2D representation) by making them distinguishable both for normal and dichromatic users.

The algorithm, shown below, has complexity of $\Theta(n \cdot m)$ when the input is an image of size $n \times m$.

Algorithm 1 Colorblind Filter Service

```

1: RedPixel  $\leftarrow$  0
2: GreenPixels  $\leftarrow$  0
3: for all pixel do
4:   if ( $R > G + 45$  and  $R > B + 45$ ) then
5:     RedPixel ++
6:   else if ( $G > R + 45$  and  $G > B + 45$ ) then
7:     GreenPixel ++
8:   end if
9: end for
10: if ( $GreenPixel > RedPixel$ ) then
11:   Color  $\leftarrow$  Green
12: else
13:   Color  $\leftarrow$  Red
14: end if
15: for all pixels with RED/GREEN components do
16:   ColorPixel = Convert { RGB to HSL }
17:   if ColorPixel is close to Color then
18:     Hue  $\leftarrow$  Hue - 30%
19:     Saturation  $\leftarrow$  Saturation - 10%
20:     Lighthness  $\leftarrow$  Lighthness + 25%
21:   else
22:     Saturation  $\leftarrow$  Saturation + 10%
23:     Lighthness  $\leftarrow$  Lighthness - 10%
24:   end if
25:   ColorPixel = Convert { HSL to RGB }
26: end for

```



Fig. 8 Original image (left) and as perceived³ (right) by a dichromatic user.

An important characteristic of this algorithm is that it is customizable, that is, each user can choose the proportion by which hue, saturation and lightness are changed⁶. The personalization of edge-services offered by SISI allows easy personal tuning to meet different color perception deficiencies.



Fig. 9 Modified image (left) and as perceived³ by a dichromatic user (right).

Fig. 8 shows an example of application of the algorithm to a well-known Paul Gauguin's painting⁷, on left as perceived by trichromatic and on the right as seen by dichromatic users. In the image on the right the details of the road in the meadow are lost, but by properly modifying colors according to the developed algorithm, colorblind viewers are able to distinguish the edge between the road and the meadow, although not being able to perceive exactly the right colors used by Gauguin. Fig. 9 shows the image as modified by the algorithm, as perceived by a trichromatic user (left), and as perceived by a dichromatic user (right⁸). The details are now evident also in the image on the right-hand side.

Besides the experiments with the Vischeck simulator [15], the results of the CBFS have been further validated through a test on several colorblind users that recognized a *significant to very significant*

⁶ The values used in the algorithm have been qualitatively tested with several color blind users and with the Vischeck simulator described in [15].

⁷ Landscape, 1890. Oil on canvas. The National Gallery of Art, Washington DC, USA.

⁸ Images on the right-hand side of Figs. 8 and 9 are obtained with the Vischeck simulator described in [15].

improvement in the perception of images that would present problems to dichromatic vision.

Finally, to measure the impact of the CBFS on the responsiveness of user navigation, a preliminary performance evaluation has performed, which is described in section 5.2.

4.6 Summary of services features

As reported in this section, PAN provides a repertoire of edge services that are tailored to address the needs of different kinds of disabilities. Table 1 summarises how each type of disability can be addressed by PAN's services. Finally, it should be emphasized that the remote, user-friendly configuration of the services applied, per-user and per-profile, offers an effective personalization of the tool. The efficiency of the services is described in the next section.

Table 1: Suggested services implemented on PAN for main kinds of disabilities.

Disability	Services	Rationale
Low vision	Gif-Deanimate AnnojanaceFilter LinkAccesskey LinkRelationship CSS-Restyling	reduces confusion by eliminating animated GIFs and advertisement, banners and pop-ups; facilitates finding links by number and finding REL links easily; restyles fonts and colors.
Color blindness	Filter ColorBlind Textcolor Restyling	Changes images and text colors so that they are easily recognizable by colorblind users.
Blindness	ImageRemoval DeleteTarget HTMLClean AnnojanaceFilter RemoveLinks	removes images so that screen-readers are more effective; disables new windows from opening when clicking on a link; improves readability for the screen readers; removes advertisements, banners and pop-ups; removes all the links in the HTML page.
Upper-limbs motor disabilities	LinkAccesskey LinkLinearizing DeleteTarget	allows interaction without mouse based interfaces; makes more easily accessible all the links in the same place; disables new windows from opening when clicking on a link.

5 Testing effectiveness and efficiency

Effectiveness and efficiency of the services that are provided by Personalizable Accessible Navigation are, of course, crucial: services must be useful, but they also need to be provided with a reasonable degree of effort and resources by Internet providers, government, associations, etc..

It should be emphasized that this research stems from 2 years long experience in a “Leonardo Da Vinci” European-funded project (ended November 2005) named WHITE (Web for Handicap Integrated Training Environment) [38]. WHITE’s goal was to improve the educational and training systems to include people with physical disabilities (in particular, visually impaired users), and the acquired experiences were very useful in the realization of PAN.

5.1 Effectiveness

The requirements and design phase of the WHITE project were supported by different volunteers of different blind associations from Italy and Romania. The project ended with the presentation of an early prototype⁹ of (some of) the techniques reported in this paper as applied to a sound engineering course.

In particular, following the categorization provided in Section 4, during the WHITE project blind users were involved during the design and realization of Text-based services (Section 4.1), Link-based services (Section 4.2) and services for filtering other objects (Section 4.4) that had the main goal of making more acceptable and efficacious the work performed by the screen reader for a blind user. In particular, one of the most popular screen readers for Windows (JAWS for Windows) was used in the tests.

⁹ It must be noticed that the prototype was an ad-hoc implementation of the filters and that it did not allow (e.g.) any configuration, profiling, and personalization.

5.1.1 User study

A in-house user study was conducted in order to test the efficacy of these services for people with low-vision disabilities and people with colorblind deficiency, involving a group of eight persons composed of both students and researchers of the University of Salerno and people from the *Accademia musicale di Caserta*¹⁰, aged between 25 and 45 years. They were either color blind or low vision users.

The users were asked to navigate through a set of pre-defined Web pages using two different browsers, one connected to the developed edge server proxy (configured to help the specific disability of each user) and another with a direct connection to the Internet. The users did not know which one between them was the browser with enhanced capabilities, that is the browser whose pages were intercepted by PAN and modified if some correction process was required. The objective of the experiment was mainly to obtain qualitative feedback about the ability of PAN to improve Web navigation for everyone.

In particular, the following questions were addressed:

1. Is the user able to find visualization improvements in the shown Web page, and such improvements are simple to understand?
2. Is the user able to get additional information when comparing the two Web pages displayed by the two different browsers?
3. Does the user lose information during the visualization of the Web pages via proxy?

All the questions are answered using the following scale: *never*, *occasional*, *sometimes*, *often*, *always*. The results are summarized in Fig. 10.

¹⁰ It was one of the partners of the WHITE project.

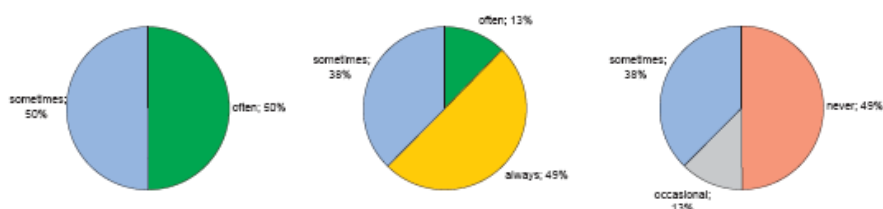


Fig. 10 Left(1) to right(3), the results of the three questions to evaluate the feedback of the Web navigation.

The subjects, after 20-30 minutes of navigation through the pre-defined Web pages, were asked to provide some comments about the usability of the provided tool. The results confirm that PAN activation and service configuration are simple and intuitive.

Users with color blindness are able to see all information they would not see otherwise, and the improvements are especially significant for graphics, buttons, tables and images with text. In particular, the services were found particularly helpful by professionals (e.g., scientist) that work heavily with colored charts with tiny colored lines that intersect each other or that are very close. Another example were the provided filters were considerably helpful was in showing graphical representations of cells and molecules to chemists.

More controversial was the usage of colorblind filters on pictures, art paintings and geographic maps. In fact, the comments were that the main advantage of the transformations directed at showing information in recognizable colors does affect (often, negatively) the overall vision of the image and in many cases, it is considered acceptable by the users to miss red/green details but retain the “originality” of the color distribution of the image. In those cases, particularly well accepted was the opportunity to easily configure and turn the filter on or off in different moments of the navigation, possibly when the user feels in doubt about the perception of the image.

Users with low-vision had the opinion that, in general, the services modifying the style of the page (i.e., CSS-Restyling) were moderately useful, since, in general, could support and improve existing tools offered by the visual interface to the operating system (such as

magnifying glass) but were not critical to the accessibility of the pages.

5.1.2 Automated test

The efficacy of all the services (except ColorBlind) was also tested by using an automated tester, namely 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. Bobby tests Web pages by using a set of accessibility checks defined in WAI [44]. The following 6 categories of Web pages were selected: Airlines, Government, Information, Museums, Research and (Computer Science) Technology. For each category, 5 significant sites (homepage) were chosen. The URL for each category are reported in Appendix A.

Bobby's results on the original Web pages were compared with Bobby's results obtained on the same Web pages adapted by PAN. The adapted pages were saved on a local server and accessed with Bobby in a subsequent step. The following Pan's services were tested: ImageRemoval, the CSS-Restyling, TextColor-Restyling, LinkAccessKey, DeleteTarget and AnnojanceFilter. The effectiveness of the Colorblind filter has been tested differently, as specified in Section 4.5.2, by applying additional filters that simulate the dichromatic vision and comparing the effect on the images altered by Colorblind.

Bobby Accessibility Tester checks if the analyzed Web pages are fully compliant with all of the automatic and manual checkpoints of the W3C Web Content Accessibility Guidelines. In particular, Bobby reports errors using the W3C Guidelines priority list, but they can be assembled in four categories: "Image Evaluation", "Insufficient Color Contrast", "Animated Objects", and "Pop-ups and new browser windows".

Figures 11-14 show the most significant results of the tests on 4 of the 6 site categories used as a sample. Each figure is composed by two graphs, indicating the number of errors and warnings resulting from the Bobby tests on the categories ‘Image Evaluation’, ‘Insufficient Color Contrast’, ‘Animated Objects’, ‘Pop-ups and new browser windows’. In particular, the topmost graph shows Bobby’s results on the original Web pages, while the bottom graph shows Bobby’s results on the adapted pages by PAN. For each graph, the Web page tested is reported on the X-axis while on the Y-axis the number of errors for each category is shown, where the lower the number, the more accessible is the page.

As shown in Figures 11-14, a substantial improvement on categories ‘Governments’, ‘Information’, ‘Research’ and ‘Technology’ is obtained with the use of PAN.

Figure 15 shows the trend of errors detected by Bobby, before and after application of PAN, on all the Web pages tested. For each page (for readability, the X-axis shows the number of the page in the list provided at the beginning of this section) the total number of errors detected in all the four error categories is reported.

These results show that accessibility is definitely improved applying PAN; for each URL, accessibility improves between 50% and 90%, in some cases improvements are over 90%. It should be emphasized that, except in one case (CNN site, URL no. 11), the total number of errors is always diminished by at least 50%.

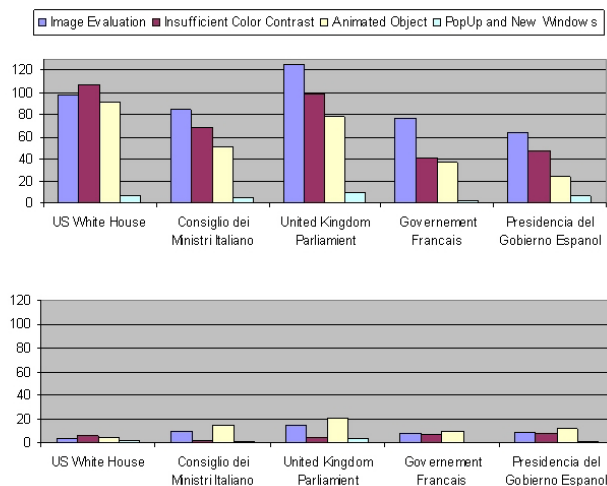


Fig. 11 Bobby results of Web pages of the ‘Government’ category.

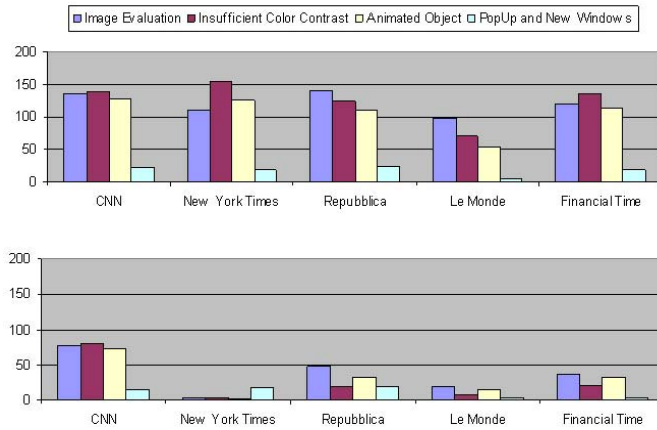


Fig. 12 Bobby results of web pages of the "Information" category.

5.2 Efficiency

PAN is based on SISI, whose efficiency leverages on highly optimized libraries for parsing and changing HTML, the Apache Web server and mod Perl. As a matter of fact, as shown in [12], SISI is particularly efficient in providing services that modify the structure of an HTML page. Obviously, different results can be obtained when the service is required to parse and change images on-the-fly. As a matter of fact, some preliminary testing of the Colorblind filter [32] using image filters showed that, while the overall architecture of PAN was able to keep up the pace with the workload, the libraries for dealing with images represented a bottleneck.

This section therefore focuses on the efficiency of the Colorblind filter service, that represents a rather heavy-weight service to be provided to a number of users. In fact, while representing a killer application for accessibility, its cost in terms of CPU is probably too much demanding for real deployment. Therefore, this section discusses the cost of the filter as is implemented in PAN, as well as potential improvements which can be obtained if additional resources are used (e.g., graphical cards that are now available at low cost on ordinary PCs).

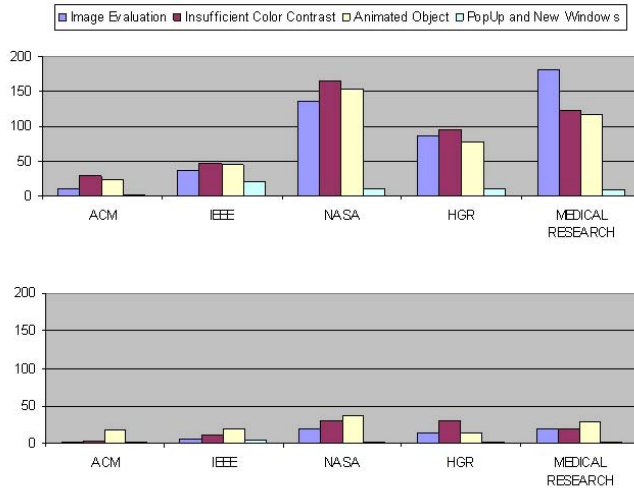


Fig. 13 Bobby results of Web pages of the “Research” category.

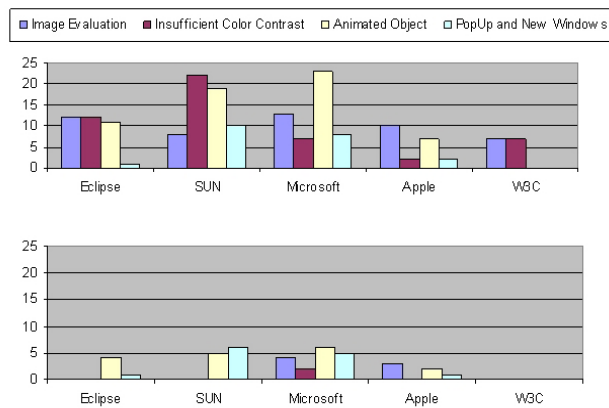


Fig. 14 Bobby results of web pages of the “Technology” category.

5.2.1 Graphical hardware

Graphics Processors Unit (GPUs) are stream processor that are highly optimized for computer games and, more generally, for real time rendering. GPUs have high memory bandwidth and more floating point units as compared to the CPUs. For example, the current top of the line GPU such as the NVIDIA 7800 GTX, available at a price of \$500, has a peak performance of 313 GFLOPS and a memory bandwidth of 56 GB/s, as compared to 25.6 GFPLOS and 6.4 GB/s, respectively, for a high-end dual core Pentium IV processor.

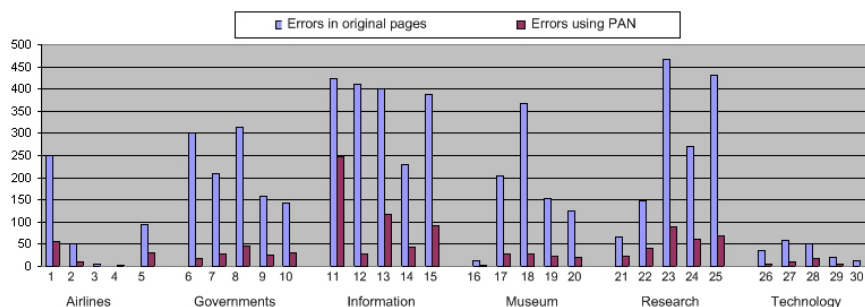


Fig. 15 Graph of the improvements obtained on sites of all categories by using PAN, as accessed through Bobby.

Recently, researchers are using this computational power as general purpose stream processing for applications beyond graphics. The term GPGPU has been coined to indicate General Purpose computation on GPUs. There are numerous examples where the GPUs have been used to perform general stream computations, for instance, in scientific computations, matrix multiplication, FFT computation, sparse linear systems, etc., as reported in [21]. In particular, Govindaraju et. all in [20] show that a classic problem like sorting ported on the GPU is significantly faster than optimized CPU-based algorithms. These results suggest that the GPU used as a co-processor can significantly improve performance of several algorithms.

The main features that makes the graphics hardware attractive are the programmability, the spatial parallelism it provide (for example, each pixel on the screen can be viewed as a stream processor) and the widespread availability on ordinary PCs.

GPU, compared to CPU, exploits a specialized computational model that can make problematic its usage in general purpose computations instead of 3D graphics rendering for which it has been designed for. Among the most significant limitations, the limited size of a program and the absence of quite useful programming paradigms such as, for example, pointers can be mentioned.

Writing general purpose algorithms on highly specialized hardware like GPU is not an easy task. The design of the programs is not intuitive at all; for example, it is necessary to map computational concepts like an input or an output to graphical computational

resources like texture and frame buffer respectively. This mapping is not always obvious, and it often has to face strict hardware limits. There are several ways of programming GPU, ranging from some ANSI C inspired language (Cg, HLSL, OpenGL Shading Language) to recent specialized language that exploit the restricted parallel programming model of the GPU (BrookGPU, Accellerator, Sh). The Cg (C for graphics) language and OpenGL graphics library was used in the work described in the next section. Cg has been designed by nVIDIA and supports different profiles to best adhere to the various hardware available from different manufacturer [16].

5.2.2 The Colorblind filter on GPU

This section describes the implementation of Colorblind filter on GPU, within the same framework previously described, i.e., the intermediary uses the local (top the proxy) GPU in order to efficiently perform the filtering.

Most of the computing power of GPUs is accessible from one particular stage of the graphics pipeline, pixel shading. Image color processing fits well into the pixel shading. In fact, pixel shaders have a particularly restricted parallel programming model in which each pixel of a texture is computed independently of every other pixel of the texture. Then, the general idea is to store images as texture map and to exploit fragment shader programs to implement the colorblind filter.

This service is built on top of the OpenGL library which is responsible to transfer each incoming image from the proxy into the graphics memory as texture in order to perform an off-screen computation (i.e., rendering). Additionally, the recent PCI-Express bus which connect CPU and GPU permits high amount of data to be transferred at up to 4GB/s in both directions. The great capability of this new bus permits to promote GPU much more as a general purpose co-processor. Furthermore, both GPU and CPU can work simultaneously, fully exploiting the system with better performances.

The Colorblind filter is written using the high level language for graphics Cg, which exposes mathematical functions implemented in the hardware, thus making the computation of complex mathematical expressions more efficient. Such mathematical functions can compute 4 components simultaneously (traditionally red, green, blue, alpha) fully exploiting SIMD instructions.

The main steps to rendering an image on the GPU are:

1. Compile and set the pixel shader program with colorblind filter to compute the value of the output pixel
2. For each incoming image from the proxy
 - (a) Load and transfer the image to video memory as texture
 - (b) Draw an image-sized quad in order to create a stream of fragments (i.e., pixels) upon which run the filter
 - (c) Pass the colors output values to system memory

This approach is general and permits easily to plug a custom color or image filter (black and white, image downgrade, dithering for Braille display, etc...) with minimum alteration, thus creating an extensible service. In order to deploy the GPU Colorblind filter, written in C++, on top of the SISI framework, written in Perl, the SWIG tool [7] was used. SWIG is a free tool designed to integrate C/C++ code with a variety of scripting languages including Perl, Python, and Tcl. The integration of Perl and C/C++ code allows programmers to easily write wrappers that convert data between the Perl interpreter and the compiled c code. SWIG has been used for the design of the low-level software interface to graphics hardware. In such a way, this interface permits to use the provided Colorblind filter on SISI within a Perl script as a standard package.

5.2.3 A comparison of Colorblind Filter and its GPU implementation

To evaluate the efficiency of the implemented Colorblind filters, this section reports the results of some tests whose goal was to measure the time required by the filters to apply transformations on Web

resources. Both implementations (Cg and Perl implementations) of the Colorblind filter apply transformations only on image resources whose formats are GIF, animated GIF and JPG, since the computationally heaviest part of the filter is on images. The tests were performed in a local setting by comparing the execution of the GPU program with the corresponding Perl program, since the framework of PAN in both cases is the same. It was anticipated that the improvement obtained would be of an order of magnitude in time, quite precious for offering the service to a number of users.

The experiments used two workload models. The first, called *Neighborhood*, follows the study described in [11], while the second, called *HeavyWorkload*, aims at stressing the services since the images to be processed had a size larger than 100 KB.

In the *LightWorkload* model, the image sizes have been defined as suggested in [11]. The 80% of GIF images are smaller than 6 KB, the 45% of them are bullets, icons, lines, banners, the 15% are animated and, finally, the 10% are *standard images*, i.e, images that do not belong to any of the categories specified above, and larger than 6KB. On average, JPEG images are larger than GIF images, the 40% of the JPEG images are larger than 6KB, the 30% of small JPEG images appear to be icons, bullets, etc., while the 35% are standard images larger than 6KB.

The cost of the color image processing was measured in terms of the response time of the requests issued as a sequential stream on a PC with a 2.5 GHZ Opteron CPU with NVIDIA GeForce 7900 GTX card, running Red Hat ES Linux. In particular, the cumulative distribution function of the response time was used instead to consider mean values, since the latter can mask important details about the behaviour of slow and fast responses, typical of systems with high variability, during experiments.

From the plot in Fig. 16 it can be seen that the Perl implementation of the Colorblind Filter is able to achieve a 90-percentile of the response time in a range between 220-230 ms, while the 90-percentile for the Cg implementation is between 4-5 ms (See Table 2).

From table 3, it can be seen that the Cg Colorblind filter is able to achieve a 90-percentile of about 160 ms, while the Perl implementation is much slower, with a 90-percentile between 16000-17000ms.

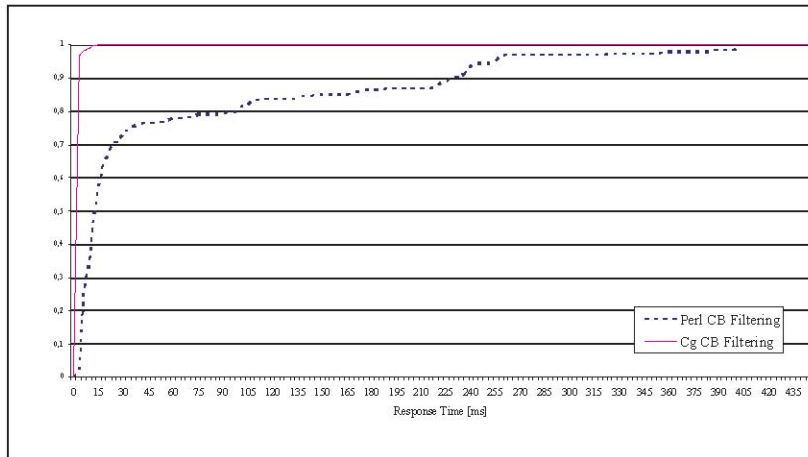


Fig. 16 Cumulative Distribution function for the Colorblind (CB) LightWorkload.

Table 2 Response Time comparison of the Cg and Perl implementations based on 50 and 90-percentiles with the LightWorkload.

	Response Time [ms]	
	Median	90-percentile
Perl CB Filtering	14	228
Cg CB Filtering	3	4,7

Table 3 Response Time comparison of the Cg and Perl implementations based on 50 and 90-percentiles with the HeavyWorkload.

	Response Time [ms]	
	Median	90-percentile
Perl CB Filtering	3200	16200
Cg CB Filtering	30	160

6 Conclusions

Personalizable Accessible Navigation represents an effective, efficient, simple and personalizable tool to provide easier access to Web resources to people with disabilities.

Its nature (being an intermediary) and the characteristics of its services make it suitable for adapting access to Web resources in different contexts, such as fixed/mobile terminals. As intermediary, Personalizable Accessible Navigation can flexibly act as a traditional

proxy, as a HTTP surrogate (placed on the server-side) or as a HTTP delegate (placed on the client side).

Being based on SISI, Personalizable Accessible Navigation can be used also for HTTPS access. A main motivation is that several important services (e-government, online banking, etc.) are only available through HTTPS access. Of course, the accessibility to these services via proxy would severely impact on the privacy of the communication, that is the main reason for the secure protocol itself. In fact, an intermediary could have access to sensitive information just because the transcoding or other services are needed. A first solution could be using Personalizable Accessible Navigation as a delegate proxy (i.e., on client machine), in such a way that HTTPS connection's privacy is preserved outside, in the network. Another more costly option could be to offer corporate surrogate proxy (i.e., in front of the Web server) so that secure and accessible services can be provided to the users.

Moreover, by implementing easy configuration and multiple profiles, PAN enhances the usability of the accessibility services in many different contexts.

The effectiveness and efficiency of PAN have been experimentally validated, which makes PAN a realistic platform to deploy services for the navigation of people with disabilities.

As a final comment, the intrinsic technological limitations encountered by a solution based, as many others, on transcoding, should be noted. Indeed, transcoding necessitates, nowadays, an increasingly complex analysis of HTML pages, as dynamic content (such as DHTML and AJAX applications) is spreading on the World Wide Web. In fact, scripts that alter the structure of the page client-side offer a challenge to transcoding techniques, which can be tackled only by approximate solutions, since computability theory shows that this kind of problems are undecidable, being easily reducible to the Turing machine halting problem. In general, transcoding applications are doomed not to provide guaranteed performances if no other

information on the semantics of the page and the modifications is available.

Acknowledgments

The authors gratefully acknowledge the support and the interesting discussions within the ISIS Lab. 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. We also thank all the WHITE’s partners, and, in particular, the Accademia Musicale di Caserta and Emilio Di Donato for his help during the testing. We thank all the users that volunteered for testing our Personalizable Accessible Navigation. Finally we thank the anonymous reviewers for insightful comments and suggestions which considerably helped us in improving the paper.

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. Text-ToSpeech: an heavy-weight Edge computing Service. In Poster Proc. of 12th International World Wide Web Conference. ACM Press, May 2003.
- 3 M. Barra, P. Maglio, A. Negro, and V. Scarano. GAS: Group Adaptive System. In Proc. of the International Conference on Adaptive Hypermedia and Adaptive Web-based Systems(AH 2002),pages 47–57. ACM Press, May 2002.
- 4 R. Barrett and P. Maglio. “Adaptive Communities and Web Places”. In Proceedings of the 2th Workshop on Adaptive Hypertext and Hypermedia, HYPERTEXT 98, 1998.
- 5 R. Barrett and P. P. Maglio. Intermediaries: An approach to manipulating information streams. IBM Systems Journal, 38(4):629–641, 1999.
- 6 R. Barrett and P. P. Maglio. WebPlaces: Adding people to the Web. In Proceedings of 8th International World Wide Web Conference, Toronto (Canada), 1999. ACM Press.
- 7 D. Beazley, D. Fletcher, and D. Dumont. Perl extension building with swig, 1998.
- 8 BBC Education Text to Speech Internet Enhancer, 1999.
<http://www.bbc.co.uk/education/betsie/>.
- 9 H. Brettel, F. Vienot, and J.D.Mollon. Computerized simulation of color appearance for dichromats. Journal of the Optical Society of America, 14(10):2647–2655, October 1997.

- 10 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.
- 11 S.Chandra,A.Gehani,C.S.Ellis,andA.Vahdat.Transcoding characteristics ofwebimages.InM.Kienzle andW.chiFeng,editors, MultimediaComputing andNetworking(MMCN'01),volume 4312,pages 135–149,SanJose,CA,Jan. 2001. SPIE - The International Society of Optical Engineering.
- 12 M. Colajanni, R. Grieco, D. Malandrino, F. Mazzoni, and V. Scarano. A scalable framework for the support of advanced edge services. In Proc. of 2005 International Conference on High Performance Computing and Communications(HPCC 2005), September 2005. (Journal version is accepted for publication by WWW Journal).
- 13 CSS Techniques for Web Content Accessibility Guidelines 1.0, November 2000. <http://www.w3.org/TR/WCAG10-CSS-TECHS/>.
- 14 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>.
- 15 R. Dougherty and A. Wade. Vischeck: Simulation of colorblind vision and images correction for colorblind viewers. <http://www.vischeck.com>.
- 16 R. Fernando and M.J. Kilgard. The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics. Addison-Wesley Longman Publishing Co., Inc., 2003.
- 17 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.
- 18 R. C. Gonzalez and R. E. Woods. Digital Image Processing, volume 2nd Edition. Prentice Hall, 2002.
- 19 A. A. Gooch, S. C. Olsen, J. Tumblin, and B. Gooch. Color2gray: Salient-preserving color removal. ACM Transaction On Graphic, 24(3), 2005.
- 20 N. K. Govindaraju, J. Gray, R. Kumar, and D. Manocha. Gputerasort: High performance graphics coprocessor sorting for large database management. In ACM SIGMOD International Conference on Management of Data, Chicago, United States, June 2006.
- 21 General-purpose computation using graphics hardware, August 2006. <http://www.gpgpu.org/>.
- 22 R. Grieco, D. Malandrino, F. Mazzoni, and D. Riboni. Context-aware Provision of Advanced Internet Services. In Proc. of the 4th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2006), Pisa, Italy, March 2006.
- 23 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.
- 24 W3C Note–Techniques for Web Content Accessibility Guidelines 1.0, November 2000. <http://www.w3.org/TR/WCAG10-TECHS/>.
- 25 W3C Working Draft – Mobile Web Best Practices 1.0, January 2006. <http://www.w3.org/TR/2006/WD-mobile-bp-20060113/>.
- 26 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.
- 27 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.

- 28 R. Han, P. Bhagwat, R. Lataire, 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.
- 29 V. L. Hanson, J. P. Brezin, S. Crayne, S. Keates, R. Kjeldsen, J. T. Richards, A. Swart, and S. Trewin. Improving Web accessibility through an enhanced open-source browser. *IBM System Journal*, 44(3):573–588, 2005.
- 30 D. Hermsdorf. WebAdapter: A Prototype of a WWW Browser with new Special Needs Adaptations, 1998.
- 31 M. Hori, G. Kondoh, K. Ono, S. Hirose, and S. Singhal. Annotation-Based Web Content Transcoding. In *Proceedings of the 9th International World Wide Web Conference, Amsterdam (The Netherlands), 2000*. ACM Press.
- 32 G. Iaccarino, D. Malandrino, M. D. Percio, and V. Scarano. Efficient edge-services for colorblind users. In *WWW '06: Poster Proceedings of the 15th International Conference on World Wide Web*, pages 919–920. ACM Press, 2006.
- 33 ImageMagick 6.2.6, 2006. <http://www.imagemagick.org/script/index.php>.
- 34 Libwww-perl. <http://sourceforge.net/projects/libwww-perl/>.
- 35 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.
- 36 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.
- 37 PerlMagick 6.22, 2005. <http://www.imagemagick.org/script/perlmagick.php>.
- 38 White project (Leonardo Da Vinci European Union Programme, 2004-2005). <http://www.progettowhite.net/applications/pag HTML/project.asp?Testo=P>.
- 39 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.
- 40 K. Rasche. Detail Preserving Color Transformation. PhD thesis, Clemson University, 2005.
- 41 K. Rashe, R. Geist, and J. Westal. Detail preserving reproduction of color images for monochromats and dichromats. *IEEE Computer Graphics & Applications*., pages 22–30, May-June 2005.
- 42 WAI Evaluation and Repair (ER) group. “Tablin: an HTML Table linearizer”. <http://www.w3.org/WAI/References/Tablin/>.
- 43 Textualise; codix.net ltd., 2003. <http://aquinas.venus.co.uk/>.
- 44 Web Accessibility Initiative (WAI), 2005. <http://www.w3.org/WAI/>.
- 45 K. Wakita and K. Shimamura. Smartcolor: disambiguation framework for the colorblind. In *Proceedings Assets '05 (7th International ACM SIGACCESS conference on Computers and accessibility)*, pages 158–165 .ACM Press, 2005.
- 46 Web Content Accessibility Guidelines 1.0, W3C Recommendation, May 1999. <http://www.w3.org/TR/WCAG10/>.

Appendix

A URLs used for Bobby's testing

- Airlines:
 - 1 - America Airlines – <http://www.aa.com/>
 - 2 - Alitalia – <http://www.alitalia.it/>
 - 3 - Airfrance – <http://www.airfrance.fr/>
 - 4 - British airways – <http://www.britishairways.com/>
 - 5 - Korean Air – <http://www.koreanair.com/>
- Governments
 - 6 - USA: White House – <http://www.whitehouse.gov/>
 - 7 - Italy: Prime Minister – <http://www.governo.it/>
 - 8 - Spain: Presidencia del Gobierno – <http://www.la-moncloa.es/>
 - 9- France: Portail du Gouvernement – <http://www.premier-ministre.gouv.fr/>
 - 10 - UK: The United Kingdom Parliament – <http://www.parliament.uk/>
- Information
 - 11 - CNN – <http://www.cnn.com/>
 - 12 - The New York Times – <http://www.nytimes.com/>
 - 13 – LaRepubblica (Italy) – <http://www.repubblica.it/>
 - 14 - Financial Times Europe – <http://www.ft.com/home/europe>
 - 15 – LeMonde (France) – <http://www.lemonde.fr/>
- Museums
 - 16 - Metropolitan Museum of Art (New York) – <http://www.metmuseum.org/>
 - 17 – Musei Vaticani (Vatican City) – http://www.vatican.va/phome_en.htm
 - 18 - Egyptian Museum (Cairo) – <http://www.egyptianmuseum.gov.eg/news.asp>
 - 19 - Grand Canyon National Park – <http://www.grand.canyon.national-park.com/>
 - 20 - Louvre – <http://www.louvre.fr/>
- Research
 - 21 - IEEE – <http://www.ieee.org/portal/site>
 - 22 - ACM – <http://www.acm.org/>
 - 23 - NASA – <http://www.nasa.gov/home/>
 - 24 - Medical Research Council – <http://www.mrc.ac.uk/>
 - 25 - Human Genome Research – <http://www.genome.gov/>
- Technology (Computer Science)
 - 26 - IEEE – <http://www.apple.com/>
 - 27 - Eclipse – <http://www.eclipse.org/>
 - 28 - SUN – <http://www.sun.com/>
 - 29 - Microsoft – <http://www.microsoft.com/>
 - 30 - W3C – <http://www.w3.org/>