# Enabling Blind Web Users to Create Accessible Web Content

Jeffrey P. Bigham
University of Washington

6 April 2007

**Abstract**

Browsing the web is inefficient for blind web users because of persistent accessibility problems. While previous efforts have tried to improve web accessibility through developer awareness, technological improvements, and legislation, these attempts have failed to adequately address accessibility concerns. Blind web users have proven quite adept at overcoming the shortcomings of the web and existing tools, but have been only marginally involved in improving the accessibility of their own web experience. In this paper, we first present a survey of existing accessibility issues and then discuss technology that has been developed to overcome them. We next discuss user studies that have been performed in order to understand the user experience of blind web users. These studies along with related work will inform new remote studies that will observe blind web users as they browse the web using their own equipment. Finally, we propose a new approach to improving web accessibility that leverages the skill and intelligence of blind web users. This approach seeks to enable blind web users to independently improve the accessibility of their own environment in a way that can be shared with both web users and developers.

# 1 Introduction

For blind individuals, the web represents not only an incredible opportunity but also a constant reminder of the remaining barriers to accessibility. Text in electronic form can be converted to an accessible format automatically, without the help of a sighted person, allowing blind users access via either voice or Braille according to their preference. In this way, blind computer users gain independent access to information that was previously unavailable and do so using an efficient process that avoids costly manual translation. Creating accessible web pages is of paramount concern because of the quantity of information available.

Blind web users use software programs called screen readers to convert web pages to voice or refreshable Braille. This ability to choose is vital because those who can read Braille generally prefer the control it provides but less than 10% of blind children master Braille [59]. Most accessibility problems are rooted in a failure to adequately express the same information that a sighted user would see visually. Screen Readers must perform the difficult task of presenting web pages in a serial fashion for voice or Braille reading and, therefore, discard the visual representation for which many web pages are designed.

As an analogy to the task a screen reader performs, imagine describing the front page of the New York Times to a friend over the phone (see Figure 1). You might reasonably begin with today's headlines and would likely skip information that you deem unimportant, such as the standard information bar along the top. When you reach the table of contents labeled *INSIDE*, you might ask your friend if he would like to hear this information or rather skip it. If a photograph or chart on the page conveys information about today's news, you would likely describe it to your friend, but you wouldn't tell your friend about other images, such as the black horizontal lines that visually separate content. You would naturally perform numerous small but intelligent tasks as you described the page, separating out the most important information and organizing it in a way that is easy to express via voice.


The New York Times

Rumsfeld Memo Proposed 'Major Adjustment' in Iraq

As Trucking Rules Are Eased, a Debate on Safety Intensifies

Russian Ex-Spy Lived and Died In World of Violence and Betrayal

Imports Spurring Push to Subsidize Fruit and Greens

Some Players Use It to Cut Glare, Others to Turn Up the Spotlight

This Time, Some Black Leaders See a Mayor They Can Stand By

Figure 1: The front page of the New York Times illustrates the implicit semantic information that is expressed visually.

When blind people read web pages, the pages are not read to them by a sighted person who can skillfully interpret visual information and summarize with respect to surrounding context. Instead, a screen reader performs this task, often poorly, and blind web users can be left to guess the meaning of a web page. Users of screen readers have hope, however, because, if the web developer that created the web page did so with accessibility and usability in mind, then the screen reader can convey the information contained in the web page quite well. Unfortunately, if accessibility and usability were not adequately considered, then browsing for the blind can be an inefficient experience [36]. We refer

to the efficiency of web browsing in this context to be the ability of blind web users to quickly and accurately locate and use desired information. This task is rendered even more difficult when the web page contains dynamic elements.

To create accessible web pages, web developers must separate semantic content from its visual presentation, but they often fail to do so. As a specific example, in previous work we found that on the 500 most-visited web sites[1], only 39.6% of informative images were assigned alternative text, meaning that the information contained in over 60% of these images is inaccessible to blind web users [30]. In 2001, Coyne and Nielsen identified 75 guidelines beyond simply providing alternative text to which web developers should adhere in order to create accessible web pages [37]. Additional accessibility features, such as providing a mechanism to skip long lists of links or redundant information, providing heading information that outlines a page, and assigning names to the rows and columns of tables are likely to be used even less often. Most web pages are partly or totally inaccessible [73] and the accessibility of a web page decreases both over time and as web pages become more complex [46].

Figure 2: All of the information from this Seattle-area restaurant's web page is presented as graphic text displayed in images, including one advertising that Braille menus are now available at the restaurant. None of the images on the page contain alternative text.

While web developers are in the best position to affect accessibility from the start, they often do not do so adequately. Lack of consideration for accessibility is rarely intentional and is most often a result of a simple lack of awareness of accessibility issues (see Figure 2). Providing accessibility information for web pages is a subjective process and existing tools are not sufficient in assisting web developers, who are mostly sighted, in creating accessible content. While the available tools can detect some accessibility violations (such as when images lack alternative text), the tools lack the ability to suggest specific improvements and subjectively rate the quality of accessibility information. Available tools still rely on web developers to be skilled in accessible web development and fail to easily integrate into the diversity of tools used by web developers. This results in a cumbersome work flow when accessibility is included [51]. In most cases, web pages can be designed in an accessible way while maintaining the desired visual presentation [64, 20, 34, 75], but tools do not explicitly help web developers with maintaining visual presentation while improving accessibility. Many systems attempt fully automatic accessibility improvement, but have thus far either met with limited success or operate on very specific problems. For example, our WebInSight system automatically provides alternative text for web images, but can only label approximately half of web images [30]. In this paper, we propose a more pragmatic, combined approach for improving web

[1]Based on statistics available at www.alexa.com.

accessibility. Our approach advocates using automated accessibility improvement when possible and enabling blind web users to independently improve the accessibility of pages they visit when it is not. Both are done in a way that other web users and web developers can later leverage.

In this research, we will create tools with explicit consideration of the users who will employ them. We will first conduct user studies that will explore the browsing experience of blind web users over time while they are using their own equipment. We will use the results of these studies to inform the design and implementation of innovative tools that will directly involve blind web users in the process of improving web accessibility. We will also continue the development of our WebInSight system in order to automatically make images more accessible when prudent and to provide specific suggestions to developers when automatic improvement is likely to fail. We believe that web accessibility will be increased by creating tools that web users can use to independently increase web accessibility in a way that allows those improvements to be shared.

The remainder of this paper is organized as follows. In Section 2, we will first outline the goals and contributions of the proposed research, and, in Section 3, we will present a background of remaining accessibility issues that lead to inefficient web browsing by blind web users. Section 4 discusses prior and related work along two main themes: work that involves understanding how blind web users utilize the web, and work that seeks to improve the accessibility of the web. The two corresponding components of my thesis proposal are outlined in Section 5. Section 5.2 outlines the user studies that will be conducted and the infrastructure that will be required to complete them. Section 5.3 outlines the tools that I will create to enable blind web users to independently create and share accessible content and, thereby, take a more active role in improving their own web experiences.

## 2 Contribution and Goals

The contributions of the research proposed within this document are two-fold. First, we seek to better understand the remaining accessibility barriers confronting blind web users. We will collect statistics concerning the accessibility problems that are encountered and their perceived impact as blind web users browse the web using their own equipment. Second, we seek to develop and test innovative tools that will help blind web users take an active role in creating and sharing accessible content.

The goals of the project are as follows:

- Understand how blind users browse the web through focus group discussions and user studies. Utilize the knowledge gained to inform the development of tools for automatically improving accessibility.
- Design and implement new tools that will allow users to increase the accessibility of their own web experience and share their improvements with other blind web users.
- Continue development of the WebInSight tool for blind web users in order to help users automatically improve the accessibility of web images and provide web developers with a more efficient means of producing accessible content through suggestions.

These goals encompass novel research in understanding how blind users access the web and in developing interaction techniques that will enable them to independently improve the accessibility

of their own web experience. The specific components of this research, including an approximate timeline for completing them, are presented in Section 5.

# 3 Accessibility Issues

> "It is a mistake to think you can solve any major problems just with potatoes."
> - *Douglas Adams*

A myriad of work has explored potential improvements to web accessibility, but, despite this effort, a number of accessibility problems remain. As the web has undergone a transformation from a world of static pages to a dynamic place more like our desktops, accessibility problems have only increased. As these new problems emerge, they underscore the need to solve existing problems while addressing new problems. In this section, we will overview many of the accessibility problems widely considered to be the most troublesome to blind web users.

## 3.1 Images and other Multimedia Content

Images and other multimedia content on the web often lack alternative text description that conveys the meaning of this visual content to users who cannot see them. Both the World Wide Web Consortium (W3C)'s Web Content Accessibility Guidelines (WCAG) and Section 508 require that such content be assigned alternative descriptions, but a number of studies have shown this is done correctly less than half of the time [30, 38, 66]. Images containing graphic text are a vital subset of web images to target for labeling, not only because they are used to convey textual information, but also because of their popularity. Over 42% of web images contain text [74]. Text in images often provides more information to sighted users than what is available in the text of the web page and images containing text often serve as buttons that are required for navigation and form input. Although, anecdotally, providing alternative text for images is one of the first problems people think of when thinking of accessibility, even those with the best of intentions often fail to implement it (see Figure 2).

## 3.2 Separating Meaning and Presentation

Web developers often assume that the users of web pages that they create will be sighted, and this flawed assumption leads to problems related to the separation of semantic meaning and presentation. For example, web developers often encode semantic information directly into the presentation of the web page, making this information unavailable to blind users. Form fields and their labels are often associated only by visual proximity, a relation that may not be reflected when viewing the page linearly with a screen reader. Hyper Text Markup Language (HTML) heading tags (`h1`, `h2`, ...) function to express a semantic outline of a page, but web developers often circumvent this potential by expressing headings by simply making the font larger and bolder, which does not convey the heading information to screen readers. Finally, HTML tables can be used to organize tabular information, and it is possible to provide semantic labels to the columns and rows that provide for easier navigation with a screen reader. Unfortunately, columns and rows headers are rarely specified using the accepted method.

These issues can be resolved with proper use of existing semantic markup, but, often, web developers do not use them correctly. We will explore ways for blind web users to independently

assign such semantic meaning to elements that lack it. Projects related to the Semantic Web share the goal of assigning semantic information to web elements and may be useful when applied in this domain [28]. Previous work has explored automatically assigning semantic roles to content in the context of the semantic web [40] and similar methods have also been applied to improving web accessibility [68, 80]. As opposed to many semantic web applications that seek to use semantically-tagged information for automated inference, applications that use Semantic Web technologies to enable web accessibility should be easier to create. Such applications need not fully understand the semantic annotations provided because blind web users can provide this intelligence. For example, Mukherjee *et al.* explored techniques that automatically recovered the implicit visual structure of web pages [62], but did not attempt to categorize them.

Problems also arise when web developers alter the presentation of information using elements that also encode semantic meaning. For example, HTML tables are useful for organizing tabular data, but can be frustrating to screen reader users when used only for presentation purposes. In order to convey the semantic meaning of tables, screen readers announce column and row information when a blind user enters a new table or switches between cells in the table. To organize complicated web pages, web developers often use many, deeply-nested tables that do not visually relate the semantic organization of the page but rather the visual structure (see Figure 3). Navigating these complex trees of information with a screen reader can be frustrating and confusing, but to sighted visitors the complexity is hidden. We will explore automatic methods for detecting the use of tables for non-tabular data by learning a model of inappropriate nesting and data consistency and will investigate the use this information to inform a process that allows blind web users to appropriately restructure tables with little manual guidance.

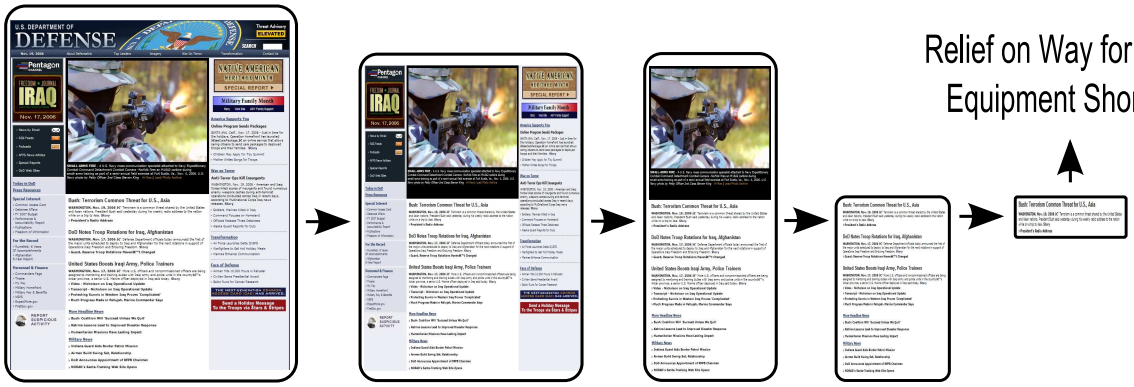



Figure 3: The Department of Defense homepage requires blind web users to navigate six levels of nested tables in order to reach the first headline because headings are not used and tables are used improperly.

### 3.3 Unannounced Context Changes

Unannounced context changes result when a user's focus and surrounding context is changed absent a specific request for the user to do so. For example, if clicking a link opens a new browser window or jumps to a different frame, blind users may not be aware that such a change has occurred. While sighted users detect this from visual differences observed away from their focus, the serial nature

of screen readers makes unanticipated changes more difficult for blind users to detect [19]. Blind users may also miss important contextual changes that occur visually away from their focus. For instance, selecting an item from a selection box may expand new content. Because the user's focus is at the selection box, they will not know that additional content has been added in the way that a sighted user would. These problems can be avoided by properly warning users of the effects of user actions and properly alerting users to non-contextual changes. We will explore methods for automatically conveying to users directly observable effects of their actions and methods that will enable them to choose how to handle such events in the future.

### 3.4 Dynamic Content

Dynamic web content is content on a web page that changes or appears in response to events triggered by input devices or timers. Dynamic web content is becoming ubiquitous as web pages begin to incorporate interface elements more commonly associated with traditional desktop applications than with static web pages. Dynamic content often uses Javascript and Cascading Style Sheets (CSS) to repurpose HTML elements in order to convey semantic information visually and can undergo numerous unannounced context changes. Often, such content is triggered by the use of a particular type of input device, such as a mouse, which makes them inaccessible to blind web users. For example, to use the CitiBank account web page, a user must use the mouse to access important account management menus, making them inaccessible to users of screen readers (see Figure 4). Dynamic content such as this is often created by repurposing existing HTML elements, and no broadly-applicable standard is currently implemented that allows web developers to semantically label the role of arbitrary elements to make them accessible. Access to content should not require the use of a specific input device, such as a mouse, and should be accessible without using dynamic menus and other non-standard controls until these can be made uniformly accessible. The working draft of the next version of the WCAG requires web developers to either annotate dynamic content or provide a separate accessible version and to separate actions from particular input devices [19].

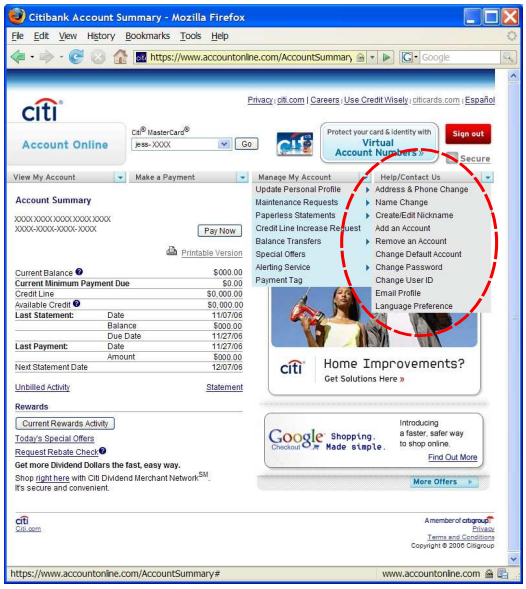



Figure 4: Blind web users do not have access to the menus on the CitiBank account page and, therefore, cannot fully manage their accounts online.

Standards that will help implement the accessibility of dynamic content have been considered in previous research [72, 42], and a working draft of a comprehensive standard for creating accessible dynamic content and controls is nearing completion by the W3C [14]. After such standards are implemented, web developers will be responsible for assigning the newly defined semantic annotations to their content and retrofitting previously created content to reflect them. Both web developers and web users need tools that will help them make sense of and correct for inaccessible dynamic content. Determining when a change occurred is relatively easy, but determining the *importance* of that change will be much harder.

### 3.5 Rich Internet Applications

Rich Internet applications are web applications that posses the features and functionality of what have traditionally been desktop applications. Rich Internet applications combine many of the accessibility issues of both dynamic content and unannounced context changes, but differ in their heavy reliance on such techniques for their core functionality. While such web applications allow mainstream users to interact on the web in new ways, they can be completely inaccessible to blind web users. Much of the content, however, could be developed in a way that is less prone to accessibility concerns. The current draft of the W3C Web Accessibility Intiative's Roadmap for Accessible Rich Internet Applications (WAI-ARIA) identifies steps required to make web applications accessible, but is currently in the planning phase and not yet implemented [14]. Nevertheless, a Rich Document Format (RDF) taxonomy for roles, states and properties has recently been published by the W3C and is already available to users with Window-Eyes 6.0 screen reader and Firefox web browser [15, 16]. The capability to fully express semantic roles for dynamic content is expected to be implemented in browsers sometime in 2007, but, to be useful, will still require web developers to use it to semantically label the elements of their web applications. We will explore automatic methods for assigning such semantic information, and methods to expose these methods to blind web users in a way that will allow them to assign them semantic meaning.

### 3.6 Flash

Flash is of particular concern for web accessibility. Accessibility in Flash is only possible in recent versions of the software and, even when available, still requires Flash developers to properly implement accessibility features. Even the basic process of verifying that a user has the proper software to utilize Flash accessibility enhancements is difficult because it relies on particular combinations of browsers and players that can be affected by the versions of both and the operating system that it is run on. Tools are being developed to gauge the accessibility of Flash in a way similar to the tools that automatically construct reports for web pages [71]. Many of the accessibility concerns from other types of web content are mirrored in Flash, such as inaccessible images and dynamic content, and approaches used to solve these problems may also benefit the improvement of Flash accessibility.

### 3.7 Summary

Blind web users face a myriad of accessibility problems, and automatically addressing these problems will require great leaps in the fundamental ability of computing systems to understand and process visual content. Instead, we plan to develop tools that leverage the abilities and intelligence of blind web users to enable them to affect the accessibility of their own web environment. The following section presents a selection of related work from which we plan to draw valuable insights to 1) better understand the problems facing blind web users and 2) implement tools that enable blind users to play a vital role in solving them.

## 4 Related Work

In this section, we will explore prior and related work in the areas of understanding the web experience of users and achieving accessibility that directly relate to the goals of this project. We

will first focus on efforts to try to understand the experience of blind users browsing the web and related technology that we can leverage in future work to better understand the user experience of blind web users. We will then explore prior work that has tried to overcome web accessibility issues during the many stages of accessible web content production.

## 4.1 Understanding the User Experience

> "Usability is like oxygen – you never notice it until it is missing..."
> - *unknown*

A vital component of improving web accessibility is understanding the user experience of those involved. Previous work has offered surprising results, such as the conclusion drawn by Mankoff *et al.* that blind web users may not be as equipped to thoroughly evaluate the accessibility of web content as sighted developers employing screen readers. In this section, we review a selection of this work and discuss the insights contained within it from which we will build when designing and implementing new studies. We hope to leverage this previous work to design new studies that will help determine which accessibility problems are the most problematic to blind web users, what can be done to solve them, and what stage in web content production and consumption is the most appropriate in which to introduce improvements. In particular, we are interested in techniques that facilitate remote user studies because we plan to leverage remote observation, augmented with in-lab focus groups and observation, in order to conduct the user studies that are proposed in Section 5.2.

Due to problems related to locating an adequately-sized population meeting specific requirements in a given geographic area and of recreating personalized setups and software configurations, sizable lab-based user studies with disabled populations can be costly, time-consuming and, in many cases, impractical. This has led some researchers to utilize remote evaluation and observation procedures instead [65, 58, 32]. Such user studies are particularly well-suited for investigating web accessibility issues because users employ a variety of screen readers and other assistive technologies that may be difficult to replicate in the lab. Because these technologies are designed for web usage, they are already connected to the Internet and are therefore more easily adapted to remote monitoring. In the remainder of this section, we first review work that has investigated remote user evaluation with disabled users in order to highlight the lessons learned. We will then discuss papers that investigate technical aspects of observing web users remotely.

### 4.1.1 Remote Studies with Blind Web Users

The most common type of remote study involving blind web users is a diary-based study. For example, Lazar *et al.* conducted a study in which blind web users recorded their web experiences in user diaries and discovered that, in contrast to sighted users, the mood of blind users does not seem to deteriorate as the inefficiency of the browsing experience increases [55]. Possible problems with diary-based studies are that users may misrepresent their difficulty in achieving tasks and may choose to report only experiences that are unusual. Another option is on-site observation in the homes and offices of participants as explored by Coyne *et al.* [37]. This type of study is expensive and is impractical for observation designed to last long periods.

Mankoff *et al.* compared the following four evaluation techniques for discovering accessibility problems in web pages: the Bobby [6] automated accessibility evaluator, web developers, web de-

velopers using screen readers, and blind web users who evaluated web pages remotely [58]. The web developers were each introduced to the WCAG accessibility standard [7]. Based on representative tasks developed in a baseline investigation with blind web users, members of each of the four conditions were asked to identify accessibility problems. The results indicated that multiple web developers employing screen readers were able to find the most accessibility problems and the automated tool was able to find the least. Remote blind users, although shown to be much less thorough at identifying web accessibility problems, most often found true problems (that is they labeled few false positives). This number was also artificially deflated due to users being unable to complete some of the tasks due to severe accessibility problems. The researchers also speculated that the remote users were not adequately encouraged to report all accessibility problems and hoped to improve on this in future work.

Petrie *et al.* investigated remote evaluation by disabled users further by comparing the results obtained via remote evaluation as compared to laboratory evaluation [65]. In this work, participants conducted the following two evaluation tasks: an evaluation of a system that converts certain technical diagrams into spoken language, and a summative evaluation of websites. Users that conducted their evaluations remotely were more likely to give high-level qualitative feedback whereas users that evaluated locally gave more specific evaluations. For example, when evaluating the same feature remotely, a remote participant said "there are a lot of problems with the description," while a local participant said, "I take it each thing in brackets is the thing they are talking about [Door to Hall and Living room]... [Door to Bathroom] it is not clear what the relationship is... I cannot technically understand these relationships...it just doesn't work." While both participants expressed their inability to understand the relationships expressed by this system, the local participant provided much more useful feedback. In other instances, usability was so poor that remote users could not determine if they had successfully completed a task and, therefore, provide an adequate evaluation. In local studies, researchers could tell participants whether or not they had successfully completed the task. An important conclusion of this work is that while remote evaluation may be appropriate for summative results, it is often not as valuable during formative studies when the technology may not work as intended and researchers benefit greatly from observing how users attempt to interact with it.

A common theme of both Mankoff and Petrie is that to maximize the value of remote evaluation, the technology used must achieve at least a base level of usability in order for users to be able to provide useful feedback. When operating away from the guidance of researchers, users may be less likely to be able to successfully complete required tasks. Furthermore, the quality and extent of qualitative feedback is likely to be much greater in local studies in which participants can interact with the researchers. The web studies that we plan to complete primarily involve recording quantitative data as users perform tasks that they already know how to do, and, therefore, a remote study should be appropriate in our case. We must still take care to extensively test any procedures that we will have remote users complete and plan to do so with local users with similar capabilities. We may need to reevaluate our methodology in the future as we develop and evaluate new technology with which users will not be familiar. In order to gather valuable qualitative data, we plan to supplement our recordings with phone interviews.

### 4.1.2 Recording User Data

In this section, we will present prior work that has used technology to record and aggregate statistics about that web usage in order to explore the user experience. Most of these systems have chosen proxy-based implementations that passively observe users as they browse the web, but such implementations are limited in their ability to collect qualitative data. As we review the work below, we will highlight techniques and technology that will plan to use in order to conduct our own user studies (described in Section 5.2).

Users have been central to the web from its inception and an extensive body of work has been dedicated to better understanding the user experience. Initially, much of this research concentrated on improving the quantitative performance of web infrastructure components. The Medusa proxy allows web researchers to go a step beyond these first systems by investigating the impact such systems have as perceived by the user [54]. The Medusa proxy is a non-caching forwarding proxy that provides several convenient features to researchers investigating the user experience. These features include the ability to simultaneously mirror requests (send a request to multiple sources) and the ability to transform user requests. The Medusa proxy was used to explore the user-perceived advantages of using the NALNR cache hierarchy and the Akamai content distribution network. The authors used the Medusa proxy for web browsing for eight days, during which they generated nearly 5000 requests. Decisions made in regard to which days and which authors would be recorded were not justified, which leaves remaining questions but should not affect the generality of their conclusions. By mirroring requests both through the NALNR cache hierarchy and to the origin server, the response times of each could be compared. They showed that the NALNR cache hierarchy improved mean latency by 15%, and that the Akamai edge servers were able to serve requests 5.7 times as quickly as the origin servers of their customers. Later work used the proxy to discover that of several HTTP optimizations, parallel connections provide the largest benefit and that persistent connections are helpful but only on the subset of pages to which they are fully utilized [27].

The ability of the Medusa proxy to record and then replay request logs may be useful in our studies by allowing us to reuse data collected by blind web users and, thereby, reduce the cost of testing variations in our studies. For example, we could test several techniques for automatically restructuring a web page in an effort to direct users to the most relevant content. We could run a study multiple times with the different layouts, or we could structure our study such that we record the information of interest to a user on a particular web page once and use it to test multiple hypotheses. For instance, we could see which method of restructuring the page leads to the recorded information of interest being placed closest to the beginning of the document in reading order. Mahmud *et al.* used a similar strategy to evaluate the improvements offered by CSurf [57].

We already employ a transformation feature similar to that offered by the Medusa proxy in our WebInSight system in order to dynamically transcode web pages to make their constituent images more accessible. While Medusa allows only requests and headers to be transformed, extending the system to allow arbitrary transformations of the web content seems straightforward. User studies could leverage this ability to transcode content in order to automatically remove or add additional accessibility features in order to determine their importance. For instance, we could remove the alternative text from all images that users view and observe the effect on their browsing efficiency.

Medusa allows researchers to accurately record quantitative data about web browsing as long as that data can be directly observed as part of an HTTP request, but the system is unable to

record finer-grained user events such as mouse movement, button clicks, or keys pressed. Such detail is important for discovering the particular web elements or components of accessibility that are particularly helpful when present or harmful when absent. Goecks and Shavlik demonstrated that by using an augmented browser, they could record mouse movement and scrolling behavior that could be used to predict a user's interaction with their web browser [43]. Claypool *et al.* utilized a similar approach in order to compare the explicit ratings that users provide to web pages with several methods of gathering implicit ratings [35]. While an augmented web browser allows recording data at a finer granularity, we would like to avoid requiring users to use specific web browsers to avoid confounding factors related to the introduction of new tools. Users in disabled populations use a wide variety of web browsers and screen readers, making it impractical to provide a version tailored to each user's individual setup.

A proxy-based approach is desirable for us because it allows users to be observed remotely using the technology to which they are accustomed and is relatively easy for participants to setup. Gonzalez introduced a Java-based system that is capable of remotely monitoring users as they browse the web [44] and later suggested its potential for conducting remote usability studies with people with disabilities [45]. The goal of this approach is to eliminate confounding factors that are often an unfortunate consequence of traditional laboratory studies, such as users being unable to use the assistive technology to which they have become accustomed. The proxy introduces Java applets into web pages that, once on a client's machine, can observe the actions of users and report back to the remote server.

UsaProxy uses the light-weight option of using Javascript to remotely record user actions and found it to be quite powerful [23]. In this approach, users connect to a proxy server which alters web pages on-the-fly to include a Javascript script that monitors the user's actions. When a page is loaded, the Javascript script attaches event listeners to most Javascript events, including the **keypress**, **mouseover**, and **mousedown** events. It also includes code that polls at regular intervals to determine the position of the mouse. For all events, the script records the event type, the time the event occurred and the element of the Document Object Model (DOM) associated with it (when applicable). To allow this information to be recorded, the system sends messages back to the server using Asynchronous Javascript and XML (AJAX). The resulting log file contains a combination of these user event recordings and the information recorded by a traditional web proxy, such as the Medusa proxy. Use of the system is unobtrusive and does not affect the normal function of the web browser. Atterer *et al.* later demonstrated how a complex AJAX application (gmail.com) can be remotely observed and evaluated using the system [22]. Javascript running in the browser is particularly suited for this type of observation because it has access to the DOM representation of the document being displayed, and automatically incorporates updates that have been made dynamically. We plan to use a similar setup in our user studies to observe low-level data that will allow us to record quantitative information associated with the browsing behavior of our users. This future direction is described in more detail in Section 5.2.

## 4.2 Achieving Accessibility

> "To know the road ahead, ask those coming back"
> - *proverb*

The accessibility of web content can be implemented and improved during the many stages between when the idea for the content is originally conceived to when the implemented web page

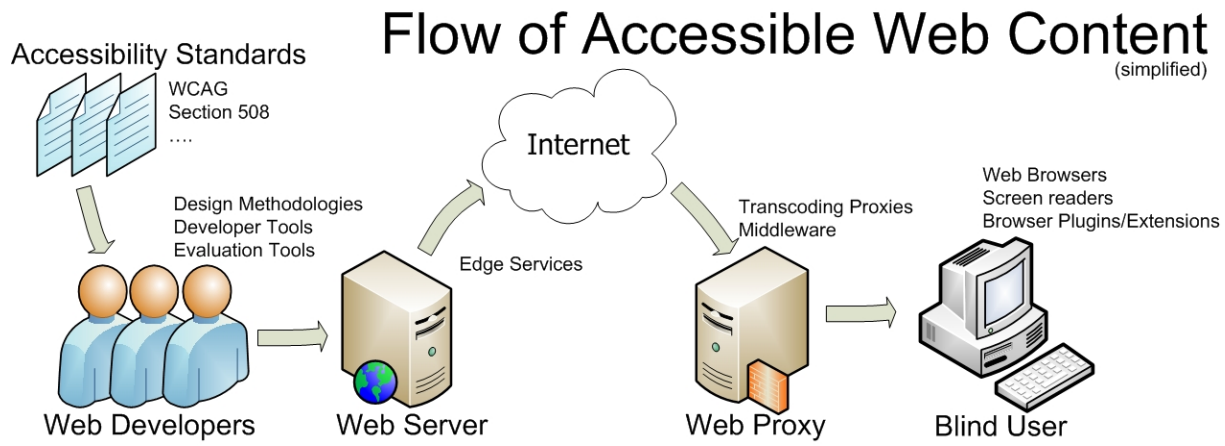



Figure 5: An overview of the flow of accessible content from web developer to blind web user, along with a selection of the components that have been explored act at each stage designed to improve web accessibility. While later stages can influence earlier stages, such change is slower and more difficult to achieve.

is conveyed to web users. The stages of this process are outlined in Figure 5 along with a selection of techniques that have been explored at each stage. This diagram can be used to place the related work that will be presented in the remainder of this section.

In the remainder of this section, we will highlight work at each stage along the web publishing pipeline. At each stage along the way, we will discuss a sample of solutions that have been explored at that stage. A summary of the systems discussed in this section is available in Table 1. Previous work has generally been implemented at one of the stages along this pipeline with little opportunity for blind web users to independently contribute. To influence this process, blind web users have been required to work against the current.

#### 4.2.1 Accessibility Standards

Web accessibility standards set guidelines that, if met, should ensure that a web page is accessible to blind users and other web users with disabilities. Achieving accessibility through a set of specific guidelines is difficult in general because implementing web accessibility requires implementing efficient access not just making information available. By requiring the web developers in the screen reader condition of the comparative study previously discussed by Mankoff *et al.* to use a screen reader, the researchers may have implicitly forced the web developers to consider usability [58]. This may partially explain why they fared better than their counterparts who did not make use of the software. Problems with static standards have been previously noted [53], but remain in use, in part, due to the difficulty of formulating, checking and enforcing more subjective usability standards.

The most important web accessibility standards for web developers in the United States are the W3C's WCAG [7] and the technology access requirements of Section 508 of the U.S. Rehabilitation Act of 1973 that were expanded and strengthened by the Rehabilitation Act Amendments of 1998. Many other countries have similar accessibility requirements, many of which are based on the W3C's WCAG [75]. In the United States, only web sites that receive funding from the Federal government

are compelled to comply with Section 508 guidelines, while private entities are exempt. A recent court case, however, may allow a wider selection of web sites to face legal trouble if they fail to implement accessibility features. In *National Federation of the Blind vs. Target Corporation*, a Federal Circuit Court ruled that Target Corporation could be sued on grounds that its inaccessible web site violated the Americans with Disabilities Act (ADA) because the web site is an extension of the physical store [13].

#### 4.2.2 Developer Tools

Numerous tools are available to web developers that seek to automatically identify accessibility issues. Some of the most popular include A-Prompt [1], UsableNet's LIFT [11], W3C Accessibility Validation Service [18], IBM alphaWork's aDesigner [4], and Watchfire's Bobby Worldwide [6]. These tools commonly report on how well a web page adheres to web standards, concentrating on easily identifiable problems such as missing alternative text for images, the failure to use row and column heading tags in HTML tables, and the use of deprecated tags to produce visual effects instead of using the preferred CSS. These tools also often provide advice to web developers on how to fix the errors that have been identified, but they cannot offer subjective suggestions or critical feedback. As an example, these tools currently lack the ability to identify images that need a non-zero-length alternative text description, and will pass a web page that uses zero-length alternative text for all images. As noted previously, web developers must be skilled in order to effectively implement standards.

The Dante approach recognizes that not all elements of accessibility, particularly those that deal with efficient use of a web page, can be met by existing HTML markup, and offers web developers the ability to semantically annotate their web pages in order to facilitate more appropriate audio presentation. Originally, this semantic knowledge was assigned manually and was introduced by Yesilada *et al.* [81]. The Web Authoring for Accessibility (WAfA) ontology (introduced as the Travel Ontology) was designed to facilitate web navigation and is the ontology that Dante uses for annotation [80]. This ontology includes such concepts as **Advertisement**, **Header**, and **NavigationalList**. Once annotations from this ontology are made to a web page, Dante allows easier navigation of it by performing transformations, such as removing advertisements, providing skip links to bypass headers and navigation links, and allowing users to easily move from one semantically-grouped section of a page to another.

A downside of this approach is that it required manual annotation by web developers. Work by Plessers *et al.* removes the manual component of creating annotating visual objects with semantic guidelines entirely by building such annotation directly into the design process [68]. This work was based on an existing web engineering approach (Web Site Design Method (WSDM) [39]), but could potentially be extended to work with any formal design methodology. Design methodologies in general, and WSDM in particular, help web developers break down the task of creating a usable web site into a series of manageable phases. They force designers to carefully consider their target audience and the tasks they will most likely want to perform before considering low-level details related to implementation. As part of this process, WSDM guides users through creating models of navigation, tasks and objects on the site. Plessers *et al.* demonstrated that 70.51% of the WAfA ontology could be automatically generated on web sites that were created using the WSDM process by directly mapping elements of the WSDM ontology to elements in the WAfA ontology (84.62% in the best case). This approach certainly has potential in that it shows web developers that are

| Name | Stage | Description |
|---|---|---|
| WCAG [7] | Accessibility Standard | W3C web content accessibility guidelines. |
| Section 508 | Accessibility Standard | Provision of the U.S. Rehabilitation Act. |
| Web Engineering Approach [68] | Developer Tool | Automatically converts elements of the WSDM ontology to the WAfA ontology for use in Dante. |
| WebInSight-Developer [31] | Developer Tool | Assists web developers in formulating appropriate alternative text for web images. |
| A-Prompt [1] | Evaluation Tool | Helps web developers meet W3C WCAG 1.0 standards. Suggests improvements. |
| W3C Access. Validation Service [18] | Evaluation Tool | Detects web pages that do not conform to W3C standards. |
| IBM aDesigner [4] | Evaluation Tool | Evaluation tool that includes a simulated screen reader. |
| Watchfire's Bobby Worldwide [6] | Evaluation Tool | Popular online accessibility evaluator that detects W3C WCAG violations. |
| UsableNet's LIFT [11] | Evaluation Tool/ Edge Service | Helps web developers identify accessibility problems. Can be run automatically as an edge service. |
| Edge-services for Colorblind Users [49] | Edge Service | Alters images and web pages to use colors better-suited for color-blind users. |
| Personalized Edge [50] Services | Edge Service | Provides a number of services that users can select to make content more accessible. |
| Altifier [79] | Edge Service/ Transcoding Proxy | Adds alternative text to images using a set of simple heuristics. |
| Dante [81] | Transcoding Proxy | Uses manually-defined ontology to aid non-visual web navigation. |
| WebInSight [30] | Transcoding Proxy | Formulates and inserts alternative text for web images. |
| Middleware [47] | Transcoding Proxy | Adds context and preview to web links. |
| GIST Summaries [48] | Browser Extension | Adds summary descriptions to web links. |
| Accessmonkey [31] | Developer Tool/ Transcoding Proxy/ Browser Extension/ User Tool | Transcodes web content using Javascript in order to handle dynamic content. Scripts can be used by both web users and web developers. |
| Hearsay [69] | Screen Reader | Screen Reader that converts HTML to appropriate VoiceXML through semantic partitioning. |
| CSurf [57] | Screen Reader | Extension of Hearsay screen reader that allows context-driven web browsing. |
| ESP Game [77] | Other | Online game that entices users to accurately label web images. |
| Phetch [78] | Other | Online game that entices users to provide accurate descriptions of images. |

Table 1: A selection of the approaches to improving accessibility presented in Section 4.2. For each entry, the stage of production in which it operates is identified in reference to Figure 5.

already using a formalized method for designing and implementing web pages that they can build in accessibility without added cost. Unfortunately, its practical benefit would seem to come largely from its ability to shift the justification for semantic annotation of content away from the merits of accessibility to a justification based on the merits of using a formal design method. Plessers *et al.* also showed that accessibility through semantic markup can be built into dynamically-generated template-based web pages, which is a powerful idea and may benefit the accessibility of a number of web sites even if they aren't employing a formal design method. The automatic method presented in this work still requires manual annotation, and still suffers from the drawbacks associated with such approaches.

#### 4.2.3 Automatic Transcoding for Improved Accessibility

Automatically transcoding content in order to render it more accessible is an approach that has been explored extensively. Tools that use this approach generally intercept content after it is retrieved from the web server (where it is stored) and before when it is read to web users. This step can occur as part of an edge-service co-located with the web server, as part of a transformation proxy to which users connect, or as part of a web browser plugin or extension.

A number of previous systems have been introduced that help screen reader users independently transform documents to better suit their needs, including several systems that use a proxy-based architecture that allows web pages to be automatically made more accessible before they are delivered to blind web users. Iaccarino described edge-services for improving web accessibility designed to be implemented at the edge between web servers and the Internet [49, 50]. Others have proposed that such transformations be implemented in a proxy to which clients can connect [47]. Harper *et al.* created an extension for the Firefox web browser that inserted GIST summaries to allow blind users the ability to get an overview of web content without reading the entire page [48]. Luis von Ahn *et al.* created online games that entice humans to accurately label images and suggested that the labels produced could be stored in a centralized database where they could be used to help make web images more accessible [78, 77]. Altifier provides alternative text for images using a set of heuristics [79].

Our WebInSight system originally used a proxy-based approach to alter web pages on-the-fly as users browse the web in order to add appropriate alternative text to many web images [30]. In later work, we proposed a Javascript-based solution that can perform a wide variety of content transcoding operations (including replicating the functionality of WebInSight) on a variety of platforms [31]. Our completed work is discussed extensively in Section 5.1.

#### 4.2.4 Screen Readers

Screen readers were originally developed in the context of console computing systems and converted the visual information displayed by a computer to text that could be read (hence the name *screen reader*) by directly interfacing with the computer's screen buffer. Since the arrival of this technology in the 1980's, a number of more advanced screen readers have been developed to handle the graphical user interfaces exposed by modern operating systems. Popular screen readers include Freedom Scientific's JAWS [10], G.W. Micro's Window Eyes [8], IBM's Home Page Reader [9], IBM alphaWorks' open source Linux Screen Reader (LSR) [12] and Screenreader.net's freeware Thunder Screen Reader [17]. Modern screen readers attempt to interface directly with the applications that

they are reading and have introduced technology, such as off-screen models of web browsing that seek to overcome limitations with that interface.

Screen readers have been developed extensively to make web content accessible and they work reasonably well when content has been appropriately annotated for accessibility, but the user experience can still be frustrating and inefficient. The hypertext environment of the web is represented by a wealth of links, multimedia and structure, and the HTML which represents much of it often lacks the necessary annotation for conveying this information non-visually. For a blind person, a screen reader currently provides the most direct control over how web content is presented. Determining the semantic purpose of web elements is difficult in general, and most screen readers report based on the underlying HTML markup instead.

The Hearsay web browser is a screen reader that is designed to be controlled by voice and transforms web content into VoiceXML for audio output [69]. Hearsay does extensive processing on the DOM of web pages to automatically transform it into a semantic tree represented in VoiceXML [5] that is ideal for audio browsing. This approach leverages automatic identification of repetition and locality in HTML documents in order to automatically derive semantic structure. The process for discovering implicit schemas on web pages was derived from earlier work by Mukherjee *et al.* that focused on isolating the schemas imposed by template-driven HTML documents [62]. This process identifies semantically-related groups of objects that appear together, such as items appearing together in a list. For instance, alternating **<h1>** and **<p>** tags in HTML often express the semantic pairings of a title of an object and its summary. In this example, these would all be siblings of their parent node, but for auditory browsing they should be paired to allow the listener to efficiently select each group. To aid in the successful reconstruction of the semantic tree, the system uses various heuristics in order to accept partial matches, and leverages semantic similarity of nodes as determined by the similarity of the words contained within them. CSurf, a recent addition to the Hearsay web browser, allows web users to browse in a context-directed manner [57]. In this system, when a user clicks on a link, the text of that link is used to automatically detect an appropriate place in the resulting web page for the browser to begin reading.

Hearsay has the potential to greatly improve the efficiency of web browsing, but does so at the cost of reduced transparency for users. The tree representation created by Hearsay may place items in unexpected levels of the tree, which may render it confusing to users. The system was shown to be very accurate in its ability to correctly create the semantic tree, but these experiments were conducted in the "news" domain for which an ontology was manually created and Hearsay was manually tuned. The authors later showed the promise of bootstrapping ontologies for new domains [61], although it remains unclear how well these techniques will perform in arbitrary domains. User studies of the system were positive, but they they too operated on this manually-tuned "news" domain and were tested only in short lab studies by (mostly) sighted users. Evaluators hinted at the issue of transparency by suggesting that they would like additional control and that they wanted Hearsay to be more explicit about the types of elements in the semantic tree. CSurf suffers similar transparency concerns because users are automatically redirected to content but provided no information concerning where on the page they have been redirected or the surrounding context.

#### 4.2.5 User Involvement

To browse the web as a blind web user, one must be both skilled and patient. Noticeably missing from this discussion has been the idea that users can directly affect the accessibility of their own web

environments. We believe that this effectively limits participation in the web to only those blind web users who are the most technically-savvy. We plan to develop technology that will help users with a wide range of technical ability effect the accessibility of the web and share improvements with other users and developers.

Currently, blind users can affect the accessibility of the web by choosing from the technological solutions falling into the categories outlined above. Iaccarino *et al.* [50] attempted to maximize the ability of users to choose by providing many options for transcoding and letting users choose the combination that worked best for them to apply. Users were still restricted to the options offered by the system. When faced with inaccessible content, blind web users can request that developers create more accessible content (or in some cases sue in order to force them to do so, as in *National Federation of the Blind vs. Target Corporation* [13] and *Maguire vs. SOCOG* [33]). The ability of blind users to directly influence the accessibility of the web is currently severely limited, and we plan to develop new tools that will allow them to more adequately do so as proposed in Section 5.3.

# 5 Thesis Proposal

This section comprises the thesis proposal component of this paper. It consists of a description of the research that will be pursued and offers a general timeline for completion. We will first describe research already completed, followed by the new research problems that we plan to explore. This proposal will focus on two complementary research areas. First, we will seek to better understand the usability of the web by automatically recording statistics concerning how blind users browse the web using their own equipment. Second, we will use this knowledge to inform intelligent tools that will actively engage blind web users in the process of making web content more accessible.

In the remainder of this section, we will first outline the research that we have already completed. Next, we will describe the new research problems that we will investigate and how we will approach them. These research problems are divided into two parts: 1) the studies we will conduct in order to better understand how blind web users utilize the web, and, 2) the technology that we will develop based on the results of these user studies in order to improve the efficiency of blind web users.

## 5.1 Completed Research

In previous work, we conducted web studies that showed that a significant number of web images remain inaccessible. To combat this problem, we introduced WebInSight, a system that automatically derives and inserts appropriate alternative text for many images. Most recently, we introduced Accessmonkey, which allows WebInSight and many other systems designed to automatically improve the accessibility of the web to be easily used by both web users and developers using a variety of platforms and software.

### 5.1.1 WebInSight

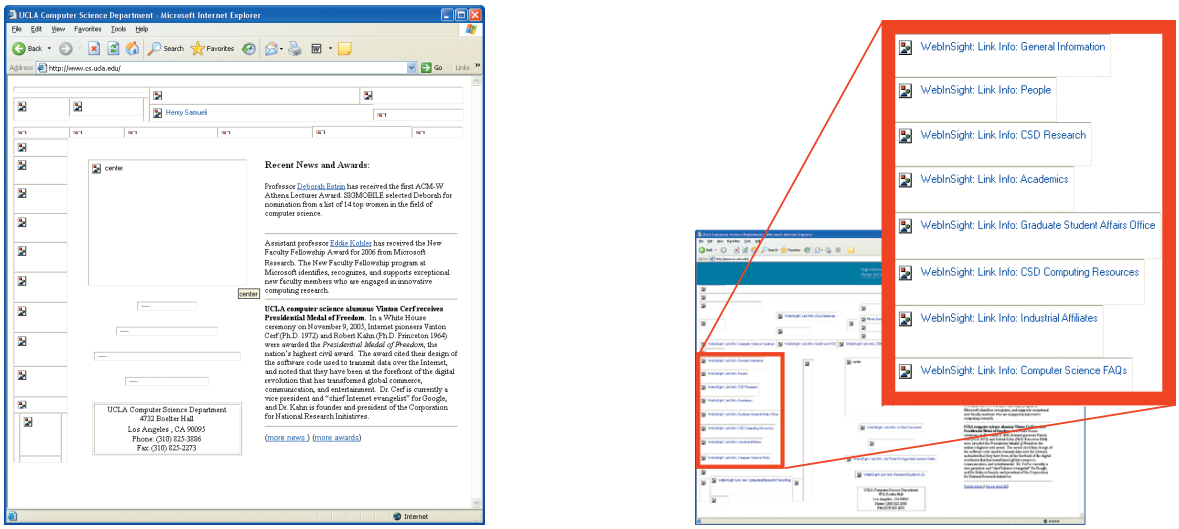
Our current WebInSight project focuses on improving the accessibility of web images. Images are the most obvious area for accessibility concern on the web, and, yet, remain one of the most prevalent accessibility problems [30, 38, 78, 66]. As mentioned earlier, we found that only 39.6%

of images that needed non-zero length alternative text on the 500 most-visited web pages were supplied alternative text. We also recorded all of the images viewed by members of the University of Washington Computer Science & Engineering Department over the period of a week. We did this by passively observing all web traffic entering the department. In this analysis, only 63.2% of the images that needed alternative text were assigned alternative text. To help ameliorate the problem, we introduced WebInSight, a system capable of automatically formulating and inserting alternative text for many images as a user browsers the web without negatively affecting the user experience [30].

In this system, alternative text for images were automatically calculated using three different labeling methods: Optical Character Recognition (OCR) labeling, context labeling and human labeling. OCR labeling is an important labeling method because many informative web images contain graphic text [74]. OCR software is targeted at images that contain black text on a white background scanned in at a known resolution, but web images come in a diversity of colors, resolutions and formats. The WebInSight system utilizes additional pre-processing that was shown to improve the recall rate of text recognition on web images by 25% [30]. The WebInSight system also uses context labeling to provide alternative text for images that are links. This method uses the observation that the title and headings of a linked page often serve as a good label for the link because, if nothing else, a user will know where they will be redirected before clicking on the image by reading such a label. The final options for labeling that is utilized by the WebInSight system is to provide alternative text by employing popular online games that compel humans to label images accurately [77, 78]. While potentially expensive in terms of latency, compelling humans to label images is attractive because humans are quite accurate.

The WebInSight system is implemented as a web proxy, which blind users can access using their current equipment. After configuring the system, all web requests that a user initiates are filtered by the WebInSight system. To avoid harming the user experience, the system initially only inserts alternative text that is already in its database. Labeling images can require a few seconds per image and forcing users to wait while all of the images on a page are labeled before presenting the page is unacceptable. Instead, the system inserts Javascript code that allows the system to add additional alternative text as it is created. This alternative text will be inserted from the beginning the next time that the user visits the page. Figure 6 shows the homepage of the UCLA Department of Computer Science both before and after applying WebInSight. The WebInSight system was found to be capable of inserting alternative text for 40.3% of the images that weren't assigned any by the containing web page's creator [30]. An important aspect of the system is that at worst it will produce the original web page with extra information and it will deliver the web page approximately as quickly as it would arrive without WebInSight. All features of the system are fully customizable by the user, allowing blind web users to decide which features are helpful and which they would prefer to suppress.

Alternative text is only valuable to blind web users if it serves as an alternative to the content provided by the image, which is often not the case [36]. Craven *et al.* found that five of the ten most frequently used alternative texts, including "1," "pad," and "*," were unhelpful [38], and blind users who are technically savvy often write custom scripts for their screen readers that help eliminate such obviously bad alternative text from being spoken. Based on these observations, the task of judging the likelihood of alternative text correctness is clearly important. An automatic method for judging the quality of alternative text would allow blind web users to suppress alternative text that is likely to only be an annoyance, allow the WebInSight system to assess the alternative text




(a) (b)

Figure 6: (a) The homepage of the UCLA Department of Computer Science showing the lack of alternative text for all images vital for navigating the site. (b) The web page as transformed by WebInSight contains correct alternative text for these images.

that it creates, and help web developers create better alternative text.

To this end, we have developed novel features based on a contextual analysis of nearby text on the web and analysis of basic features of the images that allow the system to create an accurate machine learning model of what comprises good alternative text versus bad alternative text. In two experiments, the system was able to differentiate good versus bad alternative text with precisions of 86.8% and 91.2% on two data sets respectively [29]. These preliminary results suggest that with additional training and refinement, the model could be made even more accurate and be incorporated into tools for users. Judging the subjective quality of alternative text has long been assumed to be too difficult to do automatically. These results suggest that such automatic methods could be applied successfully to other difficult and subjective tasks involved in making web pages accessible.


Figure 7: Extracting the text from this picture is difficult without correcting for distortion due to perspective.

Techniques that probabilistically associate contextual words with features of the image could be adapted to produce helpful labels for context-rich web images [24, 25]. We will seek to further improve the performance of the OCR currently used by WebInSight. For example, more complex text, such as text contained within photographs, pose additional problems for the system both because they tend to contain many colors and because of visual distortions present in real objects (See Figure 7). Recent work has improved the extraction of text from videos and photographs of documents by first correcting for distortions caused by skew, perspective and geometric variations [56, 63]. Including these technologies in the WebInSight OCR process could help extract more text appearing in web images.

### 5.1.2 Accessmonkey

We recently introduced Accessmonkey, which provides a common scripting framework, based on Greasemonkey [3], for web users and developers that is able to provide a wide range of accessibility improvements on many different browsers and platforms [31]. Scripts written for Accessmonkey can improve the accessibility of web pages as they are viewed and also assist web developers in creating more accessible web pages. For example, the WebInSight Accessmonkey script both adds alternative text to images on-the-fly as users browse the web and provides an interface that suggests alternative text to web developers for images on their pages (See Figure 8). Because the scripts used by Accessmonkey are written in Javascript, they have the potential to be able to correct a wide variety of accessibility problems, but difficulties remain. First, creating scripts is beyond the technical capabilities of most users. Second, scripts can be quite specific to particular web pages and currently there is no convenient method for users to quickly locate appropriate scripts. Section 5.3 outlines how we plan to solve these problems.

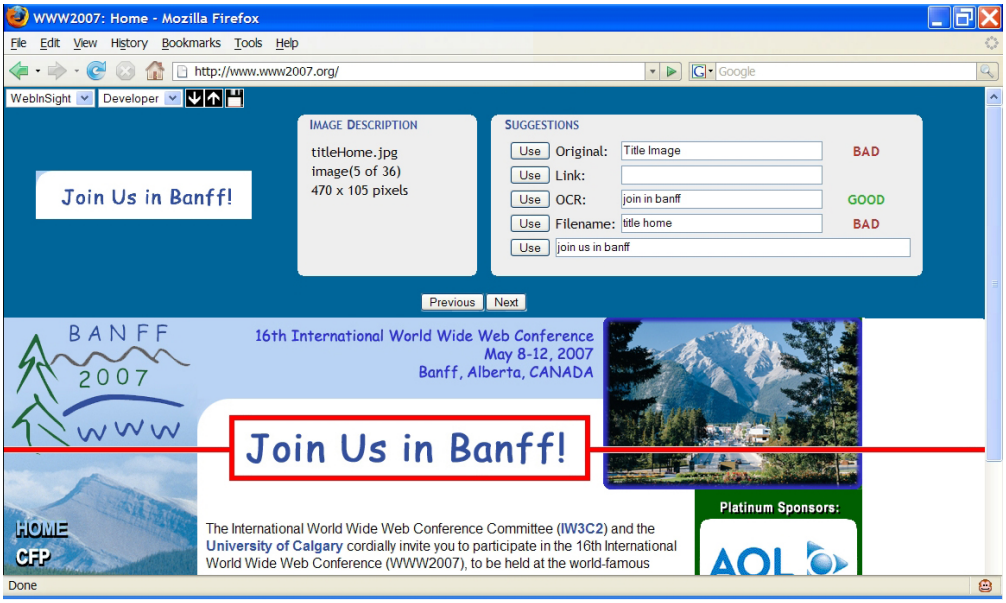



Figure 8: A screenshot of the WebInSight Accessmonkey script in developer mode applied to the homepage of the International World Wide Web Conference. This script helps web developers discover images that are assigned inappropriate alternative text (such as the highlighted image) and suggests alternatives.

## 5.2 Understanding the User Experience

In order to better understand the user experience of blind web users, we will conduct user studies investigating what leads to inefficiencies as users browse the web, what problems most need to be solved, and what approaches are likely to succeed. In this section, we will introduce the technology that we plan to use to conduct our studies and the details of the first two phases of users studies that we plan to complete. As this research continues, we will likely conduct additional user studies

using similar technology in order to evaluate the tools that we will produce and to investigate additional hypotheses.

### 5.2.1 Understanding Blind Users

In order to better understand the user experience of blind web users and how it differs from that of sighted users, we will first conduct a user study in which members of both groups will be observed. These studies will be novel in their observation of blind users because they will record low-level observations of blind web users using their own equipment and software. This is in contrast to the previous studies, which have been primarily lab- or diary-based. Additional discussion of the reasons for preferring remote studies in this situation were discussed in Section 4.1.1.

Our studies will utilize an augmented version of the UsaProxy[2] in order to record both web requests initiated by the user and actions performed by users on web pages. This software is the publicly available version of the software describe by Atterer *et al.* [23] and described in this document in Section 4.1.1. Our studies will require the existing UsaProxy to be altered in order to record additional user events and characteristics of the web pages visited by our users. These modifications will include adding the ability to record the text of links that are clicked and the ability to record when dynamic elements are added to web pages. We would also like to explore the perceived effect on the browsing experience of modifying specific features of web pages. For instance, we may alter the alternative text that is assigned to images contained within a web pages by removing it, adding alternative text known to be correct, or inserting alternative text computed by WebInSight. In addition to the quantitative feedback that the low-level measurements collected by UsaProxy will provide, we will also ask users to annotate portions of their web browsing traces. The proper way to elicit such annotations and to represent recorded traces is a sub-problem that we will need to investigate via additional user studies.

During our first study, we will seek to quantitatively investigate a number of assumptions that have guided the direction of accessibility research. For example, images that lack alternative text are likely to make web browsing difficult, but it is possible that, in practice, the content contained within them can be determined by their URLs, the surrounding context, or simply accessed in another way. In the first stage of this study, we will analyze the web traffic generated by both blind and sighted participants to test hypotheses, such as those listed in Table 2. We will augment quantitative data with qualitative data from users. Other information collected during this phase will include: URLs accessed, title of each page, time accessed, referring page (if applicable), referring link type (image, text, or other if applicable).

From the annotations made by participants during the first phase of the study, we will identify 16 common web-based tasks that different blind participants will be asked to complete during the second phase of the study. During the second phase of our study, participants will complete the tasks on web pages using alternative text for images as provided in one of four ways: 1) no alternative text provided, 2) by the web page originally, 3) by WebInSight, 4) "perfectly" by the researchers. The approach developed in this phase will be used to evaluate other tools that are developed as we continue our research. Namely, we will have users perform specific tasks on web pages that have been enhanced by our tools, not enhanced by our tools and created to properly consider accessibility concerns.

[2]Available at http://fnuked.de/usaproxy/.

| Potential Hypotheses |
|---|
| 1. Blind users will not use the mouse. |
| 2. Blind users will not simulate using the mouse. |
| 3. Blind users will access elements closer to beginning of the HTML source, tab order, etc. |
| 4. Blind users will use "find" function more often than sighted users. |
| 5. Blind users will be less likely to click on images without alternative text. |
| 6. Blind users will be less likely to click on images with inappropriate alternative text. |
| 7. Blind users will spend more time on each web page. |
| 8. Blind users will click on skip links. |
| 9. Blind users will click more often on links that contain descriptive anchor text. |
| 10. Blind users will use multiple screen readers. |
| 11. Blind users will visit fewer sites that contain dynamic content. |
| 12. Blind users will not interact with content that has been dynamically introduced. |

Table 2: Hypotheses to be tested during the first stage of planned user studies.

## 5.3 Improving Accessibility

As this research continues, we plan to investigate creating technology that will enable blind users to play an active role in the creation of accessible web content. We will investigate technology that will enable blind web users to contribute to improving the efficiency of their own browsing by implementing changes to web content and sharing those changes with others in a way that they can easily find and incorporate. We will also continue and expand work begun in our original WebInSight system to make content that is unavailable to blind web users (such as the information contained in images) accessible.

Each of the two sub-sections in this section provides a hypothesis that we will explore through the development and evaluation of new tools for blind web users.

### 5.3.1 Enabling Independent Accessibility Improvement and Sharing

**Hypothesis:** Blind web users can independently improve the accessibility of their own web experience.

A plethora of systems have been designed to automatically improve web accessibility, but these systems all fall short. Increasing the accessibility of a web page requires understanding the contents of that web page, and understanding the content of web pages requires fundamental advances in artificial intelligence and computer vision that are still far off. Previous work has shown that harnessing humans to accomplish tasks that are too difficult for computers can create startling results. von Ahn *et al.* entices users to label web images with enjoyable computer games [77, 78] and Amazon pays users to perform a variety of tasks currently beyond the capabilities of computers [2]. We propose that instead of relying only on web developers and fully-automated methods, that blind users be enabled to produce and share accessible content. Blind users directly benefit from accessibility improvement and, therefore, possess the motivation to improve content. Users who choose to improve content will benefit not only themselves but also future visitors to the improved web page.

Much of the information on web pages that is considered inaccessible to blind users can be accessed, but only with great effort and technical sophistication. Previous tools have indicated that users may be willing to go through an inefficient process once, if it means that the process will be more efficient next time. For example, the Greasemonkey Firefox extension [67, 3] allows users (both sighted and unsighted) to alter content in a way that is more pleasing or useful to them. Popular scripts remove ads, reorganized content, bypass login screens and improve accessibility (See Figure 9). Such scripts bring control back to the people consuming it and expose different views of the same content that target different users with different goals. Blind users could create scripts that render content more accessible, or offer several different "views" of the content in order to reduce its complexity, and share these changes with other blind users.

As shown in our Accessmonkey system, transforming content with Javascript to render it more accessible is quite powerful. Unlike systems that operate before the content is delivered, Javascript can easily integrate with dynamic web pages and even web applications. Gibson *et al.* showed that the templates on which web pages are created tend to change every 2-3 months on average, and we might reasonably expect scripts that are created to be applicable for at least this long. Two remaining issues need to be addressed before it can be shown to be a viable route to accessible web content for blind web users. First, users must be able to find and apply scripts designed for a particular web page seamlessly as they browse the web, and, second, users need tools to be available to help them independently create scripts that can improve web accessibility.

In order to make scripts available as users browse the web, we will first build an online repository where users can submit and find useful scripts. Next, we will create a browser extension that will allow users to retrieve scripts targeted to the web pages that they visit and give them the option to apply them. These scripts may either be chosen manually by each web user when they first visit a page or we may develop techniques for determining which script is likely to give users the best view of a page initially. We will need to do this with careful consideration of both user interface design and security. The system should be easy to use and should not make the usual browsing process less efficient. We will also need to implement standard descriptions for the scripts in order to allow script authors to inform users as to what the script does and we will also need to give users a way to provide subjective ratings of the available scripts. In order for users to be able to quickly apply these scripts, they should not be required to worry about whether or not the script is safe to use. The system will implement an appropriate security model to ensure this, for instance, by preventing user scripts from accessing content on other web pages.

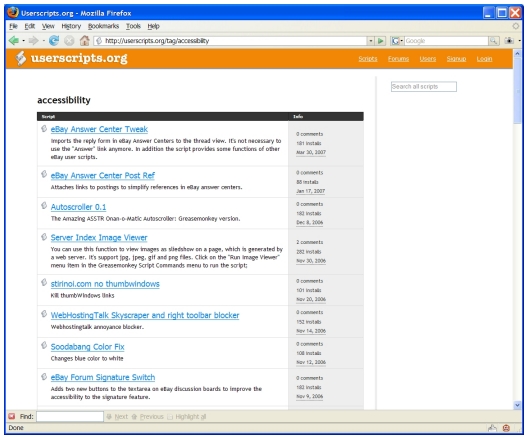



Figure 9: A screenshot of the userscript.org web page for scripts tagged with "Accessibility."

This method of creating accessible content will only be valuable if a large user base is able to create and contribute scripts. Currently, the scripts used to render content accessible are complicated and beyond the technical capabilities of most users to write. Creating and using such scripts would allow blind users more efficient, independent access to content that is currently unavailable and provide a human-centric solution to many accessibility problems without relying on

web developers. Several existing systems allow users to create scripts that can alter how they browse the web. For instance, the Platypus Firefox extension allows users to rearrange, delete, and change the appearance of page elements via a point-and-click interface [76]. Other systems bring a programming-by-demonstration interface to the web, such as Web Macros [70], WebVCR [21], Instructible Information Agents [26], Creo [41], PLOW [52], and Turquoise [60]. These systems provide convenient demonstration-based interfaces that allow even non-technically savvy users to create scripts by recording their actions and page modifications and replaying them upon subsequent visits to a web page. This work generally relies on visual interfaces and we will need to be properly adapted to both the accessibility domain and to non-visual access.

Our Accessmonkey system presents a common scripting framework for web users, web developers and web researchers as a first step at developing a platform on which improvements to accessibility can be implemented by both web users and developers [31]. By writing scripts that adhere to the requirements of the framework, members of these group can collaboratively improve the accessibility of the web and other users can take advantage of these improvements on a variety of platforms, running a variety of software. We plan to make scripts created with this system also available to web developers in a way that they can easily incorporate.

### 5.3.2 Making Unavailable Content Accessible

**Hypothesis:** Blind web users will browse more efficiently when information that is currently inaccessible is made accessible.

The above methods may enable blind web users to efficiently access much web information, but some content will be more difficult for blind web users to independently improve in this manner. Examples of such content include web images and other graphical content. To address this problem, we plan to continue developing our WebInSight system in order to make the content contained within images lacking alternative text accessible to blind users. We plan to evaluate this system with users and refine it based on their input. As part of this evaluation, we will define a formal definition of what we mean by "efficiently" based on time and perceived user frustration in order to test our hypothesis. We also plan to further our work in using the technology to assist web developers in assigning alternative text for their images through helpful suggestions. As part of this direction, we will create a web version of WebInSight that web developers can use to test the accessibility of their web images and receive assistance in improving them.

We will also explore methods for making dynamic changes to web pages available to blind web users. Screen readers do not currently convey changes to the DOM of the page directly to web users, largely because it is difficult to interpret when a user should be alerted of a change and when they should not. For instance, while a user may want to know when a new email has arrived or new results have been dynamically added to their results page, they probably don't need to know every time a dynamic clock ticks by a second, again. In this research, we will explore simple methods of detecting such changes and of alerting users when the change is unusual. This research will complement the work described above by enabling blind web users to independently detect such dynamic changes and decide what should happen upon future occurrences. For instance, our tool could alert blind users when a menu has dynamically appeared on a web page, such as in the example in Figure 4, and assist blind web users in making the menu available to them. This tool could also assist users in determining which actions cause dynamic changes to the page.

### 5.4 Timeline

I will follow the general timeline below in order to complete the steps required for my dissertation. The two main thrusts of my dissertation will be interleaved and I will take the insights from each to drive the other. I plan to graduate in Winter 2009, which is approximately 21 months from the date of my General Exam.

- **Months 0-3**
  1. Conduct first stage of user studies investigating how blind web users browse the web.
  2. Release web version of WebInSight for developers and web users.
- **Months 3-9**
  1. Release online repository for scripts intended to improve web accessibility.
  2. Conduct second stage of user studies investigating how blind web users browse the web.
  3. Develop initial version of tool for integrating available accessibility scripts as users browse the web.
- **Months 9-15**
  1. Evaluate tool for integrating available accessibility scripts as users browse the web.
  2. Develop tools that enable blind web users to independently create and share scripts using an intuitive interface.
  3. Evaluate tools that enable blind web users to independently create and share scripts.
- **Months 15-21**
  1. Finish remaining research.
  2. Prepare dissertation and defend.

## 6 Conclusion

The web is a vital platform for conveying information that is currently either inaccessible or not efficiently accessible to the large number of people who are blind. In this paper, we proposed novel user studies that will help better understand the problems facing blind web users. In contrast to previous work, these studies will combine remote observation of blind web users with user annotations of their experiences. The current approaches to achieving accessibility rest largely on convincing web developers to implement more accessible designs and on developing technology that is capable of creating accessible web content automatically. These approaches seem unlikely to adequately address accessibility problems in the near future, so we have proposed a new approach that will enable blind web users to influence the accessibility of web content independently in a way that can be shared with other web users and developers. We believe that by directly involving the users most affected by web accessibility that we can begin to break down the impasse that has stagnated progress in improving web accessibility over the past decade. Through a better understanding of the difficulties faced and by enabling blind web users to independently create and share accessible web content, we hope to create a more accessible web experience.

# References Cited


[1] A-prompt. Adaptive Technology Resource Centre (ATRC) and the TRACE Center at the University of Wisconsin. http://www.aprompt.ca/.

[2] Amazon mechanical turk. http://www.mturk.com/.

[3] Greasemonkey firefox extension. http://greasemonkey.mozdev.org/.

[4] IBM alphawork's adesigner. http://www.alphaworks.ibm.com/tech/adesigner.

[5] Voice extensible markup language (VoiceXML) 2.1. http://www.w3.org/TR/voicexml21/.

[6] Watchfire bobby. http://www.watchfire.com/products/webxm/bobby.aspx.

[7] Web content accessibility guidelines 1.0 (WCAG 1.0). World Wide Web Consortium, 1999.

[8] GW micro window-eyes, 2006. http://www.gwmicro.com/Window-Eyes/.

[9] IBM home page reader, 2006. http://www-3.ibm.com/able/solution_offerings/hpr.html.

[10] Jaws 8.0 for windows. Freedom Scientific, 2006. http://www.freedomscientific.com.

[11] Lift. UsableNet, 2006. http://www.usablenet.com/.

[12] Linux screen reader (lsr), 2006. http://live.gnome.org/LSR.

[13] National federation of the blind vs. target corporation. U.S. District Court: Northern District of California, 2006. No. C 06-01802 MHP.

[14] Roadmap for accessible rich internet applications (WAI-ARIA roadmap). World Wide Web Consortium, 2006. http://www.w3.org/TR/WCAG20/.

[15] Roles for accessible rich internet applications (wai-aria roles). W3C, September 2006. http://www.w3.org/TR/aria-role/.

[16] States and properties module for accessible rich internet applications (wai-aria states and properties). W3C, September 2006. http://www.w3.org/TR/aria-state/.

[17] Thunder screenreader, 2006. http://www.screenreader.net/.

[18] W3C markup validation service v0.7.4, 2006. http://validator.w3.org/.

[19] Web content accessibility guidelines 2.0 (WCAG 2.0). World Wide Web Consortium, 2006. http://www.w3.org/TR/WCAG20/.

[20] P. Andronico, M. Buzzi, B. Leporini, and C. Castillo. Testing google interfaces modified for the blind. In *Proceedings of the 15th International Conference on World Wide Web (WWW '06)*, pages 873–874, New York, NY, 2006. ACM Press.

[21] V. Anupam, J. Freire, B. Kumar, and D. Lieuwen. Automating web navigation with webvcr. In *Proceedings of the 9th International World Wide Web conference on Computer Networks: the journal of computer and telecommuncations networking*, pages 503–517, Amsterdam, The Netherlands, 2000.

[22] R. Atterer. Logging usage of ajax applications with the usaproxy http proxy. In *Proceedings of the WWW 2006 Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection*, 2006.

[23] R. Atterer, M. Wnuk, and A. Schmidt. Knowing the user's every move - user activity tracking for website usability evaluation and implicit interaction. In *In Proceedings of the 15th International Conference on World Wide Web (WWW '06)*, pages 203–212, New York, NY, 2006. ACM Press.

[24] K. Barnard, P. Duygulu, D. Forsyth, N. De Freitas, D. Blei, and M. Jordan. Matching words and pictures. *Journal of Machine Learning Research: Special Issue on Text and Images*, 2003.

[25] K. Barnard and D. Forsyth. Learning the semantics of words and pictures. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 408–415, 2001.

[26] M. Bauer, D. Dengler, and G. Paul. Instructible information agents for web mining. In *Proceedings of the 5th international conference on Intelligent user interfaces (IUI '00)*, pages 21–28, New York, New York, USA, 2000. ACM Press.

[27] L. Bent and G. Voelker. Whole page performance. In *Proceedings of the 7th annual Web Caching Workshop*, June 2002.

[28] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[29] J. P. Bigham. Increasing web accessibility by automatically judging alternative text quality. In *Proceedings of the 12th international conference on Intelligent user interfaces (IUI '07)*, New York, NY, USA, 2007. ACM Press.

[30] J. P. Bigham, R. S. Kaminsky, R. E. Ladner, O. M. Danielsson, and G. L. Hempton. Webinsight: Making web images accessible. In *Proceedings of Eighth International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '06)*, October 2006.

[31] J. P. Bigham and R. E. Ladner. Accessmonkey: A collaborative scripting framework for web users and developers. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility (W4A '07)*, 2007.

[32] A. B. Brush, M. Ames, and J. Davis. A comparison of synchronous remote and local usability studies for an expert interface. In *CHI '04 extended abstracts on Human factors in computing systems*, pages 1179–1182, New York, NY, USA, 2004. ACM Press.

[33] J. Clark. Readers guide to sydney olympics accessibility complaint, 2001. http://www.contenu.nu/socog.html.

[34] J. Clark. *Building Accessible Websites*. New Riders Publishing, Indianapolis, IL, USA, 2003.

[35] M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. In *Proceedings of the 6th international conference on Intelligent user interfaces (IUI '01)*, pages 33–40, New York, NY, USA, 2001. ACM Press.

[36] D. R. Commission. The web: Access and inclusion for disabled people. The Stationary Office, 2004.

[37] K. P. Coyne and J. Nielsen. Beyond alt text: Making the web easy t use for users with disabilities, 2001.

[38] T. C. Craven. Some features of alt text associated with images in web pages. *Information Research*, 11, 2006.

[39] O. De Troyer and C. Leune. Wsdm: A user-centered design method for web sites. In *Proceedings of the Seventh International World Wide Web Conference*, pages 85–94, 1998.

[40] S. Dill, N. Eiron, D. Gibson, D. Gruhl, and R. Guha. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the International Conference on the World Wide Web (WWW '03)*, 2003.

[41] A. Faaborg and H. Lieberman. A goal-oriented web browser. In *Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI '06)*, pages 751–760, Montreal, Quebec, Canada, 2006.

[42] B. Gibson and R. Schwerdtfeger. Dhtml accessibility: solving the javascript accessibility problem. In *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '05)*, pages 202–203, New York, NY, USA, 2005. ACM Press.

[43] J. Goecks and J. Shavlik. Learning users' interests by unobtrusively observing their normal behavior. In *Proceedings of the 5th international conference on Intelligent user interfaces (IUI '00)*, pages 129–132, New York, NY, USA, 2000. ACM Press.

[44] M. Gonzalez. Automatic data-gathering agents for remote navigability testing. *IEEE Software*, 19(6):78–85, 2002.

[45] M. Gonzalez, M. Gonzalez, C. Rivera, I. Pintado, and A. Vidau. Testing web navigation for all: An agent-based approach. In *Proceedings of 10th International Conference on Computers Helping People with Special Needs (ICCHP '06)*, volume 4061 of *Lecture Notes in Computer Science*, pages 223–228, Berlin, Germany, 2006. Springer.

[46] S. Hackett, B. Parmanto, and X. Zeng. Accessibility of internet websites through time. In *Proceedings of the 6th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '04)*, pages 32–39, New York, NY, USA, 2004.

[47] S. Harper, C. Goble, R. Stevens, and Y. Yesilada. Middleware to expand context and preview in hypertext. In *Proceedings of the 6th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '04)*, pages 63–70, New York, NY, USA, 2004. ACM Press.

[48] S. Harper and N. Patel. Gist summaries for visually impaired surfers. In *Proceedings of the 7th international ACM SIGACCESS conference on Computers and Accessibility (ASSETS '05)*, pages 90–97, New York, NY, USA, 2005. ACM Press.

[49] G. Iaccarino, D. Malandrino, and V. Scarano. Efficient edge-services for colorblind users. In *Proceedings of the 15th International Conference on World Wide Web (WWW '06)*, pages 919–920, New York, NY, 2006. ACM Press.

[50] G. Iaccarino, D. Malandrino, and V. Scarano. Personalizable edge services for web accessibility. In *Proceedings of the 2006 international cross-disciplinary workshop on Web accessibility (W4A '06)*, pages 23–32, New York, NY, USA, 2006. ACM Press.

[51] M. Y. Ivory. *Automated Web Site Evaluation Reseachers' and Practitioners' Perspectives*. Kluwer Academic Publishers, 2003.

[52] H. Jung, J. Allen, N. Chambers, L. Galescu, M. Swift, and W. Taysom. One-shot procedure learning from instruction and observation. In *Proceedings of the International FLAIRS Conference: Special Track on Natural Language and Knowledge Representation*.

[53] B. Kelly, D. Sloan, L. Phipps, H. Petrie, and F. Hamilton. Forcing standardization or accommodating diversity?: a framework for applying the wcag in the real world. In *Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A '05)*, pages 46–54, New York, NY, USA, 2005. ACM Press.

[54] M. Koletsou and G. Voelker. The medusa proxy: A tool for exploring user-perceived web performance. In *Proceedings of the 6th annual Web Caching Workshop*, June 2001.

[55] J. Lazar, J. Feng, and A. Allen. Determining the impact of computer frustration on the mood of blind users browsing the web. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '06)*, pages 149–156, New York, NY, USA, 2006. ACM Press.

[56] S. Lu and C. L. Tan. Camera document restoration for ocr. In *Proceedings of First International Workshop on Camera-Based Document Analysis and Recognition (CBDAR '05)*, 2005.

[57] J. Mahmud, Y. Borordin, and I. Ramakrishnan. Csurf: A context-driven non-visual web-browser. In *Proceedings of the International Conference on the World Wide Web (WWW '07)*.

[58] J. Mankoff, H. Fait, and T. Tran. Is your web page accessible?: a comparative study of methods for assessing web page accessibility for the blind. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '05)*, pages 41–50, New York, NY, USA, 2005. ACM Press.

[59] M. Maurer. The campaign to change what it means to be blind. *Braille Monitor*, 43(2), February 2000.

[60] R. C. Miller and B. Myers. Creating dynamic world wide web pages by demonstration, 1997.

[61] S. Mukherjee, I. Ramakrishnan, and A. Singh. Bootstrapping semantic annotation for content-rich html documents. In *Proceedings of the International Conference on Data Engineering (ICDE '05)*, 2005.

[62] S. Mukherjee, G. Yang, W. Tan, and I. Ramakrishnan. Automatic discovery of semantic structures in html documents. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR '03)*, 2003.

[63] G. K. Myers, R. C. Bolles, Q.-T. Luong, J. A. Herson, and H. Aradhye. Rectification and recognition of text in 3-d scenes. *International Journal on Document Analysis and Recognition*, 7(2-3):147–158, 2005.

[64] H. Petrie, F. Hamilton, and N. King. Tension, what tension?: Website accessibility and visual design. In *Proceedings of the international cross-disciplinary workshop on Web accessibility (W4A '04)*, pages 13–18, New York, NY, USA, 2004. ACM Press.

[65] H. Petrie, F. Hamilton, N. King, and P. Pavan. Remote usability evaluations with disabled people. In *Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI '06)*, pages 1133–1141, New York, NY, USA, 2006. ACM Press.

[66] H. Petrie, C. Harrison, and S. Dev. Describing images on the web: a survey of current practice and prospects for the future. In *Proceedings of Human Computer Interaction International (HCII '05)*, July 2005.

[67] M. Pilgrim, editor. *Greasemonkey Hacks: Tips & Tools for Remixing the Web with Firefox*. O'Reilly Media, 2005.

[68] P. Plessers, S. Casteleyn, Y. Yesilada, O. D. Troyer, R. Stevens, S. Harper, and C. Goble. Accessibility: a web engineering approach. In *Proceedings of the 14th international conference on World Wide Web (WWW '05)*, pages 353–362, New York, NY, USA, 2005. ACM Press.

[69] I. Ramakrishnan, A. Stent, and G. Yang. Hearsay: Enabling audio browsing on hypertext content. In *Proceedings of the 13th International Conference on the World Wide Web (WWW '04)*, 2004.

[70] A. Safonov. Web macros by example: users managing the www of applications. In *CHI '99 extended abstracts on Human factors in computing systems (CHI '99)*, pages 71–72, New York, NY, USA, 1999. ACM Press.

[71] S. Saito, H. Takagi, and C. Asakawa. Transforming flash to xml for accessibility evaluations. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '06)*, pages 157–164, New York, NY, USA, 2006. ACM Press.

[72] R. Schwerdtfeger and B. Gibson. Dhtml accessibility - fixing the javascript accessibility problem. In *Proceedings of CSUN's 20th Annual International Conference Technology and Persons with Disabilities*, 2005.

[73] D. Sloan, P. Gregor, M. Rowan, and P. Booth. Accessible accessibility. In *Proceedings on the 2000 conference on Universal Usability (CUU '00)*, pages 96–101, New York, NY, USA, 2000. ACM Press.

[74] C. H. L. T. Kanungo and R. Bradford. What fraction of images on the web contain text? In *Proceedings of the International Workshop on Web Document Analysis*, September 2001.

[75] J. Thatcher, P. Bohman, M. Burks, S. L. Henry, B. Regan, S. Swierenga, M. D. Urban, and C. D. Waddell. *Constructing Accessible Web Sites*. glasshaus Ltd., Birmingham, UK, 2002.

[76] S. R. Turner. Playtpus firefox extension, 2006. http://platypus.mozdev.org/.

[77] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI '04)*, April 2004.

[78] L. von Ahn, S. Ginosar, M. Kedia, R. Liu, and M. Blum. Improving accessibility of the web with a computer game. In *Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI '06)*, pages 79–82, New York, NY, USA, 2006. ACM Press.

[79] M. Vorburger. Altifier: Web accessibility enhancement tool, 1999.

[80] Y. Yesilada, S. Harper, C. Goble, and R. Stevens. Screen readers cannot see (ontology based semantic annotation for visually impaired web travellers). In *Proceedings of the fourth International Conference on Web Engineering (ICWE '04)*, pages 445–458, 2004.

[81] Y. Yesilada, R. Stevens, and C. Goble. A foundation for tool based mobility support for visually impaired web users. In *Proceedings of the 12th international conference on World Wide Web (WWW '03)*, pages 422–430, New York, NY, USA, 2003. ACM Press.