Behaviour & Information Technology, Vol. 24, No. 3, May–June 2005, 231–246

Taylor & Francis
Taylor & Francis Group

Achieving a more usable World Wide Web

V. L. HANSON* AND J. T. RICHARDS

IBM T. J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532, USA

Over the last few years, we have built and tested two systems designed to make Web content more accessible for people with limited vision and dexterity. The first system, based on content transcoding via a proxy server, possessed several attractive features but proved to be unacceptably complex, error prone, and slow. The second system, based on client-side transformations, worked well enough to be broadly deployed. We report here on lessons learned and on the current state of the research effort. We review the two systems, discuss their strengths and weaknesses, and examine how the second system is being used.

Keywords: World Wide Web; Accessibility; Document transformation

1. Introduction

The Web continues to evolve and Web content has, if anything, become less accessible over time. Moving from its origins as simple hypertext, the Web has become populated with pages having eye-catching designs and rich, multi-model content. Content has become increasingly dynamic, both in terms of server-side content generation and client-side script interpretation. Complex interactions such as online shopping and banking have moved to a position of prominence rivaling mere content browsing. The technical complexity underlying these dramatic changes in the Web has served to undermine its accessibility. Our initial goal was a Web Accessibility Service that would adapt to these dramatic changes.

Our work was motivated by a desire to make the Web more accessible for older adults. Specific physical and cognitive changes occur with age. Census figures show that by age 65, nearly half of us can expect to experience some disability; one quarter of us experiencing severe disability (McNeil 1997, Grundy *et al.* 1999). The term 'dynamic diversity' has been coined to describe the abilities of older adults (Gregor and Newell 2001). This phrase suitably encapsulates the dual aspects of designing for this population. They are not only diverse in their needs, but their needs can fluctuate from day to day, and even within a session, due to fatigue or a variety of other factors. Many will have a complex combination of needs.

We began our research by interviewing instructors at centres that held classes for older adults who wanted to learn to use the Internet. We asked them about the types of problems encountered by their students. These interviews confirmed computer difficulties generally reported in the literature, but also brought into focus specific problems with Web usage (Hanson *et al.* 2001).

Vision impairments provide a common source of difficulty for older adults when using the Web. Reported difficulties included small font sizes, font colours that make reading difficult (particularly in the context of certain background colours), and background images on Web pages that decrease legibility. Given the fact that acuity, contrast discriminations and colour perception are all reduced when we age, it is easy to understand the underlying reasons for this difficulty (Faye and Stappenbeck 2000a, 2000b). The ability to make some changes to content presentation is built into browsers and operating systems. The instructors we interviewed were all aware that some font enlargement was available through the browser, although few were aware of browser options for changing colours, fonts, or creating style sheets. Similarly, few were aware of operating system options for screen magnification. Thus, while the instructors often informed students that they could increase text size using the browser, other options were not presented. The fact that instructors were often unaware of these options is not surprising given that

*Corresponding author. Email: vlh@watson.ibm.com

Behaviour & Information Technology
ISSN 0144-929X print/ISSN 1362-3001 online
http://www.tandf.co.uk/journals
DOI: 10.1080/01449290412331327465

these features are buried in complex series of menus and dialog boxes.

Some of the computer centres used large computer monitors to help offset problems with the reading of small font sizes, although most could not afford the large monitors due to a combination of cost and space issues. Bifocals worn by many of the older adults provided some help with reading Web pages, but those who spent much time at the computer tended to develop stiff necks from tilting their heads at an awkward angle to read through the bifocals. This is exacerbated for people with particularly low vision, as with macular degeneration, who sit very close to computer screens. These users often develop shoulder and back pain from their position at the computer.

Using the mouse and keyboard were also reported to be difficult for older adult Web users. Some of this is due to conceptual problems in understanding the mouse or due to inexperience with a typewriter or keyboard (Czaja and Lee 2001, 2003). Other difficulties, however, are due to arthritis, tremors, or other physical problems that make keyboard entry and mouse manipulation difficult. Some modifications such as mouse and keyboard settings are built into computer operating systems. Such solutions, however, do require that users be aware of these options. Moreover, for users who have disabilities that impact their hand movement, simply trying to set the options can be extremely difficult (Trewin 2004). Although instructors at the computer centres were often aware of special keyboards and special mouses to ease difficulties with double-clicking and scrolling, none of the centres we contacted had invested in the extra cost of providing these devices. In fact, however, these special devices provide only partial solutions to the problems experienced by older adults. For example, accurate mouse usage interacts with visual ability, such that people with poor vision will have mouse difficulties even in the absence of dexterity problems (Jacko *et al*. 2000).

Although older adults often experience difficulties when starting to use computers due to the unfamiliar domain (Czaja and Lee 2001, 2003), they are avid Web users (BBC News 2002, Morrell 2002, Fox 2004). Specific page layouts create navigational difficulties, however, and visual clutter and irrelevant information are difficult for older adults to understand and navigate. Animations create distractions. In some cases, memory limitations brought about by aging can make it difficult to form the necessary mental models of the Web needed for successful browsing (Zajicek 2001).

Despite the fact that hearing problems are prevalent among older adults, hearing impairment was not considered to be a barrier to web access in our interviews, at least at this time.

In summary, changing user needs related to failing vision, limited dexterity, and difficulties dealing with the cognitive complexities of the Web characterize older adults. These were our software requirements as the project began. The software was not designed to address issues of blindness, nor was it designed to address limited hearing.

2. Proxy architecture

In previous papers, we described a prototype service about to be tested in the field (Hanson *et al*. 2001, Fairweather *et al*. 2002). The defining feature of this service was that Web page transformations were to be made by an intermediary proxy server. We realized that this proxy was going to be complex due to its multiple roles as document source analyzer and transformational engine, but we were reasonably confident that clever programming and available processing power would allow us to create a service that could rapidly transform current and evolving Web content.

Our effort has not been the only attempt to use a server intermediary to change Web page presentation for the purposes of accessibility. As one example, PDF files can be converted into HTML online.[1] As another approach, Brown and Robinson (2001) described a Web Access Gateway that allows users to make some changes to Web page presentation by passing URL requests through the server gateway. The gateway loads pages, transforms them, and sends the reformatted Web page to the user. In Japan, researchers examined the feasibility of using annotation to mark-up Web pages for accessibility (Asakawa and Takagi 2000, Takagi *et al*. 2002). These annotations not only supplied missing accessibility features, but also provided for page simplification in a manner that would be most efficiently formatted for people using a screen reader.

Our project goal was to provide more content transformations than other intermediaries, not limited to selected sites. For this purpose, our initial architecture placed a proxy server between our users' client machines and the open Web. The transformations would be performed on the proxy, with the transformed pages sent to the client. The primary advantages of this architecture were twofold. First, it would free content providers from having to modify their content. As we have discussed elsewhere (Richards and Hanson 2004), such modifications will likely remain too expensive to perform even with technological help. Second, it would free users from having to install and maintain software on their client machines. Nothing more than setting the browser's HTTP proxy would be required. Using a browser set to go through this HTTP proxy server, users access web pages the same as they would normally. Moreover, all browsers should, in principle, be able to display the transformed content since the transforms are done upstream and the browser would simply go about its

[1] http://www.adobe.com/products/acrobat/access_simple_form.html

work as usual. This would eliminate the need for costly, browser-specific coding.

Our proxy software was built on WebSphere Transcoding Publisher (http://www.306.ibm.com/software/pervasive/transcoding_publisher/). This bridges information across multiple devices and formats and is commonly used to make web content available on a handheld or other small device. Here we wanted to extend the function to include transformations for accessibility. In this architecture, the proxy intercepts the returned results and reformats the content. Based on the user's specifications for how they want web pages presented, the appropriate transformations are applied to the retrieved document and the result is then sent back to the user.

We implemented this transcoding proxy server and tested it with the SeniorNet organization (http://www.seniornet.org). SeniorNet provides not only an online community for older adults, but also supports computer centres where older adults can take classes. For the testing of our prototype, we worked with older adults who were instructors and students in Internet classes at SeniorNet Learning Centres.

The testing revealed that the proxy architecture was not feasible. While the people who tested it were positive about the project goals, their use of the system was severely limited by the following problems.

Many difficulties stemmed from the proxy's necessary complexity. In order to correctly render Web pages, the proxy had to become as capable of interpreting source content as a full-function browser. It had to correctly interpret JavaScript, cascading style sheets, markup languages, and a large and growing number of third-party plug-ins. A related problem was that the proxy needed to interpret source content as it would be interpreted by a particular browser. We quickly learned that most Web content does not fully conform to published standards. In order to 'get the job done', browsers have become quite forgiving of these errors, interpreting malformed HTML in a meaningful way. We found that we would have to discover the rules for dealing with malformed HTML in order to meaningfully render the large number pages that contain HTML errors. Given the difficulty of doing all this, it turned out that a large number of Web pages were incorrectly rendered on the users' machines. We found ourselves trying to correct errors on a page-by-page basis. Given the vast number of pages available on the Internet, it became clear that this approach was not viable. Other attempts to do intermediary-based transcoding have annotated Web pages for presentation (Asakawa and Takagi 2000, Takagi *et al*. 2002), which has the potential to overcome incorrect renderings. Annotation can never address the problem of adapting the full Web, however.

Another problem was associated with secure sites. An intermediary cannot see and modify the content flowing through a Secure Sockets Layer (SSL) connection unless that connection is broken open by a pair of SSL sockets at the proxy itself (one allowing the proxy to spoof as a client to the web server, the other allowing the proxy to spoof as the server to the requesting client). Technically it is possible to thus modify the encrypted content on the proxy. Performing such an operation on a proxy server, however, violates the end-to-end security expected of secure connections and raises a worrisome security alert (as it should) to the browser's user. It also forces the provider of the proxy service to assume responsibility for keeping the momentarily unencrypted content secure — something that is both difficult and expensive.

Simply enabling users' access to the proxy also proved to be problematic. In order to set a browser to go through a proxy, it must be possible to change the HTTP proxy setting on the browser. This is often not possible for a variety of reasons. In some cases, users were accessing the Web through modified browsers in which they were not allowed to change a proxy. More commonly, they were accessing the Web through a browser in which a proxy had already been set. Since there is presently no way to cascade proxies, this made it impossible for some users to even access our service. This was true, for example, of users accessing the Web from organizations that went through a proxy-based firewall as well as individual America Online (AOL) users.

The proxy server also created questions in relation to copyright since the server was making changes to Web pages and distributing these changed pages. As one way to address this issue, we adhered to the 'no-transform directive' (http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.9.5). This directive states that pages which include the no-transform header may not have their content modified by the proxy.

Lastly, the speed of the proxy-based service was unacceptably slow. In part, this was due to the complexity of the interpretation task itself. Presumably, this could be improved through increased processing power and ongoing optimization of the proxy's transformational software. The delays associated with creating the additional proxy to server socket connection are harder to eliminate. In principle, a large enough server would be able to cache the results of previous transforms. But since the transformations are dependent on a user's particular settings, and since much page content is dynamic, the probability of cache hits is quite low. Moreover, it is likely that a proxy serving a transformed page from cache is violating the rights of the content owner.

These problems forced us to conclude that the proxy approach to creating a Web Accessibility Service — despite its clear appeal — was not viable. In combination, the problems translated into a large number of pages being incorrectly rendered and a large proportion of users being unable to even access the service. Increasing the accuracy of

our content transformations would have been an extremely labour intensive effort, requiring testing of a vast (and probably never ending) assortment of Web pages. Even if the prototype had proved more successful, a production version of the system would have required tremendous server capacity based on the slowness of our prototype even under light load. People familiar with the Internet are used to a relatively high speed of page presentation. The extra time introduced by the proxy's additional socket connection and the transcoding of the source produced perceptible and unacceptable delays.

Our testing of the prototype caused us to abandon the proxy server approach and fundamentally reconsider our architecture. Fortunately, all the problems discussed above — errors in transforming source, necessity for browser-specific source transformations, secure connections not exposing the source to transformation, proxy setting difficulties, copyright issues, and poor performance — could be addressed by moving the point of transformation from the proxy to the client.

3. A revised client–server architecture

Our new architecture was influenced both by system considerations and by the feedback from our testers. The resulting objectives and requirements can be summarized as follows.

The central design principal for this new architecture was to let the browser do as much of the work as possible. We would not try to second-guess what the browser was going to do or impede it from doing its normal work. We would seek to inject our accessibility transformations into this work with minimal disruption.

Most of our users had computers that they used at home in addition to the computers that they used in class. It stood to reason that our users would continue to want to have the same Web browsing experience available in both locations even though the transforms were now going to be applied on different machines.

We had found in our earlier requirements gathering that users tend to prefer a standard browser with the accessibility transformations added rather than a specialized browser offering only a limited set of features (which would also tend to mark them as being disabled). Thus, we decided to make as few outwardly visible changes to the browser as possible.

Older adults experience not only a variety of disabilities; many experience a combination of disabilities. As was true with the first system, the new system needed to be flexible, allowing individual users to select any combination of transformations from a large set of possible transformations.

Accessibility settings can provide clues about underlying medical conditions. As such, they are sensitive and must be protected from inadvertent disclosure.

People desire assistance of a form not addressed by current accessibility guidelines. A prime example of this is the desire of visually impaired users to use the vision they *do* possess rather than use the speech output of a screen reader.

It is not possible to know at first encounter what a person's required transformations are. Thus, our solution needed to address the difficult problem of how to make the selection of transformations itself accessible.

Finally, our new system, like our original system, would need to support the viewing and transformation of any Web content. People do not want to limit their browsing to only a portion of the Web. Nor do they want their transformations to be applied to only a particular site or sites. This remains our most difficult challenge since Web content is increasingly dynamic, scripted, and laden with multi-media.

Our new version, called the Web Adaptation Technology, was designed to addresses all of the above objectives and requirements. In the new architecture, Web content is no longer transformed on its way through a proxy server. Instead, as shown in figure 1, all content transformations and input adaptations are now performed on the client machine using a combination of approaches. Many of the transformations are applied to the browser's Document Object Model (DOM). The DOM provides a model of an HTML document and allows the model to be changed in order to alter the page presentation. Other changes are affected by taking advantage of the browser's and operating system's built-in features or by the creation of client-side style sheets. No changes are made to the page source. A server is used only for storing user preferences.

One set of content transformations was designed to address the visual presentation of pages. Certain changes such as font enlargement, font style (sans serif), increased inter-letter and inter-line spacing, and enhanced colour contrast can increase legibility for this population (Jacko *et al*. 2000). For users with greater degrees of visual problems, we provide additional changes, augmenting the text with speech output and providing options for very large text displayed in a banner at the top of the window, enlargement of browser controls, and whole-page magnification.

The necessity for client-side code in our new design also gave us a distribution channel for software that could *only* be run on the client. A prime example of this was software that analyzed user keystrokes, building a model of keystroke filtering that would allow us to attenuate the effects of motor problems and improve the accuracy of typed input. Since some Web activities, such as writing e-mail, filling in forms, or completing login and registration information, require the use of a keyboard, we sought to make the setting of these rather complex filters much easier than is currently possible (Trewin 2004).

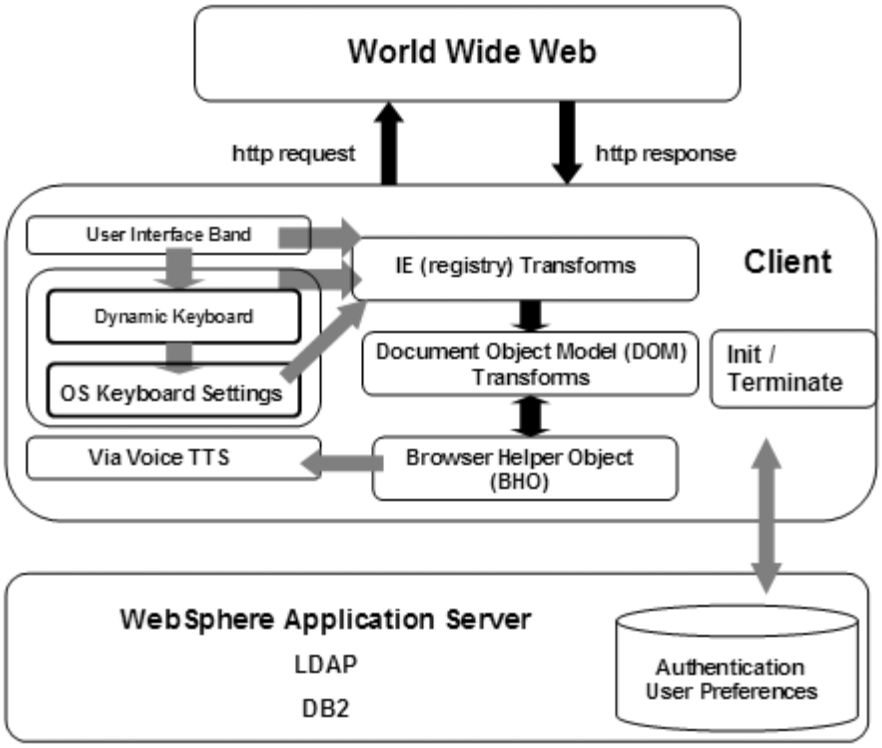

Figure 1. Revised architecture with transformations and adaptations performed on the client.

3.1 *Transformations applied to the DOM*

Some of our visual changes to Web pages are accomplished through manipulation of the DOM produced by the browser itself. Our Internet Explorer® implementation uses a Browser Helper Object (BHO) written in Java. The BHO gives us programmatic access to the DOM *before* it is rendered by the browser. Available DOM APIs support both document content manipulation and the setting of handlers for user events. As a result of this capability, we do not need to be concerned with how the HTML is to be interpreted. We simply deal with the content as an abstract tree that represents what is to be rendered. Dynamic scripts have already been applied at this point and decisions have been made about what do with malformed source.

The following transformations involve DOM manipulations:

- speak text;
- largest text size increase (larger than what is possible with the browser alone);
- banner text;
- changing colours (text, background, and links);
- image enlargement and enhancement;
- page layout (linearization).

Examples of some DOM-based transformations are shown in figures 2–4. Figure 2 shows an example of a page with banner text. The user selects text to be displayed in the banner, either by selecting the whole page or using the mouse to point and hover over a part of the page to select. Upon selection, very large text is displayed in the window one line at a time. The text advances to the next line under user control. When the speak text feature is also activated, the speech is coordinated with the banner display such that only the text in the banner is read aloud and speaking proceeds as the user advances to the next line.

Figure 3 shows an example of large text size. This is larger than is possible using the browser's built-in text size options. While this amount of enlargement is desired by some people, notice how the page content has spilled beyond the right edge of the browser window. Reading this page would require horizontal scrolling. Given that horizontal scrolling is difficult for everyone (and especially difficult for those with motor disabilities) it is quite possible that our accessibility transformation has actually made the page harder to use. To ameliorate this problem, our software also provides the ability to linearize the page content, changing a multiple column layout to a single column layout. Figure 4 shows the same content, at the same text size enlargement, rendered as a single column. This can now be read with just vertical scrolling (if desired,

Figure 2. Web page with the banner text feature.

using just the page down key, avoiding scroll bars altogether).

3.2 *Transformations affected through browser and operating system features*

In other parts of our implementation, we exploited features built into the browser (e.g. text size increases) and operating system (e.g. StickyKeys, FilterKeys, and the size of browser controls) to make Web browsing easier. Sometimes this involves a straightforward selection of these features through our new user interface. For example, StickyKeys is turned on through a button click in the keyboard panel of the settings interface (the button is labeled 'one hand' since StickyKeys is useful for people who type with one hand).

A user's needs for other typing adjustments are inferred from short samples of a user's typing (Richards *et al*. 2003, Trewin 2004). Thus, as a user types, our software makes a determination of optimal keyboard settings referencing typing models tested with users having a variety of keyboarding difficulties due to limited hand coordination (Trewin and Pain 1998). The input is analyzed, for example, as to whether key repeat delays or FilterKeys would improve typing accuracy (for more details, see Trewin 2004). Based on this analysis, the keyboard accessibility parameters are automatically adjusted to maximize typing accuracy for an individual user.

The following visual transformations and keyboard adaptations exploit browser or operating systems features:

- magnification (of whole pages, affected through a generated style sheet);
- text size increases (up to the limits of what the browser directly allows);
- text style;
- line spacing (affected through a generated style sheet);
- letter spacing (affected through a generated style sheet);

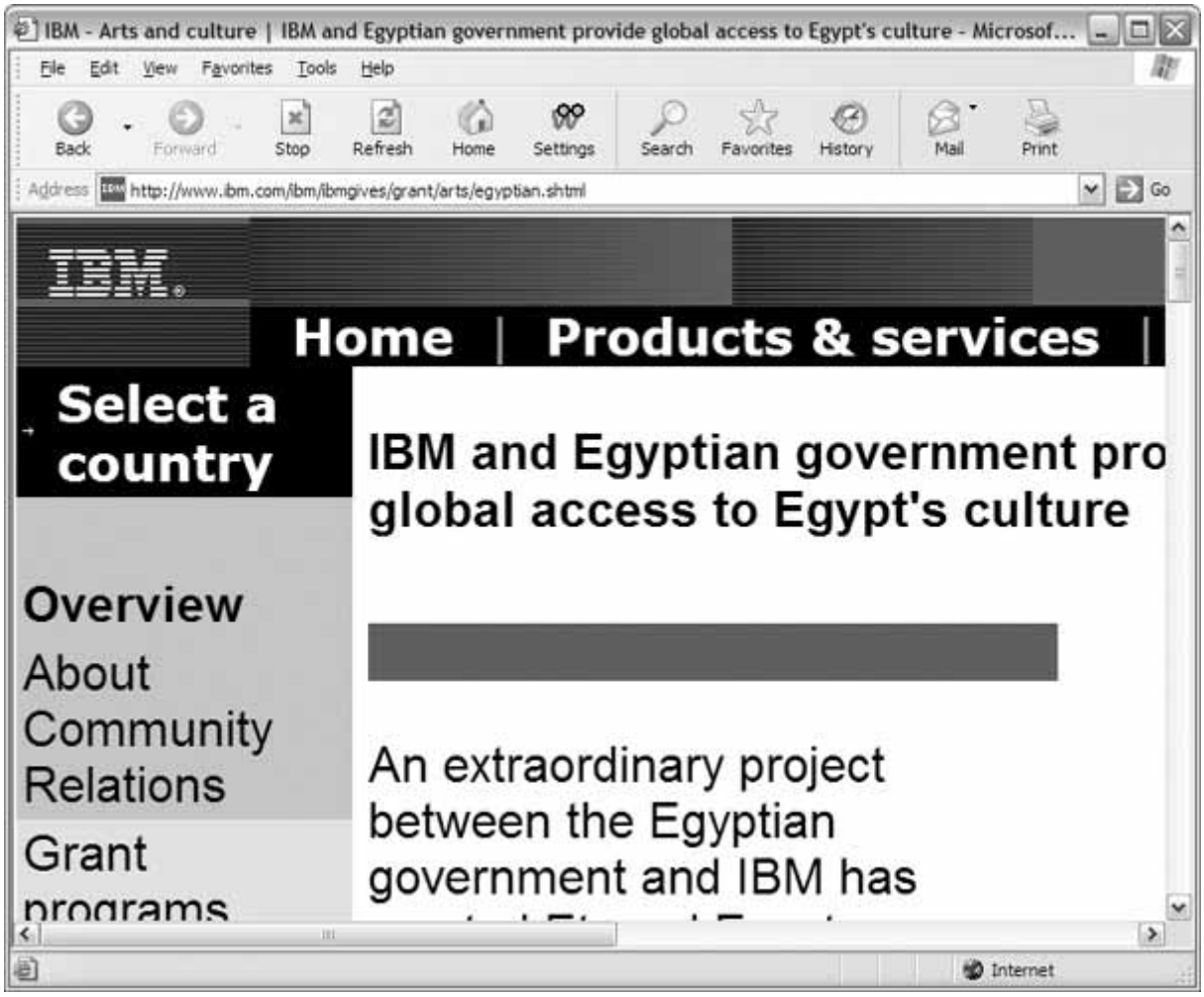

Figure 3. A Web page with text size increased.

- hide (images, animations, background);
- large browser (pointers, controls);
- keyboard (one hand, key clicks, mouse keys, plus typing adjustments).

3.3 *Setting and saving user preferences*

As shown in figure 1, user preferences are stored in a database on a server. This particular design feature arose from the needs of our users. Many of these people used the Internet in a classroom or another shared-computer environment such as a community centre. Storing user preferences on a server allows multiple users to share one machine without having the preferences from one user impact the session of another. It also allows a user to have the same settings consistently applied when they use different machines (either, for example, different machines in a classroom or machines in a classroom and at home).

For an individual user, a Web session begins when they log in to the Web Adaptation Technology server. The login screen is designed to be usable by persons with limited vision and keyboarding ability. Specifically, the screen completely fills a 640×480 display, with extremely large text indicating where people should enter their username and password. The typing adaptation software described above is active and with little typing input is capable of making adjustments to help users more accurately enter their information.

Upon successfully logging in, new users can set their preferences. Returning users have their settings preferences retrieved and applied. If they are still happy with those preferences, they use the Web as usual. If they wish, they can modify any settings previously set. At the close of each session, the user's current settings preferences are sent back to the server and saved for the next session.

3.4 *User interface design requirements*

In addition to the system requirements outlined above, there were several goals for the user interface. We needed to carefully attend to these if our software was to be

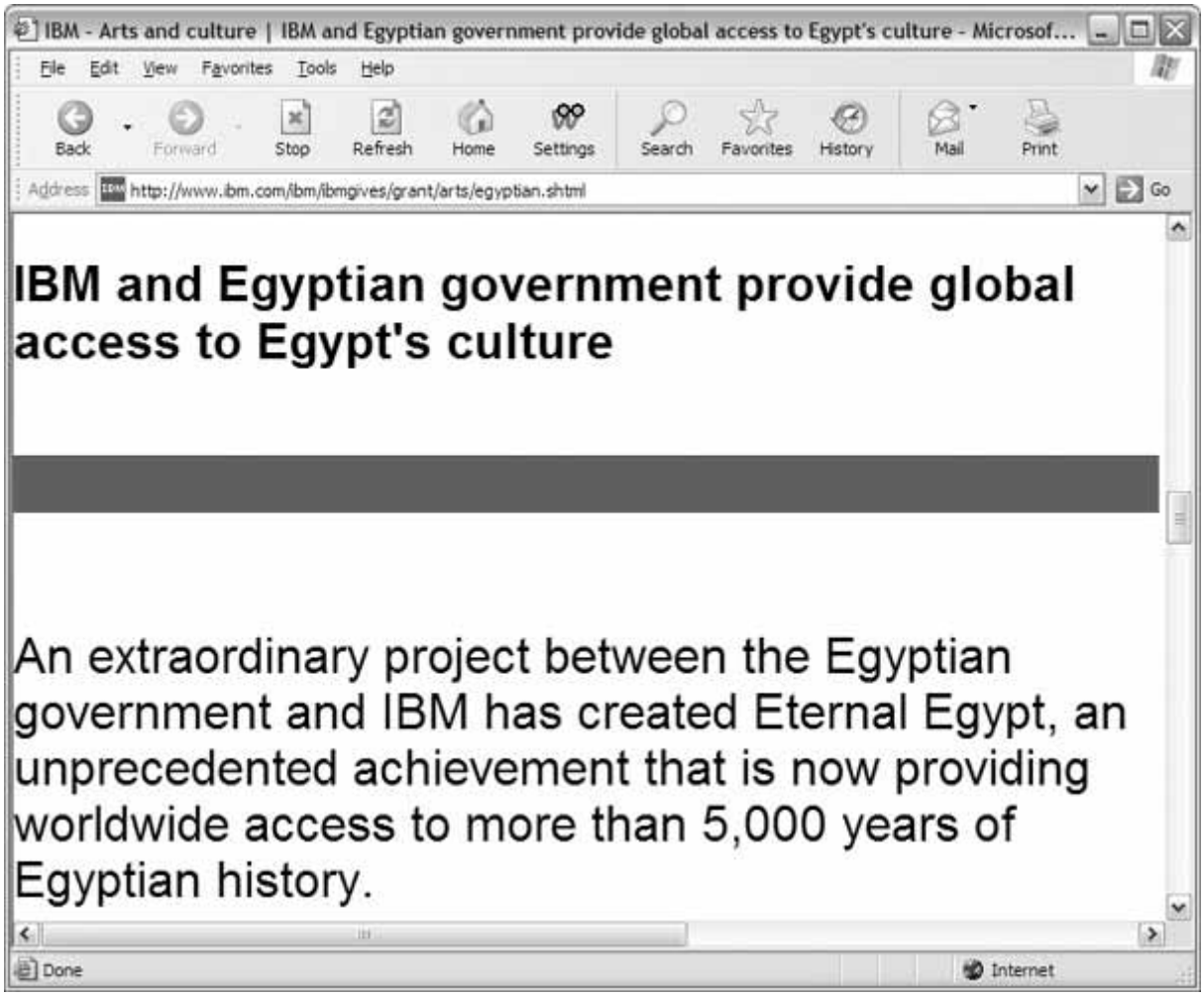

Figure 4. A Web page with the same the size increase as in figure 3, but with page linearization (one column transformation) applied.

successfully used by our target population. Our interface design was primarily shaped by two considerations.

- Many older adults are new to computing. Consequently, the specification of user preferences could not require prior computer experience.
- People are generally unable to map from an abstract characterization of a possible transformation to its actual effect. Our solution therefore needed to make it easy to directly *try out* any transformation or input adaptation on any Web page at any time.

These considerations highlighted the need to make the user interface for setting preferences extremely simple. To this end, we designed an interface in which users cycle through a set of minimalist control panels, one for each type of transformation. The panels are contained in a settings band that appears at the bottom of the browser. To bring up this band, the user selects the 'Settings' button shown in figure 5. Figure 6 shows the panel used to change text style. Some Web pages use novelty or artistically chosen text. Such text can be difficult to read by older adults and persons who have vision limitations (Arditi 1999; see also Center for Medicare Education 2000, National Institute for Aging 2002). To choose to have all Web pages displayed with an easily readable font, the Web Adaptation Technology allows users to select from among the text options shown in figure 6. As soon as one of the 'text' buttons is clicked, the requested change is applied to the current Web page. The result of such a click is shown in figure 7. If the user does not like the results of the change, they can easily try another font or switch back to the original font.

Access to each of our transformations and adaptations (except some of the typing adjustments, to be discussed in more detail below) is provided through a different control panel. Users can cycle through the panels using the arrow

Figure 5. The 'Settings' button in the toolbar is used to bring up the settings panel.

buttons on the left of the settings band. When the user is satisfied with a particular transformation or a set of transformations, they can close the band and browse the Web as usual. All subsequent Web pages will have the selected changes automatically applied. Transformations are applied whether the band is present or not (see figures 2–4 for examples of transformations applied with the settings band not present).

The ease with which users can try different changes means that they can see which transformations work for them. This is particularly important in cases where users may not know in advance which changes will prove beneficial. In fact, our users have discovered useful transformations that they would not have predicted in advance of trying them out. Consider, for example, the text style discussed above. None of our users requested this in our initial interviews. The literature, however, suggested that such adjustments would increase legibility (Arditi 1999; see also Center for Medicare Education 2000, National Institute on Aging 2002). When given the option to try out this transformation, many users did indeed discover that it made Web pages easier to read. A design that assumed that users could pick out their needed changes from a checklist, for example, would likely not lead to this outcome.

In addition, the interface we employed offered the advantage of making the system's built-in accessibility options easily available to our users. For example, there is a keyboard panel that has a button for 'one-handed' typing. Selecting this activates the StickyKeys feature from the Operating System.

In other cases, however, the setting of operating system features is not quite so direct. Take the case of debounce time. Persons with tremors, for example, may depress one key multiple times in rapid succession, causing repeated letters to appear when typing. Windows® allows users to set a 'debounce' parameter to control the length of time that repeat keys will be filtered out. In order to set this,

Figure 6. The 'Settings' band at the bottom of the screen with the panel for 'text style' visible.

however, users must know that this feature exists, must know how to set it through the operating system, and must *be able* to set it which requires the clicking of small buttons and checkboxes.

In the Web Adaptation Technology software, user requirements for options such as debounce time are inferred from short samples of a user's typing. Thus, as a user types, the software makes a determination of optimal keyboard settings referencing typing models tested with users having a variety of keyboarding difficulties due to limited hand coordination (Trewin 2004). Based on this analysis, the keyboard accessibility parameters are automatically adjusted to maximize typing accuracy for the individual. In this case, the Web Adaptation Technology software greatly simplifies the task of using built-in operating system options for the user.

In other cases, what appeared to be a straight-forward setting of the browser's built-in features was not. As an example, consider the colours option shown in figure 8. The required skills for making a change of colour include the ability to see and click small targets, and the cognitively complex task of dealing with multiply embedded menus and dialog boxes. For users new to computing, this task is made even more difficult by having to understand system settings and deal with unfamiliar terminology such as 'visited link colour' and 'hover colour'. The dual problems of user limitations and lack of computer expertise make this a daunting task. As shown in figure 8, users simply click the button in the colours panel indicating their preferred text and background colours. The browser immediately adjusts to these colours, setting, in addition, maximally contrasting visited link, unvisited link, and hover colours.

There is, however, a second level of complexity for colour changes that necessitates additional page analysis. The reason for this is that Web designers often use transparent GIFs to create visual effects, with transparent gifs often containing text designed to display over a

Figure 7. The same Web page as shown in figure 6, but with the 'text style' transformation applied.

particular background colour or image. In such cases, changing the background colour can make text or other information difficult to read. In the worst case, information can completely disappear as the text and background colours become the same when the new background colour is the same as the colour in the non-transparent part of a GIF. To deal with this potential problem, every time page colours are changed, the Web Adaptation Technology software analyzes the GIF and background colours, changing transparent GIF colours to the original background colour. Thus, the colours option, while seeming to be setting the browser's built-in features is actually doing a DOM analysis. This not only made colour changes easier for the user to set than with the browser utilities, but also made for better end results.

Finally, to improve software usability further we created an interactive help system tied directly to the various control panels in the settings band. Figure 9 shows the help panel for the text style feature. Several aspects of the help system are worth noting here. First, since the help was keyed to the individual control panels it was easy to find out what a particular control panel did just by clicking on the help button when that control panel was visible. Changing the control panel caused the help to change as well. Changing the help panel (e.g. following a hyperlink to a related transformation) caused the control panel to change. Thus, at any point, what was said about the control panel's transformations could be tried out by the user. Second, each help panel illustrated all of the features of the corresponding control panel. The text style help, for example, contains an illustration of a novel font that can be changed.

3.5 *Testing procedures*

The Web Adaptation Technology software has been in use for over a year. Although the software was originally

Figure 8. Web page showing the 'colours' transformation set to white on black.

designed to meet the needs of older adults, we were contacted by a number of organizations serving persons with disabilities who were also interested in using the software. The testing of this software was, therefore, expanded to include additional user groups, including persons with visual and dexterity limitations that are not the result of aging, young adults with developmental disabilities, and non-native speakers of English. In addition, the testing with older adults expanded beyond the SeniorNet organization to include a number of other organizations serving this population. We report here on testing by 1200 users. Approximately half were older adults.

4. Results

Table 1 shows the proportion of users who employ particular transformations. Note that the Totals column adds to more than 100%. This is due to the fact that users typically select more than one feature.

One very popular feature is the speak text option. Speak text uses IBM's ViaVoice® text-to-speech capability. Our speak text works by having the user point the mouse at the area of the page they want to have read aloud. As the mouse hovers over text, links, or images, the corresponding text is read aloud (for images, the ALT text that is present is spoken). Speak text also illustrates two important properties of our design: first, it allows people to use the capabilities they *do* have (in this case, limited but still useful vision); second, it supports a relatively effortless transition between modes of interaction (in this case, vision-based pointing and auditory playback).

The approximately 40% of our users who elect to turn on the speak text feature is a very high proportion, especially when we consider that many desktop computers used by

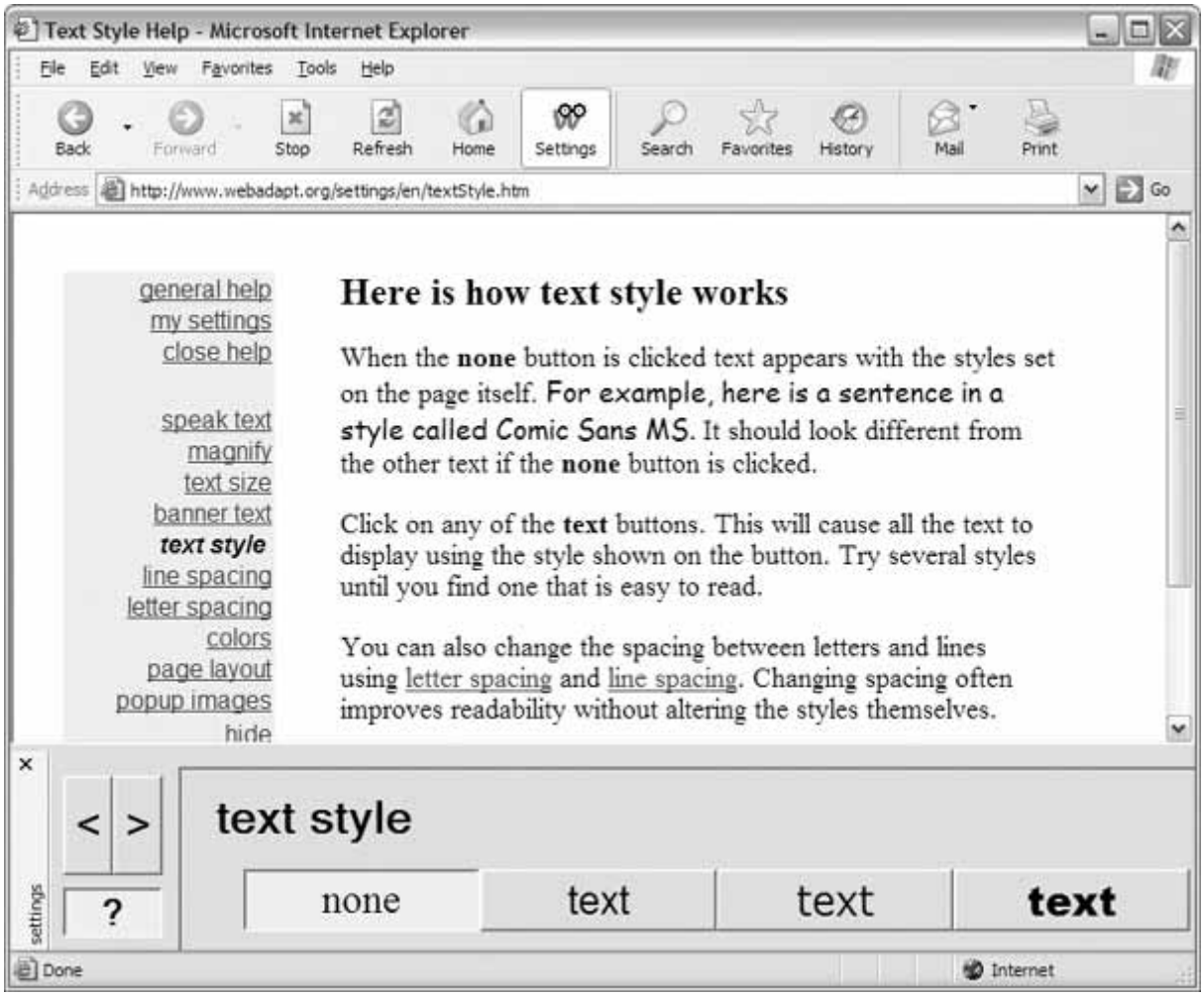

Figure 9. Interactive Help page illustrating how the 'text style' feature works.

our population do not have audio support so not everyone could take advantage of this option.

A comparably large percentage of our users, 41%, opt to have text size enlarged. This is an example of a transformation that is useful for many who would not consider themselves disabled (including the second author who uses this feature to enlarge the small fonts associated with a 1600×1200 laptop display). Most of these users choose to use our large and larger sizes (corresponding to the Larger and Largest options in Internet Explorer's text size menu option). Only about 3% choose our largest size (which is implemented by way of a DOM manipulation).

Other visual transformations are used somewhat less often but still appear to be broadly applicable. Text style (sans serif fonts with progressively heavier line weights), colour changes (mainly black on white which eliminates some distracting colour combinations, followed by the somewhat lower contrast variant of black on yellow), and whole page magnification are used by nearly a third of our users. Interestingly, the majority of users doing whole-page magnification, 58%, opt for the relatively modest 125% zoom factor. Banner text, image transformations (enlargement and contrast sharpening), and at least one of the hide options (animations, images, and backgrounds) are used by about a quarter of our users.

The use of banner text is interesting in light of the fact that only 3% of our users choose the largest of our text size transformations. Looking at the individual data for just the 27% of our users who use banner text, we found that more of them, 6%, are using our largest text size setting, and many of them, 70%, are using some text size enlargement in addition to the banner text. 64% of these users are also using bolder, sans serif fonts via our text style transformation. 58% are also using some whole-page magnification, 26% are enlarging the mouse pointer, and 40% are enlarging the menus and other browser controls. It seems that somewhat larger text, somewhat bolder text, somewhat magnified text, larger pointers, larger browser controls, and

Table 1. Percentage of our test population using particular transformations.

						TOTAL
Speak text	40.7%					40.7%
	Large	**Larger**	**Largest**			
Text size	22.8%	15.6%	2.8%			41.2%
Banner text	26.4%					26.4%
Line spacing	13.8%					13.8%
	Wider	**Widest**				
Letter spacing	11.4%	4.6%				16.0%
	Arial	**Verdana**	**Arial Black**			
Text style	12.4%	11.3%	9.6%			33.3%
	Black/White	**White/Black**	**Black/Yellow**	**Other**		
Colours	14.7%	2.1%	10.1%	2.0%		28.9%
	125%	**150%**	**175%**	**200%**	**250%**	
Magnify page	18.1%	6.6%	2.0%	1.7%	2.9%	31.3%
	Larger	Sharpen				
Enhance images	15.2%	9.9%				25.1%
Hide images	7.6%					7.6%
Hide backgrounds	6.0%					6.0%
Hide GIF animations	14.3%					14.3%
Larger pointers	16.4%					16.4%
Larger browser controls	22.1%					22.1%

very large banner text (triggered by pointing the mouse at the text of interest) tend to co-occur for a (presumably) low-vision segment of our users. Not surprisingly, a larger proportion of these banner text users — 60% compared with 40% of the entire user group — are also using the speak text feature.

Only about 15% of our users take advantage of the line spacing or letter spacing features. While this is a fairly small percentage, this fact should not minimize its importance for the people who do select it. Indeed, looking again at the users of banner text, we find that 34% are using increased letter spacing and 31% are using increased line spacing. Apparently, these features are perceived as most useful by the people in our study who have the poorest vision.

5. Conclusions

Our approach is complementary to accessible Web page design as presented in guidelines and standards. The majority of Web audits and checklists focus on the types of technical performance standards of the W3C (Brewer 2004) and Section 508 (Section 508, 2002a). These guidelines tend to address the most disabled users. In many cases, they don't directly impact page presentation, but rather address issues related to making Web pages capable of being rendered by assistive technology devices such as screen readers. Our software, rather than addressing the technical standards for accessibility, would more properly fall into the category of making Web pages usable. There are some guides to help define and measure usability (Neilson, 1999; see also UsableNet 2004), but perhaps the closest approximation to regulating usability can be found in guidelines that focus on functional performance standards for limited vision and input (Section 508, 2002b). By allowing individual users to have control over the way they wish to have pages presented and how input is controlled, the Web Adaptation Technology software is consistent in spirit with the User Agent Accessibility Guidelines (Gunderson 2004; see also Jacobs *et al.* 2002).

It is worth noting that the Web page used as an example in this paper was compliant with accessibility guidelines. As shown here, however, even this page can be difficult for persons with failing eyesight and hand control. The Web Adaptation Technology allows us to make a number of transformations and input adaptations 'on the fly' that greatly increase the usability of these pages for a substantial number of users. While usability and accessibility are sometimes considered to be two sides of the same coin, such a characterization is inaccurate. As a simple example, a page may be very intuitive to use, but inaccessible due to lack of ALT tags on the images. Thus, it is usable by persons without disabilities, but inaccessible to persons who use a screen reader. Conversely, another page may have ALT tags appropriately included for all graphics, but have a layout that is confusing. It is not uncommon to have Web pages that meet standards for technical accessibility, but are still difficult to use, whether or not a person has a disability (see, for example, Powlik and Karshmer 2002, Leporini and Paternò 2004).

The Web Adaptation Technology software is able to transform Web pages that do not meet accessibility standards. However, for pages that are compliant it can take advantage of accessibility features to deliver a better user experience. For example, if no ALT tag text is provided

for an image, the user simply hears the word 'image' when they mouse over it using the speak text feature. However, if the image has the recommended ALT tag text, the Web Adaptation Technology software will read the image ALT text aloud.

Another example of our ability to take advantage of conforming Web content is provided by our use of the skip-navigation tag. This tag was initially conceived as a way for screen readers to skip directly to the main content on a page and is recommended for just this purpose. But the skip-navigation tag also allows our software to do a better job of rendering pages for low vision users. Figure 4, discussed earlier, shows a Web page rendered as a single column. Notice that the page in figure 4 has been scrolled past the navigation bar (which is at the top of the page following linearization), and positioned at the beginning of the main content. Our page linearization feature takes advantage of a skip navigation tag, if present, to automatically position the initial view directly to this main content. For our purposes then, the skip navigation tag allows users to immediately see the main content of a page with less clutter and with fewer navigation actions.

These are examples in which conformance to standards and guidelines results in a more usable page for those not using the initially targeted assistive device. Thus, rather than eliminating the need for accessibility markup, our software capitalizes on its presence to give any person a more useable Web page.

In summary, the Web Adaptation Technology software successfully addresses the problems inherent in the original proxy server design. It eliminates the issues of secure Web sites, setting the proxy, and copyright. The use of the browser's own parsing to create the to-be-modified DOM eliminates problems with accurately rendering Web pages. Problems of scale also are eliminated in that transformations are now made on individual client machines, eliminating the need for large and powerful servers to perform multiple renderings and transformations simultaneously.

The Web Adaptation Technology has expanded from its initial use in the United States to other countries, having been translated into several languages including Spanish, French, German, Italian, Brazilian Portuguese, and Chinese. The current design appears to meet the needs of both our original user population, which was older users, and other populations that have tried the software. At this point, we are confident that the software can be used by a wide variety of users, many of whom have little experience with computers.

The Web Adaptation Technology software is not the only example of software designed to meet the needs of older Internet users or users with limited vision or dexterity difficulties. Our software does appear to be unique, however, in the range of transformations and input adaptations made available through a simple, unified interface. It is clear that the phrase 'one size fits all' does not apply to those we support. Our users have complex, interacting, and changing abilities as captured in the phrase 'dynamic diversity'. The software allows users to easily select and apply numerous transformations and input adaptations, in any combination depending on their immediate needs, to make Web pages more usable.

Acknowledgements

We would like to acknowledge the contributions of our colleagues at IBM's Watson Research Centre, Jonathan Brezin, Susan Crayne, Peter Fairweather, Cal Swart, Beth Tibbitts, and Shari Trewin, and in other IBM locations, Sam Detweiler and Rich Schwerdtfeger.

References

ARDITI, A., 1999, Effective colour contrast: designing for people with partial sight and colour deficiencies. Available online at http://www.lighthouse.org/colour_contrast.htm

ASAKAWA, C. and TAKAGI, H., 2000, Annotation-based transcoding for nonvisual Web access. In *Proceedings of ASSETS '00*, Arlington, VA, November 2000 (New York: ACM Press), pp. 172–179.

BBC NEWS, 2002, Elderly take up net surfing. Available at http://news.bbc.co.uk/2/hi/science/nature/1977823.stm.

BREWER, J., 2004, Web Accessibility Initiative (WAI). Available at http://www.w3.org/WAI.

BROWN, S.S. and ROBINSON, P., 2001, A World Wide Web mediator for users with low vision. Paper presented at the *CHI'2001 Conference on Human Factors in Computing Systems Workshop* No. 14. Seattle, WA. Available online at http://www.ics.forth.gr/proj/at-hci/chi2001/files/brown.pdf

CENTER FOR MEDICARE EDUCATION, 2000, Creating Senior-friendly websites. Available at http://www.medicareed.org/content/CMEPubDocs/V1N4.pdf.

CZAJA, S.J. and LEE, C.C., 2001, The Internet and older adults: design challenges and opportunities. In *Communication, Technology, and Aging: Opportunities and Challenges for the Future*, N. Charness, D.C. Park, and B.A. Sabel (Eds), pp. 60–78 (New York: Springer Publishing Company).

CZAJA, S.J. and LEE, C.C., 2003, Designing computer systems for older adults. In *The Human–Computer Interaction Handbook*, J. Jacko and A. Sears (Eds), pp. 413–427 (Mahwah, NJ: Erlbaum).

FAIRWEATHER, P.G., HANSON, V.L., DETWEILER, S.R. and SCHWERDTFEGER, R.S., 2002, From Assistive Technology to a Web Accessibility Service. *Proceedings of ASSETS '02*, Edinburgh, Scotland, July 2002 (New York: ACM Press), pp. 4–8.

FAYE, E.E. and STAPPENBECK, W., 2000a, Normal changes in the aging eye. Available at http://www.lighthouse.org/aging_eye_normal.htm.

FAYE, E.E. and STAPPENBECK, W., 2000b, The most common vision disorders in later life. Available at http://www.lighthouse.org/aging_eye_common.htm.

FOX, S., 2004, Older Americans and the Internet. Available online at: http://www.pewinternet.org/pdfs/PIP_Seniors_Online_2004.pdf

GREGOR, P. and NEWELL, A.F., 2001, Designing for dynamic diversity — making accessible interfaces for older people. *Proceedings of the 2001 EN/NSF Workshop on Universal Accessibility of Ubiquitous Computing: Providing for the elderly, WUAUC '01*, Alcácerdo Sal, Portugal, May 2001 (New York: ACM Press), pp. 90–92.

Grundy, E., Ahlburg, D., Ali, M., Breeze, E., and Sloggett, A., 1999, Disability in Great Britain: results of the 1996/7 Disability. Follow-Up to the Family Resources Survey. Available online at ttp://www.dwp.gov.uk/asd/asd5/94summ.asp

Gunderson, J., 2004, W3C user agent accessibility guidelines 1.0 for graphical Web browsers. *Universal Access in the Information Society*, **3,** 38 – 47.

Hanson, V.L., Richards, J.T., Fairweather, P.G., Brown, F., Crayne, S., Detweiler, S., Schwerdtfeger, R., and Tibbitts, B., 2001, Web accessibility for seniors. In *Universal Access in HCI: Towards an Information Society for All,* C. Stephanidis (Ed.), pp. 663 – 666 (Mahwah, NJ: Erlbaum).

Jacko, J.A., Barreto, A.B., Marrnet, G.J., Chu, J.Y.M., Bautsch, H.S., Scott, I.U. and Rosa, R.H., 2000, Low vision: the role of visual acuity in the efficiency of cursor movement. In *Proceedings of the Fourth International ACM Conference on Assistive Technologies, ASSETS 2000*, November 2000 (New York: ACM Press), pp. 413 – 427.

Jacobs, I. Gunderson, J. and Hansen, E., 2002, User agent accessibility guidelines, 1.0. Available at http://www.w3.org/VAAGIO/.

Leporini, B. and Paternò, F., 2004, Increasing usability when interacting through screen readers. *Universal Access in the Information Society*, **3,** 57 – 70.

Mcneil, J.M., 1997, Americans with Disabilities: 1994 – 95. Available online at http://www.census.gov/prod/2001pubs/p70-73.pdf

National institute on aging, 2002, Making your website senior friendly. Available at http://www.nlm.nih.gov/pubs/checklist.pdf.

Neilson, J., 1999, *Designing Web Usability: The Practice Of Simplicity* (New Riders Publishing, Indianapolis, IN).

Powlik, J.J., and Karshmer, A.I., 2002, When accessibility meets usability. *Universal Access in the Information Society*, **1,** 217 – 222.

Richards, J.T. and Hanson, V.L., 2004, Web accessibility: A broader view. In *Proceedings of the Thirteenth International ACM World Wide Web Conference, WWW2004*, New York, NY, May 2004 (New York: ACM Press), pp. 72 – 79.

Richards, J.T., Hanson, V.L. and Trewin, S., 2003, Adapting the Web for Older Users. In *Universal Access in HCI (Vol 4): Inclusive Design in the Information Society,* C. Stephanidis (Ed.), pp. 892 – 896 (Mahwah, NJ: Erlbaum).

Section 508, 2002a. Available at http://www.section508.gov/.

Section 508, 2002b, Summary of Section 508 standards. Available at http://www.section508.gov/index.cfm?FuscAction = content810 = 11# functional.

Takagi, H., Asakawa, C., Fukuda, K., and Maeda, J., 2002, Site-wide annotation: reconstructing existing pages to be accessible. *Proceedings of ASSETS'02*, Edinburgh, Scotland, July, 2002 (New York: ACM Press), pp. 81 – 88.

Trewin, S., 2004, Automating accessibility. To appear in Proceedings of ASSETS'04, October, 2004 (New York: ACM Press), Atlanta, GA, pp. 71–78.

Trewin, S. and Pain, H., 1998, A model of keyboard configuration requirements. *Proceedings of ASSETS '98*, April, 1998 (New York: ACM Press), pp. 173 – 181.

Usablenet, 2004. Available at http://www.usablenet.com.

Zajicek, M., 2001, Supporting older adults at the interface. In *Universal Access in HCI: Towards an Information Society for All,* C. Stephanidis (Ed.), pp. 454 – 458 (Mahwah, NJ: Erlbaum).