Web accessibility technology at the IBM Tokyo Research Laboratory

J. Maeda
K. Fukuda
H. Takagi
C. Asakawa

This paper provides an overview of Web accessibility technologies developed at the IBM Tokyo Research Laboratory. Home Page Reader provides an easy and comfortable way for the blind to read Web pages. Transcoding technology solves problems that arise when a voice browser, such as Home Page Reader, reads Web pages. It also offers more comfortable ways for visually impaired people and senior citizens to read Web pages. Accessibility Designer, the main focus of this paper, allows page designers to learn about disabilities and to create more accessible and usable Web pages at authoring time. It has two modes: a low-vision mode and a blind mode. The low-vision mode simulates the perception of people with low vision and detects inaccessible parts in the target page. The blind mode visualizes the usability of a voice browser for the target page and detects accessibility and usability problems. Accessibility Observer makes it easier for Web developers to detect accessibility problems.

Introduction

Research activity on accessibility at the IBM Tokyo Research Laboratory (TRL) started in 1985. *Accessibility* means access to information for all—focusing on people with disabilities and senior citizens. Ensuring accessibility improves the quality of life for such people by removing barriers that prevent them from taking part in many important life activities.

The history of our accessibility research began with studies on digital Braille for the blind. To solve problems with Braille punched into paper, the Braille editing system [1], the Braille information network [2], and the Braille dictionary system were developed.

In the mid-1990s, the use of the Internet became popular, and the World Wide Web became an essential information resource for our society. Today, huge amounts of information can be obtained through the Web, and the Web permeates our social activities.

As this environment developed, we shifted our focus, in 1996, to Internet accessibility technologies to enable people with disabilities to enjoy the enormous benefit of the Internet. The first success was achieved in 1997 with *Home Page Reader* [3], described in the next section. It provided an easy and comfortable way for the blind to read Web pages. Since then, we have been doing research mainly on the Web, extending our targets for content that can be made accessible and for applications to which we can apply our technologies in accordance with changes in the social and technical environments. Furthermore, in the last several years, our research area has been expanded; in addition to *accessibility*, we added *usability*. We have also expanded our target user group from the blind to also include people with low vision. For the purposes of this paper, we use the term *low-vision people* as a generic term for people who have weakened vision—such as people with amblyopia or color blindness, and senior citizens with

0018-8646/04/$5.00 © 2004 IBM

age-weakened eyesight—but who are still able to read Web pages.

In this paper, we focus on three changes in the environment:

1. *Evolution of Web content:* In the early stages of the Web, pages were simple, and most of their content was text that could easily be read by Home Page Reader. As the Web evolves, however, Web pages are gradually becoming more complicated, and their content has more and more visual elements, such as images, that can often cause accessibility problems. In addition, most business sites have recently begun providing Web pages dynamically on demand by using new server-side technologies, such as JavaServer** Pages (JSP**) and servlets. These technologies make it more difficult to improve the accessibility of pages using conventional accessibility tools.
2. *Necessity of author-side technologies:* Accessibility technologies are not only for senior citizens and people with disabilities. Changes in social and business environments, such as the enactment in the United States of Section 508 of the Rehabilitation Act Amendment, call for site owners and page designers to make their Web pages accessible.
3. *Usability:* Usability means ease of use. Ensuring usability is necessary for comfortable Web access. There exist several standardized accessibility guidelines, such as the World Wide Web Consortium (W3C**) Web Content Accessibility Guidelines (WCAG) [4], Section 508 [5], and IBM Corporate Instruction 162 [6]. They describe certain levels of accessibility. However, Web pages complying with these guidelines are not always easy to read. For example, most of the accessibility guidelines require that page designers provide an *ALT attribute* for each of the images. This provides an alternate text description of the image for user agents that cannot display images. However, providing an inappropriate ALT attribute makes the situation worse, not better.

This paper provides an overview of Web accessibility technologies invented and developed at TRL. The next section discusses Home Page Reader for the blind, followed by a section describing accessibility issues. After that, we illustrate transcoding technology to address accessibility issues of already-existing Web pages. We then demonstrate Accessibility Designer, which is the main focus of this paper. Accessibility Designer helps Web designers and developers ensure both the usability of Web pages and their accessibility. It is characterized by providing low-vision simulation and blind usability visualization features. The following section presents the server-side Java** Observer (Accessibility Observer) for improving the accessibility and usability of dynamically produced Web pages, and the final section concludes the paper.

Home Page Reader

To provide a way for blind people to read Web pages, Home Page Reader (HPR) was developed at TRL and released in Japan in 1997. HPR first *linearizes* a Hypertext Markup Language (HTML) document into one-dimensional raw text. It then reads out the resulting text using a text-to-speech engine. HPR had several features that enabled blind users to surf the Web, even if they were novice users. HPR thus helped the Web become a new but essential information resource for blind people.

Before voice browsers appeared, blind people had used regular screen readers, which were unable to handle many of the typical features of Web pages, such as hyperlinks and two-dimensional information layouts. There were several other voice browsers, such as pwWebSpeak** [7], developed in the same time frame as HPR. Compared with other voice browsers, the unique features of HPR were the following:

- *Use of a numeric keypad for Web access:* In those days, most blind people were novices with computers that were based on a graphical user interfaces (GUI). It was even difficult for blind people to use a keyboard without training. HPR, therefore, was designed for use with the numeric keypad. In more recent years, since most users have experience using a keyboard in GUI environments, the current version of HPR supports both the keyboard and the numeric keypad.
- *Reading of hyperlinks in a female voice and plain text in a male voice:* Blind people can easily find all of the hyperlinks on the Web page by listening for the female voice.
- *Table navigation function* [8]: This allows blind people to manage the two-dimensional navigation of a table. By using three features—a table cursor, a table pointer, and a cell-jumping key—users can easily move and understand the current cell position in the table. These features also enable users to easily hear the headers of both the column and the row of the current cell.
- *Fast-forward key for quick reading:* The HPR fast-forward key allows users to skim Web pages quickly and easily.

HPR contains many other features, such as a history list and form input support, and today it is still one of the most powerful tools for blind people. HPR is available in 11 languages and supports blind people throughout the world [9].

Accessibility issues

When Home Page Reader was first developed, Web pages were not as complicated as they are now, and it was relatively easy for blind people to access the Web. However, the authoring trend is becoming much more visual, and the Web is becoming even less accessible to blind people. Several issues arise when blind people try to access the Web nonvisually. One major issue is the time it takes to find the main content of a page. A voice browser, such as Home Page Reader, reads the Web content in the order in which the tags appear in the HTML file, but there are usually banners, indexes, and forms at the top of a Web page, and blind users have to listen to that heading information before the main content is read. Also, because Web page layout is designed for intuitive recognition using visual scanning, similar issues often interfere when blind users try to find specific information on a page. Moreover, page designers often forget to provide equivalent alternatives for visual elements. Problems like these make nonvisual Web access more difficult.

More complicated Web pages also create serious problems for low-vision people. Typical issues for low-vision people are the size and color of page elements, such as text blocks and images. Most Web pages are created by young people with perfect vision, and most of those authors have trouble imagining the difficulties low-vision people might experience while accessing pages they have authored.

We developed two technologies to solve these problems—transcoding technologies and Accessibility Designer. The former is applied to already-existing Web pages, while the latter is used at authoring time. They are described in the next two sections.

Transcoding technology

To improve the readability of existing Web pages on the client side, various attempts have been made, both for blind users (see the section above) and low-vision users. These attempts have included built-in Web browser features, style sheet settings for the browser, augmented browsers [10, 11], and screen magnifiers. They allow the user to change the style of the Web page according to his or her preference. However, it is difficult for these approaches to implement functions more advanced than just changing the "style" of the Web page.

On the other hand, some major Web sites, such as those of the American Foundation for the Blind [12] and the British Broadcasting Corporation [13], have their own mechanisms for changing the style of their Web content. UsableNet [14] is also offering a server-based customizing service for Web content that creates text-only views of the content of specific Web sites. A drawback of such server-side approaches is that they require the client to be accustomed to using and managing his or her preferences for each site.

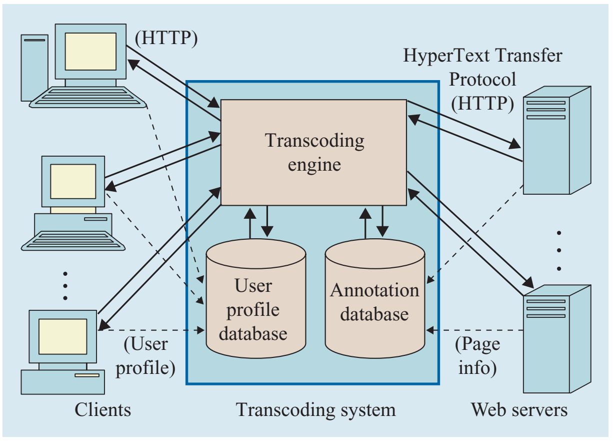


Figure 1

Basic architecture of the transcoding system.

To overcome this drawback, we adopted the transcoding approach to provide functions that change not only the style, but also the structure of a Web page. In this paper, the term *transcoding* means a technology—based on an intermediary processor interposed between Web servers and the client's browser—that can transform Web content according to a user's predefined preferences.

Figure 1 shows the basic architecture of the transcoding system. The system consists of three components: a transcoding engine, a user profile database, and an annotation database. The transcoding engine is the main component that transcodes a target HTML document. The user profile database stores the profiles of registered users. A user profile specifies the favorite settings of a user when reading Web pages. Settings include font sizes and text colors, as described below. The annotation database stores the annotations of Web pages that will be used with the transcoding process. Each annotation set for a Web page contains information on positions of the visually fragmented groups in the page, their roles (e.g., *main content*), and the relative importance of the groups. We use the IBM WebSphere* Transcoding Publisher (WTP) [15], and our transcoding system is a plug-in for WTP.

When a user sends a request to the system for a Web page, the transcoding system requests and receives the Web page on behalf of the user. The annotation of the page and the user profile are then retrieved. The transcoding engine transcodes the Web page according to the annotations and user profile, and then sends the transcoded Web page to the user. The algorithms of annotation-based transcoding are detailed in [16–20].

For blind users, the transcoding system provides the following four functions [21]:

1. *Adding a **skip to the main content** link:* This function dynamically inserts *a skip to the main content* link at the top of the target page by specifying the beginning position of the main content on the basis of heuristic rules. By following this link, users can immediately jump to the main content of the page without reading distracting menus, advertisements, etc.
2. *Simplification of the Web page:* This function obtains other HTML documents from the same Web site as the target Web page in order to calculate the difference between those documents and the target document. By considering the differences, the common parts of the target page, which are specified by a template in most cases, can be removed. When users move to a new document, they access only newly updated information, without the clutter of repeated material.
3. *Reordering the page content:* According to the annotations of the target Web page, the page can be divided into visually fragmented groups. Each group has its own role (e.g., *main content, header,* and *advertisement*), and importance. In most cases, the groups in the original page are not arranged according to their importance. The reordering function sorts them in descending order of their importance values supplied by the annotation and inserts a table of contents—with links to them—in the reordered page. This mechanism is of great benefit to nonvisual users, because they can read the most important groups first and can jump arbitrarily from group to group.
4. *Adding **ALT attributes** to image links:* This function automatically inserts ALT attributes, attempting to fill in the missing ALT attributes for image links. It does this by extracting the title of the destination page linked to the image, and provides this information about the link to the user.

Users can select which functions to invoke, and their selections are stored in their user profile. They can also select their preferences, also stored in the user profile database.

For low-vision people, five functions are provided:

1. *Changing text size.*
2. *Changing line height:* The larger the fonts and lines, the easier it is for users to read the page. However, enlarging fonts and lines also means that the amount of visible information in the browser window decreases. Users can choose their preferred settings.
3. *Enlarging images:* Two modes can be selected. In the static mode, all images are enlarged when the page is displayed. In the dynamic mode, only an image specified by using the mouse cursor will be enlarged.
4. *Changing foreground and background colors:* This function changes foreground and background colors to ensure sufficient contrast between them. Users can also choose from predefined color combinations.
5. *Creating a digest of the page (WebDigest)* [22, 23]: As the fonts and lines in a Web page become larger, a larger area is required to display all of the page content. To view the whole page, the user must awkwardly scroll through it. The WebDigest function reduces and simplifies page content while preserving the overall layout of the page. The digested page consists of several visually fragmented groups, each specified by an annotation. Users can navigate through the groups in the digested page with a keyboard or a mouse. When they want to read the full content of a group, they can simply press the enter key or click the mouse, and the complete group content pops up with the original page layout.

Our accessibility transcoding system was publicly evaluated by low-vision users, including senior citizens, through a Lycos** Japan site in 2001. It was also tested by blind students at a school for the blind in Japan, funded by the Telecommunications Advancement Organization of Japan (2002–2004). It was very favorably received by the users. However, we found it difficult to broadly deploy such a transcoding approach because of legal constraints, such as copyright issues. This approach is suitable for closed services to specific Web sites.

Accessibility Designer

The technologies introduced in the previous section deal with existing Web pages. They are required because numerous Web pages have already been created. On the other hand, it would be much more effective if page designers would ensure the accessibility of their pages at the time of authoring. Recent changes in the social environment, such as the implementation of Section 508 in 2001, motivate page designers to consider the accessibility of their pages.

Until now, programs called *accessibility checkers* have been used as author-side tools to evaluate the accessibility of Web pages. They analyze an HTML document and check whether the document is compliant with accessibility guidelines, e.g., Web Content Accessibility Guidelines. Such tools can be used as a first step of accessibility, but their effects are very limited because accessibility guidelines do not consider usability very well, and analyzing HTML tags and attributes is insufficient to evaluate the accessibility and usability of the Web page as displayed by a Web browser.

For Web authors, one of the most effective ways to learn about Web accessibility is to experience how disabled people access the Web by themselves using assistive technology software, such as Home Page Reader, JAWS** [24], or ZoomText** [25]. However, it is not very easy for nondisabled authors to have experiences similar to those of people with disabilities, even if they use such assistive applications. For example, it would take a lot of time for sighted users to learn how to access the Web nonvisually using applications like Home Page Reader, because it is designed for the needs and preferences of blind users. Another approach would be to observe the real environments that disabled people use to access the Web with their assistive technology software. It would also be difficult for authors to have such experiences.

To overcome these difficulties, we developed Accessibility Designer, a disability simulator for Web authors to use as they work on their own computers at authoring time. It has functions to simulate two disabilities, low vision and blindness, and it can automatically detect usability problems.

Low-vision mode

The low-vision mode evaluates the accessibility and usability of Web pages for low-vision people. One of the notable features of this mode is its *presentation-based approach*; that is, it simulates the way that low-vision people see a Web page and evaluates accessibility and usability on the basis of the simulation.

Without this tool, page designers cannot avoid using accessibility checkers that analyze HTML tags and attributes. But applying such *text-based approaches* to the evaluation of accessibility for low-vision people has little effect, because HTML analysis has very limited capability to examine the appearance of the rendered Web page. For example, tools adopting the text-based approach can detect images without their ALT attributes, but they cannot detect characters whose foreground and background colors do not have enough contrast.

There are several low-vision simulators. Vischeck [26] is a Web application that simulates color-blind vision. Xproof [27] is an application program that also simulates color blindness. However, these applications simulate only low vision; they cannot automatically detect accessibility problems from simulated Web pages or images. For example, Vischeck has a function for color-blind image correction, but it cannot be used to correct HTML documents. Therefore, checking the accessibility of a number of Web pages using only these simulators is an exhausting task for page designers. Nor can these tools be used to analyze the total accessibility of a Web site.

Functions

The basic idea of the low-vision mode of Accessibility Designer is to make an image by rendering a Web page, to simulate the view of low-vision people by using image transformation techniques, and to detect inaccessible parts of the page by applying image analysis techniques. **Figure 2** shows a block diagram of the main functions of this mode. As can be seen in the figure, this mode consists of five parts: the low-vision model, and the rendering, simulation, image analysis, and result presentation functions.

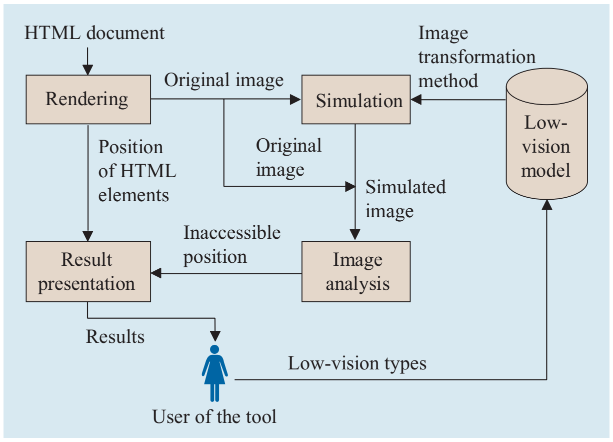


Figure 2

Main functions of the low-vision mode of Accessibility Designer.

The low-vision model is a database that stores image transformation techniques and their parameters for simulating various types of low vision. Examples of the stored data are shown in **Table 1**. These types and parameters can be arbitrarily combined, though none of the image transformation techniques are new.

The rendering function renders an HTML document to create what we call an *original image.* In the rendering process, the correspondences between HTML elements and their positions in the image are calculated and stored. This function can be provided by well-known technologies, such as the rendering engine of a Web browser.

The simulation function applies image transformation techniques to the original image according to designated low-vision types and parameters. The output image of the transformation is called the *simulated image.*

The image analysis function analyzes both the original image and the simulated image to detect characters that cannot be read by people with the designated vision problems. There are two necessary conditions for a character to be judged as readable: Its foreground and background colors must be distinguishable from each other, and its shape must be retained even after the simulation. To determine whether a character satisfies these conditions, the following steps are followed.

First, for each color appearing in the original image, a binary image is produced in which that color becomes the foreground and all of the other colors become the background. A connected component-labeling technique is applied to the binary image to extract all of the connected

Table 1 Examples of data records for the low-vision model.

Low-vision type	*Image transformation technique*	*Parameters*
Weak vision	Convolution of a cylindrical function that represents a blur	Radius of the cylinder
Degradation of crystalline lens transparency	Degradation of luminance values of colors according to their wavelength	Degraded luminance values
Color blindness	Substitution of a color for a different color on the same "confusion line"	Type of color blindness

components. The connected components are screened to extract components that can be regarded as characters. A thinning algorithm is applied to each character to extract its shape, and the shape is stored. To describe the shape of a character, the crossing numbers of pixels in the connected component are used.

For each character in the original image, the connected component at the same position in the simulated image is then extracted. Foreground and background colors of the connected component are tested to determine whether they can be distinguished from each other. If there is no color problem, the shape of the connected component is calculated in the same manner as in the original image. The shapes of the character in the original image and of the corresponding connected component in the simulated image are compared in order to judge whether the shape of the character is retained after the simulation.

The result presentation function in Figure 2 acquires the positions of the unreadable characters from the image analysis function. It also acquires the correspondences between the HTML elements and their positions from the rendering function. Using this information, this function specifies the HTML elements that contain unreadable characters and shows them to the user (a Web page designer).

User interface

Figure 3 presents the user interface for the low-vision mode of Accessibility Designer, which is implemented using the Standard Widget Toolkit. Most of its components are used in common with the blind mode. In the upper left area is an embedded Web browser, not an image, so that the user can actually navigate the Web. This part is common to both the low-vision and the blind modes.

The upper right part is peculiar to this mode. There are two buttons at the top of this area, the *simulation button* and the *setting button*. The setting button opens the setting panel with the low-vision types that can be chosen. The current prototype can deal with three types, each of which has its own parameters:

1. *Weak eyesight.* Parameter: quality of eyesight (20/20 to 20/200).
2. *Color blindness.* Parameter: type of color blindness (*protan*, the abnormality of red-sensitive cone pigment; *deutan,* the abnormality of green-sensitive cone pigment; or *tritan,* the abnormality of blue-sensitive cone pigment.)
3. *Degradation of crystalline lens transparency.* Parameter: age bracket (20s to 60s).

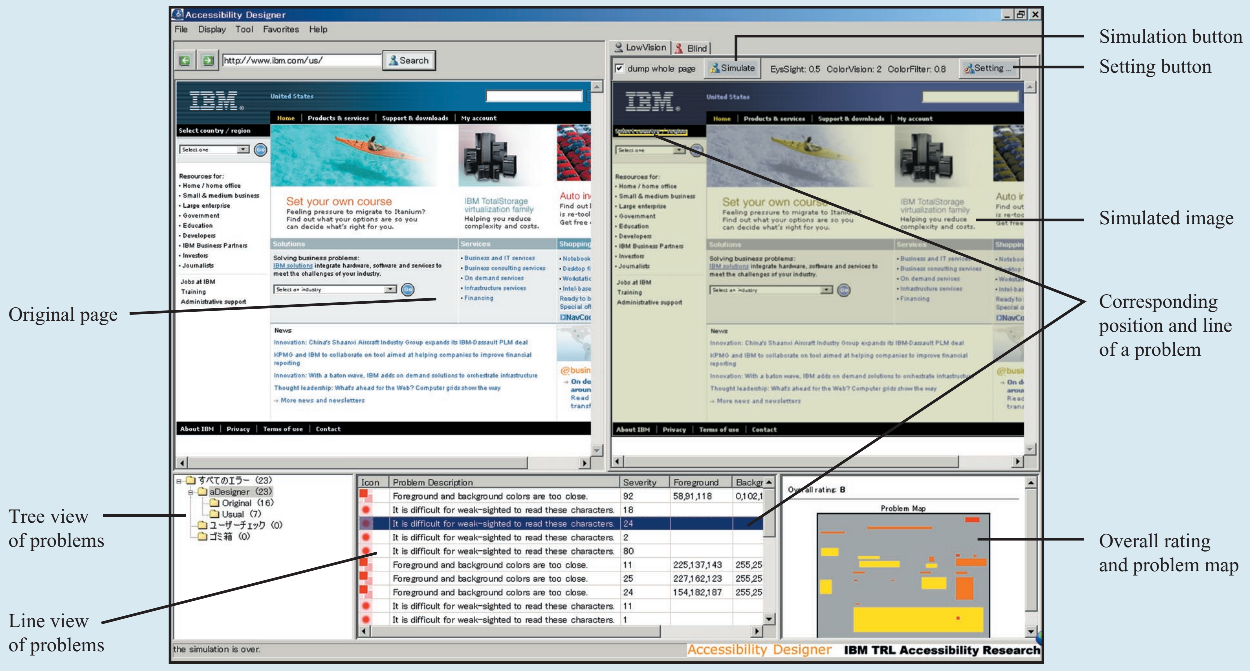
These types can be arbitrarily combined by using check boxes. When the settings have been chosen, the appropriate simulation functions are immediately invoked. The simulate button also invokes the simulation and follow-on functions with the current settings that are displayed between the simulate button and the setting button. The simulated image of the page is displayed under these buttons.

When problems are detected as a result of the image analysis function, a tree view of the categorized problems is displayed in the lower left area (Figure 3). The user can select the category of problems to be listed in the line view at the bottom center. The line view presents the problems of the selected category line by line. Each line consists of a type, description, severity, position, area, and foreground and background colors (optional) for a problem. By selecting lines in this view, the positions of the corresponding problems are displayed in the simulated image. The lower right area provides overall page rating for accessibility and usability for low-vision people. This area also presents a *problem map* that visualizes the distribution of all problems in the page.

Examples

Figure 4 shows two examples of the simulation of weak eyesight. Figure 4(a) is a sample page in which simple characters and complex characters are lined up in two columns. The font size of the characters in the first row is 5 points, and the font size increases by 5 points from line to line.

Figure 4(b) shows a result of a simulation of the view of a person with 20/40 eyesight who is looking at a 19-inch



Icon	Problem Description	Severity	Foreground	Backgr
■	Foreground and background colors are too close.	92	58,91,118	0,102,1
●	It is difficult for weak-sighted to read these characters.	18		
●	It is difficult for weak-sighted to read these characters.	24		
●	It is difficult for weak-sighted to read these characters.	2		
●	It is difficult for weak-sighted to read these characters.	80		
■	Foreground and background colors are too close.	11	225,137,143	255,25
■	Foreground and background colors are too close.	25	227,162,123	255,25
■	Foreground and background colors are too close.	24	154,182,187	255,25
●	It is difficult for weak-sighted to read these characters.	11		
●	It is difficult for weak-sighted to read these characters.	1		

Figure 3

User interface for the low-vision mode of Accessibility Designer.

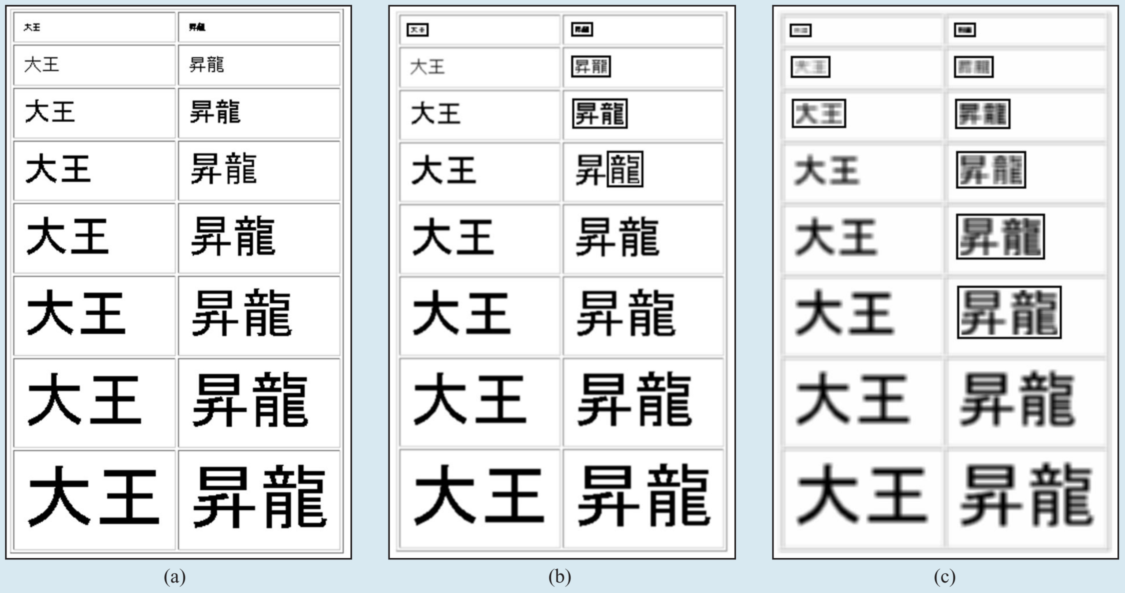


Figure 4

Simulations of weak eyesight: (a) Original page; (b) eyesight 20/40; (c) eyesight 20/200.

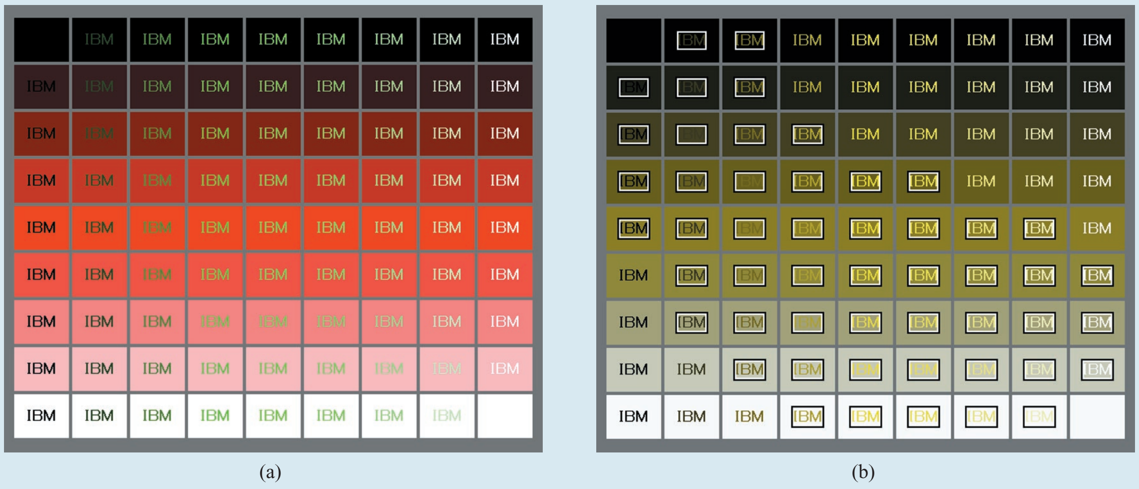


Figure 5

Simulation of color blindness: (a) Original page; (b) deutan color blindness.

Table 2 Comparison of our presentation-based approach and conventional text-based approaches.

	Presentation-based approach	*Text-based approaches*
Ability to check visual attributes	Yes	No
Ability to check non-HTML content	Yes	No
Education and training for the authors	Yes	No
Applicability to non-Web media	Yes	No
Ability to check nonvisual attributes	No	Yes

monitor with the resolution set to 1280 × 1024 pixels, at a viewing distance of 30 cm. The characters judged to be hard to read are surrounded by rectangles. Figure 4(c) presents a simulation of the view of a person with 20/200 eyesight under the same conditions. The results show that the low-vision mode of Accessibility Designer can simulate the fact that the weaker the eyesight, the wider the range of characters that become difficult to read, and that, as the characters become smaller and more complex, they become harder to read.

Figure 5 demonstrates an experiment on color blindness. Figure 5(a) is a sample page. In the table, the foreground colors vary from black to white by way of green and the background colors vary from black to white by way of red. Figure 5(b) is a simulated image of the view of color blindness (deutan). Rectangles indicate the characters whose foreground and background colors are judged to be hard to distinguish from each other. The estimation is based on an algorithm from "Techniques for accessibility evaluation and repair tools" [28]. Characters at the upper left corner (black characters on black background) and lower right corner (white characters on white background) could not be recognized as characters in the first step of the image analysis function, and therefore were not processed further.

Advantages

Table 2 illustrates a comparison between our presentation-based approach and conventional text-based approaches as accessibility- and usability-checking tools for low-vision people. First, our approach can check visual attributes, such as colors and font sizes, but text-based approaches cannot. For example, one of the most popular accessibility checkers, Bobby** [29], cannot check colors. It therefore extracts all of the lines from an HTML document that include images (⟨IMG⟩ tags) and color-related attributes, and requires the authors to check them manually.

Second, the presentation-based approach can check any content displayed by a Web browser, regardless of the source (for instance, even the characters within an image), but the text-based approach can check only what is written in an HTML document.

Third, the user of our approach can see simulated images of various types of low-vision problems and can learn how easy or difficult they are to read by people with the selected type of visual impairment. This provides a good experiential education for page designers.

In addition, and digressing a little from the Web, our presentation-based approach can be applied to non-Web media, such as presentation materials, portable data format (PDF) files, and scanned images from printed documents, as long as they can be converted into digital images. On the other hand, the presentation-based approach cannot check nonvisual attributes, such as the ALT attributes of images.

As described above, our presentation-based approach is much more powerful than conventional text-based approaches in checking accessibility and usability for low-vision people. However, it is basically unable to check nonvisual attributes. Therefore, complementary use of a presentation-based approach (the low-vision mode of Accessibility Designer) and a text-based approach (the blind mode described below) is very effective in comprehensively evaluating the accessibility and usability of a Web page.

Blind mode

One of the most crucial factors in improving usability for the blind is *navigability*. Sighted users can easily recognize the main content in a page, since it is often located in the center of the page, with a white background, a large font, and a large image as visual cues indicating important information. In the voice format, there are no visual cues, such as font sizes, colors, or images. Only text information can be extracted from pages, and they have a serialized format without any visual layout information. Blind users need to access such serialized formats and need to explore within the serial structure without using any pointing devices, only keyboards. This severe issue is well known as the navigability problem for nonvisual Web access.

To check these Web issues, accessibility guidelines have recommended that sighted Web designers access target pages by using voice browsers. However, they have been facing essential difficulties with this method. One is the time it takes to read a Web page out loud, often several minutes per page. Another is the difference in hearing ability between the sighted and the blind. Since blind users are always being exposed to speech information rendered by voice browsers, their listening ability on average is much more developed than that of sighted users. In addition, voice browser have specialized interfaces for nonvisual operation, so it is very hard to learn how to use them effectively.

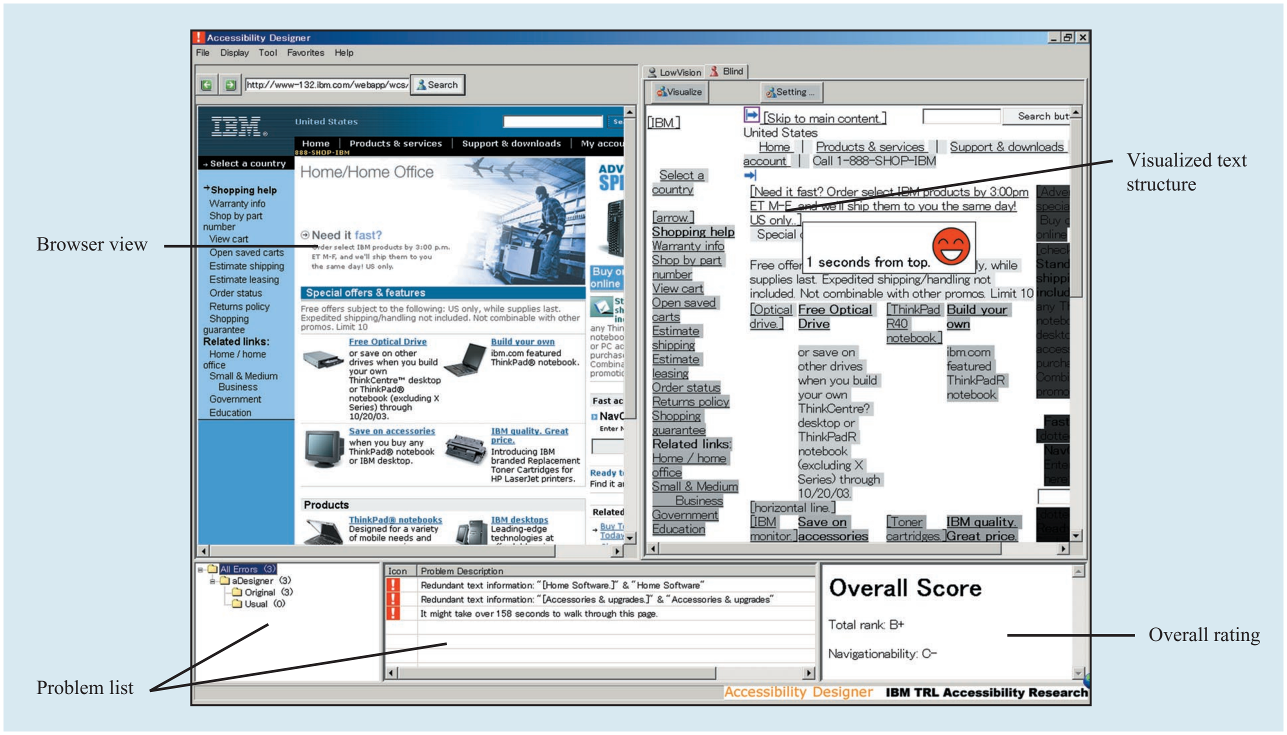
Another way to check navigability is to use accessibility checkers. However, in spite of the severity of the navigability issue, most of the accessibility checking tools have no function for checking that aspect. The WAVE [30, 31] has a function to indicate the reading order of each text block in a page by using small numbered labels on each text block. This makes it easy for Web designers to check the reading order of the text blocks. However, they cannot check the usability more realistically—for example, by being able to answer questions such as *How long does it take from the top to the main content?* or *How effective are the skip-navigation links?* In addition, the numbering method is not intuitive for the sighted.

Therefore, our Accessibility Designer has a function to visualize the *reaching time* for each element in a page. **Figure 6** presents the user interface for the blind mode of Accessibility Designer. When a user of Accessibility Designer selects the blind mode and presses the *visualize button,* a simulated view is displayed in the upper right area. In this simulation, we are focusing especially on the issues of navigation ability.

Figure 7 shows an example of this visualization function. We call this function *blind usability visualization*. It shows the times required to reach each element in a page with standard voice Web browsers. It calculates the times from the top of a page to each element when a user uses voice output, then visualizes these times by using graduated shading. In the figure, the left side shows a page without any intrapage links, such as the skip-to-main-content link and the page index links. The main content (top story) area is shaded black, showing that it will take a long time to arrive at this area. The right side of the screen shows an improved page with a skip-to-main-content link. The main content area is shaded light gray, showing how blind users can now access the top story easily. These two pages have the same appearance when rendered by standard browsers such as Microsoft Internet Explorer, but they are significantly different from the blind usability point of view. The blind mode allows its users, e.g., Web designers and site administrators, to evaluate the navigability of their pages at a glance. The blind mode of Accessibility Designer creates this visualization on the basis of text output from a voice browser and page structure analysis, as follows:

1. Create a text rendering based on a voice Web browser engine. Text elements are associated with the document object model nodes.
2. Analyze the page structure, such as intrapage links, heading tags, and list tags.
3. Calculate the reaching times on the basis of the user operation model (type of voice browser, heading navigation overhead, etc.).
4. Assign shades to each text element and insert information about cascading style sheets (CSS) into the HTML file.
5. Convert each image into text with its ALT text.
6. Visualize accessibility-related tags (table headers, heading tags, form elements, intrapage links, etc.).

We are currently evaluating the effectiveness of these blind usability visualization functions by using them for



All Errors (3)
aDesigner (3)
Original (3)
Usual (0)

Icon	Problem Description
!	Redundant text information: "[Home Software.]" & "Home Software"
!	Redundant text information: "[Accessories & upgrades.]" & "Accessories & upgrades"
!	It might take over 158 seconds to walk through this page.

Overall Score

Total rank: B+

Navigationability: C−

Figure 6

User interface for the blind mode of Accessibility Designer.

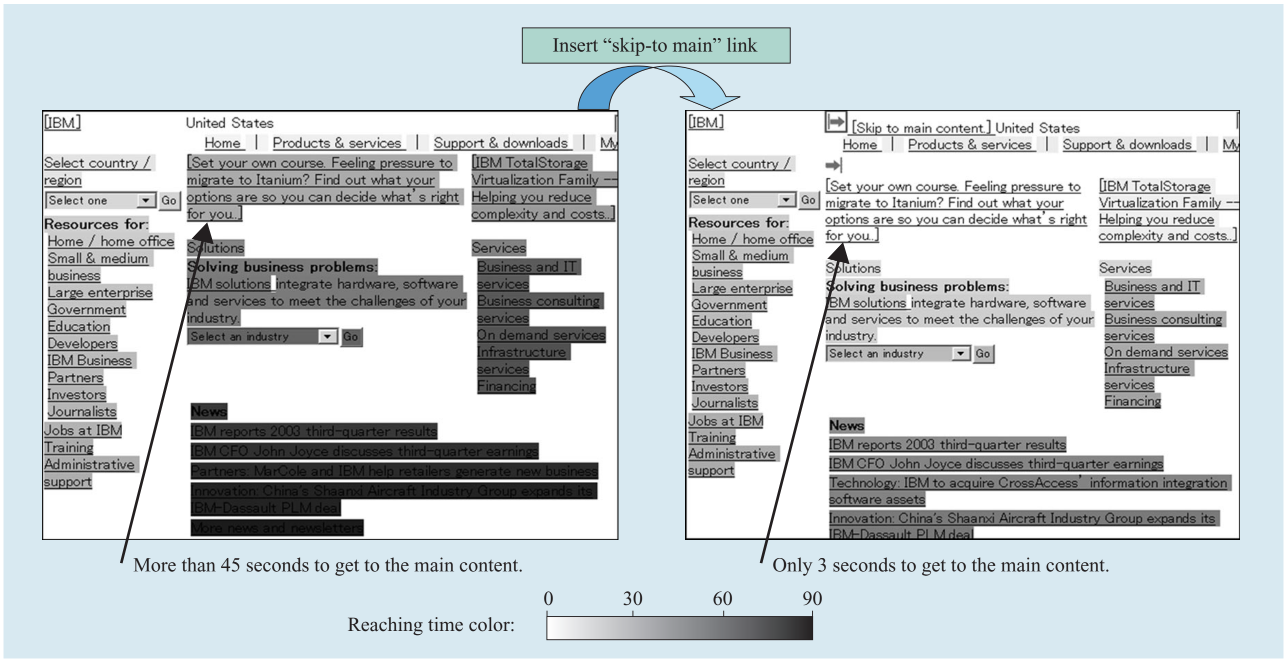
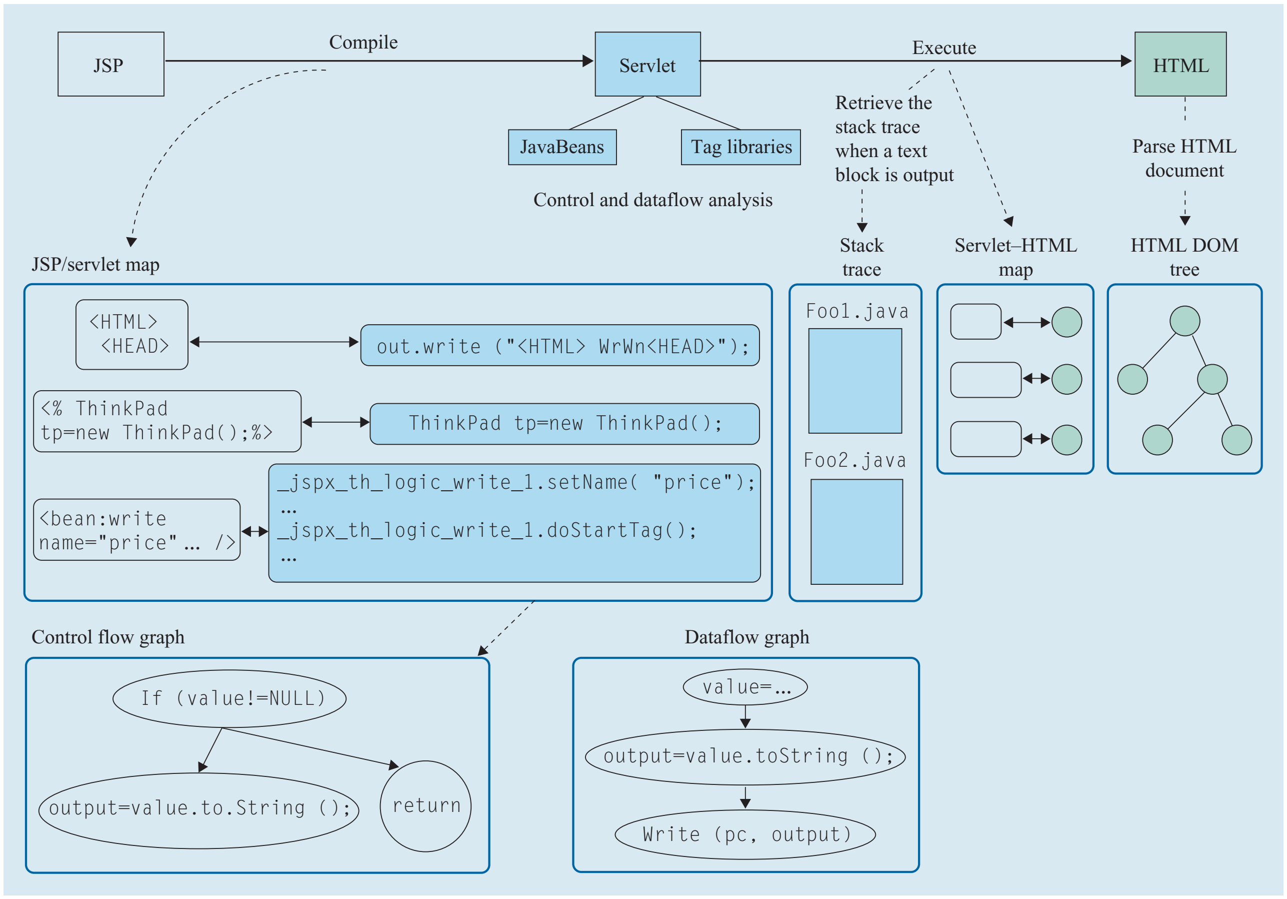

Insert "skip-to main" link
More than 45 seconds to get to the main content.
Only 3 seconds to get to the main content.
Reaching time color:
0
30
60
90
[Skip to main content]

Figure 7

Example of blind usability visualization.



JSP
Compile
Servlet
Execute
HTML
JavaBeans
Tag libraries
Control and dataflow analysis
Retrieve the stack trace when a text block is output
Parse HTML document
JSP/servlet map
Stack trace
Servlet–HTML map
HTML DOM tree
<HTML> <HEAD>
out.write ("<HTML> WrWn<HEAD>");
<% ThinkPad tp=new ThinkPad();%>
ThinkPad tp=new ThinkPad();
<bean:write name="price" ... />
_jspx_th_logic_write_1.setName("price"); ... _jspx_th_logic_write_1.doStartTag(); ...
Foo1.java
Foo2.java
Control flow graph
If (value!=NULL)
output=value.to.String ();
return
Dataflow graph
value=...
output=value.toString ();
Write (pc, output)

Figure 8

Accessibility Observer system architecture.

practical page renovation tasks. We are also conducting some usability testing.

Accessibility Observer

Server-side Java, such as servlets and JSP, are becoming widely used for Web application platforms. These technologies enable developers to provide various applications through the Web and to extend the role of the Web from being an information infrastructure to being a social infrastructure. Such applications are widely used, but many of them are not accessible for people with disabilities and senior citizens. If they were to become accessible, it would be extremely useful and valuable for these groups of people.

However, the improvement of accessibility for these server-side Web applications by using existing accessibility checkers is quite difficult. For server-side Java, each HTML document is generated from a Java program called a Java servlet. If an accessibility checker—one that provides problem reports based on the line numbers of the HTML document—detects an inaccessible part in the generated HTML, Web developers have to search through the Java code to find the corresponding positions where the HTML section was generated.

For an application using JSP, the matter becomes even more complicated. JSP allows a Web developer to easily create a Web application by separating the application logic from its design. Therefore, a typical JSP page integrates several modules, such as JSP tag libraries or JSP includes, for the sake of fast development and easy maintenance of the system. For example, developers can break a Web page down into reusable parts—such as `header.jspf`, `footer.jspf`, or `navigation.jspf`—and they can then assemble Web pages using these parts as JSP includes. Moreover, there are some frameworks for making templates and assembling presentation pages from component parts [32–34]. In these frameworks, many JSP-based pages use more than ten components. Before the actual execution of the JSP, it has to be compiled into a servlet. Thus, it is

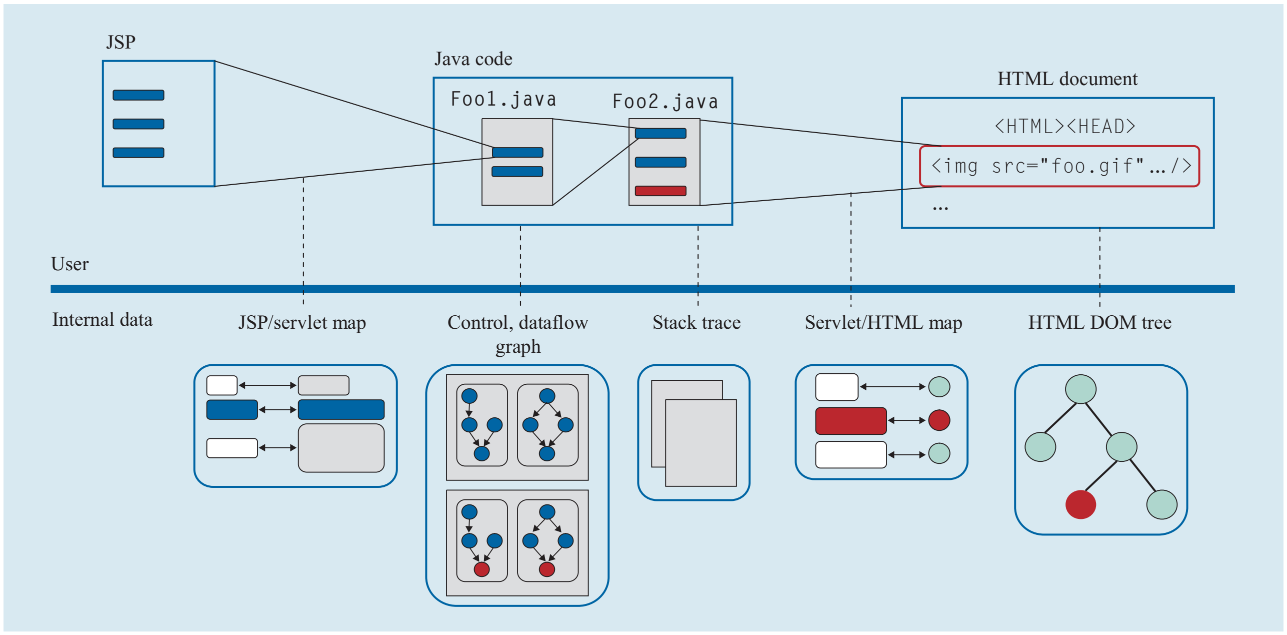


Figure 9

Tracing back to the point of origin.

difficult for developers to find the generating position of the inaccessible part.

To address this situation, Accessibility Observer for server-side Java was developed at TRL. Before execution, Accessibility Observer prepares a mapping between the JSP and the compiled servlet. At runtime, Accessibility Observer observes the JSP/servlet writer and creates a mapping between fragments of generated HTML and the corresponding code positions of the servlet (**Figure 8**).

By using these mappings, Accessibility Observer narrows the search range down to a point. For instance, if a user indicates the position in an HTML document (the red sections in **Figure 9**), Accessibility Observer traces it back to the servlet and back to the related JSP or taglibs. Analysis of control flow and dataflow can also be used to narrow the range of corresponding modules and source code.

Although the development of Accessibility Observer was originally motivated by accessibility needs, it can also be used as a general support tool to analyze, debug, or modify Java-based dynamic Web content. It allows Web developers to easily modify the attributes of the objective element or to find the position of the source code that creates the data in a particular table.

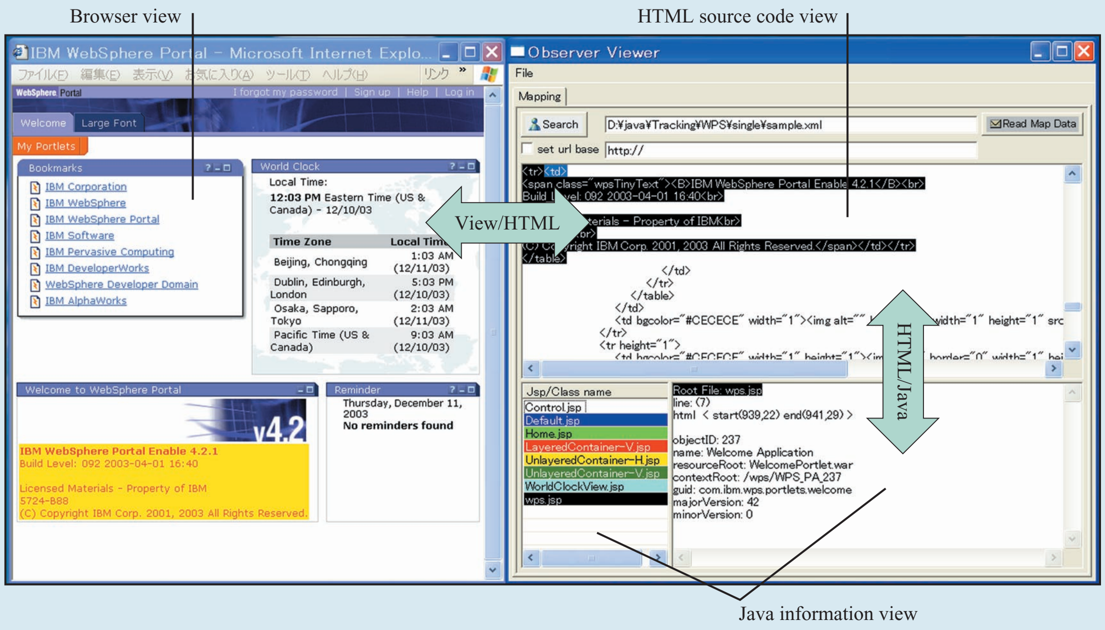
Figure 10 shows an example using Accessibility Observer. This sample client has a browser view, an HTML source code view, and a Java information view. In the browser view, users can click any position to highlight it, and the corresponding position of the HTML is also highlighted in the HTML source code view on the right side. At the same time, in the Java information view, the designated JSP/servlet information, such as the package name, file name, line number, and so on, appears. Users can click on some part of the HTML to locate the corresponding position in the browser view while also seeing the associated JSP/servlet information.

In this way, users can easily understand the correlations among the browser view, the HTML, and the Java source code. This helps developers to analyze, debug, and modify Java-based dynamic Web content.

Conclusions

This paper presents an overview of accessibility technologies developed at the IBM Tokyo Research Laboratory. Home Page Reader and accessibility transcoding technology were reviewed, and Accessibility Designer was described. Accessibility Designer makes it possible for page designers to learn about disabilities, because it demonstrates the accessibility issues faced by blind and low-vision people. With it, designers can create much more accessible and usable Web pages at the time of authoring. Accessibility Observer makes it easier for Web developers to detect accessibility problems by mapping an HTML position and the corresponding Java source code position.

There are many research topics to be considered in the future. First, techniques for client-side dynamic content,



```
D:¥java¥Tracking¥WPS¥single¥sample.xml
http://

<tr><td>
<span class="wpsTinyText"><B>IBM WebSphere Portal Enable 4.2.1</B><br>
Build_L  vel: 092 2003-04-01 16:40<br>

       terials - Property of IBM<br>
         br>
(C) Co  yright IBM Corp. 2001, 2003 All Rights Reserved.</span></td></tr>
</table>
                    </td>
                  </tr>
                </table>
              </td>
              <td bgcolor="#CECECE" width="1"><img alt=""      width="1" height="1" src
            </tr>
            <tr height="1">
```

Jsp/Class name
Control.jsp
Default.jsp
Home.jsp
LayeredContainer-V.jsp
UnlayeredContainer-H.jsp
UnlayeredContainer-V.jsp
WorldClockView.jsp
wps.jsp

```
Root File: wps.jsp
line: (7)
html < start(939,22) end(941,29) >

objectID: 237
name: Welcome Application
resourceRoot: WelcomePortlet.war
contextRoot: /wps/WPS_PA_237
guid: com.ibm.wps.portlets.welcome
majorVersion: 42
minorVersion: 0
```

Figure 10

Example of the use of Accessibility Observer.

such as JavaScript**, are widely used these days, and making the resulting pages accessible presents significant and difficult challenges. In some cases, the accessibility technologies described in this paper can be applied to such content, but much more effort is still required to ensure the accessibility of Web pages based on these techniques. Second, from the point of view of the Web site owner, site-wide technologies that deal with a number of—or all of—the pages in the Web site are also useful. They can improve the accessibility of the Web site, for instance by detecting problems with page templates and by finding specific subdirectories that have larger numbers of inaccessible pages than other parts of the site. Finally, as the Web becomes a place not only to acquire information, but to conduct general activities, such as shopping, we need many more studies of technologies to find ways to make it easier for senior citizens and people with disabilities to interact with the Web and conduct activities online.

The environment surrounding accessibility will change in the future as well. However, our goal is not to simply follow such changes, but to establish the process of ensuring accessibility for every aspect of information technology. We are striving to make steady progress toward that goal through our research activities.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Sun Microsystems, Inc.; W3C host institutions: Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, or Keio University; isSound Corporation, Carnegie Mellon University; Freedom Scientific; Ai Squared, Inc.; or Watchfire Corporation in the United States, other countries, or both.

References

1. A. Koide, C. Asakawa, and N. Suzuki, "Research on Computer Aids for the Visually Disabled," *Computers as Our Better Partners: Proceedings of the IISF–ACM Japan International Symposium*, March 1994, pp. 66–69.
2. C. Asakawa, "Braille Information Network: Braille Forum," *Computers as Our Better Partners: Proceedings of the IISF–ACM Japan International Symposium*, March 1994, pp. 148–149.
3. C. Asakawa and T. Itoh, "User Interface of a Home Page Reader," *Proceedings of the 3rd International ACM SIGCAPH Conference on Assistive Technologies (ASSETS '98)*, April 1998, pp. 149–156.
4. Web Content Accessibility Guidelines 1.0; see *http://www.w3.org/TR/WCAG10/*.
5. Section 508 of the Rehabilitation Act; see *http://www.section508.gov/*.
6. Developer guidelines for Web Accessibility; see *http://www.ibm.com/able/guidelines/web/accessweb.html*.
7. J. C. De Witt and M. T. Hakkinen, "Surfing the Web with pwWebSpeak," *Proceedings of the 1998 Technology and Persons with Disabilities Conference*; see *http://www.csun.edu/cod/conf/1998/proceedings/159.htm*.
8. C. Asakawa and T. Itoh, "User Interface of Nonvisual Table Navigation Method," *CHI '99, extended abstracts on Human Factors in Computing Systems*, May 1999, pp. 214–215.

9. IBM Corporation, Home Page Reader 3.0; see *http://www.ibm.com/able/solution_offerings/hpr.html*.
10. J. T. Richards, V. L. Hanson, and S. Trewin, "Adapting the Web for Older Users," *Proceedings of HCI International 2003 (Universal Access in HCI)*, June 2003, pp. 892–896.
11. IBM Corporation, Easy Web Browsing Solution; see *http://www.research.ibm.com/trl/projects/access_c/ewb.htm*.
12. American Foundation for the Blind, My AFB; see *http://www.afb.org/*.
13. BBC Education, Betsie Home Page; see *http://www.bbc.co.uk/education/betsie/*.
14. UsableNet Inc., LIFT Text Transcoder; see *http://www.usablenet.com/products_services/text_transcoder/text_transcoder.html*.
15. IBM Corporation, WebSphere Transcoding Publisher; see *http://www.ibm.com/software/webservers/transcoding/*.
16. H. Takagi and C. Asakawa, "Transcoding Proxy for Nonvisual Web Access," *Proceedings of the 4th International ACM SIGCAPH Conference on Assistive Technologies (ASSETS 2000)*, November 2000, pp. 164–171.
17. C. Asakawa and H. Takagi, "Annotation-Based Transcoding for Nonvisual Web Access," *Proceedings of the 4th International ACM SIGCAPH Conference on Assistive Technologies (ASSETS 2000)*, November 2000, pp. 172–179.
18. T. Sakairi and H. Takagi, "An Annotation Editor for Nonvisual Web Access," *Proceedings of HCI International 2001 (Universal Access in HCI)*, August 2001, pp. 982–985.
19. H. Takagi, C. Asakawa, K. Fukuda, and J. Maeda, "Site-Wide Annotation: Reconstructing Existing Pages to be Accessible," *Proceedings of the Fifth International ACM SIGCAPH Conference on Assistive Technologies (ASSETS 2002)*, July 2002, pp. 81–88.
20. K. Fukuda, H. Takagi, J. Maeda, and C. Asakawa, "An Assist Method for Realizing a Web Page Structure for Blind People," *Proceedings of HCI International 2003 (Universal Access in HCI), June* 2003, pp. 960–964.
21. H. Takagi and C. Asakawa, "Page-Customization Allowing Blind Users to Improve Web Accessibility by Themselves," *Proceedings of HCI International 2001 (Universal Access in HCI)*, August 2001, pp. 1003–1007.
22. J. Maeda and M. Kobayashi, "WebDigest: Visually Enhanced Web Page Representation with Layout Preservation," *Research Report RT-0345*, 2000; IBM Tokyo Research Laboratory, 1623-14 Shimo-tsuruma, Yamato-shi, Kanagawa-ken 242-8502, Japan.
23. J. Maeda, K. Fukuda, H. Takagi, and C. Asakawa, "WebDigest: Layout-Preserving Visually Enhanced Web Pages," *Proceedings of the 2003 Symposium on Applications and the Internet (SAINT 2003)*, January 2003, pp. 418–421.
24. Freedom Scientific, JAWS; see *http://www.freedomscientific.com/index.html*.
25. Ai Squared, Inc., ZoomText; see *http://www.aisquared.com/*.
26. Vischeck; see *http://www.vischeck.com/*.
27. Colorfield Software, Xproof; see *http://www.colorfield.com/xproof/*.
28. Techniques for Accessibility Evaluation and Repair Tools; see *http://www.w3.org/TR/AERT#color-contrast*.
29. Watchfire Corporation, Bobby; see *http://bobby.watchfire.com/*.
30. L. R. Kasday, "A Tool to Evaluate Universal Web Accessibility," *Proceedings of the 2000 Conference on Universal Usability (ACM)*, November 2000, pp. 161–162.
31. WAVE 3.0 Accessibility Tool; see *http://www.wave.webaim.org/*.
32. Apache Software Foundation, Jetspeed; see *http://jakarta.apache.org/jetspeed/site/*.
33. Apache Software Foundation, Tiles; see *http://jakarta.apache.org/struts/userGuide/dev_tiles.html*.
34. IBM Corporation, WebSphere Portal Server; see *http://www-106.ibm.com/developerworks/websphere/zones/portal/*.

Received October 16, 2003; accepted for publication February 17, 2004; Internet publication September 16, 2004

Junji Maeda *IBM Research Division, IBM Tokyo Research Laboratory, 1623-14 Shimo-tsuruma, Yamato-shi, Kanagawa-ken 242-8502, Japan (maeda@jp.ibm.com).* Mr. Maeda received an M.S. degree in information science from the University of Tokyo in 1989. He joined the Tokyo Research Laboratory in 1992. His research interests include Web accessibility, image and video processing, and digital media. He was named a Master Inventor at IBM Research in 2003. Mr. Maeda is a member of the IEEE Computer Society, the Association for Computing Machinery (ACM), the Information Processing Society of Japan (IPSJ), and the Institute of Electronics, Information and Communication Engineers (IEICE).

Kentarou Fukuda *IBM Research Division, IBM Tokyo Research Laboratory, 1623-14 Shimo-tsuruma, Yamato-shi, Kanagawa-ken 242-8502, Japan (kentarou@jp.ibm.com).* Dr. Fukuda received M.E. and Ph.D. degrees in information and computer sciences from Osaka University, Japan, in 1998 and 2000, respectively. He joined the Tokyo Research Laboratory in 2000. His research interests include Web accessibility, Web application development, and multimedia systems. Dr. Fukuda is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) and the Information Processing Society of Japan (IPSJ).

Hironobu Takagi *IBM Research Division, IBM Tokyo Research Laboratory, 1623-14 Shimo-tsuruma, Yamato-shi, Kanagawa-ken 242-8502, Japan (takagih@jp.ibm.com).* Dr. Takagi joined the Tokyo Research Laboratory in 1999. He has conducted research and development on nonvisual computer interfaces. In 2000 he received a Ph.D. degree from the Department of Science, University of Tokyo. Dr. Takagi is a member of the Association for Computing Machinery (ACM), the Information Processing Society of Japan (IPSJ), and the Japan Society for Software Science and Technology (JSSST).

Chieko Asakawa *IBM Research Division, IBM Tokyo Research Laboratory, 1623-14 Shimo-tsuruma, Yamato-shi, Kanagawa-ken 242-8502, Japan (chie@jp.ibm.com).* Dr. Asakawa joined the Tokyo Research Laboratory in 1985. Since that time she has conducted research and development on nonvisual computer interfaces for the blind, including the Home Page Reader in 1997. She received recognition from the Japanese Ministry of Health and Welfare in 1999 and was inducted into the Women in Technology International Hall of Fame in 2003. She received a Ph.D. degree from the Department of Advanced Interdisciplinary Studies in the Graduate School of Engineering at the University of Tokyo in 2004. Dr. Asakawa is a member of the IBM Academy of Technology, the Association for Computing Machinery (ACM), the Institute of Electronics, Information and Communication Engineers (IEICE), and the Information Processing Society of Japan (IPSJ).