Accessibility Designer: Visualizing Usability for the Blind

Hironobu Takagi, Chieko Asakawa
Tokyo Research Laboratory, IBM Japan
1623-14 Shimo-tsuruma
Yamato-shi, Kanagawa-ken, Japan
+81-46-215-4557, +81-46-215-4633
takagih@jp.ibm.com, chie@jp.ibm.com

Kentarou Fukuda, Junji Maeda
Tokyo Research Laboratory, IBM Japan
1623-14 Shimo-tsuruma
Yamato-shi, Kanagawa-ken, Japan
+81-46-215-4659, +81-46-215-4899
kentarou@jp.ibm.com, maeda@jp.ibm.com

ABSTRACT

These days, accessibility-related regulations and guidelines have been accelerating the improvement of Web accessibility. One of the accelerating factors is the development and deployment of accessibility evaluation tools for authoring time and repair time. They mainly focus on creating compliant Web sites by analyzing the HTML syntax of pages, and report that pages are compliant when there are no syntactical errors. However, such compliant pages are often not truly usable by blind users. This is because current evaluation tools merely check if the HTML tags are appropriately used to be compliant with regulations and guidelines. It would be better if such tools paid more attention to real usability, especially on time-oriented usability factors, such as the speed to reach target content, the ease of understanding the page structure, and the navigability, in order to help Web designers to create not simply compliant pages but also usable pages for the blind. Therefore, we decided to develop Accessibility Designer (aDesigner), which has capabilities to visualize blind users' usability by using colors and gradations. The visualization function allows Web designers to grasp the weak points in their pages, and to recognize how accessible or inaccessible their pages are at a glance. In this paper, after reviewing the related work, we describe our approach to visualize blind users' usability followed by an overview of Accessibility Designer. We then report on our evaluations of real Web sites using Accessibility Designer. After discussing the results, we conclude the paper.

Categories and Subject Descriptors

K.4.2 [**Social Issues**]: Assistive technologies for persons with disabilities, H.3.4 [**Online Information Services**]: Web-based services, H.5.2 [**User Interfaces**]: Evaluation/methodology, Graphical user interfaces (GUI), Screen design (e.g., text, graphics, color), Standardization, Style guides

General Terms

Design, Reliability, Human Factors, Standardization, Verification.

Keywords

Accessibility, visually impaired, blind, voice usability, accessibility checker.

1. INTRODUCTION

In the late 90s, Web accessibility received broad attention, and regulations and guidelines were published and adopted. One of the major focuses of these rules is blind usage. This contributed to increase attention on accessibility issues for the blind. In the United States, Section 508 of the Rehabilitation Act [1] was amended in 1998 and has been in effect since June 2001. This regulation has the power to force IT vendors to deliver accessible Web sites and Web applications to federal agencies. It was epoch-making regulation in the area of Web accessibility. Quite a few efforts have been made to create compliant Web pages, and various tools have been developed, such as authoring tools, evaluation tools [2, 3], and repair tools[4, 5]. Many organizations have been trying to enlighten people connected to Web development, not just the Web developers and designers, but also the site owners and business people. These efforts are invaluable for improving Web accessibility and widening the production of compliant pages.

In spite of these efforts, we unfortunately find that serious usability issues still exist in many Web pages, even for some that are compliant with the basic regulations[14]. For example, all of the regulations and guidelines call for Web designers to link an alternative text (ALT text) with each image. However, it is well known that this instruction is easily misunderstood and many Web designers tend to insert meaningless words for the obligatory ALT text, such as "blank space", "shadow of image", "photo", or "image". In spite of the severity of these issues, most accessibility checkers do not have any function to check these inappropriate ALT texts, because most of the regulations and guidelines do not cover these practical issues.

Another example is problems with heading tags. Heading tags (H1, H2, etc.) are defined as key elements to give structure to pages for ease of navigation by users, including blind users. If heading tags are "appropriately" embedded into pages, blind users can easily grasp an overview of the content in a page, and navigate through each chunk of content. However, most of the accessibility checkers do not have functions for checking heading tags. Some checkers warn Web designers about missing heading tags, but do not have any ability to check the appropriateness of these tags.

We classified these problems into three categories:

1) Too much focus on compliance to the guidelines, but not on real usability.

Regulations have contributed to improving accessibility technology, but compliance is becoming the objective of the checkers, even though "usability" is supposed to be the most important objective for accessibility technologies.

2) Relying on syntactic checking of Web pages.

Current checkers only rely on syntactical checking technique to detect accessibility issues, so the checkable errors are limited to the level of the tag description layer.

3) No attention to "time-oriented aspects" of users' operations.

Users almost never sit listening to speech output passively. They move and jump in a page by using various types of jump keys that are built into voice browsers. They create their mental models of the pages through this process and try to logically navigate through the pages to get to their target information. This "time-oriented aspect" of usability is crucial to achieve voice usability, but current checkers do not consider this aspect.

It is certain that the most effective method to detect and solve these issues is to conduct usability testing on target pages. This method, however, takes time, costs too much, and is inconsistent, and therefore it is almost impossible to test thousands of pages on a real site. The guidelines recommend that Web designers check their pages by accessing their pages by using screen readers [6, 7]. However, it's quite impractical for Web designers to even attempt to experience the same situation as blind users. For example, screen readers have a lot of complicated combination keys (similar to keyboard shortcuts), and blind users become experienced not only in the operation of their screen readers. In addition, they usually develop stronger listening abilities than the average sighted person [8].

Therefore, we decided to provide an easier and more effective way to evaluate a page at a glance by developing a visualization feature that tries to supplement or solve these three problems with the current checkers. We developed the Accessibility Designer (aDesigner), which has capabilities to visualize blind users' usability by using colors and gradations. The visualization function allows Web designers to grasp weak points in their pages, and to recognize how accessible or inaccessible their pages are at a glance.

In this paper, we will first give an overview of current accessibility checkers and related work. Then we will propose a new approach to check blind usability, Blind Usability Visualization, followed by an implementation example with a disability simulation tool, the Accessibility Designer. We then evaluate real sites on the Web using Accessibility Designer.

2. RELATED WORK

Ivory and Hearst [9] classified automatic usability checking methods into five categories with each category classified into several subcategories. They mentioned existing accessibility checkers, such as Bobby [2] and Lift [3], under "Inspection – Guideline Review." These tools focus on checking compliance to guidelines and reducing the workload to repair them to be compliant. From the usability point of view, there are wide varieties of possibilities in other categories.

We think that one of the most promising approaches is "disability simulation." Persons with disabilities are accessing the Web using totally different environments from non-disabled users. This is the point of how Web designers imagine, recognize and understand the disabled Web access experience. For this purpose, disability simulation has possibilities to help Web designers to experience a similar experience to having disabilities.

For low-vision simulation, Vischeck [10] and our Accessibility Designers' low-vision mode [11] have functions to create low-vision users' views based on image-processing techniques. However, for blind simulation, no tool was available before we created ours. Guidelines recommend checking pages by using voice browsers, such as Jaws [7] or Homepage Reader [6], but it is very hard to have an experience similar to being blind, since these tools for blind users require learning a lot of key combinations, and since the cognitive abilities of blind users are developed differently compared with sighted users.

A-Prompt [4] offers a function to check the reaching order for tables by presenting each table cell one-by-one, like the way blind users access Web pages as a serialized format. The WAVE [12, 13] has a function to indicate the reading order of each text block in a page by using small numbered labels on each text block, so Web designers can check the reading order of the text blocks. However, the numbering method is not intuitive for the sighted. In addition, it is very hard for Web designers to recognize practical usability, such as "How long does it take from the top to the main content?" or "How effective are the skip-navigation links?"

3. PROPOSAL FOR BLIND USABILITY VISUALIZATION

In Introduction, we described three problems of current checkers, too much focus on compliance to the guidelines, relying on syntactic checking of Web pages, and giving no attention to the time-oriented aspects of users' operations. We also indicated that usability testing on target pages is very effective but not practical for developing commercial-level sites. Only automatic checkers still have the power to create accessible Web environments for the blind. With such automatic methods, it could be possible to make a Web site compliant with the guidelines, but it is still hard for Web designers to learn the real problems of blind usage. The compliance is the letter of the law, but it is much more important to understand the real meaning of the law to establish the real usability of the Web for the blind.

Therefore, we have proposed a new method, "Blind Usability Visualization". This method aims at allowing Web designers to recognize their pages' usability "at a glance", including the time-oriented aspects as well as to understand the real usability of blind usage. This method consists of the following three features.

1) Presenting the reaching time to each part of a page by using background "colors".

As discussed in Introduction, the time-oriented aspect, especially the navigability of a page, is a key factor to provide usable Web pages. Therefore, this method fills in colors on a Web page by analyzing how long it takes from the top of the page to each part of the page. In this paper, we define this time as the "reaching time". Figure 1 shows the visualized snapshot that is referenced later in this section. This function allows Web designers to recognize various types of usability issues, such as the lack of a skip-to-main content link (abb. skip-link), missing heading tags, and so on, as well as to learn about the importance of these elements. The calculation method for reaching time is described in the next section.

2) Indicating accessible or inaccessible "areas" with color-filling

Current checkers mainly use "icons" to indicate accessible or inaccessible parts in a browser view [2, 3]. In addition, they insert icons to indicate areas, such as the beginning or the end of a heading tag, a label, or a table. Therefore, we used a coloring approach to indicate these areas.

3) Presenting the text information extracted or generated by standard voice browsers, while retaining the fundamental visual layouts.

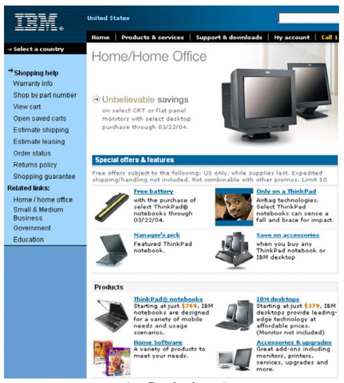
The text information here means the undisplayed information from the regular browser view, which is extracted or generated by the voice browsers to provide verbal information about specific elements for blind users. (Examples are alt texts for images and image maps, notification messages for starting or ending forms, etc.) It is recommended to test pages to check this verbal information as well by using voice browsers, but it is difficult for sighted Web designers to recognize the corresponding positions in the visual browser view with such verbal information just by listening to it. Therefore, the basic visual layout structure is retained in our visualization view for ease of checking, while presenting this verbal information simultaneously in the same view.

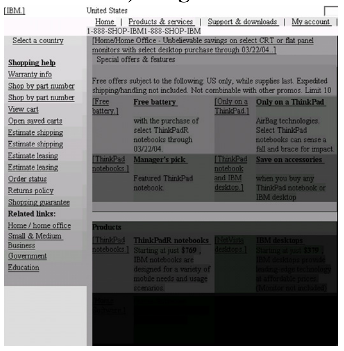
Among these features, 1) the reaching-time presentation is the most effective and novel feature to help improve the blind users' usability, and the detailed description follows.

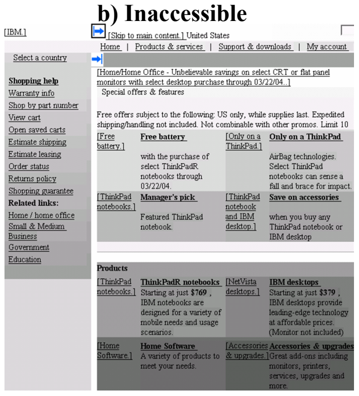
Figure 1 shows an example of visualization. The gradations of the background color show the times to reach each element in a page with standard voice Web browsers, white for 0 seconds and black for 120 seconds or longer from the top.

The system calculates the times from the top of a page to each element when a user uses voice output, then visualizes these times by using color gradations. Figure 1 (b) shows an inaccessible page, since there are no heading tags or any intra-page links, such as the skip-link and the page index links. The main content area (here including a title and a large image) is shaded as black, showing it will take a long time to arrive at this area. Figure 1 (c) shows an improved page with a skip-link. The main content area is shaded with a light gray color, showing how blind users can now access the main content easily. In this way, Web designers can recognize how fast or slowly blind users can reach the main content by referring to the differences in these colors. The darker color indicates taking a longer time, while a lighter color indicates a shorter time.

The insertion of a "skip-to-main-content" link, however, only helps accessing the main content area. You can see other areas filled with black, which means that these areas require more than 120 seconds for users to reach them. On the other hand, appropriate heading tags help blind users to build their mental model of a page by providing quick access to each part of the page. In Figure 1 (d), there are several headings (highlighted with blue) followed by brighter color areas. The standard voice browsers provide functions to jump directly to each heading tag in a page, so this allows blind users to navigate through each heading quickly and it helps them to grasp the page overview


a) Original


b) Inaccessible


c) With skip-link

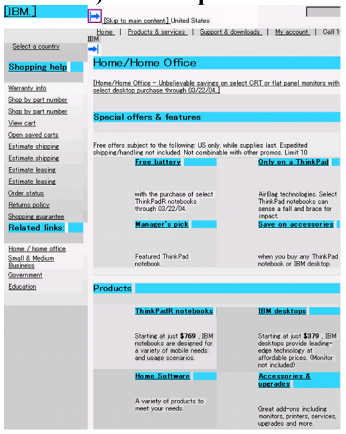

d) With headings

Figure 1. Effectiveness of reaching-time visualization
(white for 0 seconds and black for 120 seconds or longer from the top of a page)

effectively. An important point is to have lighter colors at the beginning of each block of information. It is not an issue that the later part of each block becomes darker, since this is unavoidable as blind users listen to the information in one dimension, unlike the two dimensions of vision. These three pages have just the same appearance when rendered by standard browsers (Internet Explorer, etc.), but they are significantly different from the blind usability point of view when they are rendered by standard voice browsers.

Our background-color-based reaching-time visualization method has the power to expose the following problems:

- Existence, availability and appropriateness of skip-links.

  As mentioned above, Web designers can easily recognize the existence of a skip-link by looking at the visualization view. If a skip-link exists at the top of the page and the main content is filled with a dark color, it means there is a problem, such as a broken skip-link or a missing anchor. In addition, Web designers can easily recognize the appropriateness by looking at the color of the main content (target) area. Our visualization is the first method to allow Web designers to check or evaluate these "skip-link" related issues.

- Existence and appropriateness of heading tags

  Figure 1 (d) shows an example of this function. Web designers can check the existence of the blue-areas and their appropriateness by looking at the background colors.

- Inappropriate content orders

  In the visualization view, the color gradation indicates the reading order of standard voice browsers. If visually closely laid out and undividable pairs of text elements have different gradation levels, it means they will not be read as paired texts. Therefore it may be necessary to redesign the tag structure to combine these text elements.

We considered two other types of reaching time visualization, a position-based method and a "time map method". The position-based method split the visual page into blocks and arranged them to fit in time-oriented graph based on the reaching time calculations. This method can allow Web designers to check the "exact" reaching times, but it mixes up the original page's layout. The "time map" is used to visualize geographical information, such as arrival times from Tokyo for each major city in Japan, by distorting the original map. We think this method can also be applied for the purposes.

4. IMPLEMENTATION – THE ACCESSIBILITY DESIGNER

We implemented the visualization method into a visual accessibility checker, "Accessibility Designer" [15]. This tool aims at providing Web designers an environment to gain experience in how low-vision people see a Web page, and in how blind people access a Web page by using voice browsers. In other words, this tool aims at simulating disabilities to check the pages real usability while authoring.

This tool has two modes, one for a low-vision simulator and another for blind usability visualization. The low-vision simulator can create simulated views of original page by using image processing techniques [11]. First, it creates an image of the target page by rendering the page. Then, it simulates the view of low vision people and detects inaccessible parts of the page by applying image analysis techniques. In this paper, we are focusing on blind usability, so we will describe the blind mode in greater detail.

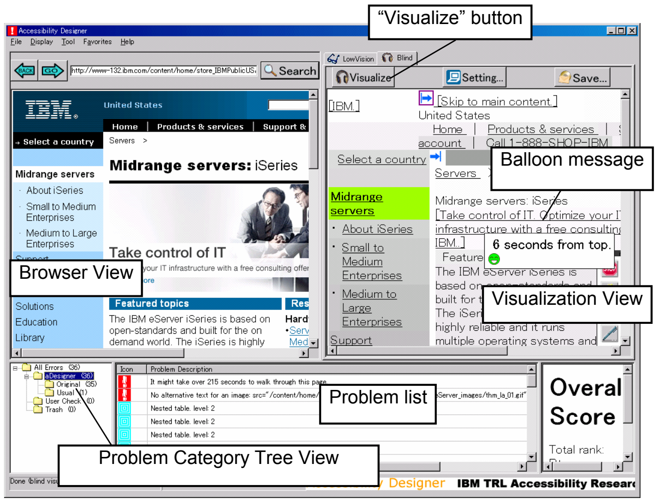


Figure 2. User interface of aDesigner

4.1 User Interface

Figure 2 presents the user interface of the blind mode. The upper left area is an embedded Web browser, not an image, so that the user can actually navigate the Web. When a user (Web designer) selects a target page in the left pane and presses the "Visualize" button, a visualization view will be displayed in the upper right pane. When a user moves the mouse over the visualization view, a balloon message and icon, which indicate the exact reaching time and acceptability of the reaching time, will be shown and follow the movement of the mouse. Only three types of icons are used to indicate an error position, an intra-page link, and an intra-page link destination. When a user clicks an intra-page link icon, an "arrow" appears between the link position and destination position.

Accessibility Designer also has an original error detection capability by using the voice browser engine (see Section "Automatic Error Detection"). When problems are detected, they will appear in the lower area, as shown in Figure 2. A tree view of the categorized problems is displayed in the lower left area. 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 and description. By selecting lines in this view, the positions of the corresponding problems are displayed in the visualization view.

4.2 Implementation

Most of the modules are written in Java, and C++ and VisualBasic are used to access the HTML source code from the browser view (left pane). We also used JavaScript to control the arrows and balloon messages. The SWT (Standard Widget Toolkit) [16] was used for the GUI (Graphical User Interface) library.

4.3 Visualization Process

Figure 3 shows the process to convert the original HTML source code into the visualization view.

1. When a user presses the "Visualize" button, the HTML source code of the target page (in the Browser View) is sent to the HTML parser, and the parser converts the HTML into a DOM tree structure.
2. The Voice Browser Simulation module creates a text rendering by using the voice browser engine [6]. Text items are associated with nodes in the DOM structure, so the following modules can utilize this information.
3. The Intra-page Link Analysis module analyzes the linkage structure of the intra-page links.
4. The Reaching-time Calculation module calculates the reaching times based on the user's operation model. The model provides basic values to estimate reaching time, such as default speech rate, time to switch between reading mode and heading navigation mode, and so on.
5. The CSS & JavaScript Insertion module inserts CSS to fill each element with graded colors corresponding to the reaching times. JavaScript is used for balloons and arrows.
6. Image-text Replacement replaces each image and form element with corresponding voice browser output text. Other visualizations (table headers, heading tags, form elements, adding intra-page link icons, etc.) are done by this module.
7. The final HTML file is sent to the Visualization View.

4.4 Reaching-time Calculation

Basically, the system calculates the reaching time to get to the target elements by using the voice browser engine.

Blind users usually use various types of commands to get to the specific information quickly, so usability should consider such accelerating operations in calculating the reaching time, not only the reading time for the text information. With the current implementation, the reaching time is effected by the existence of intra-page links and heading tags. So when these tags exist, the reaching times become shorter.

Figure 4 shows an example of typical intra-page link connections. There are 12 areas and 7 intra-page links. The calculation of reaching time can be regarded as a shortest path problem for a weighted directed graph. Figure 5 shows a graphic representation of Figure 4. Nodes are the indicated areas, the numbers are area identification numbers, the thin lines show the tag order, the bold lines are forward links, and the dotted lines are backward links. Each node has a weight representing the time required to read through the corresponding area. For example, the shortest path to get to Area 6 is the route 1, 2, 8, 9, 10, 11, and then 6. This will take a minimum of 16 seconds (2 + 2 + 3 + 3 + 3 + 3). In addition, heading navigation time should be taken into account, if headings exist. Access keys also should be taken into account. Therefore, we used a custom shortest-path-finding algorithm.

The reading time could be calculated for various speech rates. When a user is advanced, he or she might use a faster rate, such as 360, 380, or 400 words per minute (wpm) or even faster [8]. However a novice user might use a slower rate, such as 180 or

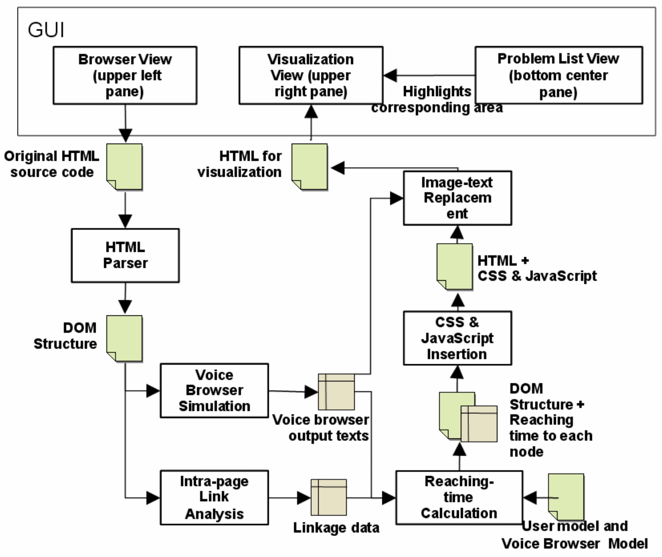

GUI
Browser View (upper left pane)
Visualization View (upper right pane)
Problem List View (bottom center pane)
Highlights corresponding area
Original HTML source code
HTML for visualization
Image-text Replacement
HTML Parser
HTML + CSS & JavaScript
CSS & JavaScript Insertion
DOM Structure
Voice Browser Simulation
Voice browser output texts
DOM Structure + Reaching time to each node
Intra-page Link Analysis
Linkage data
Reaching-time Calculation
User model and Voice Browser Model

Figure 3. Block diagram for blind usability visualization (for Accessibility Designer)

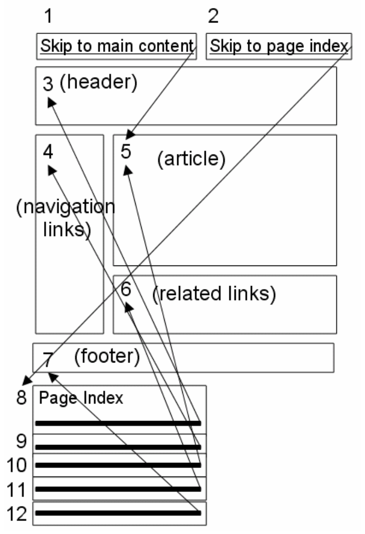


Figure 4. Example of intra-page link connections

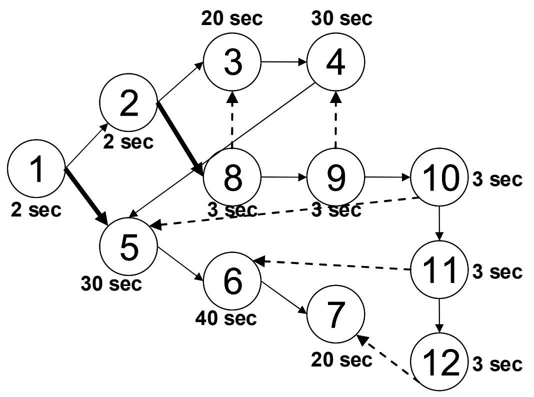


Figure 5. Graph representation of intra-page link connection in Figure 4

200 wpm. With the current implementation, only the novice user operations are simulated, so the reading time is calculated using 180 wpm.

4.5 Automatic Error Detection

In addition to visualization, this tool has original error detection capabilities, not only for compliance but also for usability.

Existence and availability of a skip-link.

The tool checks if a skip-link exists in a page. In addition, when it exists, it checks if the destination anchor exists and is appropriate. To evaluate the skip-link, several heuristic rules are used. When the intra-page link is located at the top of a page it is assumed to be the skip-link. In addition, it checks if that ALT text includes specific words, such as “skip”, “main”, or “jump”. We will describe a test of this function in the section “Evaluation”.

Redundant texts.

The tool has a function to detect repetitive texts that appear only when a user uses voice browsers [14]. This is possible by using the voice browser engine directly to extract its internal text information. Such features are not integrated into other current checkers. The redundancy often happens because equivalent text or a text link is often available near the images or image links. This might happen because there are various descriptions in the guidelines related to alt texts and it confuses Web designers. Therefore, they may provide both alt text and equivalent text for an image. In such case, blind users are forced to listen to the same texts repeatedly, e.g. "search" and "search". If Web designers could understand correctly how blind users use the alt texts, such redundant texts would not be used. Instead, they could input "null" text for some of the alt texts. The tool has visualization functions as well as automatic checking, so a user can interactively check the severity of the issues visually, and then confirm the effectiveness of modification afterwards.

Inappropriate ALT texts.

The tool has a function to detect inappropriate ALT texts based on a customizable taxonomy. The inappropriateness of alt texts most often happens because Web designers understand that all images need to have alt texts and current checkers check if there is some alt text for each image. The result is that some useless alt texts such as "spacer", "line", "button", and so on are often used. Instead, those images should also be described with "null" for the alt texts, since those images are not actually important. Standard voice browsers ignore "null" alt texts. We check for these problems with alt texts to notify the designers about proper usage of alt texts.

Too long reaching time.

The tool warns if the maximum reaching times on a page exceed a threshold time.

5. EVALUATION

We conducted three trials for evaluating the effectiveness of the Accessibility Designer and its visualization functions. The first trial was an investigation of reaching times to the main content areas in news articles on popular news sites, the second was an investigation of the number of broken or missing skip-links on sites that have a policy of providing skip-links, and the third was a practical trial to apply this tool in the development process of a Web application. We will briefly introduce and discuss these trials.

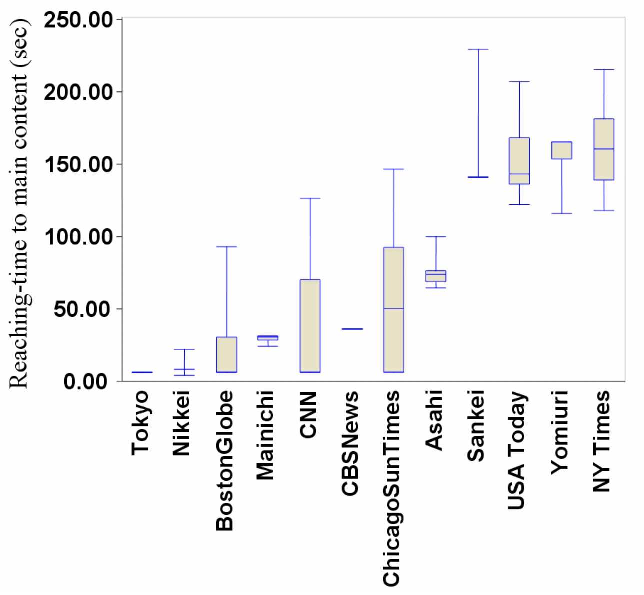
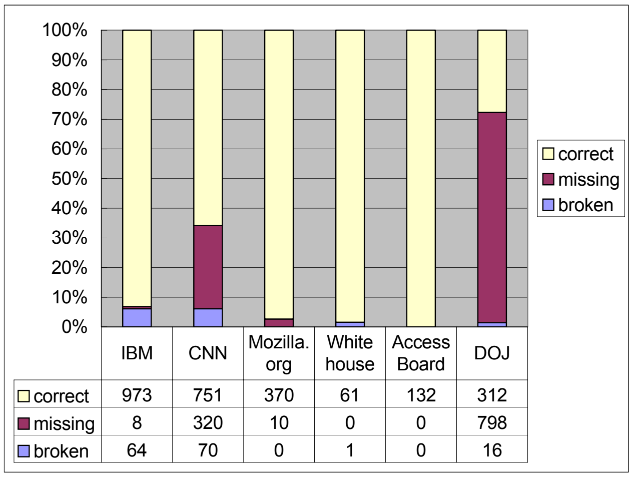


Figure 6. Comparison of average reaching time to main content area in article pages (box plot, order by average value)

5.1 Reaching Times to the Main Content Area in Popular News Sites

Each news site has visually well designed page layouts, but the level of accessibility consideration varies depending on the site owner’s policies. Therefore, we tried to measure the average reaching times to the article areas in news article pages in order to compare their accessibility. We selected twelve major newspaper sites, six in the U.S and six in Japan. We randomly selected 20 article pages, which have typical (or standard) layout on each target site. Then each of these article pages was opened using Accessibility Designer. When the page was opened and visualized, our research assistant looked for the main content and moved the mouse cursor over it to find out the reaching time from the top of the page in the visualization view. She recorded the reaching-time for each page.

Figure 6 shows comparative results for these twelve sites. The Nikkei Shimbun is the most accessible newspaper site among the tested sites. These pages have heading tags and three skip links including a link to the article line. In these pages, blind users can easily navigate to the main content as well as to each article title by using heading navigation functions. The Tokyo Shimbun is also fastest, but these pages have basic accessibility issues such as missing ALT texts. The Boston Globe is also fast because of heading tags. The Mainichi Shimbun is relatively faster than other sites, since the site navigation links are located in the best place on the page. They might have selected such a layout by considering the accessibility, but if heading tags or a skip-link were used, the usability would be more effectively improved. CNN also has skip-links or heading tags, but the ratio of missing skip-links was high and the results (see plot) become scattered. The other tested sites do not use any heading tags or skip-links,



	IBM	CNN	Mozilla. org	White house	Access Board	DOJ
🟨 correct	973	751	370	61	132	312
🟥 missing	8	320	10	0	0	798
🟦 broken	64	70	0	1	0	16

Figure 7. Checking results of skip-links in accessible sites

and as a result, it takes about one to three minutes to get to the main content areas.

5.2 Number of Broken Skip-links in Accessible Sites

In Section 508 of Rehabilitation Act, it says that a method shall be provided that permits users to skip repetitive navigation links [1]. Therefore, most of federal agency sites have skip-links in their pages, and many private companies and non-profit organization sites have them. Skip-links can offer great usability, but it is very hard to maintain all skip-links so they work perfectly. We could easily find broken skip-links, which existed but which did not work or which only jumped back to the top of the page again. Such broken skip-links bother blind users and make the usability worse.

One reason might be a technical issue. It is necessary to maintain two elements, the link and the corresponding destination anchor to keep these links working. Another reason might be lack of a effective tool for checking the availability of skip-links. Therefore, we evaluated how many skip-links are broken or missing in a site where all pages have skip-links. These sites are accessible sites, since they have the skip-links, so the site owners are probably giving high attention to accessibility. From this trial, we would like to show the importance of the maintenance process and the importance of tools that can support appropriate skip-links.

We checked six sites, two from the private sector, one for a non-profit organization, and three for government sites. We used the Accessibility Designer Site-wide Edition, which has a function to crawl a target site and create visualization results for all of the pages, and generate a site-wide report of automatically detected errors. We filtered the results to make the results more accurate by checking visualization result manually.

Figure 7 shows the results of our evaluation. The “broken” means the number of pages with broken skip-links, and “missing” means pages without skip-navigation links. The “missing” pages include pages that do not require skip links, so the number of “broken” links is relatively more important in this case. Two private sector sites, IBM (www.ibm.com) and CNN (www.cnn.com) have almost the same broken-link rate, 6.1 percent. These two sites have a very large number of pages, so we can see how hard it is to maintain skip-links. However, we think that 6% is high enough for blind users to lose faith in the skip-links. The non-profit organization (mozilla.org) doesn’t have any broken links, and shows a great effort to create an accessible site. Two major government sites, Whitehouse (www.whitehouse.gov) and the U.S. General Services Administration (www.section508.gov) do not have any pages without skip-links, and we found only one broken link. The U.S. Department of Justice site (www.usdoj.gov) showed that many pages exist without “skip navigation.” We manually checked some pages and we found that more than half of them have simple layouts, and do not require skip-navigation links. However, the site still has a large number of missing-skip pages, and they are still needed to manage the content and layout more neatly.

Through this trial, we demonstrated the difficulty of maintaining skip-links and the effectiveness of the visualization and automatic checking functions of Accessibility Designer.

5.3 Applying for the Practical Development Environment

We used Accessibility Designer practically while developing an Intranet Web application. This application consisted of over 200 JSP (Java Server Pages) files and a few static HTML files. After the initial evaluation for accessibility of this Web application, it was predicted it would take more than a couple of months to make it compliant. However, it actually required only a few weeks to make it even better than merely compliant. Through this experience, we found the following advantages of the tool:

- It reduced the workload and costs for repairs. Web designers and developers, who did not know much about accessibility, could participate in the repair process, so the repair tasks were performed effectively. Even so, accessibility professionals still needed to join in the tasks, but the tool contributed to reducing the necessary number of such experts.

- It established a high level of accessibility with less usability testing with real users. The blind usability visualization could substitute for the initial stage of real usability testing, so that the time was reduced. Of course, real usability tests are still needed to expose unforeseeable problems. Nevertheless, it was effective in helping to complete the usability testing process in a short amount of time.

- Finally, the web application became really accessible and usable by focusing on the user experience and actual productivity. We received favorable feedback from blind employees immediately after the application was released.

6. CONCLUSION

This work was motivated by improving the Web’s accessibility, not only for providing compliance, but also for improving the usability and blind users’ productivity. After investigating problems and missing features in the current Web accessibility checkers and reviewing related work, we proposed a method for visualizing the blind users’ usability in a visual layout view. This fundamentally consists of three features, one for presenting the reaching times to each part of a page by using background “colors”, one for indicating accessible or inaccessible “areas” with color-filling, and one for presenting the text information extracted or generated by standard voice browsers, while retaining the fundamental visual layout. This new approach allows Web designers to check how fast or slowly blind users can reach the main content area of their pages. It also indicates how well and effectively the skip-links and heading tags can be used by blind users. Both user categories are visualized for sighted designers to recognize problems at a glance, instead of using long verbal descriptions. Providing such visually intuitive interfaces, we aimed at providing an effective learning tool for designers to understand the importance of the Web accessibility and the real meaning of the guidelines and regulations for Web accessibility. After proposing the new visualization approach, we described the implementation of the method on the Accessibility Designer.

Finally, we evaluated real sites using accessibility Designer. The first trials on investigating the reaching times to the main content showed that some sites take about one to three minutes, while others take only 10 seconds or so. This is because some sites provided accessibility features, such as skip-links and heading tags, but others do not consider accessibility at all. In the second group of trials, we checked skip-link related issues. Some sites provide skip-links, but some of them did not work because there were no destination anchors or the anchors were broken. The third trial, which was the use in a practical development process, clearly showed that it reduced the workload for repair and saves time and cost, while establishing the high usability.

In our future work, we are planning to develop a more flexible user model, which has an ability to simulate a variety of user behaviors. When a user is more experienced, he or she will use various operating commands to find the main content, not only listening to the voice output or using simple commands. Therefore, the user model needs to cover frequently used operations by advanced users, intermediate users, and novice users more precisely.

We have already distributed Accessibility Designer to Web designers and developers [15]. So far, we have received a variety of favorable feedback. Most of them requested us to make it seamlessly usable with other authoring tools. Therefore, we are considering how to integrate the Accessibility designer into a Web authoring tool.

The Web accessibility effort has been spread all over the world, but we are still far from realizing a truly accessible and usable environment on the Web for people with disabilities. We believe that our Web accessibility enabling tool will be one of the key factors to achieve this goal. We hope that we can deliver this tool to more Web designers and developers.

7. REFERENCES

[1] Section 508 of the Rehabilitation Act; see http://www.section508.gov/.

[2] Bobby, Watchfire Corporation; see http://bobby.watchfire.com/.

[3] LiFT, http://www.usablenet.com/

[4] A-Prompt, University of Toronto, http://aprompt.snow.utoronto.ca/

[5] Asakawa, C. et al, Automatic Web Content Accessibility Compliance Tool for Section 508, Proc. Technology and Persons with Disabilities conference (CSUN 2002), 2002.

[6] Asakawa, C., Itoh, T., User Interface of a Home Page Reader, Proc. the 3rd international ACM SIGCAPH conference on Assistive Technologies (ASSETS '98), pp.149-156, 1998.

[7] JAWS, Freedom Scientific Inc., http://www.freedomscientific.com/

[8] Asakawa, C., Takagi, H., et al. Maximum listening speeds for the blind, Proc. Conf. of International Community for Auditory Display 2003, 2003.

[9] Ivory, M.Y., Hearst, M.A, The state of the art in automating usability evaluation of user interfaces, ACM Computing Surveys (CSUR), Vol. 33 Issue 4, pp. 470 – 516, 2001.

[10] Vischeck, http://www.vischeck.com/

[11] Maeda, J., Fukuda, K., Takagi, H., Asakawa, C., Web accessibility technology at TRL, IBM Research and Development Journal, 2004. (on press)

[12] Kasday, L.R., A Tool to Evaluate Universal Web Accessibility, Proc. the 2000 conference on Universal Usability (ACM), pp. 161-162, 2000.

[13] WAVE 3.0 Accessibility Tool, see http://www.wave.webaim.org/.

[14] Theofanos, F.M., Redish, J., Bridging the gap: between accessibility and usability, Interactions, Vol 10 , Issue 6, pp. 36 – 51, 2003.

[15] aDesigner, http://www.alphaworks.ibm.com/tech/adesigner

[16] Standard Widget Toolkit, http://www.eclipse.org/swt/