ABSTRACT
At its most basic, the web allows for two modes of access: visual and non-visual. For the most part, our design attention is focused on making decisions that affect the visual, or surface, layer — colors and type, screen dimensions, fixed or flexible layouts. However, much of the power of the technology lies beneath the surface, in the underlying code of the page. There, in the unseen depths of the page code, we make decisions that influence how well, or poorly, our pages are read and interpreted by software. In this paper, we shift our attention beneath the surface of the web and focus on design decisions that affect nonvisual access to web pages.

Categories and Subject Descriptors
H.5.4 [Information Interfaces and Presentation (e.g., HCI)]: Hypertext/Hypermedia—Navigation, User issues.

General Terms
Design, Human Factors, Standardization

Keywords
Accessibility, Usability, Universal Usability, Universal Access

1. INTRODUCTION
In a web transaction, the first read of a page is by software that parses and acts upon the source code: for example, by rendering the page visually, reading the page aloud, or extracting and storing information about the page. The accuracy and effectiveness of software’s rendering and actions is affected by the quality and design of the source code.

In striving to achieve good design, we generally focus on visual presentation. Our efforts are aimed at designing a visual display that is usable and appealing, with little attention given to the source code since visual users are only indirectly affected by its design. However, nonvisual users, such as vision impaired users and search engine software, do not work with the visual display. Unlike visual users, their experience is directly affected by the design of the underlying source code. Organization, quality, and clarity influence how well software can read and interpret the source code. Nonvisual web access can be improved by applying the following guidelines for source code design.

Shneiderman defines universal usability as an approach to design that is focused on “enabling all citizens to succeed in using information and communication technologies to support their tasks” [18]. A focus on page code design improves the universal usability of web pages by addressing access challenges in a variety of contexts. For instance, the small viewport on mobile devices presents many of the same challenges as nonvisual access. This paper concludes with a discussion of how these guidelines can be applied to improve web access for mobile users.

2. GUIDELINES
Several factors influence the effectiveness of nonvisual web access. As with most applications of universal design [5], these factors improve access for all users, including visual users and users of mobile devices.

2.1 Integrity
The soundness and stability of document structure.

2.1.1 Use markup to describe document structure
Designers have a toolset of time-honored principles to communicate the structure of a document. Alignment and proximity convey information about the relatedness of elements. Typographic emphasis draws attention to important elements [10]. While effective, visual design is not a science. Structure is only implicit in the application of these principles. Additionally, principles are often misused, or abandoned in favor of more avant-garde approaches. At the end of the day, visual design conventions are meaningful to people who can see them, and even among those who can see them, are open to interpretation.

On the other hand, markup provides a means to explicitly define document structure. With markup, designers can embed information structure and relationships among page elements into content of a document. Encoded structure can be read by software, making possible “a web of data that can be processed directly or indirectly by machines,” envisioned by Berners-Lee [1].

HTML offers designers a set of tags for use in describing information structure of web documents. While limited, these tags add a layer of meaning, thereby enhancing software’s capacity to read and interpret web documents. In addition, software can use structural markup to provide additional functionality to the user, such as providing a document overview using heading tags, or announcing the number of items in a list.
Use the HTML toolset (Table 1), along with design principles (see subsection 2.2.2, below), to communicate document structure to nonvisual users.

<table>
<thead>
<tr>
<th>Element</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1, h2, h3, h5, h6</td>
<td>Headings</td>
</tr>
<tr>
<td>p</td>
<td>Paragraphs</td>
</tr>
<tr>
<td>blockquote</td>
<td>Quoted text</td>
</tr>
<tr>
<td>ul, ol, li</td>
<td>Unordered and ordered lists</td>
</tr>
<tr>
<td>table, th, tr, td</td>
<td>Tabular information</td>
</tr>
<tr>
<td>fieldset, legend, label</td>
<td>Form labels</td>
</tr>
<tr>
<td>em, strong</td>
<td>Emphasized words and phrases</td>
</tr>
<tr>
<td>cite</td>
<td>Citations (e.g., book titles)</td>
</tr>
<tr>
<td>abbr, acronym</td>
<td>Abbreviations and acronyms</td>
</tr>
<tr>
<td>code</td>
<td>Computer code</td>
</tr>
<tr>
<td>dfn</td>
<td>Defined term</td>
</tr>
</tbody>
</table>

2.2 Usability

The ease with which users can access content.

2.2.1 Eliminate unnecessary clutter

Signal-to-noise is a principle that measures the ratio between the signal, or relevant information, and the noise, or extraneous information, contained in a design. A high signal-to-noise ratio results in more effective communication, while a low ratio reduces usability because it asks users to filter out noise in order to receive and interpret information [10]. Many web pages have low signal-to-noise, as much of the display area is given over to the browser interface, advertising, navigation, and branding (Figure 1) [13].

2.1.2 Avoid meaningless and misleading markup

Web standards provide a formal grammar for constructing machine- and human-readable documents. If all web designers apply the same syntax, then all web documents are written in the same language, increasing the potential for connections among documents and data exponentially [2].

Coding to standards [22] is a relatively new practice among web designers, and is certainly not one that has been universally adopted. Many designers use markup to accomplish visual design effects, exert control, and achieve consistency across browsers. Additionally, web authoring tools don’t always encourage good coding practices or generate standards-compliant code. Many popular web sites are not designed according to web standards. Visual users are largely unaffected by non-standard code. A user cannot tell just by looking whether a page uses tables for layout. On the other hand, nonvisual users cannot help but experience the source code that lies below the surface. When reading web documents, software cannot discriminate between markup that is used purposefully and markup that is pressed into service for other than its intended purpose. For example, software cannot distinguish a table that is used for layout from a table containing data. Therefore, for nonvisual access:

- Avoid presentation markup. Tags such as <font>, <i>, and <b> do not convey meaning that can be interpreted by software. Instead, use structural markup to describe page elements and CSS to describe their presentation.
- Avoid using text for visual purposes. This includes punctuation such as horizontal bars or brackets to separate links. Software cannot recognize when punctuation is only relevant visually. Instead, use images and CSS for visual effects.
- Avoid misappropriating tags. Using tables to create columns or <blockquote> to indent text weakens the integrity of document structure and is likely to be misinterpreted by software. Instead, use CSS for layout [8].

![Figure 1: Visual signal-to-noise ratios for articles on popular news sites](image-url)
For nonvisual users, contending with clutter is difficult because software cannot readily discriminate between signal and noise. There is no <advertisement> or <company logo> tag that software can quickly skip over, and no <content> or <navigation> tag that software can use to find its way directly to content. Unlike visual users, who can identify and largely ignore clutter, nonvisual users must consider each element before making a determination whether it is signal or noise [7].

Minimize or eliminate elements that are not directly related to page content.

2.2.2 Communicate relationships among elements

Navigating an information space requires that we understand how elements in the space relate to one another. In a visual context, we use design to delineate elements and describe their relationship to one another—for example, by enclosing elements within a bounding box and labeling the box.

Differentiate elements using structural markup: for example, use the FIELDSET tag to group related form elements and label the group using LEGEND. With markup fails, use the following design principles to group elements [10, 21]:

- Proximity: Elements that are close to one another will be perceived as related. To design proximity into page code, put related elements in sequence. For example, code an image caption directly following the image.
- Similarity: Elements that are similar will be perceived as part of a group. Design similarity into page code by using the same method to mark up like elements. For example, use lists to mark up navigation links.
- Continuity: Elements that are presented without interruption will be perceived as part of a group. In designing page code, avoid breaks in the flow of discourse.

2.2.3 Apply a consistent design

Design consistency allows users to apply what they know to different contexts, making new tools easier to learn and use. External consistency can be accomplished by applying common standards to a design [10]. Armed with learned knowledge of conventions—what Norman calls “knowledge in the head”—users are able to form a conceptual model of a new design based on existing rules and constraints [14].

With no formal guide or stylistic conventions, the web as a whole lacks consistency. However, as the technology matures, conventions are evolving naturally. Pages generally begin with a site logo, followed by navigation and search, and end with provenance and contact information. Search features often include a “Search” text label, text input field, and submit button. Adopting these and other established conventions improves efficiency by allowing users to apply existing knowledge to new pages. In addition, consistency within a site further enhances usability by allowing users to form a mental map of a site that they can apply to all pages. In a nonvisual context, conventions and internal consistency allow users to predict the location and design of page elements.

Use established conventions wherever possible. Design for internal consistency among pages, particularly with regards to functional elements—where they are in the page code and how they are designed.

2.3 Functionality

The degree to which users can operate functional elements.

2.3.1 Make functional elements workable via the keyboard

Along with conveying information, web pages provide access to functionality, such as filling out and submitting forms and activating links. For universal access, functional elements must be workable using keyboard commands. An interface that requires point-and-click interaction will not be usable by users who cannot work a pointing device. Specifically, nonvisual users do not work with a rendered page and therefore cannot see to point and click. On the other hand, keyboard commands for movement, input, and activation [19] can be issued using a variety of input devices [3]. For universal usability, all functional elements must be operable from the keyboard.

In addition, functional elements must adhere to the guidelines discussed elsewhere, in particular:

- Use structural markup. Software can use structural markup to make elements easier to operate. For example, use labels to explain the purpose of form fields.
- Communicate relationships. Define the boundaries of functional groups, such as navigation and forms—whether explicitly using markup or implicitly using design principles.
- Accommodate serial access. Functional elements are commonly accessed in sequence (see subsection 2.4.2, below). For example, arrange form elements so they follow a logical sequence of label and field, and position the “submit” action at the end of the form (Figure 2).

![Figure 2: Functional elements must be sequential to support serial access](image)

2.4 Readability

The degree to which content can be understood.

2.4.1 Use text for essential information

Text is the most effective format for conveying information to visual and nonvisual users because text can be read by software. Additionally, software can adapt text, so that users who cannot access a text-based design can transform the information into a format that is accessible—for example, by having the text read aloud by software. On the other hand, information conveyed using other formats, such as images or audio, cannot be readily accessed by software, making it less likely to be readable by all users.

HTML has provisions for providing non-textual content with accompanying fallbacks, so that users who cannot access information in its primary format have access to a text-based equivalent. The most common fallback is the ALT attribute of the IMG tag, which allows for a short textual description of the information contained in an image. Other non-text formats require more effort, such as a text transcript of an audio presentation [4]. Equivalents are certainly preferable to inaccessible content.
However, for universal design, “same means of use” is preferable to “equivalent use,” making text the preferred format for nonvisual access [5]. Whenever possible, use text to convey information. When a non-text format is central to the message, provide a text equivalent.

2.4.2 Accommodate serial access to page content

Document order is the order in which elements appear in the page code. Software generally starts at the beginning of source code and works its way through sequentially. The result of this serial access is that information contained at the top of the page is the first accessed by software [7, 11].

The inverted pyramid is a way of conceptualizing and communicating information. In writing, the inverted pyramid style presents the most important information at the beginning of a text [12]. Overlaying an inverted pyramid structure onto the page code of most web pages shows that the base of the pyramid is commonly composed of branding, advertising, and navigation, with content appearing well toward the tip. In a visual context, the “first read” is not affected by document order because visual users can skip over marginal elements and go directly to the main content. However, due to serial access, document order largely determines the quality of nonvisual access. Readability suffers with pages that are top-heavy with irrelevant information.

Provide relevant content near or at the top of the page code, followed by navigation, advertisements, and other marginal content.

3. Discussion

The principles and guidelines discussed affect web access in multiple contexts. Mobile access works better with text than with images. In addition, small screens present many of the same challenges as the limited “viewport” of nonvisual access, requiring that we communicate the relationships between elements, minimize clutter, and design for serial access [16]. The efficacy of search engine software also improves with structural, standards-based markup [20]. And all users enjoy the usability gains derived from the consistent application of conventions and design principles.

Here we focus on four guidelines for nonvisual web access that also improve access for mobile users.

- **Use markup to describe document structure.** One approach to providing web access on small screen devices is to extract content from pages designed for large screens and apply a mobile-friendly design. Opera’s Small Screen Rendering (SSR) works by adapting attributes such as layout, color, and type to display in the mobile context. With structural markup, SSR can apply different visual styles based on the structural information contained in the document [15].

- **Eliminate unnecessary clutter.** The small viewport and serial access to content increases the usability costs of low signal-to-noise in the mobile context. Additionally, large pages take longer to load and display, and many mobile providers charge for data transfer. Limiting or removing clutter reduces costs in usability, time, and money.

- **Use text for essential information.** Images can be problematic in the mobile context. Images that are wider than the screen must either be scaled to fit by the client software or accessed full-size via horizontal scrolling. When images are scaled, important details such as text may not be readable [17]. On the other hand, horizontal scrolling is generally viewed as a suboptimal method for accessing information. Because of the high cost of loading images and the relatively low gains, mobile users may opt to disable image loading, making it essential that designers supply equivalent text for important content.

- **Accommodate serial access to page content.** Mobile browsing is affected by document order. For instance, SSR converts a multi-column large-screen design into a single column layout. The sequence of elements in the single-column design is defined by the order in which they appear in the source code [15]. Typical web designs have branding and navigation before content, meaning users have to scroll, often extensively, to get to the main content of the page [17]. Additionally, studies [9, 17] have shown that users can be disoriented by pages that begin with standard elements. With top-heavy designs, the first screen looks the same across all pages of a site. Mobile users do not receive visual feedback when a new page has loaded with unique content. The inverted pyramid style for source code design works well for mobile devices by highlighting relevant content on the first screen and providing direct access to content.

4. Conclusion

Web technology allows for adaptive interfaces that meet the needs of diverse users accessing pages in multiple contexts. The beauty of the web is more than screen deep. By attending to design at the source code level, we further the goal of universal usability for both visual and nonvisual users.

5. References


OPENING UP ACCESS TO ONLINE DOCUMENTS USING
ESSENTIALITY TRACKS

Matthew T. Atkinson
Research School of Informatics
Loughborough University
Loughborough, Leicestershire, LE11 3TU
England, UK
M.T.Atkinson@lboro.ac.uk

Jatinder Dhiensa
Research School of Informatics
Loughborough University
Loughborough, Leicestershire, LE11 3TU
England, UK
J.Dhiensa@lboro.ac.uk

Colin H. C. Machin
Department of Computer Science
Loughborough University
Loughborough, Leicestershire, LE11 3TU
England, UK
C.H.C.Machin@lboro.ac.uk

ABSTRACT
This paper discusses extensions to the previously developed “essentiality and proficiency” approach to increasing usability and accessibility of websites. The existing approach is introduced, as is a new application in the processing of DocBook XML documents. The current principles are extended to make them more appropriate for increasing the usability of long documents. Techniques for allowing organisations to efficiently disseminate information based on the proposed application are discussed – increasing productivity for both non-disabled and disabled users.

Categories and Subject Descriptors
H.3.5 [Information Storage and Retrieval]: Online Information Services—Web-based services; H.5.4 [Information Interfaces and Presentation (e.g., HCI)]: Hypertext/hypermedia—User issues; I.7.2 [Document and Text Processing]: Document Preparation—Hypertext/hypermedia

General Terms
Documentation, Human factors, Standardization

Keywords
Accessibility, DocBook, Essentiality, Proficiency, Usability, Web, XML

1. INTRODUCTION
A significant amount of research has been carried out into providing access to websites for computer users with disabilities (such as sight loss). Some of this research [1, 2] focuses on finding ways to provide the user with only the information they need, in the format that is most suitable for them. This “universal design” [3, 4] style of approach is applicable to providing access for non-disabled users who are, for example, using embedded devices or simply do not have time to view anything but the essential information a website has to offer.

We present some new applications of and extensions to this research that are relevant to both on and offline document access (via integration with the DocBook XML typesetting system). Utilising this approach will allow corporations and educational institutions to provide websites, training material and manuals that can be automatically filtered and rendered to meet the needs of users in various roles. This will negate both the cost of transcribing such materials into alternative formats and that of maintaining the transcribed versions. It will also allow new methods of enhancing productivity for non-disabled users.

2. CHALLENGES AND CURRENT WORK
Dhiensa et al [5] point out that the problems of web access (which is equally applicable to offline electronic documents in web formats) is three-fold:

- Information exclusion (the fact that either people do not have access to computers or, even when they do, the cost of assistive technology is too great or the sites they wish to access are not standards-compliant).
- Though standards and legislation are now in place, they do not guarantee usability [6].
- “Information Overload”. The problem, and potential ways to cope are described in more detail in [7] and [8].

Research on solving these problems has been targeted at different points within the lifecycle of websites. Mohamad et al [6] presents a solution that highlights standards-compliance and accessibility problems so that they can be fixed by site developers. Hanson and Richards [2] modify existing sites on-the-fly – enabling users to have sites rendered in their chosen format.

The Disability Rights Commission Report [9] shows that websites are 35% easier to use for everyone if they are accessible1. This tells us that accessibility is a valid metric

1i.e. compliant with accessibility [10] and other web standards
by which we can estimate the usability of websites for all people. This implies that organisations adapting the ideas presented here should be able to improve the productivity of all people wishing to access their website and web-based documents.

3. PRIMER ON ESSENTIALITY AND PROFICIENCY

To solve the problems discussed above, work carried out by Dhiensa [1] lead to the creation of the “Essentiality and Proficiency Tool”. The basic concepts of this tool are:

- A web page contains information that has varying levels of importance for its visitors.
- The author of a web page should attribute such levels of *essentiality* to sections of the page. Essentiality ratings (integer values from 1-10) surround page elements and can be nested (e.g. more essential elements/text can be found within less essential ones).
- When the page is rendered for a user, content is filtered based on its essentiality level in relation to the level that the user wishes to view the page at. Only content with an equal, or higher, essentiality rating is displayed.
- Users have varying preferences and needs for how they view the page – it should be rendered according to the *proficiency* of the output device and/or disability of the user.²
- Profiles can be used to store settings (essentiality levels, formatting preferences) for users of the system.
- The tool is designed to act as a proxy service, modifying pages as they are requested and transforming them according to the user’s profile. Later we will discuss the challenges associated with this approach and how they are inherently overcome in the problem domain this paper is concerned with.
- The two ideas of essentiality and proficiency are highlighted as separate factors involved in making sites accessible. These ideas fit well with DocBook XML documents, as we will discover later.

The rest of the work in this paper adapts and subsequently builds on these concepts.

3.1 Usage of Essentiality and Proficiency

The tool requires input in the form of a profile and essentiality markup on the pages it is to filter and transform. Therefore the process involved with its use is split into separate activities for page authors and users of the tool.

3.1.1 For Authors

Authors are required to indicate the essentiality levels of elements within the pages they create. Originally, tags had to be added manually to the source code of pages. An example web page with some essentiality markup is shown in figure 1.

![Figure 1: A sample web page with essentiality tags.](image)

Recently a Mozilla Firefox plugin was created [11] that allows this task to be completed via a GUI interface. Tags are still added to the document’s source, but this process is hidden from the author – making it usable by a wider range of authors.

The GUI interface allows authors to obtain feedback on their markup in the following ways:

- Parts of their pages marked up as essential are highlighted. The colour varies according to essentiality rating.
- They can request that the page be displayed at a given essentiality level. All information marked up at this level, and below is then displayed.

It should be noted that the adoption of essentiality tags by the World Wide Web Consortium (W3C) is an ultimate goal of the project. Before this happens, however, the use of microformatting³ has been used to ensure that pages are still regarded as valid. This technique will not be required by the process for marking up DocBook XML documents.

3.1.2 For Users

The system is designed to make its use as transparent as possible. In the current prototype, the user visits a web page where they can select and tailor their profile. The profile records their preferred levels of essentiality and proficiency settings.

From the profile page, the user can enter a URL to visit in a textbox. The system then retrieves and transforms that page, finally presenting it to the user. For more information, please consult the previous work [1].

3.2 Benefits

This system enables content producers to maintain only one version of their website – there is no longer a need to

---

³This is the practise of marking up the essentiality levels of document elements using their CSS “class” property, instead of using a dedicated essentiality tag. This ensures that the page is still regarded as valid and (ironically?) passes accessibility checks.
create separate “accessible” or text-only variants. The Essentiality and Proficiency Tool automatically adapts sites to users’ needs with more flexibility than such alternative versions have historically provided – and at significantly lower cost.

Additionally, the work of Cheng provides a friendly, cross-platform, GUI-based method for authors to mark up their content.

The extensions and techniques proposed in later sections of this paper inherit the process model and general approach used by Dhiensa et al, but implement them in the context of automatic translation of documents written in the DocBook XML typesetting system.

3.3 Limitations of the Current System

Two main limitations of the current system are currently being investigated. These are:

Dynamic Content — A great deal of web content is generated dynamically by scripts (which may include the output from database lookups). A way to mark up this content with the appropriate essentiality levels should be devised.

Scalability across User Types — Some documents (especially corporate guidelines, user/developer manuals and “procedure documentation”[5]) are aimed at groups of people with a number of different roles. These documents inevitably contain information of varying importance to people in these differing roles. There is currently no way to denote this difference in interest.

This paper is mainly concerned with the latter of these limitations, which will be revisited shortly.

4. DOCBOOK XML

So far the work described has been applied to make websites accessible. However, it could equally be applied to documents that use web standard formats. DocBook XML is a Document Type Definition (DTD) and set of output filters designed for the creation of technical documentation. It is employed by many companies and institutions in the creation of their internal and external documentation.

The ethos of DocBook is to:

- Provide separation of content from formatting.
- Be as extensible as possible, allowing users to customise both output filters and the DocBook DTD.

[5] “Procedure documentation” refers to the type of documentation that is created within corporations to describe business processes, health and safety guidelines, software test procedures and so on. It often targets roles at different levels within an organisation, so finding relevant information in them can be difficult.
What makes DocBook particularly suitable, as far as Essentiality and Proficiency is concerned, is its close integration with web and on-the-fly translation technologies. Previous work ([12] and the use of DocBook for creating accessible lecture notes) found that DocBook provided a means to generate output in a number of useful formats from one source file. Most notably: the effort required to customise the existing XSLT code that produced (X)HTML output, in order to improve accessibility, was minimal. PDF and RTF output can also be produced easily.

5. APPLYING ESSENTIALITY TO DOCBOOK

Before we propose extensions to the existing approach, we will describe how it was implemented for DocBook documents. This demonstrates how useful the principles can be when applied in this way and provides a stepping stone for extending them. A very simple DocBook document is shown in figure 3. "Essentiality" is currently a one-dimensional grading of importance, as shown in figure 1.

In a simplistic essentiality-filtering algorithm, the essentiality level of the \texttt{essn} tag is checked against the user-defined filter level. The idea is that all elements which have an equal or higher rating to that selected by the user should be displayed (i.e. we create a high-pass filter). The content bounded by the \texttt{essn} tags should be displayed only if this condition is met.

However, we must beware of the situation where a more essential piece of information is nested inside a less essential section. If the essentiality level of a given element is not sufficient to get through the filter, but it contains a nested element that \textit{would} get through the filter, we must display the nested element.

The following DocBook source could produce misleading output:

\begin{verbatim}
<essn level="7">
 <para>Under no circumstances must any employee:</para>
</essn>
<essn level="5">
 <itemizedlist>
  <listitem>
   <para>Stay in the building after the alarm has sounded. <essn level="9">Leave the building immediately if there is a fire.</essn>
  </para>
</listitem>
</itemizedlist>
<br>...
</essn>
\end{verbatim}

Sample output from the above source (if user-specified essentiality level is 6-9):

\begin{verbatim}
<p>Under no circumstances must any employee:</p>
<ul>
<li>Leave the building immediately if there is a fire.</li>
</ul>
\end{verbatim}

Figure 4: An example of the importance of context.

5.1 The Problem of Context

A further issue is that of information being taken out of context due to the way it has been filtered. Consider the situation where one vitally important sentence has been displayed that was in the middle of a less important paragraph. If the parent paragraph is filtered but the sentence is displayed (due to the situation described above), then it could be taken out of context and have potentially serious ramifications – see figure 4 for an example.

There are two ways to ensure this does not happen:

- Authors should ensure essential information is marked up with the appropriate contextual information (e.g. the entire bullet point should have been at level 9 in our example).
- As the above may not always be appropriate, the algorithm should ensure that the output makes breaks in context clear to the user.

6. PROPOSED EXTENSIONS TO ESSENTIALITY MARKUP

Adapting the ideas of essentiality and proficiency to the world of DocBook documentation could further improve the accessibility and usability of such documents. However, the scalability of the current system can limit its usefulness, especially when long documents, aimed at people in multiple roles, are concerned.
6.1 Essentiality Tracks

We propose extending the current tags to create a number of tracks through the document; each denoting how important the information is for people of different roles and/or groups of people. Figure 5 shows a simple example of the tags that may be used for this.

In the case that a certain element, or set of elements may be of interest to multiple roles, they can be marked up with tags that may be used for this. Figure 5 shows a simple example of Essentiality Tracks.

Figure 5: Simple example of Essentiality Tracks.

```xml
<essn track="manager" level="3">
  <p>...</p>
</essn>
<essn track="developer" level="4">
  <p>...</p>
</essn>
```

Figure 6: Simple algorithm for selecting elements to process from the correct essentiality track. Also includes code to add context break warnings where necessary (see 5.1).

```java
ProcessEssnTag(tag) is
begin
  if tag track == global track
    if tag level >= global level
      display elements inside tag
    else
      if nested tag
        print 'Context Break'
        ProcessEssnTag(nested tag)
        print 'Context Break Finished'
      end if
    end if
  end if
else
  if nested tag
    ProcessEssnTag(nested tag)
  end if
end if
```

6.2 Track Coordination

The extensions proposed above have been informally tested, however further improvements may be necessary before the system could be deployed for very large documents. This section lists some further improvements that should be considered in future work.

Error Checking — By requiring authors (or their authoring tools) to declare all tracks for the document in its header, the `<essn>` tags may be validated to ensure they refer to a track that exists.

Track Grouping — It may be useful to group tracks at certain points to make the markup more readable and ensure it is easy to update in the future. One way to provide grouping of essentiality tracks extends the above suggestion to include a section in the document header that links tracks to their groups.

Track Relationships — Providing a means to express how tracks are related could aid marking up and revising the markup in long documents. Allowing authors to use expressions which link the ratings between different tracks could allow them to more effectively partition the document. For example, a simple “invert” rule could be used to ensure that anything marked up as very important for developers should be marked as of little importance for users. This feature could introduce too much complexity, however, so its usefulness will need to be tested.

Overview — Future essentiality editors could include a visual “map” of where different tracks can be found. This would present the author with a quick overview of how the document is structured.

Figure 7 gives an example of how the first two suggestions above may be expressed in terms of document markup. It presents a more intuitive way of marking up an element or sub-tree’s essentiality across different tracks – simple `<essn>` tags have to be nested; `<grade>`s can be applied in parallel.

6.3 Applying Essentiality Tracks to DocBook in the Real World

As discussed, DocBook is designed with customisation in mind. It is trivial to edit the XSLT code so that truly accessible (X)HTML output is produced\(^7\) so this will not be discussed further.

Following basic accessibility improvements, proficiency may be implemented by further enhancing the stylesheets to enhance the fonts, choose user and/or device-compatible colours and reorder navigation links according to the values specified in the user’s profile.

Adding support for our extended essentiality tags involves editing the DocBook DTD\(^8\). As the DTD itself is an XML document this is also a relatively easy task. However, it should be borne in mind that adding elements to a DTD (as opposed to taking them away) results in incompatibility with the original format because the new standard is a superset of the original one. This is not a problem when an organisation uses a format internally, but it is still important to submit these changes for inclusion in future versions of the standard if they are to be widely promoted and used.

---

\(^7\) Replacing all layout tables with CSS2, as the AGRIP Documentation Project have done, for example.

\(^8\) In the experiments conducted for this paper, the DTD was not edited and the “-novalid” option was passed to the XML processor to disable validity checks. This allows quick tests to be made but is not as robust as editing the DTD.
Figure 7: Example of some more practical DocBook DTD Extensions. Being able to group tracks could make marking up long documents considerably easier. This scheme also provides some built-in error-checking.
but we do not consider them here and offline, via the use of XSLT and other related technologies. This may be done both on track and level.

Figure 8: Sample webform used to select essentiality track and level.

6.4 Online Transformation

As discussed, many different output formats can be generated from a DocBook document. This may be done both on and offline, via the use of XSLT and other related technologies. As an example of where on-the-fly processing may be useful, consider an organisation publishing the documentation for some software it has developed for its employees.

In such an example, people in many different job roles could be interested in the documentation; ranging from managers to users and developers. There would likely be a number of essentiality tracks in the source files of the document. It is very likely that users would have different proficiency requirements too. If we consider the combination of essentiality track, filter level and proficiency requirements, we realise that a vast amount of output would have to be developed to cater for the needs of everyone in advance.

An efficient way to disseminate this documentation would be to have readers fill in a (very) short webform such as that shown in figure 8. This would select the parameters for the transformation, which would then occur in real time, generating (X)HTML (or PDF/RTF) output for the reader’s chosen essentiality track, level and proficiency profile. There may be good reasons for having this generation take place either on a server running Apache Cocoon or within the user’s web browser.

6.5 Challenges of the Proxy Approach and How they are Alleviated

Though they promise great benefits (almost no modification necessary on client machines, centralised administration and upgrades), traditional proxy services may run into problems. Some popular criticisms (including those highlighted by Hanson and Richards [2] and Mohamad et al [6]) are:

- The method they use to transform pages may be overridden by features of the page (such as the use of embedded CSS instead of external stylesheets).

- Bandwidth constraints may slow the system down.

- Such systems often have to “undo” the inaccessible work done when the site was created; this could be inefficient (at least from a design elegance point of view).

- Proxies have to be tolerant of non-standards-compliant sites.

- Most web sites are not annotated to provide extra accessibility information.

These problems are not present (or are averted) within the proposed DocBook transformation system for the following reasons:

- All style information is embodied in the XSLT and external CSS stylesheets (in the case of generated (X)HTML).

- The system is envisaged to be deployed within a web service such as Apache Cocoon [13] (which could cache the results) on an organisation’s LAN. However, there is nothing to stop the process happening client-side, within the user’s browser (both Mozilla Navigator and Mozilla Firefox support the required XSLT standards). Client-side translation may be of use when server resources are at a premium.

- No “undoing” work is necessary; only the transformations appropriate to the user are carried out.

- No tolerance of non-conformance is required as all DocBook XML documents supplied to the system should be valid – they are checked for validity before all other processing takes place.

- Though there is no absolute need for annotation, we have already established a business case for it (increased productivity and usability for all), so it need not be seen as an extra chore.

The proposed approach works because an organisation would have control over the standards and transformations in use at all stages – this is rarely the case when trying to improve the accessibility of third-party websites.

7. CONCLUSIONS AND FUTURE WORK

Use of essentiality tracks and the above techniques for transformation are proposed as an efficient and effective method for organisations such as companies and educational institutions to disseminate their materials. There is still a lot of potential for future work, however:

- Other output formats than traditional HTML and PDF could have the principles of essentiality and proficiency applied to them. For example: translation of DocBook into Braille is not impossible and – given the essentiality filter – could be an ideal format for accessible hard copies.

- The semantics of essentiality tracks could also be improved. The examples given in this paper indicate that they could be of great benefit to users, however the task of marking up content for the authors could be made easier. Section 6.2 has already suggested some work that could be carried out to further improve the proposed system.

Though in the case of lecture notes and similar material, the tool should be used to improve general accessibility, not to give students direct answers (thus reducing their ability to think critically) – research into the balance between these effects should be carried out.

\[\text{There is an older SGML standard and DSSSL stylesheets, but we do not consider them here.}\]
The essentiality editor [11] should be extended to support tracks.

More tests should be carried out with respect to the most natural scale for essentiality. The one used here is taken from previous work (levels 1-10, with 10 representing the most essential information) but it may be that a different scale is more useful.

The ideas proposed so far are being formally tested. The community of AGGRIP [14] is participating in these tests as (a) a large amount of DocBook documentation is already used within this project and (b) as most users are blind, the effects on accessibility as well as general productivity could be assessed. So far, informal user feedback has yielded positive results.

Additionally, our future research will be touching on some of these areas.

8. ACKNOWLEDGEMENTS

This work was made possible by the sponsorship of the Grundy Educational Trust and Loughborough University.

9. REFERENCES


Transforming Web Pages to Become Standard-Compliant through Reverse Engineering

Benfeng Chen
Computer Science Department
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
bfchen@cs.ust.hk

Vincent Y. Shen
Computer Science Department
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
shen@cs.ust.hk

ABSTRACT
Developing Web pages following established standards can make the information more accessible, their rendering more efficient, and their processing by computer applications easier. Unfortunately, more than 95% of the existing Web pages today are not “valid” in that they do not follow some of the recommendations (standards) of the World Wide Web Consortium (W3C). Fixing any Web page to make it standard-compliant is a major undertaking. There is now an open-source tool called HTML Tidy which will attempt to fix the invalid HTML code automatically. However, Tidy often changes the Web page’s appearance after processing. It is not an effective tool to transform existing Web pages to make them standard-compliant.

In this paper we report the design and implementation of PURE, a tool that cleans up an HTML document through reverse engineering. PURE starts with the rendering result of a given Web page and generates valid HTML code and CSS automatically to produce the same appearance. It is found to be effective for many existing Web pages. A prototype is now available for public testing and comments.

Categories and Subject Descriptors
H.4.3 [Communications Applications]: Information Browsers

General Terms

Keywords
W3C recommendations, Web page, HTML, HTML Tidy, Cascade Style Sheets, rendering engine, browser.

1. INTRODUCTION
The World Wide Web Consortium (W3C) is an organization that develops and promotes the use of standards on the Web. However, as much as 95% of existing Web pages today are “invalid” [9] in that they fail to conform to the W3C “recommendations” (commonly considered “Web standards” by the community) published since 1995 [16]. These invalid Web pages may not be rendered consistently across platforms or by different browsers, may not be accessible to some users, and may cause problems for Web-based applications. The seriousness of the problem spawned the Web Standards Project (WaSP) [10], which was established to encourage people to design Web pages which conform to standards in order to reduce the cost and complexity of development, while increasing the accessibility and long-term viability of any site published on the Web [21].

The basic idea of Web standards is to develop Web pages with valid HTML or XHTML code and to separate content from presentation. A Web page is composed of three parts: content, presentation and behavior. According to Web standards, the content should be written in valid HTML or XHTML code; the presentation (e.g., layout, font, color, etc.) should be specified by valid CSS code; and the behavior should be controlled by valid JavaScript (officially, the ECMAScript [2]) through the DOM interface [17]. Figure 1 shows the “trinity” of Web standards [21]. It is necessary for a Web page to be compliant with Web standards because of the following reasons [5][11]:

- Its accessibility is wider because the CSS-based presentation is more flexible for different devices, browsers or handicapped people.
- Its size is smaller and bandwidth for access is reduced. For example, ESPN.com is saving 2TB of traffic per day by redesigning Web pages according to Web standards [7].
- It is friendly for machine processing. Without the misused table and font tags for layout and appearance in the code, other computers can “understand” the Web page’s content better.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

W4A at WWW2006, 23rd-26th May 2006, Edinburgh, UK
Copyright 2006 ACM 1-59593-281-x/06/05...$5.00.

Figure 1. The trinity of Web standards
However, as we noted earlier, most Web pages on the Internet today are not standards-compliant [9]. A very common problem is that the HTML TABLE element is widely used for page layout. Note that the TABLE element is intended to mark up truly tabular information (“data tables”). But, due to its flexibility, content developers often use the TABLE element to layout pages (“layout tables”). They should have used the Cascade Style Sheets (CSS) to layout pages for all the presentation effects [12]. People believe that about 99% of existing websites are obsolete as far as Web standards are concerned [21]. Rewriting these legacy Web pages will incur a huge cost of time and money. Therefore, an automated tool will be very useful if it can help Web developers transform existing Web pages to those that are standards-compliant while keeping the appearance consistent with the original.

Fixing an existing HTML document to make it standard-compliant automatically is a difficult problem. The W3C has endorsed an open-source tool called “HTML Tidy” which can automatically fix a Web page’s invalid code [8]. But it does not convert presentation-related elements from existing HTML code to using style sheets. Furthermore, it often fails when fixing a complicated Web page, especially if its layout is designed with nested HTML tables. Unfortunately, nested layout tables are widely used in legacy Web pages and various tricks are frequently used to extend the presentation effect of tables. A quick study shows that of the 500 most popular websites [1], about 470 of them use the TABLE element to control the layout. The appearance of these Web pages often changes drastically after they are processed by Tidy. Figure 2 shows an example (http://www.imdb.com, No. 46 of the top 500 most popular websites on September 1, 2005): Figure 2 (a) is the original page and Figure 2 (b) shows parts of the resulting page after processing by Tidy, which produced a long thin page. This is indeed an extreme example but, since many of the Web pages of the top 500 websites are quite complicated, Tidy messes up the appearance of at least 80% of them.

This paper presents an automated tool called “PURE” (“Page clean-Up through Reverse Engineering”) which uses the reverse engineering approach to transform legacy Web pages to make them compliant with Web standards while keeping the appearance consistent with the original. PURE first uses a popular Web browser to render the original Web page to be processed (the “preprocessing” step). It then examines the layout of the Web page presented in the browser’s window. It segments the original Web page’s content into small rectangles (called “boxes” thereafter). It retrieves the position and size of each box through the browser-generated page structure (in the form of a DOM tree [17]). A recursive algorithm is designed to reconstruct the layout of the Web page using the CSS box model [15]. At the end of this step a new Web page is generated which contains a number of empty boxes (the “layout reconstruction” step). In the last step, PURE fills in the content for each box. It takes the source code for each box from the original Web page, transforms it into standard-compliant code, and then puts the resulting code into the corresponding box in the new page (the “box filling” step). The home page of www.imdb.com after processing by PURE is shown in Figure 3.
Our preliminary evaluation shows that over 50% of the home pages of the top 500 websites can be successfully transformed through this three-step approach automatically. The failures are mostly due to some special features of the browser-generated page structure which is inconsistent with the DOM tree as defined by W3C. Most of these problems could be fixed with some human interaction. We therefore believe that PURE can become a useful tool to help Web developers to transform legacy Web pages.

The highlights of our approach are:

- We are able to separate the presentation from content. To understand a Web page’s presentation from its source code is a very difficult task unless we build a rendering engine ourselves. In our reverse engineering approach, we can obtain the presentation from a popular browser (such as the Microsoft Internet Explorer (IE)). The major task is to build the same presentation using CSS, which is much easier than building a rendering engine.

- We avoid the difficulty of parsing the original HTML code by getting the HTML DOM tree from the chosen browser.

- We avoid the difficulty of handling ambiguous code and guessing the author’s intention. The invalid code is often ambiguous. For example, the developer may have left out certain closing tags and the browsers must guess where to place them if some are missing. It is reasonable to assume that the authors have tested the Web pages using IE before they upload them to a Web server. So we could use IE’s rendering result as the starting point for reverse engineering.

- We use a divide-and-conquer strategy by first segmenting the Web page into small boxes and then reconstructing these boxes separately. If a certain box has problems, it will not affect the others and the author can manually fix the problematic boxes one by one.

This paper is organized as follows: Section 2 explains PURE’s approach in detail. Section 3 describes our evaluation of PURE. In Section 4, the future work to make PURE more effective is discussed.

2. THE REVERSE ENGINEERING APPROACH

2.1 Background
The content of an HTML document is contained in the BODY element. All elements which may appear in BODY can be classified into 2 kinds: the “block-level” elements and the “inline-level” elements. Generally, block-level elements may contain inline-level elements and other block-level elements. Inline-level elements may contain only data and other inline-level elements. For example, the elements P, DIV, TABLE, UL and H1 are block-level elements and B, SPAN, FONT, A and IMG are inline-level elements. During rendering, block-level elements always begin on new lines, but inline-level elements do not [13].

Generally, a Web browser parses the HTML document into a tree structure and then begins rendering based on the tree. The tree structure is often called the HTML DOM (Document Object Model) tree. The W3C’s visual formatting model gives the idea on how a browser renders an HTML document based on its DOM tree [14]. Each node in the DOM tree generates one or more boxes on the screen if it is visible. A visible inline-level element may generate more than one box when the text within it spans several lines. A visible block-level element always generates one box because it will begin with a new line. We can see an element’s box boundaries by adding the STYLE attribute of CSS to specify the border of the box. For example:

```html
<P style="border: 1px solid red">some text</P>
```

If we specify the border of each element in the document body, we can see that a Web page’s appearance is composed of many nested boxes. Modern browsers provide a friendly programming interface for developers to access a page’s DOM tree structure and each element’s box information (e.g., position, size, source code, etc.). Therefore, it is possible for us to reverse engineer a Web page box by box through the browser’s rendering result.

2.2 System Overview
The basic idea of PURE’s approach is to reconstruct the Web page’s layout using the CSS box model and then fill in the content for each box. When designing page layout with CSS, people commonly use the DIV element to represent a box and use the style sheet to position the box of the appropriate size. Since the Web page is composed of many nested boxes, a straightforward way is to use DIV to build every box in the original page and then render the new CSS-based page. But this will be messy. For example, the code "<P>some <B>bold</B> text</P>" generates two boxes. One box is for element P and the other is for element B. Wrapping each box with DIV results in the code
"<DIV><P>some <DIV><B>bold</B></DIV> text</P></DIV>". which breaks the paragraph. This is not acceptable. Another problem of this simple method is that many redundant boxes will be generated due to the HTML table. For example, the code "<TABLE><TR><TD>some text</TD></TR></TABLE>" generates three boxes because there are three elements here: TABLE, TR, and TD. This will lead to redundant code as "<DIV><DIV><DIV>some text</DIV></DIV></DIV>". Actually, we only need one DIV. From this example, we can also see that HTML table-based layout is not efficient. The above two examples show that we cannot simply use every box in the original page to build the layout of the new page. Instead, we should choose only the necessary boxes. We call these boxes “primary” boxes. The primary boxes cover all the content and are not overlapping with each other. As a result, the primary boxes segment the page into individual rectangles and they will not affect each other. The element which satisfies any of the below conditions is considered a primary box:

- The TD/TH elements within the layout table (layout table cell)
- The leaf block-level elements outside the layout table

A leaf block-level element is a block-level element whose descendant elements in the DOM tree are all inline-level elements. In other words, there is no block-level element inside the leaf block-level element.

In the new Web page, we use CSS positioning techniques [15] to make these primary boxes appear in the same positions as they are rendered by the chosen browser. At the end of this step, the layout reconstruction is finished and the next step is to fill in the content for each primary box. Since each box is corresponding to an element in the DOM tree, the content inside the box is the code of that subtree which has that element as root. Therefore, the approach to fill in the content of each box is to go through its corresponding subtree top-down and generate standard-compliant code.

Figure 4 shows the flow chart of the PURE system. These three major steps are explained further below.

2.3 Preprocessing

This step has two tasks: (1) generate the input Web page’s structure by passing it to a popular browser’s rendering engine; (2) format the page’s structure for later processing. The primary boxes are supposed to cover all the page content. But without formatting the document structure, they may fail to do so. For example, the code "<BODY><P>text one</P> text two <B>text three</B></BODY>" will only generate one primary box, which is the P element. Therefore, the text “text two” and element B will be missing according to the primary box coverage. Our Lemma 1 can solve this problem.

**Lemma 1:** In the HTML DOM tree, if inline-level elements or text nodes are the siblings of a block-level element, they are implicitly enclosed by the block-level element.

It is easy to prove this lemma. The block-level element always begins with a new line and ends with a line break, which means any element before or after the block-level element will end or begin with a line break. In other words, there is an implicit block-level element containing the inline-level elements or text nodes around a block-level element. As for the above example, the code "<BODY><P>text one</P><P>text two</P><B>text three</B></BODY>" has the same appearance with the
original one. The document formatting process inserts the general block-level element DIV to those places where there are implicit block-level elements. As a result, the above example code will become “<BODY><P>text one</P><DIV>text two</DIV></BODY>”. Figure 5 shows the process. After formatting, two elements will be marked as primary boxes: the elements P and DIV. We can see that they cover the whole content of the page.

Figure 5. An example of formatting code

2.4 Layout reconstruction

The tasks of this step are: (1) identify the primary boxes; (2) create a new page with the primary boxes in the same position. The primary box is defined as the layout table cell or the leaf block-level element outside the layout table. The primary box identification process goes through the DOM tree top-down and checks every element. If an element is considered a primary box, its descendant elements will not be processed. This guarantees that none of the primary boxes overlaps each other. Note that it is often difficult to distinguish between a layout table and a data table unless we develop an intelligent algorithm which can “understand” the relationship among the content of table cells. This is not the major task of PURE. Instead, we use a simple but effective way. From our observation, when a cell of table contains block-level elements (e.g., P, TABLE, etc), the table is often used for layout. On the other hand, the cells of data tables mostly only contain inline-level elements or text. To identify the leaf block-level element is easy. After the document formatting process, all elements at the same level (i.e., they have the same parent element) are either all block-level or all inline-level. If a block-level element outside the layout table has inline-level elements, it will be made a primary box.

The primary boxes form the layout of the Web page. The next task is to create these boxes in a new Web page with the same positions. As mentioned before, we use DIV element to represent a box and using CSS positioning techniques to control the location of each box. This is a typical CSS box-model based layout design method, which is widely used by the Web developer community. There are three positioning schemes in W3C’s CSS recommendation: “normal flow”, “floats” and “absolute positioning” [18]. These position schemes are so flexible and powerful that there are always various ways to construct a single layout by using a combination of these schemes. PURE provides two different positioning methods to reconstruct the layout. One is to use absolute positioning and the other is to use a combination of normal flow and floats.

Using the absolute positioning scheme to construct layout is simple and straightforward. PURE generates the boxes one by one and specifies the position and size of each box in the external CSS file. The body of the HTML code is like:

```
<BODY>
<DIV id="FirstBox"> ... </DIV>
<DIV id="SecondBox"> ... </DIV>
....
<DIV id="LastBox"> ... </DIV>
</BODY>
```

And the external CSS code is like:

```
#FirstBox {position:absolute; top: 10px; left: 10px; width: 800px; height: 100px;}
#SecondBox {position:absolute; top: 110px; left: 10px; width: 200px; height: 400px;}
...
#LastBox {position:absolute; top: 660px; left: 10px; width: 800px; height: 200px;}
```

We can see that the document body structure is a two-level tree and all the primary boxes are in the same level with the same parent element BODY. This document structure may not be identical to the original page. Another problem is that, when developers change a box’s size, they need to recalculate the positions of its neighbor boxes and update them in the CSS file. So this approach is not flexible. If the number of primary boxes is not big, this approach is acceptable. But for a complicated page with many primary boxes, this approach may not be suitable.

The second approach, which uses a combination of normal flow and floats schemes to construct the layout, is more flexible than the first one. It does not specify the position of each box. Developers are free to edit an individual box and do not need to update the positioning code of other boxes.

The normal flow is the default scheme in CSS box positioning. In this scheme, boxes flow vertically starting at the top of their containing block, with each one of them placed directly below the preceding one. The effect is illustrated in Figure 6.

```
#box1 {width: 300px; height: 300px;}
#box2 {width: 200px; height: 300px;}
...
<DIV>
<DIV id="box1">text one</DIV>
<DIV id="box2">text two</DIV>
</DIV>
```

Figure 6. Normal Flow Positioning
The floats scheme works in the horizontal direction. It is achieved by setting an element’s `float` style to either `left` or `right`. A box with floats style specified is shifted as far to the right or left of its containing block as possible. If two or more adjacent elements are floated, their tops are positioned on the same line (side by side) if there is sufficient horizontal space to accommodate them. Figure 7 illustrates this effect.

Figure 7. Floats Positioning

With normal flow and floats positioning techniques, we can use nested `DIV` elements to construct any kind of layout of existing Web pages. By inserting sub-boxes with normal flow positioning, a box is split horizontally into rows. By inserting sub-boxes with floats positioning, a box is split vertically into columns. Therefore, we can construct layout like the way an HTML table does it. But this CSS approach is more flexible than the HTML table. In the HTML table, the border lines of columns are aligned across rows and border lines of rows are aligned across columns. In this CSS approach, a box can be freely split into rows and columns without forced alignment.

Considering the process in which Web developers design page layout with nested `DIV` elements and CSS, we designed a top-down recursive algorithm to construct the layout automatically. In the beginning, the Web developers commonly segment a page vertically or horizontally into several major rectangles (boxes). Then they segment each major box into several sub-boxes if necessary. After that, they may segment each sub-box into even smaller boxes. By repeating this process recursively and creating boxes level by level, the desired layout is finally achieved. The PURE’s algorithm to construct page layout with nested `DIV` elements and CSS is similar to this process. We assume that all the primary boxes are already at their right places (see example in Figure 8 (a)). At first, we segment the page from top to bottom with boxes. The segmentation method is: we set a horizontal virtual line on the screen. If this line does not cut across any primary box, we consider this line the border line between two boxes. We move the virtual line down and check whether it should be a border line or not. After all the horizontal border lines are detected, the first level segmentation is done and the page is composed of a number of boxes stacked from top to bottom (Figure 8 (b)). Then we try to segment each of these boxes individually. A vertical virtual line inside a box moves from left to right and checks whether it should be a border line. After all the vertical border lines within each box are detected, the second level segmentation is done (Figure 8 (c)). The boxes generated in the second level segmentation are further segmented in a different direction to generate the third level boxes (Figure 8 (d)). This recursive process is repeated until no more boxes can be segmented, which means that the last level boxes are all primary boxes. Thus, the page layout construction is done.

Figure 8. Layout reconstruction algorithm

As for the example in Figure 8, PURE will generate the code below. For simplicity, the corresponding CSS code is not shown here. The code forms the layout of the new Web page and reserves the space for primary boxes. The next step is to fill in the code for each primary box.

```
<BODY>
<!-- first level -->
<DIV>
<!-- code for primary box A -->
</DIV>
<DIR>
<!-- second level -->
<DIR>
  <!-- code for primary box B -->
</DIR>
<DIR>
  <!-- code for primary box C -->
</DIR>
<!-- code for primary box D -->
</DIV>
```

Original page with primary boxes
(a)  
First level segmentation  
(b)  
Second level segmentation  
(c)  
Third level segmentation  
(d)  

Figure 8. Layout reconstruction algorithm
Figure 9 shows the layout of the example page in empty boxes following this process.

2.5 Box filling
The last step is to fill in the content of each primary box in the new Web page. Because each primary box is corresponding to a subtree in the DOM tree of the original Web page, PURE goes through the subtree top-down and generates valid code for each node of the tree. Some visual effects (e.g., background color, text alignment and font, etc.) are inherited from ascend nodes which are outside the primary box. Therefore, the ascend nodes should be scanned to extract their visual effects and then we use style sheet to format the nodes inside the box. The presentation-related elements inside the box are removed by using style sheet during this process. For example, the HTML elements `FONT`, `B`, `I` and `U` will be replaced by the `SPAN` element and style sheet is used to specify the text appearance. By doing so, all the presentation code in the original Web page is stripped out so that the new HTML document is purely for content and its presentation is purely controlled by CSS. PURE then uses HTML Tidy to make the final clean-up of the generated code for each box.

3. IMPLEMENTATION AND EVALUATION

3.1 Selection of Web page rendering engine
As discussed above, the reverse engineering approach first uses a browser’s rendering engine to generate a Web page’s structure. Table 1 lists the rendering engines used in well-known browsers and their popularity statistics in October 2005 [4].

<table>
<thead>
<tr>
<th>Rendering Engine</th>
<th>Used by browsers</th>
<th>Popularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSHTML-Modern</td>
<td>Internet Explorer 6.0</td>
<td>82.51%</td>
</tr>
<tr>
<td>MSHTML-Legacy</td>
<td>Internet Explorer 4.0/5.0/5.5</td>
<td>3.94%</td>
</tr>
<tr>
<td>Gecko</td>
<td>Mozilla, Firefox, Netscape 7.x</td>
<td>9.97%</td>
</tr>
<tr>
<td>KHTML</td>
<td>Safari, Konqueror</td>
<td>1.62%</td>
</tr>
<tr>
<td>Opera</td>
<td>Opera – all versions</td>
<td>0.83%</td>
</tr>
<tr>
<td>Netscape</td>
<td>Netscape 3.0/4.0</td>
<td>0.07%</td>
</tr>
</tbody>
</table>

The rendering engine “MSHTML-Modern”, which is used by Microsoft Internet Explorer 6.0, is by far the most popular. It is reasonable to assume that most Web page developers would test their creation with this rendering engine before uploading it to a public Web server. Microsoft also provides sufficient documentation about programming with MSHTML-Modern [6] so that we can obtain enough information about the Web page’s structure. For these reasons we use it in PURE. Note that our approach will work equally well with any other browser as long as the structure information is available.

In order to embed MSHTML-Modern in PURE, the software is developed on Windows XP in C++.

3.2 Graphic user interface
Since we do not expect PURE to be perfect in fixing invalid Web pages automatically, it is designed to have a graphical user interface (GUI) so that the user may fix some problems interactively whenever needed. Through PURE’s GUI the user may adjust the resulting layout conveniently. PURE uses HTML Tidy to make sure the code used to fill in the boxes is valid HTML code. However, the user is given the opportunity to change that code manually if Tidy’s result is not suitable. Figure 10 shows an example of a PURE pop-up window which the user may use to fix the HTML code easily.
The results show that PURE is able to successfully reconstruct over 50% of the Web pages collected from the top 500 (or rather, 440 of the top 500) websites. We studied the failure cases and found that most of the failures are caused by the inconsistent page structure between the one generated by MSHTML-Modern and W3C’s definition of the DOM tree.

4. DISCUSSION

Today, people have realized that most of existing Web pages are not compliant with Web standards, which hinders the accessibility and viability of the Web. Transforming those legacy Web pages to be standard-compliant is necessary but will require a lot of effort. Our PURE tool is developed to help Web developers to do the transformation automatically. Because some legacy Web pages are very complicated and the browser may not generate the page structures according to W3C’s definition of the DOM tree, PURE may not work well automatically in some cases. But with some manual interaction, PURE can be effective for most cases. Furthermore, the PURE tool will be released as an open source and free software so that the community can help improve it. We believe that PURE will evolve into a useful tool for Web developers to transform legacy Web pages. With additional work, PURE may be made an option for Web page authors to pass the code through before leaving the authoring tool used for the page’s creation.

Web page reverse engineering can be applied to many other applications. One possible application is to transform existing Web pages to make them compatible with mobile devices. Today, mobile Web access suffers from interoperability and usability problems that make the Web difficult to use for most mobile phone subscribers [19]. One well-known problem is that traditional Web pages do not provide a good browsing experience on mobile devices due to their small screen size. This problem may be solved by using a new presentation scheme (style sheet) with which the browsers on mobile devices will use to display the Web page nicely. Some popular websites now provide a version designed specially for mobile devices such as http://www.google.com/pda and http://wap.oa.yahoo.com/. But it would be difficult to keep the content consistent with the original Web page unless there is a tool to generate a mobile version automatically from the traditional website.

The reverse engineering approach introduced in PURE can be adopted to help Web developers build Web pages that are compatible with mobile devices based on existing ones. That is: (1) segment the original Web page into primary boxes; (2) remove some primary boxes if necessary (e.g., large images or navigation links on the top of the page [20]); (3) rearrange the placement of primary boxes to fit the mobile device’s screen. Therefore our reverse engineering approach will become a valuable foundation for research work on Web standards and Mobile Web.

5. PROTOTYPE

Rather than providing additional evaluation data, which are necessarily subjective, a PURE prototype is now available for public testing and comments at http://webproject.cs.ust.hk:8004/. It is free and open source software and published under General Public License (GPL) [3]. Readers are welcome to improve it or develop applications based on it.

Table 2. Evaluation result of PURE

<table>
<thead>
<tr>
<th>Similarity</th>
<th>100%</th>
<th>90%</th>
<th>80%</th>
<th>≥80% (Success)</th>
<th>&lt; 80% (Failure)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>70</td>
<td>51</td>
<td>103</td>
<td>224</td>
<td>216</td>
</tr>
<tr>
<td>Rate</td>
<td>16%</td>
<td>12%</td>
<td>23%</td>
<td>51%</td>
<td>49%</td>
</tr>
</tbody>
</table>
6. ACKNOWLEDGMENTS
This research was supported in part by Sino Software Research Institute grant SSR101/02.EG14, “W3C Office” and by Research Grant Council grant AOE/E-01/99, “Information Technology for a 21st Century Hong Kong”. We want to thank Dan Hong, Yongzhen Zhuang and Shan Chen for their valuable contributions to this work.

7. REFERENCES
http://www.alexa.com/site/ds/top_500


http://johnhaller.com/jh/useful_stuff/browser_statistics


http://msdn.microsoft.com/workshop/browser/prog_browser_node_entry.asp


http://www.w3.org/TR/REC-html40-971218/struct/global.html


http://www.w3.org/TR/1999/WD-positioning-19990902


Personalizable Edge Services for Web Accessibility

Gennaro Iaccarino
ISISLab
Dipartimento di Informatica ed
Applicazioni “R.M. Capocelli”
Università di Salerno, Italy
iaccar@dia.unisa.it

Delfina Malandrino
ISISLab
Dipartimento di Informatica ed
Applicazioni “R.M. Capocelli”
Università di Salerno, Italy
delmal@dia.unisa.it

Vittorio Scarano
ISISLab
Dipartimento di Informatica ed
Applicazioni “R.M. Capocelli”
Università di Salerno, Italy
vitsca@dia.unisa.it

ABSTRACT
Web Content Accessibility guidelines by W3C [29] provide several suggestions for Web designers on how to author Web pages in order to make them accessible to everyone. In this context, we are proposing the use of edge services as an efficient and general solution to promote accessibility and breaking down the digital barriers that inhibit users with disabilities to actively participate to any aspect of our society.

To this aim, we present in this paper PAN: Personalizable Accessible Navigation, that is a set of edge services designed to improve Web pages accessibility, developed and deployed on top of a programmable intermediary framework [8].

The characteristics and the location of the services, i.e. provided by intermediaries, as well as the personalization and the opportunities to select multiple profiles make PAN a platform that is especially suitable in accessing the Web seamlessly also from mobile terminals.

Categories and Subject Descriptors
H.4 [Information Systems Applications]; H.5 [Information Interfaces and Presentation]; K.4 [Computers and Society]: Social issues—Handicapped persons/special needs

General Terms
Human Factors

Keywords
Web Accessibility, Programmable edge servers, disability, universal access.

1. INTRODUCTION
The World Wide Web, with its ability to is a “24x7 source of information”, and the services it provides, is a remarkable reality in our society. Many different explanations can motivate this astonishing success, from technological to sociological to economical ones. Among the former, very important is the emphasis on the capabilities of the standards to accommodate a wide range of services, therefore, pushing the World Wide Web as a universal access portal to the information, wherever located and however accessed.

In spite of this tremendous and strengthened success, an important challenge affects the structure of the pages that the Web increasingly provide every day: most of them are published without any consideration about an important question: “is this Web page accessible for everyone!”.

More specifically, Web accessibility means that people with disabilities can easily navigate and interact with the Web. Conversely, currently most Web sites have accessibility barriers that make difficult or impossible for many people with disabilities to use and access the Web. Using keyboards and mouse, hearing video and audio multimedia files, browsing through some intrusive Web pages (i.e. Web pages with pop-ups, advertisements, etc.) could appear as a normal activity for “non-disabled” people, but, on the other hand, it appears as a not simple task for users with some type of disability.

Currently there are no universally accepted categorizations of disability, but as described in [10] they can be classified into the following main categories: Visual disabilities, that include blindness (uncorrectable loss of vision) low vision and color blindness (a lack of sensitivity to certain colors); Hearing Impairments that include deafness (uncorrectable impairment of hearing) or a mild hearing impairment; Physical disabilities that include motor disabilities (weakness, limitations of muscular control, such as lack of coordination, limitations of sensation in hands and arms); Speech disabilities that include difficulty in producing speech that is recognizable by software for voice recognition; Cognitive disabilities that include, for example, dyslexia, attention deficit disorder, intellectual and memory impairments.

In particular, in the above classification, visual impairments produce the most common source of difficulty for users accessing the Web. Small fonts, some text and background color combinations, background images, and blinking text represent a problem for Web users. Moreover, blind people notice several problems if Web pages are composed of a lot of images, tables, and not only of linear text. In addition, for users with hearing impairments, the most critical difficulty comes from the increasing popularity of audio and video multimedia applications that, on the other hand, represent an increasingly important part of many Web sites today.

Of course, the accessibility problems are greatly enhanced by the navigation by mobile terminals, since the “traditional” transcoding operates only taking into account the limitations of the devices and not taking into account the potential disabilities of the user. In literature, we can find several examples of intermediary adaptation systems for mobile terminals such as iMobile by AT&T [25], the
must be emphasized that improvements on Web pages could position (a new profile from scratch), that mainly suit their needs. Finally, it is with disabilities can take advantage of the provided edge services any time, anywhere and by using any client device. Users with disabilities can take advantage of the provided edge services by simply choosing a profile, among the provided ones (or to define a new profile from scratch), that mainly suit their needs. Finally, it must be emphasized that improvements on Web pages could positively affect all users, for example, support for speech output could not only benefit blind users, but also Web users whose eyes are busy with other tasks.

Organization of the paper: In the next section, we present the important issue of Web accessibility and, therefore, strongly motivate our research. Moreover, we present some examples of accessibility systems that exist in literature, and some example of intermediary frameworks that could be programmed to provide accessibility services. In particular, we first describe how programmable proxies can be employed to promote accessibility, and in general, tackle a relevant part of the problems that are generated by the dynamic nature of the Web, then, we briefly we introduce the platforms for programmable HTTP proxies that are available nowadays. In Section 3 we provide some details about PAN and its configuration and the multiple profiles options. In Section 4 we present PAN’s services that can be used for efficiently providing accessibility to people for disabilities, in several contexts, included the mobile Web. Finally, in Section 5, we conclude with some final comments.

2. WEB ACCESSIBILITY

The W3C Consortium is employing a lot of efforts and is leading a lot of activities to make the most famous information space accessible for anyone, and then allow people with disabilities to actively perceive, understand, navigate, and interact with the Web.

Within this area our interest is mainly devoted to the provision of personalized applications, that is, applications that are able to customize Web content according to users’ preferences and needs. In particular we have implemented some examples of services that make Web pages more accessible for users with visual and motor disabilities.

Some disabilities, such as hearing impairments, do not require any transformation on HTML pages accessed by users. Visual and motor impairments could be addressed by using screen readers or assistive technologies, respectively (modified keyboards or software for keyboard and mouse emulation). But our aim is how to address these challenges with an instrument that is able to personalize the Web without any modification both on client and server systems.

Some surveys\(^3\) on accessibility in Web design estimate that there are 180 million people worldwide who are visually impaired, 45 million are blind and 135 million are partially sighted. In the US, 16.5 million people suffer of vision impairments. Finally, the efforts in making the Web a more accessible place is very important since the the rapid aging of the population and the estimated increase of disabled users (a reasonable estimate is a doubling by 2020 worldwide and a total of 20 million US people by 2010).

2.1 Examples of Web accessibility systems

Assistive devices, like screen readers and audio browsers, are being used by blind users to access the Web. However, these systems show limitations since they are not able to filter Web pages for removing useless information involving users, bothered by more and more intrusive elements, to quickly leave their frustrating navigation.

In order to address this issue, several systems have been developed with the aim at providing Web content accessible to everyone. These systems, that can be classified as filter and transformation tools, include for example, systems that build an only-text version of Web documents (BETSIE and textualise), systems that apply transformation according to users’ preferences (Web Access Gateway and Tablin), systems that apply transformations according to specific rules and heuristics (WAB).

BETSIE, acronym of BBC Education Text to Speech Internet Enhancement [6], is a simple CGI Perl script whose main goal is to tackle the difficulties experienced by people using text-to-speech systems for Web browsing. BETSIE produces on-the-fly text-only version of every Web page navigated by the end user and, in particular, it modifies Web pages by handling frames, by linearizing tables, by removing images, Java and Javascript code, etc. Finally, it allows any user to choose among some color-theme, specific sizes and font for text. An important limitation of BETSIE is that it has not been possible for the BBC to offer a full text-only version of the entire Internet, this means that BETSIE only works with BBC sites.

The \{textualise;\} system [27], transforms a Web site from a graphics-heavy and inaccessible version, to a text-only version that is easily accessible for visually impaired users. By following the W3C’s Web Content Accessibility Guidelines, it removes elements that screen readers cannot handle, allows user customization of fonts and background colors, it removes Java applets, Javascript code, graphics and fixes ALT attributes, replaces Shockwave, Flash, and other plug-in applications with a link to them. etc. A server component, within the textualise; system, performs the required transformations by using a set of transformation rules.

The Access Gateway system [7], specifically designed for people with low vision or dyslexia, is able to handle frames (by serializing them), tables (by linearizing them), to remove images (by substituting them with missing ALT attributes), JavaScript codes, Flash, cookies. Another goal is how to allow speakers of other languages to view Web pages written in these languages, when the encodings are not supported by their browsers. It is implemented as CGI script in C++.

WebAdapter [19] is WWW browser that provides accessibility functionalities for blind, visual and physically impaired people. Their idea is to include these functionalities within a browser without affecting the UI for non disabled users. In particular the adaptations provided by WebAdapter include adaptation for physically impaired users (customization of images sizes), adaptations for visually impaired users (turning-off of background images), and finally, adaptations for blind users (sequential presentation of tables). Finally, WebAdapter use an integrated speech synthesizer to read HTML documents for people with visual disabilities.

The Accessibility Transformation Gateway [23] is a transcod-
ing gateway designed to apply on-the-fly transformations of Web pages in order to adapt them for users with disabilities. The Web pages, filtered by the gateway, can be easily handled and rendered by screen readers or assistive technologies. The access gateway intercepts requests/responses by applying transformations according to the accessibility and usability rules [29]. An important step include the construction of the Document Object Model (DOM) tree representation of the requested HTML page. Finally, some chosen rules include: (a) providing alternate text for images, applet and for each Object, (b) linearizing tables, (c) avoiding any URL redirect and automatic page refresh, etc. Other functionalities, implemented as part of the accessibility transcoding gateway project, are simplification (i.e. deletion of clutter information on Web pages) and summarization (building of a preview of Web pages) [22].

The IBM system described in [17] enables universal access to the Internet content by allowing different types of devices, and people with different abilities, to receive content that is suitable for their needs. It removes comments and JavaScripts, handles images (by modifying colors or removing them at all), linearizes tables, allows text summarization. The evolution of this system led to a most famous IBM WebSphere Transcoding Publisher technology that dynamically translates Web content and applications to meet different client devices capabilities and users preferences.

Crunch [15, 16] is a Web proxy that uses a set of heuristics to extract content from HTML pages in order to make it accessible according to W3C Guidelines. It employs a set of heuristics, that filters operating on a DOM representation of a Web page, to remove all extraneous of useless information (not recognizable, for example, by screen readers). In particular, the content extractor navigates the DOM tree recursively and remove and adjust specific nodes by leaving only the content behind. Crunch provides filters for removing images, links, scripts, and more complex filters to remove advertisements, link lists, empty tables. Once entirely parsed and adapted, the page can be rendered both in HTML and plain text (for example for text to speech and summarization). Finally, the Crunch is not able to handle Flash and dynamic generated codes and to distinguish between different accessing users.

Other examples include WAB, a not customizable HTTP proxy (based on CERN httpd) that modifies Web pages to assist blind users. The page is transformed in a text-only version to make it accessible to screen readers. Tablin [26] is a filtering system, developed by the WAI Evaluation & Repair (ER) group, that can be used to linearize HTML tables and render them in a form suitable for their reading from screen readers.

2.2 Intermediary Frameworks

Most Web sites, today, are designed to follow the “one-size-fits-all” philosophy, by providing content and services without taking care of the abilities of users that access them. In fact, because of its increasingly complex infrastructure, the Web does not really provide equal access and equal opportunity for users with disabilities. The documents available on the Web exhibit a growing complexity, especially for people with visual disabilities, that often are unable to access, summarize and distill information on a Web pages or groups of pages. For people with motor disabilities the WWW represents a very important source of information, a familiar and ubiquitous environment where to get information about any aspect of life: education, employment, shopping, business, government and more.

Hence, as the Internet continues to evolve with an increasing diversity and heterogeneity, there is a growing demands for logical solutions that are able to allow the universal access to the Web content, by breaking down the accessibility barriers that inhibit the access by users with disabilities. These technologies could be provided server-side, client-side or on intermediary systems. While server-side solution could involve several efforts for Web designers for writing an re-designing accessible Web pages, and client-side solution require the development of adaptive browsers or other assistive technologies, intermediary solutions could be provided without any intervention both on client and server systems, by allowing transcoding and Web personalization on-the-fly, transparently for users with disabilities [1, 20].

By summarizing, the main advantages of edge servers systems are that they can be transparently deployed without involving hardware and/or software changes both on client and server systems, they can reduce both complexity on servers (that will only take care of providing the requested resources) and system requirements on clients (independently from device’s capabilities), add new services without stopping the servers, by ensuring their fault-tolerance, offer increased flexibility (where new components can be deployed) and scalability, and, finally, improve the quality of access to the resources available on the Web.

Increasingly often, intermediaries have been used to develop and deploy services for accessibility. Famous examples of intermediary systems that can be easily programmed to promote accessibility include Muffin [http://muffin.doit.org], a Web HTTP proxy that provides functionalities to remove cookies, kill GIF animations, remove advertisements, add, remove, or modify any HTML tag, remove Java applets and JavaScript code, etc.

Rabbit [http://rabbit-proxy.sourceforge.net] is a Web HTTP proxy that accelerates the delivery of Web contents to end users by compressing text pages and images, by removing unnecessary parts of HTML pages (background images, advertisements, banners, etc.) and, finally, by caching filtered documents before forwarding them to the clients.

Web Based Intermediaries (WBI) [3, 4, 5] is a dynamic and programmable framework, developed at IBM Almaden Research Center [http://www.almaden.ibm.com/cs/wbi], whose main goal is to personalize the Web by realizing an architecture that simplifies the development of intermediary applications. WBI defines a programming model that can be used to implement all form of intermediaries, from simple server functions to complex distributed applications.

On top of the WBI programmable proxy, as part of the more complex SEsS: Scalable Edge-computing Services system [12], has been developed the Test-To-Speech service [2], that allows the speaking of the text of HTML pages during their displaying towards end users. Moreover, the Test-To-Speech Service can help the comprehension of documents that are written in foreign languages, since a reader that is partially familiar with the spoken language and not with its written form can be supported in getting, at least, the meaning of the information contained in the document. Finally, the service can be used as a support toward kids that want to learn a language since provides a written/spoken presentation of material of their interest during a natural experience as navigating the Web.

The framework that has been use to develop PAN is Scalable Intermediary Software Infrastructure for edge services (SISI): a brief description of the framework and its functionalities are presented next.

2.3 Scalable Intermediary Software Infrastructure (SISI)

The Scalable Intermediary Software Infrastructure for edge services (SISI) [8] is a flexible, dynamic and programmable intermed-
This framework has been designed with the goal of guaranteeing an efficient and scalable delivery of personalized services at intermediate edge server on the WWW.

SISI programmability is a crucial characteristic since it allows an easy implementation and assembling of adaptation services that enhance the quality of services perceived by users during their navigation. To allow programmability, the SISI framework provides a programming model and a set of APIs that can be used for a quick prototyping and a easy development of new services to improve the navigation on the Web for not-disabled users as well as for disabled ones. Services can be assembled and configured to enhance the set of pre-defined functionalities.

The architecture is innovative since it provides programmability without giving up efficiency, offers users profiles management primitives, deployment/un-deployment and authentication/authorization mechanisms. The work is placed on top of existing open-source applications such as Apache Web server and mod_perl because of their quality and their wide popularity.

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

The **ProxyPerl Module** intercepts all clients requests and initializes the transaction process. Its most important task is to fetch the requested URLs manipulating, if necessary, HTTP headers. If no transformation has to be applied on the HTTP flow, the requested document will be sent back unchanged to the client. Otherwise, the transaction will be managed by the handlers invoked according to the user’s profile.

The **Authorization Module** is useful both to restrict access to the proxy server as well as to distinguish between users, so that each user can have different SISI services applied to her requests.

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

Finally, the **Deploy Module** is used by the programmer to add new services to the framework. It consists of an automatic modules generation process that implements intermediary services starting from simple Perl files.

The SISI framework also provides a Graphical User Interface implemented to simplify the management and the debugging of the SISI components. The goal is to relieve the system administrator and programmers from learning or remembering complex and tedious commands during the administration phase and the debugging of new deployed modules.

SISI user-friendly configuration of services is an important feature since in this phase the users can easily provide information about the required services and personalize their navigation on the Web. In fact, by allowing services configuration, SISI is able to affect the adaptation of a given delivery context, and to change accordingly the user experience.

### 3. Personalizable Accessible Navigation

Personalizable Accessible Navigation, developed on top of the SISI framework, offers to people with disabilities important functionalities described below.

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

**Easy deployment**: client-side configuration is quite simple, since it is enough to use the proxy setting on the browser and providing the authentication to PAN to use services. Moreover, it provides ubiquitous services from any node and does not require installation (and therefore, administrator privileges) which makes it particularly useful when accessing the Web from public terminals as well from mobile terminals. Our main goal is to avoid complex and computationally onerous browser-dependent technology [18] and provide the same set of services to each user, in any context (be it home, workplace or public terminal) with different devices (such as fixed or mobile terminals).

**Efficiency**: the accessibility services are executed on the path from server to client, therefore amortizing the cost of implementing expensive transcoding techniques onto the “natural” delay that is considered physiologic.

#### 3.1 Installation

To use the PAN’s set of accessibility services, users have, firstly, to install the SISI framework, that is available as raw source code for Unix/Linux platforms and in a pre-compiled version for Windows. The installation of the SISI framework requires the installation of the Apache Web server and mod_perl, and some packages for image manipulations (PerlMagick library) and for developing Web client applications (LWP User Agent perl package).

The installation and the deployment of PAN is accomplished by simply using the deployment mechanism provided by the SISI framework (see [8] for more details).

#### 3.2 Configuration and profiles

Once deployed, the PAN set of services is accessible through a specific configuration page (See Fig. 2).

In fact, our approach to manage user profiles is to explicitly ask the user what s/he needs and use this information with a rule-based approach to personalize the content. In particular, users have to fill-out forms to create new profiles and to modify or delete the existing ones, as shown in Fig. 2.

Our user-friendly configuration of services is an important feature since in this phase the users can easily provide information about the required services and personalize their navigation on the Web. In fact, by allowing services’ configuration, it is possible to affect the adaptation of a given delivery context, and to change accordingly the user experience.

When a new user is added to the system, a default profile is automatically generated and the user can modify it the first time s/he logs into the system to choose his/her preferences.
As an example, a user with a moderate low-vision disability may be able to navigate with a modest help of PAN when using his/her usual terminal (due to good indoor illumination and large-size screen) but may need more help when accessing the network by public terminals (i.e., ordinary-sized screens) or even more help is needed when browsing the Web by a mobile terminal. In each context, the user can select a different profile, previously set and configured on PAN, obtaining the right amount of support, right there, when he/she needs it.

Finally, it must be mentioned that PAN is the only system that allows personal configuration of services as well as multiple profiles when compared to the systems described in section 2.1.

4. PAN SERVICES FOR ACCESSIBILITY

The services provided by PAN are grouped in four main categories depending whether they act on text, links, images or other objects on the HTML page (such as pop-up windows, etc.) according, also, to the classification implicitly provided in [29]. Of course, many of the guidelines provided by W3C for making accessible a Web site can be also found on the corresponding Best Practices for Mobile Web [14]: as a matter of fact, dealing with user limitations or terminal limitations can be seen as the two faces of the same goal: promote universal access to Web resources.

4.1 Text-based edge services

Here we describe two services that adapt Web pages by taking into account the rules suggested by W3C to improve accessibility [29] and to enhance, in general, the navigation of Web pages [28], and, more specifically, of CSS files [9].

4.1.1 The CSS-Restyling edge service

Cascading Style Sheets (CSS) files benefit accessibility since they separate document structure from its presentation. More specifically, they allow to define how different elements, such as headers, text, links, etc., should appear to end users. They establish a precise control over the structure of Web pages, by providing rules for characteristics such as spacing, alignment and positioning. They also allow to define a precise control over text style effects such as font size, color, and color-contrast, etc.. Finally, style sheets allow Web designers to simplify the structure of Web pages and clean up them by useless information and dynamic elements, by making them more slight and more accessible at the same time (i.e. more readable from screen readers).

Our CSS-Restyling edge service accesses the Web page retrieved by the origin server, parses it and, for each HTML tag of the CSS file, modifies its attribute by replacing it with accessible values as suggested by [9, 14]. In particular, in order to deliver information to a greater number of users, we perform the following changes:

- Font size (text and links) is changed according to service parameters (Guideline 11 in [13]);
- Text and link colors, background and foreground color combinations are changed to provide sufficient contrast when viewed by users with color deficiency (Checkpoint 2.2 Color contrast, Guideline 2 Text formatting and position in [13]);
- Blinking content is turned off (Checkpoint 7.2 Text style effects, Guideline 7 in [13]);
- Textual cues instead of images (by always providing for each image the corresponding alt attribute with the content attribute of the img tag, if present, or the name of the image) (Checkpoint 3.3 Text instead of images, Guideline 3 in [13]).

The CSS-Restyling edge service applies transformations on Web pages by taking into account both internal and external CSS files. The values provided by the CSS-Restyling service can be customized by the user according to his/her needs and abilities by accessing a form on the proxy side (see Fig. 3).

An example of application of the CSS-Restyling edge service is shown in Fig. 4.
Figure 4: The CSS-Restyling edge service. The original WWW 2006 Website (top) and the same page with the application of the CSS-Restyling service by PAN (bottom).
The main goal of this service is to adapt Web pages in order to make them more accessible for people with color deficiencies.

## 4.2 Link-based Services

This section include services that act on links of Web pages in order to make Web pages more readable when users use assistive technologies such as speech synthesizers, screen readers, etc.

### 4.2.1 The RemoveLink Service

By following the Guideline 1 defined in [29], the RemoveLink edge service brutally removes all anchors (the HTML `<A>` tag) from HTML pages, by replacing them with plain text (the text between the open and close of the anchor tag).

The main advantage of this service is that screen readers or other vocal browsers will avoid to read useless information in Web pages when users that access them are not able to see links and follow them. Examples of Web pages include Web pages news, meteorology, encyclopedias pages, etc., in which no links to other resources are required.

### 4.2.2 The LinkAccessKey Service

The W3C Guideline 9 Design for device-independence [29], in section Keyboard access, and the section 5.2 Navigation and Links (5.2.6 Access Keys) in [14] state the need to use access keys HTML elements to allow users with some disabilities to browse the Web.

Content developers should always ensure that users may interact with a page with input devices other than a mouse. Unfortunately, since this limitation is not often considered, we provided a solution by developing a service that allows both motor and visual disabilities users to browse the Web without limitations.

This service adds to any link embedded in a Web page a numeric Access Key in such a way to make it accessible through a simple combination of keyboard keys, ALT+Access Key+Return for example (See Figs. 5 and 6). Pressing the access key assigned to an element gives focus to the element, and the corresponding action will be executed. In particular by pressing the access key, the browser will follow the corresponding destination link. Moreover, the numeric value of the access key is also added into the HTML source of the Web page so it can be easily read by any screen reader. The LinkAccessKey edge service could be useful to improve the Web navigation on devices with limited display capabilities.

### 4.2.3 The LinkRelationship edge Service

As defined by the W3C guideline 13 Provide clear navigation mechanisms and by section 5.2 Navigation and Links (5.2.2 Navigation Bar) in [14] content developers should use the LINK element and link types to describe document navigation mechanisms and organization. Some user agents may synthesize navigation tools or allow ordered printing of a set of documents based on such markup. The LINK element may also be used to designate alternative documents. Browsers should load the alternative page automatically based on the user’s browser type and preferences. For example, content developers can produce a different content for browsers that support “braille” rendering.

Our service adds a toolbar containing the LINK attributes on top of each HTML page, as shown in Figure 5. It can also be useful to make HTML pages more accessible and more readable by screen readers, and for improving the Web navigation trough devices with limited capabilities.

The W3C defines, other than the traditional set of relationship links (Start, Preview, Next, Help, Bookmarks), another useful set for Web accessibility. It includes, for example, the relationship link Alternate, designed to provide alternative versions of documents, the Contents link relationship, that refers to a summary or as site map, the Glossary, that refers to a glossary of terms contained in the current, the Copyright, that it refers to a copyright of the current document, etc.

### 4.2.4 The DeleteTarget edge service

The target attribute of the HTML A tag specifies the frame where the document will be opened. In particular, the _blank value allow to open a new browser window for displaying the corresponding page.

As defined by the W3C Guideline 10 Use interim solutions [29], new browser windows represent a critical problem for screen readers because of their impossibility to jump among different pages. Moreover, when the focus comes back on the original window, the screen reader starts again from the begin of the page to speech the corresponding content.

The main goal of the DeleteTarget edge service is, therefore, to avoid any target attribute in both link and anchor elements, by displaying all pages in the same browser window.

### 4.2.5 The LinkLinearizing edge service

The LinkLinearizing edge service allows to organize and to lexicographically order links embedded in Web documents in order to simplify their direct access by users with visual and motor disabilities, by also adding a numerical access key (See Fig. 6).

This service parses HTML pages and places all discovered links in a table on the top/bottom of them (as specified by the user during the configuration and customization of the service).

### 4.3 The FilterImages edge services

Web images represent a serious problem for blind users since they cannot be read by screen readers. Moreover, particular images formats or compositions, represent a barrier also for cognitive disabled people, as shown next.

#### 4.3.1 The ImageRemoval edge Service

The ImageRemoval edge service removes any image embedded in a Web page by replacing it with a link to it, as suggested in section 5.4 Page Definition (5.4.5 Non Text Items) [14].

In particular, this service parses the Web page looking for HTML `<img>` tags, and replace them with a `<A>` tags whose content attribute will contain the link information (See `<image>` followed by the original `<img` content attribute, if it exists, and only the `<See image>` link
message, conversely. In this way, only if requested, the image will be displayed to end users (See Fig. 7).

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

4.3.2 The GIF-Deanimate Service

GIF animated images represent a digital barrier for users with cognitive disabilities. In fact, these images reduce attention from the content, involving important problems to users that suffer of dyslexia or of general attention deficit pathologies.

By following the Guideline 7: Ensure user control of time-sensitive content changes [29] we have developed a GIF-Deanimate edge service that replaces each animated image embedded in a Web page with a static one, by showing only its first frame.

The GIF-Deanimate service has been implemented by using the well-known PerlMagick [24] library, that is an object-oriented Perl interface to ImageMagick [21], a free software that allows to create, edit, and compose images with different formats.

4.4 Easy and smooth navigation service

4.4.1 AnnojanceFilter edge service

As defined in the Guideline 10: Use interim solutions [29] navigating with several pop-up windows, banners, scripts etc., can be difficult for people with visual and cognitive disabilities.

The main goal of the AnnojanceFilter edge service is to get rid of particularly annoying abuse during the navigation on the Web. More precisely, the AnnojanceFilter service provides functionalities for removing advertisement, banners, pop-ups in JavaScript and HTML, JavaScript code, for disabling unsolicited pop-up windows, etc. During service’s configuration users can choose the functionality to enable by providing parameters if required.

4.4.2 The HTMLClean edge service

The main goal of this service (based on the HTML::Clean Perl library) is to clean HTML pages from useless or redundant code. It is very useful for Web pages built with programs that silently add internal code. Other functionalities include removing white spaces (Checkpoint 3.3: Text formatting and position [13]), useless (i.e. non-standard) META elements, HTML comments, replace tags with equivalent shorter tags. By using the provided methods, the service is able to reduce the size of Web pages thus speeding up the download.

The HTMLClean service is able to ensure that documents are clear and simple so they may be more easily understood (Guideline 14: Ensure that documents are clear and simple. [29]).

5. CONCLUSIONS

Personalizable Accessible Navigation represents an efficient, effective, simple and personalizable tool to provide easier access to the Web resources to people with disabilities. Its nature (being an intermediary) and the characteristics of its services make it suitable for adapting the access to Web resources in different contexts, such as fixed/mobile terminals. Moreover, by implementing easy configuration and multiple profiles PAN enhancing the usability of the accessibility services in many different contexts.

We have realized preliminary tests in order to evaluate the effectiveness of Personalizable Accessible Navigation accessibility services. We selected a set of Web pages of international relevance, i.e., the US White House, CNN and New York Times Web portals, the Eclipse and Sun Web sites, and finally, the IEEE, and ACM Organization sites.

Their Web pages are tested using the Bobby Web Accessibility Tester (available at the URL: [http://webxact.watchfire.com]) that is a comprehensive Web accessibility tool designed to aid Web masters to test Web sites in order to improve their accessibility.

Moreover, Bobby tests Web pages by using the guidelines established in [28].

We compare the results obtained on the original Web pages by using Bobby with the results obtained on the Web pages adapted by PAN (by saving the adapted pages on a local server and accessing them with Bobby later on). The following PAN’s services were tested: ImageRemoval, the CSS-Restyling, LinkAccessKey and AnnojanceFilter. Results are showed in Fig. 8 for the original Web pages and in Fig. 9 for the pages adapted by PAN.

In these figures we report on the Y-axis the number of errors or warnings obtained by Bobby on each page for the categories “Image Evaluation”, “Insufficient Color Contrast”, “Animated Objects”, “Pop-ups and new browser windows”, i.e., the lower the number, the more accessible is the page.

The Bobby Accessibility Tester checks if analyzed Web pages are fully compliant with all of the automatic and manual checkpoints of the W3C Web Content Accessibility Guidelines. PAN,

As defined in the Guideline 10: Use interim solutions [29] navigating with several pop-up windows, banners, scripts etc., can be difficult for people with visual and cognitive disabilities.

The main goal of the AnnojanceFilter edge service is to get rid of particularly annoying abuse during the navigation on the Web. More precisely, the AnnojanceFilter service provides functionalities for removing advertisement, banners, pop-ups in JavaScript and HTML, JavaScript code, for disabling unsolicited pop-up windows, etc. During service’s configuration users can choose the functionality to enable by providing parameters if required.

4.4.2 The HTMLClean edge service

The main goal of this service (based on the HTML::Clean Perl library) is to clean HTML pages from useless or redundant code. It is very useful for Web pages built with programs that silently add internal code. Other functionalities include removing white spaces (Checkpoint 3.3: Text formatting and position [13]), useless (i.e. non-standard) META elements, HTML comments, replace tags with equivalent shorter tags. By using the provided methods, the service is able to reduce the size of Web pages thus speeding up the download.

The HTMLClean service is able to ensure that documents are clear and simple so they may be more easily understood (Guideline 14: Ensure that documents are clear and simple. [29]).

5. CONCLUSIONS

Personalizable Accessible Navigation represents an efficient, effective, simple and personalizable tool to provide easier access to the Web resources to people with disabilities. Its nature (being an intermediary) and the characteristics of its services make it suitable for adapting the access to Web resources in different contexts, such as fixed/mobile terminals. Moreover, by implementing easy configuration and multiple profiles PAN enhancing the usability of the accessibility services in many different contexts.

We have realized preliminary tests in order to evaluate the effectiveness of Personalizable Accessible Navigation accessibility services. We selected a set of Web pages of international relevance, i.e., the US White House, CNN and New York Times Web portals, the Eclipse and Sun Web sites, and finally, the IEEE, and ACM Organization sites.

Their Web pages are tested using the Bobby Web Accessibility Tester (available at the URL: [http://webxact.watchfire.com]) that is a comprehensive Web accessibility tool designed to aid Web masters to test Web sites in order to improve their accessibility.

Moreover, Bobby tests Web pages by using the guidelines established in [28].

We compare the results obtained on the original Web pages by using Bobby with the results obtained on the Web pages adapted by PAN (by saving the adapted pages on a local server and accessing them with Bobby later on). The following PAN’s services were tested: ImageRemoval, the CSS-Restyling, LinkAccessKey and AnnojanceFilter. Results are showed in Fig. 8 for the original Web pages and in Fig. 9 for the pages adapted by PAN.

In these figures we report on the Y-axis the number of errors or warnings obtained by Bobby on each page for the categories “Image Evaluation”, “Insufficient Color Contrast”, “Animated Objects”, “Pop-ups and new browser windows”, i.e., the lower the number, the more accessible is the page.

The Bobby Accessibility Tester checks if analyzed Web pages are fully compliant with all of the automatic and manual checkpoints of the W3C Web Content Accessibility Guidelines. PAN,
instead, addresses at this time, a subset of them. This explain the reason whereby the accessibility degree is not highly improved in all the Web pages tested.

More precisely, the accuracy that comes out form tests is inconstant for Insufficient Color Contrast, Image Evaluation and Animated Objects categories since Bobby observes errors, closely related to image color contrast and Flash animations, that we do not still have provided as services in our system. On the other hand, we have already envisioned these functionalities, and we can quickly implemented them, since the programmability of the SISI’s framework where PAN is leveraging on.

Moreover, since the accessibility degree is highly improved in almost the Web pages tested, our immediately important steps include further automatic tests and, in addition, detailed usability tests in order to also estimate the effective benefits that users can perceive during their navigation on the Web.

Finally, we must mention that the PAN location in the World Wide Web architecture makes it usable in three different ways: the first one is the traditional proxy setting and could be suitable for providing public services to communities. But, PAN can also act as an HTTP surrogate, that is as an intermediary that acts on behalf of an origin server to provide complex functionalities, as shown in Fig. 10 therefore adding to a non-accessible Web site the opportunity to become accessible.

On the other hand, PAN, can also be placed at the other end of the client-server path, by playing the role of an HTTP delegate on behalf of client systems, by providing personalized functionalities that can be also used for accessing off-line HTML repositories as well as intranet contents (see Fig. 11).

Acknowledgments: The authors gratefully acknowledge the support and the interesting discussions within the ISISLab. In particular, we thank Angelo Esposito for collaborating in the implementation of CSS restyling PAN service. Part of the research was supported by EU “Leonardo da Vinci” project Web for handicap integrated training environment (WHITE), I/03/B/F/PP-154143, 2003.

6. REFERENCES


Structure Benefits All

Aaron Leventhal
IBM, 14 Crosby Street
Arlington, MA
781-643-5083
http://www.mozilla.org/access
aleventh@us.ibm.com

ABSTRACT
Accessibility for the Dynamic Web is now possible due to new standards being developed at the W3C and being implemented in Firefox. The technology allows today’s web pages to contain additional markup describing semantics. An often-cited benefit of this technology is the ability to describe scripted widgets with dynamic behaviour. However, another major benefit is to differentiate the sections of a web page, via human-readable labels or predefined semantics such as “main”, “contentinfo”, “navigation” and “search”. Marking the sections of a web page offers significant improvement for users who need access to today’s web with a small device or an assistive technology.

Categories and Subject Descriptors

General Terms
Human Factors, Design, User Agents

Keywords
Accessibility, Semantics, Dynamic HTML, User Agents

1. INTRODUCTION
This history of software development is filled with unsuccessful attempts to converge onto a single popular technology that can provide write once, run anywhere, while most likely unattainable. However, the web today is getting closer to that reality – or, at least to write once, test and debug everywhere. There are now reasonable choices for developing applications that work on both the desktop and on mobile devices, albeit each with a different cost/usability tradeoff.

This paper will discuss techniques for authoring websites that work for a variety of users on a variety of devices.

The focus is on making mainstream websites work better on small devices, using simple techniques which provide accessibility advantages for free.

2. DECIDING WHICH MARKUP TO USE
First, lets establish that we are talking about content which will be served in either HTML, XHTML or a variant such as XHTML-MP (mobile profile). In reality the mobile web is converging on HTML, which is nearly a superset of the other two languages. Why? All mobile browsers need to support HTML for competitive reasons – its become a must have feature that end users can browse the real web. Because of this, mobile developers are not necessarily seeing a driving need to use a more restrictive markup dialect when targeting mobile platforms.

Depending on the situation a developer faces, it is reasonable to use HTML, XHTML Basic or XHTML-MP, but in fact this choice matters less and less as the mobile web evolves. Those developers writing from scratch, concerned with standards or who believe the future belongs to true device independence via XForms, X+V or SVG, should currently choose XHTML Basic. Those developers who simply wish to be in sync with the web today, may simply follow basic HTML authoring techniques for small devices and accessibility.

3. DECIDING HOW MUCH TO TAILOR THE CONTENT FOR SMALL DEVICES
A bigger choice point is how much effort will be put into tailoring the content for small devices.

3.1 The Easiest and Least Effective Choice: Do nothing
In what’s unfortunately the most likely scenario, the author will not concern themselves with the fact that small devices may use of their content. What happens to this content on a small device? It depends on the content, the browser and the device.

Opera’s Small Screen Rendering
Two pieces of software consistently do a good job formatting mainstream web pages for small devices – Opera and Opera Mini[1]. These two browsers use Operas SSR (Small Screen Rendering), and generally do a great job of unrolling the content into a long column that makes maximum use of the screen size[2]. The SSR rendering removes the need for users to scroll back and forth horizontally in order to read the
page. Navigation through the content can be accomplished just with vertical scrolling. Unfortunately Operas SSR may not always provide a satisfactory version of the content, if the developer:

- Structured the document out of order from its natural reading order
- used images which are too large to work on a small screen (and require too much download bandwidth)
- used dynamic effects like pop out submenus which may not fit on the screen
- relied on mouse pointers—there may be no pointing device or none which produces events like mouseover
- used frames

In addition, not all mobile browsers have as advanced technology as SSR for dealing with real web pages. Even if SSR handles a given web page perfectly, it is very likely that the layout will not be satisfactory on other mobile browsers.

This kind of technology will greatly help format web sites for many devices, without requiring the author to consider every possible configuration. However, it cannot be even close to optimal without some cooperation from the author.

And, although SSRs layout is an improvement, it does not reorder the content to show the most important items first. It is still very difficult to find a web page that does not require paging down through many screens before reaching the main content. It can also be difficult to just find the search box for a web site.

The problem is that most websites position the link navigation sections to the top and left of a site. Since these are usually first in the document, SSR displays them first even though they are arguably the least important content. Opera recommends putting most the navigation links to the right side of the page. Unfortunately, most web sites go against this recommendation, and most authors are not willing to change their design because of this recommendation.

As we discuss at the end of this article, it is currently possible to mark navigation, search and main content sections, and ultimately this will allow mobile users to find what they need more quickly.

Example of Small Screen Rendering

Here is the ideal before (desktop) and after (on a small device) of Small Screen Rendering:

![Figure 1: The Eclipse web site on the desktop, before transformation.](image)

The Eclipse example shows the reasonable limit for SSR for the usual case. The normal layout from figure 1 changes dramatically in figure 2, which shows the content rendered vertically: a header, then the search box, then a dozen links followed by the main content. The main content is not on the screen when you first enter the site using Opera. However, the user has only a small distance to scroll down before the main content begins.

Many things can and usually will go wrong, and even in this ideal case usability is impaired by all of the links cluttering the beginning of the page, rather than coming after the main content.

Although far from perfect, innovations such as Small Screen Rendering do prove that a lot is possible, even when dealing with web content as it exists today. If only desktop screen magnification software such as ZoomText, MAGic and Lunar knew this trick, it would help low vision users quite a bit! As it stands low vision users often have a difficult time finding what they’re looking for on web sites, as they pan and scroll both horizontally and vertically—finding the link you want can be like an exercise in mowing the lawn. It will be interesting to see how browsers on small devices evolve their reformatting techniques.

3.2 A More Effective Choice: Follow Basic Tips

The following tips have been collected from Operas developer pages[3], with italicized comments provided by the author. Again Opera has shown leadership in the field by putting together a complete, simple to understand resource for developing content which works well on small devices. These techniques are not expensive to implement in a website being developed from scratch. Adding to their usefulness, over 90 of these tips are also good for accessibility.

Coding tips

1. Use terse, efficient markup—also good for accessibility!
2. Avoid frames – also good for accessibility!
3. Avoid pop-ups – also good for accessibility!
4. Avoid using proprietary features, or use fallbacks also good for accessibility!
5. Specify image height and width Images load slowly on small devices – this helps the web browser know ahead of time how much screen real estate they will need.
6. Use alternative text on images – also good for accessibility! It is common to browse with images off because this greatly decreases the wait and cost for the end user.
7. Have fallbacks for JavaScript and dynamic effects – also good for accessibility!

Testing Tips
1. Test in Opera, in Small-Screen view (shift+F11) also good for accessibility!
2. Test with graphics turned off also good for accessibility!
3. Test with JavaScript turned off – also good for accessibility!
4. Test with no mouse – also good for accessibility!

Small-Screen design tips
1. Design with document order in mind – also good for accessibility!
2. Design the small-screen version for maximum readability – also good for accessibility!
   (a) Use headings. These can be used to the users advantage, via navigation shortcuts which move the user between headings also good for accessibility!
   (b) Avoid tables for layout use CSS instead also good for accessibility!
   (c) Do not rely on color in text it will all be shown black on light gray also good for accessibility and printing
3. Only use images suited for a small screen, hide the rest in mobile media style sheet Large images do not work well on a small screen, so some images are removed and others are scaled. Images slicing will not work.
4. Be careful with the use of colors, font sizes, and alignment – also good for accessibility!

Modern devices usually have a color screen, but they offer less contrast than a normal PC screen. Devices are also often used in sunlight and other difficult conditions while PC screens are used in more controlled environments. Reducing screen contrast can be used to conserve battery length. For these reasons you should use good contrast between foreground and background. Expect subtle color differences to disappear.

Utilizing Operas tips in your mainstream web site is the least expensive solution which still works fairly well.
On the minus side, it still really provides the user with too much information for a small screen. The user will often need to scroll vertically to find the important content. Thus, this technique will not bring millions of mobile customers to a portal.

3.3 A Bit More Optimal: Create a Mobile Style Sheet
SSR is clever enough that it will not take action when a mobile style sheet is present. In that case SSR assumes that the mobile style sheet has been designed for efficient mobile browsing.
Creating a mobile style sheet has a few advantages:
1. The author can code to remove content which is redundant, purely decorative or of lesser importance
2. The author has complete control over layout
3. This solution does decrease the download size significantly in that images with the CSS rule display: none will not be loaded. However, the HTML document will still be loaded.

Here is some sample code to insert a mobile style sheet:

<!-- Stylesheet for devices -->
<link rel="stylesheet" href="phone.css" media="handheld">

Unfortunately, it is still difficult to avoid having the navigation links show up at the top of the content without removing them altogether, if the desktop version has them to the top or left. Web authors could put link navigation panels at the end of their document and still have them rendered at the top or left, but are unlikely to do so because CSS positioning is not rendered consistently in desktop browsers.
Worst of all, not all mobile browsers will pick up on the mobile style sheet.

3.4 Optimal but Expensive: Create an Alternate Version
Finally, if the mobile market for a particular website is crucial, the alternate version of the content is most likely needed. This will provide the ideal solution for mobile users, in terms of small download size, most readable layout and best usability, an alternative version must be created. This alternative version will be tested and optimized for a variety of mobile devices. Providing a unique URL for mobile users is not recommended, in that mobile users will not as easily remember the special URL.
It is better to detect the presence of a small screen browser by checking the user agent string or the UA Profile. It is difficult, but it can be done. The server can then be configured to present the alternative version of the content.
Providing an alternate version is an option for content providers with a large commercial stake in the mobile space. And, as the accessibility world has discovered, alternative versions of content often lag behind their mainstream counterparts functionality.
3.5 A Newly Evolving Compromise: Add Semantics

The solutions discussed up to this point all have concrete advantages and disadvantages. The first three options are affordable for a wide variety of organizations and developers. The fourth option, creating an alternative version of the content, is only practical where significant resources are available to develop, test and maintain that version.

On the other hand, the affordable options all commonly share one important problem: they tend to result in lost and frustrated users. It’s a bit too tedious to find what you need with a small window within the elongated column of content. If it takes too long to find what they want, the user is most likely going to give up before completing the task.

For example, try to find the main content on the following BBC front page (for this exercise pretending that BBC does not have a mobile version) (see Figure 3).

Start in the Main Content

Clearly, reaching the BBC sites main content requires too much effort. The biggest usability hurdle with this and many sites is allowing the small device user to see or navigate to what they want quickly.

This is also a known problem for users with disabilities. Screen magnification, screen reader and keyboard-only users all have difficulty getting to the main content of a website efficiently after a page loads.

The main content is generally the first place you want to visit, so it would be ideal if screen magnifiers, screen readers, keyboard navigation extensions and small device browsers could all start the user in the main content.

The Importance of Landmarks

All of these user types also need streamlined navigation to other key features of a web site, such as the main search box or a navigation links panel. It would be excellent if the user agents for small devices and users with disabilities could provide shortcuts for navigating to these parts of a page.

Setting Landmarks

Fortunately, there is a potential way to enable these features because web sites can mark up the main content, navigation panel or search field as such. The necessary semantics are called document landmarks. The most important document landmarks are defined in the XHTML 2 [5] Role Access module:

main This defines the main content of a document.

secondary This is any unique section of the document. In the case of a portal, this may include but not be limited to: show times; current weather; or stocks to watch.

navigation This is the navigation bar on a web document. This is typically a list of links to other pages on the site or other areas of the same document.

banner A banner is usually defined as the advertisement at the top of a web page. The banner content typically contains the site or company logo and other key advertisements for the site.

Figure 3: The BBC website unrolled via Operas Small Screen Rendering. The main content shows up somewhere around the 3rd screen full.
The author believes this success is only a matter of time. The industry can build on its successes with dynamic widget role support in screen readers such as JAWS and Window-Eyes. Web pages developed using roles can now provide desktop-like user interfaces to visually impaired Firefox users, as well as the mainstream [7].

As the user agent support becomes more of a reality, developers can begin truly working on techniques to develop content meant for both desktops and small devices.

4. CONCLUSION

Accessibility for the Dynamic Web is now possible due to new standards being developed at the W3C and being implemented in Firefox. The technology allows today’s web pages to contain additional markup describing semantics. An often-cited benefit of this technology is the ability to describe scripted widgets with dynamic behavior. However, another major benefit is to differentiate the sections of a web page, via human-readable labels or predefined semantics such as “main”, “contentinfo”, “navigation” and “search”. Marking the sections of a web page, in principle a simple concept, can potentially offer significant gains for web usability with handheld devices and assistive technologies.

5. REFERENCES

Capability Survey of Japanese User Agents and Its Impact on Web Accessibility

Takayuki Watanabe
Dept. Communication,
Tokyo Woman's Christian University
2-6-1 Zenpukuzi, Suginami-ku,
Tokyo, 167-8585, JAPAN
+81-3-5382-6524
nabe@lab.twcu.ac.jp

Masahiro Umegaki
Dept. Systems Engineering,
Graduate School of Electro-Communications
1-5-1 Chofugaoka, Chofu-shi
Tokyo, 182-8585, JAPAN
umegaki@se.uec.ac.jp

ABSTRACT
Capabilities of major Japanese user agents, three screen readers and one voice browser, were investigated with the following test files: W3C UAAG 1.0 Test Suite for HTML 4.01, an accessible PDF file, an accessible Flash file, and test files which test Japanese specific issues. Using the UAAG 1.0 Test Suite, 20 out of 48 Priority 1 checkpoints were met by all user agents, while all of the user agents failed to meet 11 of the checkpoints. Test results of all test files were assigned into three categories: capabilities satisfied by almost all user agents, capabilities not satisfied by any of the user agents, and capabilities that were satisfied by some of the user agents only. The test results indicated that 1) two major Japanese user agents do not have enough functions to navigate through a Web page using the structure information of the content, and 2) none of the user agents have enough functions to control multimedia and time-dependent interactions. These results provide an objective evidence to define the Japanese baseline, a set of technologies that a user agent is assumed to support, which is required in the WCAG 2.0 working draft. Accessibility responsibility between Web content and user agents is also determined by the current survey.

Categories and Subject Descriptors
K.4.2 [Social Issues]: Assistive technologies for persons with disabilities, H.5.2 [User Interfaces]: Evaluation/methodology, Standardization, H.5.4 [Hypertext/Hypermedia]: User issues

General Terms
Measurement, Human Factors, Standardization, Verification

Keywords
Web, accessibility, user agent, W3C, UAAG, MSAA, DOM, Japan, visual disability.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
W4A at WWW2006, 23rd-26th May 2006, Edinburgh, UK
Copyright 2006 ACM 1-59593-281-x/06/05...$5.00.

1. INTRODUCTION
The importance of Web accessibility has been recognized worldwide. The W3C (World Wide Web Consortium) developed the first version of the Web Content accessibility Guidelines (WCAG 1.0) [3] in 1999. WCAG 1.0 was subsequently used to guide or regulate Web content accessibility in many countries. In the United States, Section 508 of the Rehabilitation Act [11], which includes Web accessibility requirements, came into effect in 2001.

In Japan, private companies developed their own guidelines based on WCAG 1.0. In the public sector, in 1999, "Guidelines for the Creation of Internet Web Content Accessible by People with Disabilities" were jointly announced by the Ministry of Posts and Telecommunications and the Ministry of Health and Welfare. In 2004, JIS X 8341-3 "Guideline for older persons and persons with disabilities -information and communications equipment, software and services- Part3: Web Content" [7], was published as a Japanese Industrial Standard. JIS X 8341-3 was developed paying attention to the harmonization with WCAG 1.0, and with the subsequent WCAG 2.0 working draft [17,18]. Section 67 of the Industrial Standardization Law of Japan states that "When the nation and local public bodies determine standards, they must pay attention to JIS." Thus the JIS X 8341 guidelines affect public sector activity in Japan.

In spite of the above guidelines, Web accessibility remains deficient, even in the public sector. In the United States, according to "The State of Federal Websites: The Pursuit of Excellence" [12] published by PricewaterhouseCoopers Endowment for The Business of Government, only 13.5% of federal web sites were fully accessible (i.e., the Bobby accessibility testing software reported no error for those sites.). Web accessibility has also been evaluated in Europe, as reported by the UK Cabinet Office in a document titled "Accessibility of public sector services in the European Union" [13]. According to that document, detailed assessment of 436 government service websites across Europe showed that only 3% of them achieved Level A conformance with WCAG 1.0. In Japan, "Nikkei Passocon (Personal Computer)" carried out a survey of government websites and found that even the most accessible of them lacked fundamental considerations of accessibility [10].

Improvements of Web accessibility require web site design, authoring, and programming to conform to Web content accessibility guidelines such as WCAG 1.0, Section 508, or JIS X
Web authors, however, sometimes cannot make accessible content if they do not know how user agents render their content. For example, even when "accesskey" attributes are used to make easy access to some important parts of the content, if a user agent does not support an access key function, a user cannot directly access that part with use of an accesskey. Another example is alternative information about an image: an image can have an "alt" attribute which is used to specify alternate text of the image, a "title" attribute used to offer advisory information about the image, or a "longdesc" attribute used to specify a link to a long description of the image. Even if an author uses a "longdesc" attribute to describe the image in detail, a user agent which does not support "longdesc" attributes cannot make use of that information. Another example is heading elements: even if an author uses heading elements such as "h1" and "h2", a voice browser which does not use different voice when reading heading elements cannot convey the structure information to the user. Therefore, improvement of Web accessibility requires knowledge of user agent capabilities.

Japanese users who are blind cannot use user agents (screen-readers or voice browsers) sold in the United States because these user agents do not speak Japanese. Japanese users need either user agents developed in Japan or non-Japanese user agents that are localized to Japan. However, the Japanese market is small relative to the global market for English language user agents. Thus developers of Japanese screen-readers and voice browsers face difficulty generating sufficient revenue to implement enough capabilities to their user agents. As a result Japanese language user agents suffer in comparison to their English language counterparts.

Harmonization of Web accessibility guidelines is of great interest today. For example, Judy Brewer, domain leader of W3C/WAI, argues for the importance of standards harmonization in her presentation [1]. In order to use one international guideline, we must know how Japanese user agents differ from English ones, since Web accessibility is accomplished both by accessible Web content and by accessible user agents [5].

2. OBJECTIVES OF THE CURRENT SURVEY

Based on the evidence and arguments presented in the preceding section, a capability survey of Japanese user agents is a necessary step towards improved Web accessibility in Japan. Carrying out this survey can also provide objective data concerning the issues discussed in the following subsections.

2.1 Conformance to UAAG 1.0

Capabilities of user agents used in the United States have been investigated with the UAAG 1.0 Test Suites. Testing Japanese user agents with the same test suites enables us to show 1) how well Japanese user agents conform to UAAG 1.0 and 2) the differences in capabilities between Japanese and English language user agents. This testing also shows us to examine the applicability of UAAG Test Suites to Japanese user agents.

2.2 Japanese Baseline

Some checkpoints of WCAG 1.0 include the phrase "Until user agents ...". "Checkpoints that contain the phrase 'until user agents ...' require content developers to provide additional support for accessibility until most user agents readily available to their audience include the necessary accessibility features."[3].

In order to make WCAG independent of the capabilities of user agents and Web technology, a new concept of "baseline" was incorporated into the WCAG 2.0 working draft [2]. "About Baselines and WCAG 2.0" [8] explains the baselines as follows:

A "baseline", as used in WCAG 2.0, is the set of technologies that an author assumes are supported and turned on in accessible user agents. Authors must ensure that all information and functionality of the Web content conform to WCAG 2.0 even when a user agent supports and uses only the technologies in the baseline.

This document [8] says

A baseline may be set by a government body, client, organization, author, or combination of these.

In order to apply WCAG 2.0 in Japan, the Japanese baseline must be specified by some authoritative body with objective evidence. In order to specify a baseline in Japan, we need to examine the capability of Japanese user agents, i.e. what technologies are supported by user agents, in detail. If every Japanese user agent can handle JavaScript technology so as to make it accessible to users, then authors of Web content do not have to write alternative content and function given by JavaScript. On the other hand, if some user agents cannot make JavaScript accessible, authors are required to prepare alternative content.

PDF and Flash are widely used in the current Web. To set a Japanese baseline, we must know how well current Japanese user agents adapt these new technologies to the requirements of users with disabilities.

2.3 Accessibility Responsibility Between Content and User Agents

Content and user agents are dependent on each other. Poor user agents require content authors to provide a "repair" content to fix a problem in the content side, while poor content hinders Web browsing even with excellent user agents. This joint dependency between user agents and content hinders the progress of Web accessibility because, when problems arise, authors can blame poor user agents and developers of user agents can blame poor content. Thus methods are needed to distinguish between problems caused by content and by user agents respectively.

3. RELATED WORK

W3C/WAI UAAG (User Agent Accessibility Guidelines Working Group) has developed "a second generation test suite of UAAG 1.0 and implementation reporting system" [4]. "UAAG Implementation report for HTML 4.01" [15] shows test results of various English language user agents such as Internet Explorer, Firefox, Opera, Safari, and Home Page Reader. "Evaluations" [16] shows test results of various English language user agents and English language assistive technologies such as Safari, Window-Eyes with Internet Explorer, Mozilla, Opera, Internet Explorer, JAWS, Home Page Reader, and Firefox.

4. RESEARCH METHODS

Capabilities of user agents were examined using test files. Every test was carried out on Windows XP SP2 with all Windows-
update patches implemented. The following subsection shows three kinds of test files used in the current survey.

4.1 UAAG 1.0 HTML 4.01 Test Suite
As shown in the previous section, UAAG had developed test suites which check if the user agent conforms to UAAG 1.0 [6]. Draft "UAAG 1.0 Test Suite for HTML 4.01" [14] consists of about 400 test files, which are classified into checkpoints of UAAG 1.0.

As illustrated in the sample screenshot shown in Figure 1, each test file consists of a Requirement Reference, Procedure, Test, Source Code, and Results. Testing is done with the Test section of the test file and results are examined in the Result section.

A sample of user agents was tested to see if they meet the UAAG 1.0 checkpoints.

The results in each test file were evaluated according to the following categories given in the Test Suite:

- C (Complete): The user agent passes all tests.
- AC (Almost Complete): The user agent passes all tests but one.
- PI (Partial Implemented): The user agent passes some tests.
- NI (Not Implemented): The user agent passes no tests.
- NR (Not Rated): The reviewer did not or could not run this test.
- NA (Not Applicable): The test does not apply to this user agent.

Secondly, results of every test file were summarized as a result of each checkpoint. When there was no test file for a checkpoint, that checkpoint was evaluated by a human judge based on the textual content of the checkpoint.

4.2 PDF and Flash
The PDF file¹ made by Adobe Systems Inc. as a reference accessibility test was used to our test. In using this test file, the following points were examined.

1) Every piece of content is read.
2) Heading elements are read.
3) Caption and alternative information for graphs is read.
4) Tables are read and navigation is enabled inside a table.
5) Alternative information for images is read.

The Flash file² made by Fujitsu Inc. to show the accessibility features of Flash was also used. With use of this test file, the following points were examined.

1) Every piece of content that was made accessible is read.
2) Objects which were made hidden from a display but were set to be read are read.
3) Tab key navigation is provided between buttons.
4) Reverse Tab key navigation is provided between buttons.
5) All button labels are read.
6) Objects which should be skipped are skipped.
7) Links are activated.

4.3 Japanese Specific Issues
WCAG 1.0 and UAAG 1.0 do not pay attention to language specific issues. JIS X 8341-3 includes some requirements which are important in Japanese and languages which use Han characters (Kanji ideographs). Test files were created³ to check the following issues:

1) How Japanese symbols are read.
2) How language attribute is used to read content.
3) How ruby elements are read.
4) How words that contain whitespace characters are read. (e.g. Word)
5) How ambiguous date, time, and money representations are read. (e.g. 2006/03/22 09:50)
6) If list number of "ol" element is read.

Some of these tests are not Japanese specific issue but we created these tests because the UAAG Test Suites do not test these issues.

All tests were carried out in the default reading-mode, reading line by line, of the user agents.

5. SUBJECT USER AGENTS
The current survey investigates Japanese user agents for person with visual disabilities. The current survey treats a user agent as a combination of Web browser and assistive technologies. Three user agents most popular in Japan, and one user agent (JAWS) known to have high capability, were selected for the survey.

1) IBM Home Page Reader 3.04 SP3 (Japanese edition)
2) PC-Talker XP Version 3.04
3) 95 Reader Version 6.0
4) JAWS for Windows Professional Version 6.2 (Japanese edition)

Home Page Reader is a voice browser, while other three user agents are screen readers which read Internet Explorer. PC-Talker and 95 Reader are domestic screen readers developed in Japan.

6. SUMMARY OF TEST RESULTS
This paper describes summary test results for each user agent.

Detailed test results for the UAAG 1.0 test files and for the PDF, Flash, and Japanese specific issues are shown on our Web site [19].

Test results for each UAAG 1.0 checkpoint are summarized in Table 1. We could not evaluate some checkpoints because evaluating these checkpoints required the knowledge of internal specifications of the software. Those checkpoints were rated as NR and were not included in the following summary and discussions.

The results described in the following subsections are assigned into three categories: capabilities satisfied by almost all user agents, capabilities not satisfied by any of the user agents, and capabilities that were satisfied by some of the user agents only.

---
¹ http://www.comm.twcu.ac.jp/~nabe/data/UAResearch2005/TestData/wordsample.pdf
² http://jp.fujitsu.com/museum/pdp/plasma1.html
³ These test files are at: http://www.comm.twcu.ac.jp/~nabe/data/UAResearch2005/TestData/Japanese/
ACCESKEY attribute for TEXTAREA

Requirement Reference

Test Suite: UAAG 1.0 Test Suite for HTML 4.01
  Guideline 1: Support input and output device-independence
  Checkpoint 1.1: Full keyboard access

Test References
  HTML Accesskey definition

Test is required for conformance

Procedure

- Using the standard keyboard or an assistive technology that emulates the keyboard, move focus to the TEXTAREA using accesskey "T".

Test

Textarea 1 [Accesskey "T"]:
Use the keyboard to activate the textarea.

Source Code

```html
<form action="/action/return.html" method="post">
  <label for="textarea1">Textarea 1 [Accesskey "T"]</label>
  <textarea id="textarea1" accesskey="T" name="textarea1" rows="5" cols="40">Use the keyboard to activate the textarea.<textarea>
</form>
```

Results

- The keyboard or an assistive technology that emulates the keyboard moves focus to the TEXTAREA using accesskey "T".

Copyright 2004 University of Illinois at Urbana-Champaign, Disclaimer

Figure 1. Example of test file (Test 97) in UAAG 1.0 Test Suite for HTML 4.01 [14]
Table 1. Results of UAAG 1.0 Test Suite for HTML 4.01

<table>
<thead>
<tr>
<th>UAAG 1.0 Checkpoint</th>
<th>Priority</th>
<th>PC-Talker 3.04</th>
<th>95 Reader 6.0</th>
<th>JAWS 6.2</th>
<th>HPR 3.04</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1: Full keyboard access</td>
<td>1</td>
<td>AC</td>
<td>AC</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>1.2: Activate event handlers</td>
<td>1</td>
<td>AC</td>
<td>PI</td>
<td>AC</td>
<td>AC</td>
</tr>
<tr>
<td>1.3: Provide text messages</td>
<td>1</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>2.1: Render content according to specification</td>
<td>1</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>2.2: Provide text view</td>
<td>1</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>2.3: Render conditional content</td>
<td>1</td>
<td>PI</td>
<td>PI</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>2.4: Allow time-independent interaction</td>
<td>1</td>
<td>NI</td>
<td>NI</td>
<td>PI</td>
<td>PI</td>
</tr>
<tr>
<td>2.5: Make captions, transcripts, audio descriptions available</td>
<td>1</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>2.6: Respect synchronization cues</td>
<td>1</td>
<td>PI</td>
<td>PI</td>
<td>PI</td>
<td>PI</td>
</tr>
<tr>
<td>2.7: Repair missing content</td>
<td>2</td>
<td>PI</td>
<td>PI</td>
<td>PI</td>
<td>AC</td>
</tr>
<tr>
<td>2.8: No repair text</td>
<td>3</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>2.9: Render conditional content automatically</td>
<td>3</td>
<td>PI</td>
<td>PI</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>2.10: Don’t render text in unsupported writing systems</td>
<td>3</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>3.1: Toggle background images</td>
<td>1</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>3.2: Toggle audio, video, animated images</td>
<td>1</td>
<td>PI</td>
<td>PI</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>3.3: Toggle animated or blinking text</td>
<td>1</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>3.4: Toggle scripts</td>
<td>1</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>3.5: Toggle automatic content retrieval</td>
<td>1</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>3.6: Toggle images</td>
<td>2</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
</tr>
<tr>
<td>4.1: Configure text scale</td>
<td>1</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>4.2: Configure font family</td>
<td>1</td>
<td>PI</td>
<td>PI</td>
<td>PI</td>
<td>PI</td>
</tr>
<tr>
<td>4.3: Configure text colors</td>
<td>1</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
</tr>
<tr>
<td>4.4: Slow multimedia</td>
<td>1</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>4.5: Start, stop, pause, and navigate multimedia</td>
<td>1</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>4.6: Do not obscure captions</td>
<td>1</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>4.7: Global volume control</td>
<td>1</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>4.8: Independent volume control</td>
<td>1</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>4.9: Configure synthesized speech rate</td>
<td>1</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>4.10: Configure synthesized speech volume</td>
<td>1</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>4.11: Configure synthesized speech characteristics</td>
<td>1</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>4.12: Specific synthesized speech characteristics</td>
<td>2</td>
<td>PI</td>
<td>PI</td>
<td>PI</td>
<td>PI</td>
</tr>
<tr>
<td>4.13: Configure synthesized speech features</td>
<td>2</td>
<td>AC</td>
<td>AC</td>
<td>C</td>
<td>AC</td>
</tr>
<tr>
<td>4.14: Choose style sheets</td>
<td>1</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
<td>AC</td>
</tr>
<tr>
<td>5.1: No automatic content focus change</td>
<td>2</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
<td>PI</td>
</tr>
<tr>
<td>5.2: Keep viewport on top</td>
<td>2</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
</tr>
<tr>
<td>5.3: Manual viewport open only</td>
<td>2</td>
<td>NI</td>
<td>NI</td>
<td>NI</td>
<td>PI</td>
</tr>
<tr>
<td>5.4: Section and focus in viewport</td>
<td>2</td>
<td>NI</td>
<td>NI</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>5.5: Confirm form submission</td>
<td>2</td>
<td>PI</td>
<td>PI</td>
<td>PI</td>
<td>PI</td>
</tr>
<tr>
<td>6.1: Programmatic access to HTML/XML infoset</td>
<td>1</td>
<td>NR</td>
<td>NR</td>
<td>NR</td>
<td>NR</td>
</tr>
<tr>
<td>6.2: DOM access to HTML/XML content</td>
<td>1</td>
<td>NR (NI/PI)</td>
<td>NR (PI)</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>6.3: Programmatic access to non-HTML/XML content</td>
<td>1</td>
<td>NR</td>
<td>NR</td>
<td>NR</td>
<td>NR</td>
</tr>
<tr>
<td>6.4: Programmatic access to information about rendered content</td>
<td>1</td>
<td>NR</td>
<td>NR</td>
<td>NR</td>
<td>NR</td>
</tr>
<tr>
<td>Section</td>
<td>Status</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>--------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.5: Programmatic operation of user agent user interface</td>
<td>NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.6: Programmatic notification of changes</td>
<td>NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.7: Conventional keyboard APIs</td>
<td>NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.8: API character encodings</td>
<td>NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.9: DOM access to CSS style sheets</td>
<td>NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.10: Timely exchanges through APIs</td>
<td>NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.1: Respect Focus and selection conventions</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.2: Respect input configuration conventions</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.3: Respect operating environment conventions</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.4: Provide input configuration indications</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.1: Implement accessibility features</td>
<td>NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.2: Conform to specifications</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.1: Provide content focus</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.2: Provide user interface focus</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.3: Move content focus</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.4: Restore viewport state history</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.5: No event on focus change</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.6: Show event handlers</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.7: Move content focus in reverse</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.8: Provide text search</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.9: Allow structured navigation</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.10: Configure important elements</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.1: Associate table cells and headers</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.2: Highlight selection, content focus, enabled elements, visited links</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.3: Single highlight configuration</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.4: Provide outline view</td>
<td>PI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.5: Provide link information</td>
<td>PI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.6: Highlight current viewport</td>
<td>PI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.7: Indicate viewport position</td>
<td>PI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.1: Current user input configuration</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.2: Current author input configuration</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.3: Allow override of bindings</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.4: Single-key access</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.5: Default input configuration</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.6: User profiles</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.7: Tool bar configuration</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.1: Provide accessible documentation</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.2: Provide documentation of accessibility features</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.3: Provide documentation of default bindings</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.4: Provide documentation of changes between versions</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.5: Provide dedicated accessibility section</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.1 Capabilities Satisfied by Almost All User Agents

6.1.1 UAAG 1.0 Test Suites
Almost all user agents cleared the UAAG 1.0 Test Suite for HTML 4.01 at the following functions of the test file:

- Accesskey attribute. However, access keys where the key combinations were identical to the short cut key of the operating system or Internet Explorer could not be used.
- Button function in the form element. However, 95 Reader did not handle input button of the submit type.
- Tabindex. Except that 95 Reader did not use "tabindex".
- Activation of event handlers such as "onchange", "onfocus", and "onblur". Except that 95 Reader sometimes did not activate "onchange" event handlers.
- Alt attributes of "img" and input elements.
- "Title" attribute of "img" element. Except that PC-Talker and 95 Reader did not read "alt" attributes when both "alt" and "title" attributes were specified.
- "Caption" element of table. But "Caption" elements were read as normal text with no extra voice characters or voice attributes being specified.

The following functions were also satisfied by every user agents as a function of Internet Explorer:

- Display "noscript" element.
- Display alternate information for the "object" element.
- Activation of "link" element.
- Navigation of "link" elements, "textfield" elements and button controls within a form by the Tab key.
- Change of text size.
- Not display images and background images.
- Disabled scripts.
- Application of user style sheets.

6.1.2 PDF and Flash
Results for PDF were as follows:

- Sequential reading of headings and body from the top of the page to the end of the page. But Home Page Reader did not read headings.
- Every user agent sequentially read a table from the left-top cell to the right-bottom cell. But only JAWS could navigate within a table.

Results of Flash were as follows:

- Forward and backward navigation of button focus using the Tab key.
- Reading of button labels.
- Focus skip

Home Page Reader did not read Flash well in comparison to the other user agents. User agents other than Home Page Reader also had the following functions:

- Reading of displayed information.
- Reading of objects which were hidden from a display but were set to be read.

6.1.3 Japanese specific issues
No specific results were obtained for Japanese specific issues.

6.2 Capabilities Satisfied by None of the User Agents

6.2.1 UAAG 1.0 Test Suites
None of the user agents satisfied the following tests.

- Activation of "ondblclick" event when both "onclick" and "ondblclick" were specified.
- Toggle (stop) animated image.
- Control of multimedia objects embedded in "object" (or "emblld") elements.
- Toggle (stop) animated or blinking test.
- Toggle (stop) redirect and refresh. (JAWS could stop refresh in virtual view when configured.)
- No automatic focus change to a new window.
- Navigation among "th" elements.
- Navigation among "thead", "tbody", and "tfooter". Every user agents read "tfooter" before "tbody" because "tfooter" was written before "tbody" in the HTML.
- Only JAWS navigated among "list" elements.
- Only JAWS read tables using the "axis" attribute.
- Only JAWS read tables using "col" or "colgroup" attributes.
- Only JAWS had functions to list every heading elements ("h1", "h2", ...) and "caption" elements of the table in a separate window.
- Only Home Page Reader had functions to list all access keys used in the page.

The above lists do not include tests where results were ambiguous because of incomplete test files or ambiguous test procedures.

6.2.2 PDF and Flash
As for PDF, no user agents read headings differently from other text so as to make users distinguish which were headings and which was not. No user agents had navigation function among heading elements.

No Problems were found in Flash.

6.2.3 Japanese specific issues
Results of Japanese specific issues were as follows:

- Reading (pronunciation) of Japanese symbols were different among user agents. For example, Japanese (Zankaku) symbol "dash" was read as "bar", "dash", or "hyphen".
- "Lang" attribute did not change language of text-to-speech voice.
- No user agents could read ambiguous date, time, and money representations appropriately.

6.3 Capabilities Differing Among User Agents

6.3.1 UAAG 1.0 Test Suites
Test Results for the UAAG 1.0 Test Suites showed that JAWS and Home Page Reader had the following functions which were not supported by the other two Japanese user agents (PC-Talker and 95 Reader).

- Navigation of heading elements inside the page.
- Reading table with use of structure markups such as "summary" elements, "th" elements, and "scope" attribute.
- Link to anchors in the same page.
Text search in a page.
Reading "title" attributes of "abbr" and "acronym" elements.
Selection of arbitrary parts of text in a page using a keyboard.
Customization of reading functions.

6.3.2 PDF and Flash
For PDF,

- All user agents except 95 Reader inserted a line break between displayed lines, which resulted in unnatural reading of Japanese text. (e.g., one line was read as "All u/ser a//ents// exce//pt 95 ..")
- Only JAWS had dedicated functions for reading a table.
- PC-Talker did not read alternative information of a graph. It only read the caption of a graph.
- Home Page Reader did not read headings.
- 95 Reader and JAWS read alternative information of images. PC-Talker did not read this information. Home Page Reader read that information in some images.

With Flash, only Home Page Reader did not read well.

6.3.3 Japanese specific issues
Results of Japanese specific issues were as follows:
- JAWS and Home Page Reader read list numbers of ordered list elements.
- 95 Reader did not read Japanese (Zenkaku) symbols even if it was configured to read them.
- Home Page Reader read character entity references of &copy; (copyright) and &reg; (registered trademark) as "C Tyosakuen" ("Tyosakuen" means copyright) and "R Toroku Syohyo" (Toroku Syohyo means registered trademark).
- PC-Talker and JAWS read characters inside the "rp" elements of ruby.
- JAWS ignored whitespace characters inserted inside a word.
- JAWS read the word as one word even if a part of that word was modified by a "span" element.

7. DISCUSSION
The current survey examined capabilities of Japanese user agents in detail and found the following facts.

7.1 Conformance to UAAG 1.0
Each checkpoint of UAAG 1.0 is assigned to one of three priorities. Priority 1 is the basic requirement and is described as "If the user agent does not satisfy this checkpoint, one or more groups of users with disabilities will find it impossible to access the Web. Satisfying this checkpoint is a basic requirement for enabling some people to access the Web."[6]

As shown in Table 1, among 48 Priority 1 checkpoints, the following 20 checkpoints (CP) were met\(^4\) by all user agents:
- CP 1.1: Full keyboard access
- CP 1.3: Provide text messages
- CP 2.1: Render content according to specification
- CP 2.2: Provide text view
- CP 3.1: Toggle background images
- CP 3.4: Toggle scripts
- CP 4.1: Configure text scale
- CP 4.3: Configure text colors
- CP 4.7: Global volume control
- CP 4.9: Configure synthesized speech rate
- CP 4.10: Configure synthesized speech volume
- CP 4.11: Configure synthesized speech characteristics
- CP 4.14: Choose style sheets
- CP 7.1: Respect focus and selection conventions
- CP 7.2: Respect input configuration conventions
- CP 9.1: Provide content focus
- CP 9.2: Provide user interface focus
- CP 12.1: Provide accessible documentation
- CP 12.2: Provide documentation of accessibility features
- CP 12.3: Provide documentation of default bindings

Thus, all of the user agents had basic capabilities to ensure Web accessibility.

None of the user agents met\(^5\) the following 11 Priority 1 checkpoints:
- CP 2.4: Allow time-independent interaction
- CP 2.5: Make captions, transcripts, audio descriptions available
- CP 2.6: Respect synchronization cues
- CP 3.2: Toggle audio, video, animated images
- CP 3.3: Toggle animated or blinking text
- CP 3.5: Toggle automatic content retrieval
- CP 4.2: Configure font family
- CP 4.4: Slow multimedia
- CP 4.5: Start, stop, pause, and navigate multimedia
- CP 4.6: Do not obscure captions
- CP 4.8: Independent volume control
- CP 9.4: Restore viewport state history

The above list shows that current user agents lack functions relating to the control of multimedia and time-dependent interactions.

Of the preceding checkpoints, we recommend that the following functions should be implemented in Internet Explorer rather than user agents:
- Toggle (stop) animated image.
- Control of multimedia objects embed in "object" (or "embd") elements.
- Toggle (stop) animated or blinking text.
- Toggle (stop) redirect and refresh.
- No automatic focus change to the new window.

These functions are basic functions of a web browser and thus related to not only accessibility but also usability.

7.2 Capability Difference
As shown in Section 6.3.1, there was a clear difference between the capability of JAWS and Home Page Reader and that of PC-Talker and 95 Reader.

The following Priority 1 checkpoints of UAAG 1.0 were met by JAWS and Home Page Reader but not met by PC-Talker and 95 Reader:
- CP 2.3: Render conditional content
- CP 6.2: DOM access to HTML/XML content

\(^4\) Test result is C or AC.

\(^5\) Test result is PI or NI.
Evaluation of checkpoints was ambiguous procedures, etc. Evaluation of checkpoints was difficult if no test file was prepared for that checkpoint. We also found that the checkpoints of UAAG 1.0 were difficult to understand because of its abstract representation. We sometimes could not definitively decide if a user agent met a checkpoint.

6 These problems were reported to UAWG.
This ambiguity in the UAAG 1.0 Test Suite for HTML 4.01 prevents comparison of the results of Japanese and English user agents.

7.5 Accessibility Responsibility Between Content and User Agents

As shown in subsection 7.1, there are Japanese user agents which meet many of UAAG 1.0 checkpoints. These Japanese user agents did not meet checkpoints of multimedia and time-dependent interactions but did meet important checkpoints such as full key board access, activation of some event handlers, rendering conditional content, configuration of text size, content focus movement, navigation in data table, text search, and structured navigation.

Therefore, content authors should use markup to specify the structure and should require the use of user agents which make use of structure information. In other words, content should meet WCAG or JIS X 8341-3 requirements and user agents should try to meet UAAG requirements, resulting in a happy combination of accessible content and accessible user agents.

8. CONCLUDING REMARKS

Capability of four Japanese user agents for person with visual disabilities was examined with test files.

The results of the survey with UAAG 1.0 Test Suite for HTML 4.01 showed no user agents fulfilled all Priority 1 checkpoints of UAAG 1.0. All user agents, however, met 20 out of 48 Priority 1 checkpoints. All user agents met important checkpoints such as full key board access, activation of some event handlers, rendering conditional content, configuration of text size, and content focus movement. None of the user agents met 11 of Priority 1 checkpoints, which showed current user agents lacked functions relating to the control of multimedia objects and time-dependent interactions. We found there were two kinds of user agents in Japan. JAWS 6.2 and Home Page Reader 3.04 had functions of navigating heading elements inside the page and reading table with use of structure markups such as “summary” and “th” elements and “scope” attribute. They also had functions of linking to anchors in the same page, text search in a page, reading list numbers of ordered list elements, reading “title” attributes of “abbr” and “acronym” elements, selection of arbitrary part of text in a page using a keyboard. On the other hand, 95 Reader 6.0 and PC-Talker 3.04 did not have these functions.

Current Web standards and accessibility guidelines such as JIS X 8341-3 require content authors to use markup to specify content structure. Japanese Web content can count on former user agents because they make use of structure markups.

There was no UAAG 1.0 test file that could not be applied to Japanese user agents. There were, however, some test files and checkpoints in which objective evaluation was very difficult. Test results of Japanese specific test files showed no significant issues in those tests.

Every user agents barely read PDF, after it was made accessible. The results, however, show that PDF is not accessible as (X)HTML. What could be done by all user agents was sequentially reading paragraph text and table cells from the top of the page to the bottom of the page. No user agents used heading elements for accessibility or usability usage. As for Flash, every user agents except Home Page Reader 3.04 read accessible Flash well.

Web accessibility is improved not only with accessible content. User agents which have enough accessibility functions and authoring tools which help authors to write accessible content is also important. As described in the introduction, content and user agents is not independent. This dependency requires both content and user agents to conform to standards. W3C/WAI developed accessibility guidelines of content (WCAG) and user agent (UAAG). Japan has Web content accessibility guidelines as JIS X 8341-3, which is almost same as WCAG 2.0 working draft. The results of the current survey show that there are user agents which meet many requirements of UAAG 1.0. Therefore, we can say that content authors should conform to accessibility standards such as WCAG and JIS X 8341-3 and users use Web with user agents that have enough accessibility capabilities. These requirements will enhance Web accessibility.

Japanese users who are blind use PC-Talker and 95 Reader more often than JAWS, which costs much more than the other two screen readers. Home Page Reader can be used in addition to a screen reader. There, however, are many Japanese users who use only PC-Talker or 95 Reader when using Web. Thus, improvement of Japanese screen readers and education of users to use user agents which have enough capabilities are necessary to improve Web accessibility in Japan.

We did not carry out how user agents treat Java (Java applications and Java applets). Use of Java will be increasing because Java is used in electronic certification services of e-Government. Use of JavaScript and DHTML also will increase in Web 2.0. In addition to the current survey, survey of these techniques is needed.

All user agent under survey use Internet Explorer. In the United States, in addition to Internet Explorer, JAWS 7.0 and Window-Eyes 5.5 can use Firefox 1.5, which has built-in accessibility functions including DOM access [9]. Using Firefox and Internet Explorer enables users to choose an appropriate Web browser and enhance accessibility and usability of Web browsing.

As described in the last paragraph of the Introduction, harmonization of Web accessibility guidelines requires the knowledge of user agent capabilities. Discussion of international accessibility standards should pay attention to the different capabilities of user agents among countries.

The objective of the current research is not to compare user agents or to point out weakness of any user agents. Every result was obtained with the test files described in this paper and shows the results of that test file. It is possible that these results cannot generalize to other cases.

9. ACKNOWLEDGMENTS

The author thanks Makoto Ueki, Masaya Ando, Ai Omote, and Masako Muraoka for examining user agents with many test files. This work had not finished without their support.

7 Retail price of Japanese user agents are: $333 for PC-Talker, $305 for 95 Reader, $131 for Home Page Reader, and $1243 for JAWS. Popular screen reader is about $300, whereas JAWS costs 4 times of these screen readers.
This work was partially supported by the Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Scientific Research on Priority Areas, #16091210, 2004.

10. REFERENCES


Dialog Generation for Voice Browsing

Zan Sun  Amanda Stent  I.V. Ramakrishnan

Department of Computer Science
Stony Brook University
Stony Brook, NY 11794, USA
{zsun, stent, ram}@cs.sunysb.edu

ABSTRACT
In this paper we present our voice browser system, HearSay, which provides efficient access to the World Wide Web to people with visual disabilities. HearSay includes content-based segmentation of Web pages and a speech-driven interface to the resulting content. In our latest version of HearSay, we focus on general-purpose browsing. In this paper we describe HearSay's new dialog interface, which includes several different browsing strategies, gives the user control over the amount of information read out, and contains several different methods for summarizing information in part of a Web page. HearSay selects from its collection of presentation strategies at run time using classifiers trained on human-labeled data.

1. INTRODUCTION
The World Wide Web has become an indispensable aspect of our society, used for education, commerce, medicine and entertainment. However, the primary means of accessing the Web is via browsers designed for visual modes of interaction (e.g., Internet Explorer, Firefox, etc.). This limits access for an entire community of people with visual disabilities. This target population faces particular difficulties in accessing, scanning and summarizing/distilling information on a Web page or group of pages, filling out Web forms, and using Web search facilities.

Creating audio browsable Web content has become the focus of intensive research efforts by industrial enterprises (e.g., IBM) and standardization organizations (e.g., W3C). New markup languages, such as VoiceXML [9], SALT [8] and XHTML+Voice [11], and new voice browser systems, such as IBM’s WebSphere Voice Server, have emerged to facilitate the creation, publishing, and exchange of audio browsable Web content. However, adapting to voice browser technology still remains a significant burden for many Web content providers. Furthermore, while current screen readers and voice browsers are useful for reading HTML documents, they impose significant overhead on users. These systems provide almost no filtering of Web page content to eliminate “noise” (e.g., advertisements), and do not provide the user with a semantic view of the pages being browsed. As a result, the user is forced to arrow down or page down through a single columned presentation of all the links on a given page including the navigational links and ads.

In previous work, we developed a voice browser system, HearSay [29]. HearSay provided access to the content of news, commerce and educational Web pages in an efficient and simple way and had a uniquely flexible, controllable interface. However, because HearSay relied on ontologies and hand-built templates for content extraction and presentation, its scope was limited. In this paper we present our revised, general-purpose HearSay system. We focus on HearSay’s dialog interface, which includes several different browsing strategies, gives the user control over the amount of information read out, and contains several different methods for summarizing information in part of a Web page. HearSay selects from its collection of presentation strategies at run time using classifiers trained on human-labeled data.

The rest of this paper is organized as follows. In Section 2 we describe the HearSay system. In Section 3 we present HearSay’s general purpose browsing strategies. We describe HearSay’s content presentation strategies in Section 4. We discuss related work is described in Section 5 and finally, we conclude in Section 6.

2. INTRODUCTION TO HEARSAY
2.1 HearSay Architecture
The architecture of the HearSay voice browser is shown in Figure 1. It includes three basic components: the Browser Object Interface, the Content Analyzer, and the Interface Manager. The Browser Object Interface\(^1\) fetches pages from Web servers. Special features include automatic form fill-outs and retrieval of pages pointed to by navigable links that require execution of JavaScript.

The Content Analyzer partitions an input Web page into a logical structure of segments containing related content elements by analyzing the page’s structure and content. The output of the Content Analyzer is a partition tree of the content in the input page. The Interface Manager labels each partition in the partition tree using pre-trained classifiers, described in Section 4. These labels are used by the Dialog Generator that auto-

\(^1\)http://msdn.microsoft.com/library/default.asp?url=/workshop/browser/prog_browser#odintry.asp
automatically generates a VoiceXML dialog interface to the Web page content. We use our own VoiceXML interpreter, along with freely available text-to-speech synthesizers and speech recognizers, to execute this VoiceXML dialog.

2.2 Using HearSay

HearSay’s main output modality is speech. It accepts input in text or speech, so can be used both on small form-factor devices such as PDAs, and on desktops/laptops. Suppose Alice is a student who has visual disabilities. She often browses the Web using HearSay. Today she opens the web site of New York Times (http://www.nytimes.com) by saying “New York Times”. After loading the page (shown in Figure 2), HearSay automatically partitions it into segments of related content using structural and semantic information.

HearSay automatically creates a VoiceXML dialog interface to the partitioned content. For the Web page in Figure 2, HearSay might say, “There are two sections. Section one (size: 2 percent) keywords are New York Times, Personalize Your Weather, Updated Friday.... Navigate it?”. From the description Alice decides that section one is the header of the Web page and replies “No”. HearSay continues to the next section, saying “Section two (size: 98 percent) keywords are news, international, business, ... Navigate it?”. Alice chooses to browse this partition (labeled 2 in Figure 2), which itself contains 3 partitions. After further browsing, Alice may navigate to a headline story linked to in the partition labeled 3 in Figure 2, at which point HearSay will read out the story. Alice can also invoke common navigation commands such as “Go back”, “Skip” or “Repeat” at any time.

Unlike our first version of HearSay [29], which was based on using domain-specific ontologies and dialog templates, the current HearSay is designed for generality. Consequently, we have modified both our Content Analyzer and our Interface Manager. We briefly describe our Content Analyzer, then look in detail at our Interface Manager.

2.3 Content Analysis

Here we describe the content analysis algorithm that HearSay uses to partition a Web page into semantically related segments. It is based on our previous work on structural and semantic analysis of Web content [24, 29, 25, 26]. Content analysis (see [24] for details) is based upon the observation that semantically related items in content-rich Web pages exhibit consistency in presentation style and spatial locality. Exploiting this observation, a pattern mining algorithm working bottom-up on the DOM tree of a Web page aggregates related content in subtrees. Briefly, the algorithm initially assigns types, reflecting similarities in structural presentation, to leaf nodes in the DOM tree and subsequently restructures the tree bottom-up using pattern mining on type sequences. The DOM tree fragment for the page in Figure 2(a) is shown in Figure 3(a). The type of a leaf node is the concatenation of HTML tags on the root-to-leaf path and that of an internal node (or partition) is composed from the types of its child nodes. In the restructured tree, known also as the partition tree, there are three classes of partition: (i) group - which encapsulates repeating patterns in its immediate children type sequence, (ii) pattern - which captures each individual occurrence of the repeat, or (iii) block - when it is neither group nor pattern. Intuitively the subtree of a group node denotes homogenous content consisting of semantically related items. For example, observe how all the headline news in the central part in Figure 2(a) are rooted under the group node in the partition tree. The leaf nodes of the partition tree correspond to the leaf nodes in the original DOM tree and have content associated with them. The partition tree resulting from structural analysis of the DOM in Figure 3(a) is shown Figure 3(b). The partition tree represents a logical organization of the page’s content.

3. HEARSAY’S BROWSING STRATEGIES

HearSay’s dialog creation component takes as input a partition tree constructed from Web page content. It walks over this tree, constructing a menu-based dialog for browsing the content using speech. In previous versions of HearSay, this dialog was constructed using a set of domain-specific VoiceXML templates. Our focus in this version of HearSay is generality; the system should provide a reasonably efficient interface to any Web page. This involves a) permitting efficient navigation; and b) presenting content efficiently. Because the data structure the system operates over is a partition tree, HearSay’s general-purpose navigation strate-
3.1 Breadth-First and Depth-First Navigation

In BFN, all the child partitions in a partition are presented to the user, who then selects one for further browsing. This strategy is straightforward and gives users an overview of the available selections from which they can choose. However, if a partition has many children, it can be hard for a user to listen to and remember all the browsing choices. Consider the category news section of the New York Times shown in Figure 4(a). The partition tree of this particular section is shown in Figure 4(b). There are in fact 20 child partitions of this partition, too many for the user to remember [23].

DFN is used in cases like these. In DFN, each child of a partition is presented individually, with the user given a yes/no choice about whether to navigate into that partition right after it is presented. An alternative to DFN would be to use BFN with barge-in, so that a user could interrupt the system with "navigate" right after hearing about a partition of interest. However, with speech input the use of barge-in leads to more speech recognition errors. In addition, with DFN the user never has to listen to the children of a partition more than once (because the system resumes presenting children at the location where the user last made a choice), whereas with BFN+barge-in, the user would have to listen to the whole list of children of a partition at each return to the root of the partition.

Table 1 shows the BFN and DFN dialogs output from HearSay for the category news presented above.

HearSay2 uses a simple method to choose between BFN and DFN based on the number of children a partition has. However, the user can also switch between BFN and DFN at any time to fit personal preference.

3.2 Verbosity

When first browsing a Web page, the user may need lots of information to make navigation choices. However, for expert users this information may be unnecessary and annoying. HearSay now lets users adjust the amount of information it provides about each partition:

- Level 1 – non-verbose mode: provide just the depth and type (group, pattern or block) of the partition.
- Level 2 – partly-verbose mode: provide all the information from level 1 plus information about the partition's structure.
- Level 3 – full-verbose mode: provide all the information from level 1 plus information about the partition's structure AND content.

The user may change the "verbosity level" at any time. Once the level is set, HearSay will use it until it is changed again. A sample dialogue involving changes to the verbosity level is shown in Figure 5.

4. HEARSAY’S PRESENTATION STRATEGIES

In our current version of HearSay, users can set their own browsing strategy (BFN or DFN) and choose the amount of information about the partition tree they want to hear. However, HearSay should also be able to modify its presentation strategy by itself according to the changing requirements imposed by the content it is presenting. In the
sections below, we identify three aspects of content presentation in HearSay. We then describe how we used human-annotated data to automatically train classifiers to determine HearSay’s content presentation strategy at run time.

4.1 Navigation: Searching vs. Browsing

It is well-known that user activities over Web pages during navigation consist of two basic types: searching and browsing. Previously, researchers have looked at how users switch between strategies across sequences of Web pages [12, 33].

Here, we apply these ideas to navigation across partitions (possibly within a single Web page).

In the New York Times homepage shown in Figure 6(a), there are basically two big partitions (labeled 1 and 2). Partition 1 is the header of the page, while partition 2 contains the main content. Partition 2 is further divided into three partitions: a menu on the left-hand side, a set of headline news items in the middle, and a set of other news stories and related content on the right-hand side. A visitor to this page looking for news is probably not interested in listening to partition 1 or the menu in partition 2. Instead, the user will search to partition 3, the headline news items. At this point, her activity will turn from searching to browsing, i.e. listening to the news stories.

As illustrated in this example, each partition in one of HearSay’s partition trees can be classified as either a searching or a browsing partition. For browsing partitions, HearSay can simply read out the partition’s contents. We created a VoiceXML dialog template to perform this task. However, for searching partitions, HearSay needs to provide a summary of information about the partition so the user can decide whether to search inside it. The type of summary depends on the structure and content of the partition itself.

4.2 Partition Summaries: Structure vs. Content

We distinguish between two basic types of partition summary for searching partitions: structural, and content-based.

In a structural summary, HearSay describes the structure of the partition: its location in the partition tree, its size, etc. In a content-based summary, HearSay presents key words in the content of the partition or gives a short extractive summary based on the text in the partition. For partitions containing heterogeneous content, structural summaries are more informative. For partitions containing semantically related items, content-based summaries are more useful.

We designed a VoiceXML dialog template for each type of summary. However, when constructing a content-based summary for a partition, HearSay must decide which text to include.

4.3 Partition Summaries: Content Selection

Common text summarization techniques use term frequencies in documents to identify important words/phrases/sentences [22]. These documents are usually fairly large, e.g. a news report, or an academic paper, etc. However, our partitions are generally small, so extractive summarization methods work poorly. Our summarization method is based on the observation that, in Web pages, visual hints are commonly used to emphasize important elements (e.g., the titles of news articles are usually in big fonts). Each sentence in a searching/content-based partition is labeled as either important or unimportant. The important sentences/phrases are used in the text summary for that partition.

4.4 Evaluation

We used machine learning methods to train classifiers for each of these three binary classification tasks:

- Classify each partition as browsing or searching
- Classify each searching partition as best-suited for a structural or content-based summary
- Classify each sentence/phrase in a searching/content-based partition as either important or unimportant

4.4.1 Partition Types

We collected about 50 partition trees from different Web pages (mainly in the news domain) and manually labeled a subset of partitions selected at random from each partition tree (the trees are very large, so we could not label all the partitions). Each partition was labeled as browsing, searching/structural or searching/content-based. Separately, we automatically extracted 44 features (shown in Table 2) for each partition in each partition tree. These features represent information about the structure of the partition tree and the content in the original Web page.

We built two support vector machines [14] using these features. One classifies partitions as searching or browsing; the second classifies searching partitions as structural or content-based. We used libSVM [13] to train our classifiers. We used a sequence of binary classifiers rather than a single one-versus-all SVM because: (a) binary classifiers usually have better performance than multi-class classifiers; and more importantly, (b) the two problems do not necessarily follow the same feature-space mapping and distribution.

We used five-fold cross-validation to evaluate the performance of our classifiers for this task. We experimented with different kernels; our best results, obtained using the Radial

<table>
<thead>
<tr>
<th>BFN</th>
<th>DFN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HEARSAY2:</strong> You are at level 2. There are 20 sections in this level. Section 1, Business; Section 2, National; Section 3, Washington; Section 4, Health ...(may takes a while to finish all the 20 sections) please choose from the above 20 sections.</td>
<td><strong>HEARSAY2:</strong> You are at level 2. There are 20 sections in this level. Section 1, Business. Navigate it?</td>
</tr>
<tr>
<td>USER: If she still remembers the second item)</td>
<td>USER: No.</td>
</tr>
<tr>
<td><strong>HEARSAY2:</strong> Section 2, National. Navigate it?</td>
<td><strong>HEARSAY2:</strong> Section 2, National. Navigate it?</td>
</tr>
<tr>
<td>USER: Yes.</td>
<td>USER: Yes.</td>
</tr>
<tr>
<td><strong>HEARSAY2:</strong> National. (continue with the national news)</td>
<td><strong>HEARSAY2:</strong> National. (continue with the national news)</td>
</tr>
</tbody>
</table>

**Table 1:** Breadth-first navigation vs. depth-first navigation
Basis Function kernel, are 86% correct for navigation type (browsing or searching) and 88% correct for summary type (content-based or structural).

In HearSay’s Dialog Generator, a VoiceXML dialog template is applied to each partition based on its classification and the user-selected verbosity level. When the user selects verbosity level 2, the content-based or structural template is used for searching partitions. Both structural and content-based information are presented for searching partitions at verbosity level 3; no description is provided for searching partitions at verbosity level 1. The browsing template is used for browsing partitions at all verbosity levels.

4.4.2 Partition Summaries

We manually labeled 700 leaf nodes from our 50 partition trees. Each node was labeled as important or unimportant. Separately, we automatically extracted the 13 features given in Table 3 for each node. These features include information about the position of the leaf node as well as the formatting information.

We trained decision tree classifiers for this task using Weka [32] implementations of the decision tree algorithms (i.e., J48, ADTree[16], NBTree[19] and LMT[20]). As a baseline, we took the first sentence/phrase in a partition as the summary of that partition. For testing, we used 10 partition trees from 10 new news websites. Our results are shown in Table 4. The best decision tree (i.e., J48) works significantly better than the baseline for partitions where there are clear visual hints. However, we found that performance on this task is highly dependent on the performance of the underlying partitioning algorithm. For example, sometimes boundaries of par-

---

**Table 2: Features for SVM.**

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>NODE</td>
<td>The type of the partition.</td>
</tr>
<tr>
<td>6-11</td>
<td>PAR</td>
<td>The type of the parent.</td>
</tr>
<tr>
<td>12-17</td>
<td>PREV</td>
<td>The type of the left sibling.</td>
</tr>
<tr>
<td>18-22</td>
<td>NEXT</td>
<td>The type of the right sibling.</td>
</tr>
<tr>
<td>23</td>
<td>NUM_W</td>
<td>Number of words.</td>
</tr>
<tr>
<td>24</td>
<td>NUM_LK</td>
<td>Number of links.</td>
</tr>
<tr>
<td>25</td>
<td>NUM_L</td>
<td>Number of leaf nodes.</td>
</tr>
<tr>
<td>26</td>
<td>NUM_LS</td>
<td>Number of left siblings.</td>
</tr>
<tr>
<td>27</td>
<td>NUM_RS</td>
<td>Number of right siblings.</td>
</tr>
<tr>
<td>28</td>
<td>LEVEL</td>
<td>Number of levels to the root.</td>
</tr>
<tr>
<td>29</td>
<td>LONG_LV</td>
<td>Number of levels to the deepest leaf node.</td>
</tr>
<tr>
<td>30</td>
<td>SHORT_LV</td>
<td>Number of levels to the nearest leaf node.</td>
</tr>
<tr>
<td>31-36</td>
<td>NUM_C</td>
<td>Number of children with the same type.</td>
</tr>
<tr>
<td>37</td>
<td>TOTAL_W</td>
<td>Total number of words in the tree.</td>
</tr>
<tr>
<td>38</td>
<td>PAR_W</td>
<td>Total number of words in the parent.</td>
</tr>
<tr>
<td>39</td>
<td>TOTAL_L</td>
<td>Total number of leaf nodes in the tree.</td>
</tr>
<tr>
<td>40</td>
<td>PAR_L</td>
<td>Total number of leaf nodes in the parent.</td>
</tr>
<tr>
<td>41</td>
<td>MAX_LV</td>
<td>Level of the deepest branch in the tree.</td>
</tr>
<tr>
<td>42</td>
<td>PAR_MAX</td>
<td>Level of the deepest branch in the parent.</td>
</tr>
<tr>
<td>43</td>
<td>MIN_LV</td>
<td>Level of the most shallow branch in the tree.</td>
</tr>
<tr>
<td>44</td>
<td>PAR_MIN</td>
<td>Level of the most shallow branch in the parent.</td>
</tr>
</tbody>
</table>
### Table 3: Features used in labeling problem.

<table>
<thead>
<tr>
<th>Feature</th>
<th>CNN</th>
<th>GoogleNews</th>
<th>LATimes</th>
<th>MSNBC</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of words in current node</td>
<td>84%</td>
<td>78%</td>
<td>85%</td>
<td>83%</td>
<td>80.8%</td>
</tr>
<tr>
<td>No. of words in the parent node</td>
<td>69%</td>
<td>73%</td>
<td>88%</td>
<td>83%</td>
<td>77.7%</td>
</tr>
<tr>
<td>Number of previous and next siblings</td>
<td>97%</td>
<td>97%</td>
<td>97%</td>
<td>97%</td>
<td>97%</td>
</tr>
<tr>
<td>Max number of levels to the root</td>
<td>97%</td>
<td>97%</td>
<td>97%</td>
<td>97%</td>
<td>97%</td>
</tr>
</tbody>
</table>

### Table 4: Primary evaluation for labeling problem.

<table>
<thead>
<tr>
<th>Web site</th>
<th>Baseline</th>
<th>J48</th>
<th>ADTree</th>
<th>NBTree</th>
<th>LMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NY Times</td>
<td>84%</td>
<td>100%</td>
<td>97%</td>
<td>92%</td>
<td>81%</td>
</tr>
<tr>
<td>CNN</td>
<td>71%</td>
<td>80%</td>
<td>74%</td>
<td>50%</td>
<td>39%</td>
</tr>
<tr>
<td>GoogleNews</td>
<td>78%</td>
<td>100%</td>
<td>100%</td>
<td>97%</td>
<td>97%</td>
</tr>
<tr>
<td>LATimes</td>
<td>85%</td>
<td>88%</td>
<td>88%</td>
<td>88%</td>
<td>78%</td>
</tr>
<tr>
<td>MSNBC</td>
<td>83%</td>
<td>81%</td>
<td>81%</td>
<td>81%</td>
<td>71%</td>
</tr>
<tr>
<td>Average</td>
<td>79.5%</td>
<td>92.3%</td>
<td>92.4%</td>
<td>92.4%</td>
<td>81.4%</td>
</tr>
</tbody>
</table>

5. RELATED WORK

The issue of promoting Web accessibility for persons with visual disabilities has become increasingly prominent. As early as 1997, W3C launched the Web Accessibility Initiative (WAI) [10] to promote the development of browser standards and guidelines (e.g., HTML authoring guidelines and user agent guidelines) to make the Web more accessible to individuals with visual disabilities. Similar initiatives have been developed by industry: examples include Microsoft’s accessibility initiative [1], IBM’s Special Needs Systems program [4], and Sun Microsystems’ Java accessibility API [5].

Several studies have highlighted the ineffectiveness of existing screen readers for Web browsing tasks [15, 17]. As a result, several specialized Web voice browsers have been developed to adapt HTML-based contents. For example, the JAWS system [6] and IBM’s Home Page Reader [3] allow navigation via hyperlinks. The BrookesTalk system [2] uses NLP-based text abstracting and summarization techniques to facilitate voice browsing of the Web. The voice feature in the Opera browser [7] provides a set of voice-based commands.

A key difference between HearSay and these systems is that HearSay performs extensive analysis of the content of HTML documents, while other systems do minimal processing of Web page content. Content-based analysis enables segmentation of Web page content into related blocks and facilitates efficient speech-driven browsing.

As accessibility becomes a more mature research area, the design of specialized voice-driven interfaces for is receiving more attention (e.g., [31, 18]). For example, in [30] the authors examined the accessibility issues relating to the Web and proposed solutions in the context of a screen reader system. More guidelines relating to improving the accessibility of search engines were proposed in [21]. However, in most research relating to accessibility the proposed solution is for content authors to add additional tags to Web page content, or for the engineers of the browser to provide specialized ontologies and rules to facilitate content presentation [29, 27]. By contrast, we take the markup of content on the Web as is, and use automatic analysis to make the content accessible. Our approach is to provide efficient access to as much of the Web as possible; we are willing to sacrifice some elegance for coverage.

6. CONCLUSION AND FUTURE WORK

We have described our new HearSay Web browser for people with visual disabilities. HearSay is designed for efficient, broad-coverage voice-driven Web browsing. In this paper, we focused on the general-purpose browsing and content presentation strategies employed in HearSay. In future work, we plan to conduct a complete evaluation of HearSay and refine our browsing and presentation strategies accordingly.

7. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under grant number IIS-0534419 and by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCCHD030010. The authors would like to thank the anonymous reviewers for their very helpful feedback.

8. REFERENCES

http://www.microsoft.com/enable/.
http://cms.brookes.ac.uk/computing/speech/.
http://www.saltforum.org/.
http://www.w3.org/TR/voicexml20/.
http://www.w3.org/WAI.
http://www.voicexml.org/specs/multimodal/x+v/12/.
(Browsing strategy: BFN. Verbosity level 2.)

(Partition type: Searching. Structural summary.)

HearSay: Level 2, 3 sections.
Section 1 (size: 13 percent) has 169 words and 51 links. Frequent words are News, Opinion, Features, Services, Member.
Section 2 (size: 63 percent) has 819 words and 133 links. Frequent words are International, National, Washington, Business, Editor, Complete Coverage.
Section 3 (size: 24 percent) has 312 words and 39 links. Frequent words are Market, Complete News, Forum, World, Reviews.
Please choose from the three sections.

User: Section 2.

(Browsing strategy: DFN.)

HearSay2: Level 3, 6 sections.
Section 1 (size: 18 percent) has 147 words and 8 links. Frequent words are court, prosecutors, sunni, Iraq, centers. Navigate it?

User: Yes.

(Browsing strategy: BFN.)

(Partition type: searching. Content-based summary.)

HearSay2: Level 4, 4 sections.
Section 1 is about Plant Operators Must Install Pollution Controls, Court Rules.
Section 2 is about Judge Gives Prosecutors New Chance in Terror Case.
Section 3 is about Sunnis Denounce Plan by U.S. and Iran to Hold Talks on Iraq.
Section 4 is about New Yorkers Will Pay to Sweat at 6 Previously Free Centers.
Please choose from the four sections.

User: Section 3.

(Partition type: browsing.)

HearSay2: Level 5. Link: Sunnis Denounce Plan by U.S. and Iran to Hold Talks on Iraq. By KIRK SEMPLE, 7:11 PM ET. Sunni leaders today said the talks would amount to “unjustified interference” by foreign nations in Iraq’s domestic affairs. Link: Kurds Destroy Shrine, Navigate it?

User: No.

HearSay2: (continue) Link: Photos, Navigate it?

User: No.

HearSay2: (continue) Link: Complete Coverage: The Reach of War, Navigate it?

User: No.

HearSay2: (continue) You’ve finished browsing this section. Now going back to upper level.
Level 4, 4 sections.
Section 1 is about Plant Operators Must Install Pollution Controls, Court Rules.
(repeating the description of block 3)

Table 5: Sample HearSay dialog based on the Web page in Figure6(a)


GraSSML: Accessible Smart Schematic Diagrams for All

Z. BEN FREDJ
Oxford Brookes University
Department of Computing
Wheatley Campus, OX33 1HX
+44 (0)1865 485603
zbenfredj@brookes.ac.uk

D.A. DUCE
Oxford Brookes University
Department of Computing
Wheatley Campus, OX33 1HX
+44 (0)1865 484528
daduce@brookes.ac.uk

ABSTRACT
Graphical representations are a powerful way of conveying information. Their use has made life much easier for most sighted users, but people with disabilities or users who work in environments where visual representations are inappropriate cannot access information contained in graphics, unless alternative descriptions are included.

We describe an approach called Graphical Structure Semantic Markup Languages (GraSSML) which aims at defining high-level diagram description languages which capture the structure and the semantics of a diagram and enable the generation of accessible and “smart” presentations in different modalities such as speech, text, graphic, etc. The structure and the semantics of the diagram are made available at the creation stage. This offers new possibilities for allowing Web Graphics to become “smart”.

Categories and Subject Descriptors

General Terms
Design, Experimentation, Human Factors, Languages, Theory, Legal Aspects.

Keywords
Accessibility, Smart Diagrams, Structure, Semantics, XML, SVG, Semantic Web, RDF, OWL, SPARQL.

1. INTRODUCTION
Diagrams are a fundamental component in the exposition of scientific research and are unavoidable in professional life. Despite their importance, not much has been done to provide accessibility support for information of this kind. To allow full access to the web it is also important that people with disabilities can create accessible web content containing accessible web graphics. It is important to understand that accessibility is not only for people with obvious disabilities but also for people who simply access information and learn in different ways.

The emergence of SVG has changed the way 2D graphics are created on the web. SVG offers many advantages and has introduced accessibility features that raster formats do not offer [1]. When using SVG, information about the diagram is available to the browser in terms of the objects it is composed of. Nevertheless, a number of limitations remain as it only captures diagrams at a low level of abstraction. It is more a “final form” presentation, which involves some drawbacks in the direct creation, modification and access to complex, highly structured, diagrams.

The next section describes the problem in more detail and presents the limitations of SVG and of some of the approaches previously taken to resolve the problem of graphic accessibility. Then we present our approach to the problem which generates presentations from high level descriptions. The following section discusses the current state of the GraSSML development and system architecture. The final section contains conclusions and thoughts on future work.

2. PROBLEM AND RELATED WORK
Many workers have explored different methods to make graphics accessible to blind or visually impaired people by representing graphics through an auditory interface, tactile drawings, text description, etc. Although these approaches partially address the accessibility problem of graphics, they present some limitations.

The approaches taken by the “Blind Information System” (BIS) [2] and “Graphical User Interfaces for Blind People” (GUIB) [3] projects depend on human intervention by a moderator, not necessarily the author of the graphic. In both cases the resulting description of the picture depends on the analysis and indexing of a third party. It is an important responsibility for the moderator who decides what information to convey and thus indirectly imposes a view when the picture is being read (e.g. inadvertently omitting important information).

The TeDUB project [4] (Technical Drawings Understanding for the Blind) explored the possibility of a semi-automatic analysis of diagrams. Their approach for the presentation and navigation of graphical information offers many advantages for blind users. But the (semi-) automatic analysis of the diagram information might produce wrong results and might need active human intervention and time.

The W3C Recommendation for 2D graphics is SVG. For the remainder of this paper we focus on approaches that use SVG as the presentation format. There are profiles of SVG adapted to the needs of mobile devices and technologies. SVG presents many advantages and provides many accessibility benefits [1]. However, some important issues still remain to be addressed.

SVG does not capture diagrams at a high level of abstraction. It is a “final form” presentation, which involves some drawbacks in the direct creation of complex, highly structured, diagrams. It is difficult at this level to handle the resizing and positioning of different shapes in complex diagrams. A simple modification such
as changing the alignment from vertical to horizontal can be awkward. SVG does not allow flexible readjustment of layout in response to viewer requirements and the viewing environment, such as different screen formats (PDA, mobile phones etc.).

The issue is that the intentions of the author are not totally captured. The structure of the SVG may reflect the sequence of operations used to create the diagram, rather than the intrinsic object structure within the diagram itself. This is likely to be the case for diagrams created with a general-purpose drawing tool.

Although SVG stores structural information about graphical shapes as an integral part of the image and allows metadata to be attached to primitives, there is little real scope for generating alternative presentations from the description at this level. An additional issue is that an SVG document contains the semantics of the diagram only implicitly. The “alternative equivalents”, which allow the author to include a text description for each logical component and a text title to explain the component's role in the diagram, could become tedious for the creation of complex diagrams. If the metadata added by the author are not accurate enough, the semantics of the diagram could differ from the description obtained from the metadata.

Some research groups have explored the accessibility features of SVG and have attempted to address some of the SVG limitations. The “Science Access Project” research group has successfully explored many accessibility features of SVG in their ViewPlus project [5] but it also has identified some of its limitations [6]: some SVG documents become less accessible when created without <title> and <desc> elements and some are very badly structured and therefore less informative. The ViewPlus project has a good approach by exploring the information behind the picture but the solutions proposed to overcome SVG limitations need too much effort in adding information and/or reorganizing it. It can be tedious and time consuming. It illustrates the fact that SVG is too low level and not informative enough regarding the structure of the graphical information.

An extension to SVG, called Constraint Scalable Vector Graphics (CSVG) [7], has been proposed. It partially addresses some of the SVG limitations by proposing additional capabilities which allow alternate layouts for the same logical group of components in a diagram. But whilst permitting a more flexible description of figures, CSVG remains very close to SVG and still captures diagrams at a similar low level of abstraction.

The SVG linearizer tool [8] generates a textual linear representation of the content of an SVG file by using a metadata vocabulary describing it. The author has to describe the SVG content using this RDF vocabulary and to add textual descriptions to all elements that constitute primary RDF resources. Then, from this information plus information contained in the SVG file itself, an HTML file is generated. This operation can be tedious and not very efficient in very complex diagrams and adding the RDF annotations is an onerous task for the author. This method is too dependent on the creator’s patience and willingness to produce appropriate metadata.

These previously presented approaches that we categorize as “bottom-up approaches” have been looking at the problem upside down. They start with the graphical representation of the diagram. The diagram is analyzed and interpreted by a predefined system and/or a moderator (who is not usually the creator of the diagram). The latter is required to add “metadata” to help understand the information. Some of these approaches involve writing difficult programs in order to “discover” the structure or semantics of the graphics.

In 1997, John A. Gardener [9] gave an overview of the concepts of “smart graphics” (information behind a picture) and intelligent graphics browsers for accessing such information. He highlighted the fact that “Nearly every part of smart graphics technology exists today, but to our knowledge there is no complete package that incorporates everything necessary to author a smart picture, incorporate it into an electronic document, and display it intelligently”. Our system GraSSML aims, by means of its family of languages, to be such a package by providing access to this most valuable “information behind the diagram”. Our hypothesis in the GraSSML approach is that many of the limitations involved in current approaches can be overcome if the information on the structure and semantics of the graphics were made “part of the graphics”, i.e. take a top-down approach.

3. THE GraSSML APPROACH

Our approach aims at facilitating the creation, modification, access and adaptation of diagrams as well as making the information “behind” the diagram available at the creation stage. The availability of this information is then explored to generate alternative representations improving accessibility of diagrams.

The main idea behind our approach is to reduce a task to a sequence of transformations between inputs and outputs expressed in different “Little Languages” [10]: “The GraSSML family of Languages” (Figure 1).

3. THE GraSSML APPROACH

An important aspect of this project is concerned with the syntax and semantics of diagrams. In natural language, words can be mapped into a set of meanings whereas in a visual language, geometric objects do not have a unique semantic interpretation. There are no universal visual conventions and each person interprets graphical information using his own mental schema and/ or imagination. For a computer life is not so easy. A computer needs a set of formal representational conventions to carry out this interpretation. Each domain has its own notation, syntax and semantic rules. A set of syntactic and semantic rules needs to be defined. As a starting point, we should analyze a wide range of diagrams in a specific domain in order to develop a diagrammatic semantic grammar. Following this specific predefined grammar, the user should be able to define the syntax and semantics of his diagram. Technologies emerging from the Semantic Web Activity will be a base to define such a grammar (RDF / RDFS / OWL).
There are many classification schemes for diagrams. In this work we have used the structural taxonomy proposed by Lohse [12] and have chosen the class of structural diagrams (which includes process and hierarchical diagrams) to study in this project.

4. THE GraSSML SYSTEM

4.1 A Simple Example

```
<ZineML type="Hierarchy">
 <box>Director John</box>
 <box>IT Manager Peter</box>
 <box>IT Developer Sarah</box>
 <box>IT Developer Linda</box>
 <box>Finance Manager Sue</box>
</ZineML>
```

Sets of rules S to P

Semantic Level

```
<ZineML type="Hierarchy">
 <box>Director John</box>
 <box>IT Manager Peter</box>
 <box>IT Developer Sarah</box>
 <box>IT Developer Linda</box>
 <box>Finance Manager Sue</box>
</ZineML>
```

Sets of rules S to S

Structure Level

```
<ZineML type="Hierarchy">
 <box>Director John</box>
 <box>IT Manager Peter</box>
 <box>IT Developer Sarah</box>
 <box>IT Developer Linda</box>
 <box>Finance Manager Sue</box>
</ZineML>
```

Sets of rules S to P

Textual Representation

```
Structure Level:
The box labelled “Director John” is connected to the box labelled “IT Manager Peter” and the box labelled “Finance Manager Sue”.

Semantic Level:
The director named John is at the top of the organisational chart. He manages the IT Manager named Peter and the Finance Manager named Sue.
```

Presentation Level

```
Director John
IT Manager Peter
Finance Manager Sue
IT Developer Sarah
IT Developer Linda
```

Figure 2: The Organizational Chart example

4.2 The GraSSML Family of Languages

4.2.1 Structure Level: ZineML

“ZineML” aims to be at a higher level than SVG by representing the structure of the diagram. It facilitates the creation and modification of diagrams. ZineML documents aspire to be readable by humans and to give good overviews of diagram structures. The language seeks to be rich enough to allow accessible alternatives of the structural representation of a diagram.

ZineML postulates a set of basic shapes selected to cover a wide range of possibilities for structured diagrams common in different domains (e.g. business, computing…). At this level of abstraction the language designed “ZineML” is not domain dependent, but proposes some options to express and apply rule sets when creating the diagrams. These rule sets could be used to tailor the diagram specifically to a domain.

ZineML offers the possibility to determine and to adjust positions and sizes semi-automatically, with a minimum of effort from the author. The derivation of graphical and other representations from ZineML is done in accordance with predefined rule sets (Figure 2) called “Sets of rules S to P” (Structure to Presentation).

4.2.2 Semantic Level: MyLanguage

“MyLanguage” is the XML language (e.g. XML Schema) used to capture the semantic intent behind the diagram. It does not aim to be universal at this level but domain-dependent, hence the name! It has to be applied to a specific domain where clear conventions are followed when creating diagrams (e.g. Organization charts, UML).

MyLanguage does not make any commitment to graphical presentation, but aims to capture the concepts and relationships between concepts (ontology) that are to be expressed in pictorial or other representational form. At this level, “Who knows better than the authors?” We should bear in mind that “A diagram is worth a thousand words”. Indeed, the semantics of the diagram can be complex, and without the help of the author the interpretation generated may be both verbose and shallow in the sense that the author is the one who knows “exactly” what the main message behind the diagram is and therefore can give concise information.

The information concerning the different notations used, the possible relationships between them and the meaning behind each of them, is the information which is required in order to create such a language. Hence, it is necessary to first study the domain and the class of the diagrams with the aim of identifying what concepts and properties these diagrams seek to represent in the domain. As a result of this study a particular “MyLanguage” (e.g. Org_hierarchy) is created. This language aspires to be an intuitive domain specific language; it employs the notations and concepts familiar to practitioners of the domain and by doing so it makes the semantics explicit to the user. Consequently a domain expert can easily understand the information. Even if all the information needed has been gathered, an important issue remains unsolved: the semantics is only implicit for the computer. Indeed, XML covers only the syntactic level and does not provide any means of talking about the semantics of data. There is a need to make the semantics explicit to the computer.

There is an obvious link to activity in the Semantic Web area concerned with the expression of subject ontologies and relationships expressed over terms defined in ontologies (RDF / RDFS / OWL). We are currently working on creating and using ontologies to make the semantics of the diagrams explicit and to underpin “MyLanguage”.

Along with the specification of a particular MyLanguage, notational conventions are created. The notational conventions govern the diagrammatic representation of the elements of MyLanguage, and are captured in sets of rules governing the generation of ZineML from MyLanguage (Figure 2: Sets of rules “S to S”). Example: Director (an element in Org_hierarchy) will be represented by a box filled in yellow and a centered label.
4.2.3 Presentation Level: SVG, XHTML...

The availability of the information “behind the graphic” allows us to generate alternative representations. In order to explore this information in the best possible way and to provide representations as accessible as possible we refer to studies on what kind of information is needed and how to describe graphics textually or verbally [e.g. (13) (14)].

• Graphical representation (Figure 2): SVG is used as the graphical output renderer at this level. For complex diagrams we are aiming at allowing an interactive exploration of the diagram by using some of SVG’s facilities and the available information concerning the structure and semantics of the diagram (the user can hide some details).

• Textual representation (Figure 2): a Verbalization Model has been implemented. It allows the generation of a textual representation of the structure and semantics of the diagram.

• Query System: The structure and the semantics of the diagram being available, it becomes possible to express queries concerning specific parts of a diagram in novel ways: “smart diagrams” become a reality. At a later stage we aim at developing the query system based on the explicit structure and semantics of the diagram. Example (Figure 2): What is the total number of employees reporting to Peter? Who is reporting to Peter? Show me the levels that directly report to Peter?

4.3 Implementation

Implementation of ZineML for two classes of diagrams: process and hierarchical diagrams is in hand. The Java Programming Language has been used to implement the presentation algorithms allowing at generating the graphical representation of the diagram (SVG). The graphical representations of ZineML and MyLanguage are done by applying XSLT transformations to each document by respecting rule sets. The verbalization model allowing the generation of the structure and semantic of the diagram based on the syntax of the corresponding XML language used (ZineML or MyLanguage) has been implemented using an XSLT Transformation on ZineML and MyLanguage. The output is an XHTML document containing the textual representation of the structure and the semantic of the diagram. Other functionalities of GraSSML are at the design stage ready for implementation. Once the implementation of the first prototype is completed, experiments will be carried out in order to evaluate the usability and accessibility of the system.

5. CONCLUSION AND FUTURE WORK

Graphics on the Web have significantly improved but a number of issues still remain unsolved. This paper has outlined these issues and has presented some related work aiming at resolving them. These only address some of the problems and they emphasize the need to make the graphical information (information on the structure and the semantics of the graphics) part of the graphic.

We have looked at the problem of creation, modification, access and exploration of Web graphics from a different perspective. Using the GraSSML system, the semantic and structural information is made available at the creation stage. The availability of this information offers new possibilities. Web Graphics become “smart”, they carry their knowledge with them, and intrinsically they know their structure (ZineML) and semantics (MyLanguage). Any application smart enough to access, explore and present this knowledge in the right way should be able to propose solutions for identified problems (e.g. adaptive and accessible graphics for all). This could substantially improve the accessibility of web graphics. GraSSML could be the starting point for many projects currently aiming to access, present, explore and adapt graphical information [15] [16].

6. ACKNOWLEDGMENTS

Financial support for ZBF from Oxford Brookes University is gratefully acknowledged.

7. REFERENCES

The Meaning of ‘Life’: Capturing Intent from Web Authors

Rhys Lewis
Volantis Systems Ltd
1 Chancellor Court, Occam Rd,
Surrey Research Park, Guildford UK
+44 1483 739 775
www.volantis.com
rhys@volantis.com

ABSTRACT
Interest in accessing the Web from small, mobile devices, such as cell phones, is increasing rapidly. The challenge of delivering content to such devices is similar in many ways to the challenge of delivering it to users with disabilities. There is a real synergy between these use cases which offers the hope that solutions applicable to one will also be applicable to the other. This presentation will examine the ways in which recent work in standards, being driven by the need to support mobile Web users, may also help to improve accessibility.

Categories and Subject Descriptors

General Terms
Human Factors, Standardization, Languages.

Keywords
Semantics, Adaptation, Web, Authoring, Mobile.

1. INTRODUCTION
For the majority of users, the Web is primarily a visual experience. The availability of network bandwidth and highly capable display technology mean that many web sites are lively, colorful and compelling. However, in achieving sites that are interesting and usable by the majority, authors and designers often use techniques that rely heavily on visual cues. The placement of material within a page, the color and styling of elements, and juxtaposition are all commonly used to convey meaning. Such techniques work well if the page is used in the context originally envisaged by the designer. However, these techniques tend to fail if the user context differs from that envisaged. For example, users with certain kinds of disability may not be able to view the site at all, or may need to zoom the page to perceive its text and graphics. In this case, the visual context and cues that rely on juxtaposition may be lost. Similarly, users accessing the site from small, mobile devices will only be able to see a small part of each page at any one time, losing the context and relationship between the sections that it contains.

In a number of cases, the constraints placed on delivery of web pages to small mobile devices mimic those that have been experienced by users with disabilities since the inception of the Web. Improvements in user experience require more explicit representations of meaning than are currently in common use by web authors.

The title of this paper, while more than a little contrived, emphasizes the issue of the lack of semantic information in most web pages. According to one dictionary [11], the word ‘life’ can be used as a noun or an adjective. There are a dozen or so different meanings for the word when used alone, and several additional meanings when it is paired with other words. The terms ‘life preserver’ and ‘life science’ are examples.

The problem of trying to analyze the meaning of the word ‘life’ within a sentence is akin to the problem of trying to analyze part of a web page. Context is often of vital importance in allowing the true meaning to be understood. The problem with much of today’s Web is that the context is not explicit. It relies on a particular interpretation of the rendered page by the user. Anything that alters that interpretation, such as a disability, or a device that has characteristics that the designer did not expect, can inhibit the user’s understanding of the page.

It is worth examining for a moment the kinds of semantic information conveyed by web pages. It is possible to think of these in two broad categories. First, there are the semantics associated with the application itself. For example, a web site dedicated to movies is likely to encapsulate concepts such as reviews, tickets, prices, movie theatres and times of shows. These concepts are likely to be represented in the user interface of the site. However, they will almost certainly be expressed as some combination of markup and script. These representations will not directly encapsulate the application level concepts. Rather, they will implement a user interface for viewing or manipulating the data associated with the concepts. For example, the proposed movie site might provide forms from which users can access a movie review or purchase tickets.
This specific application of the user interface leads to the second general category of semantics. Even at the simplest level, sites embody general user interface semantics. By convention, particular arrangements of user interface elements have become commonly understood. A set of links arranged horizontally across the page near its top edge is understood to be the primary navigation for the site, leading users to major sections. It has become common to term such an arrangement a ‘navigation bar’. However, there is no explicit representation of this abstraction in current markup languages, such as XHTML Version 1. The implication that such a set of links has some specific meaning is conveyed purely by visual cues, such as position, color and background.

Another example of user interface semantics conveyed visually relates to simple XHTML forms. It is very common for the relationship between an input field and its label to be defined simply by juxtaposition. The label is placed so that it is adjacent to the field when the page is viewed in the way envisaged by the designer. As with the navigation bar, unanticipated changes in the way in which the page is rendered can defeat the intent of the author in conveying information.

<table>
<thead>
<tr>
<th>Logo</th>
<th>My Account</th>
<th>Login</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Logo" /></td>
<td><img src="image" alt="Login" /></td>
<td><img src="image" alt="My Account" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Book Tickets</th>
<th>City</th>
<th>Movie</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Book Tickets" /></td>
<td><img src="image" alt="City" /></td>
<td><img src="image" alt="Movie" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>New this week</th>
<th>Search for Movies</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="New this week" /></td>
<td><img src="image" alt="Search for Movies" /></td>
<td><img src="image" alt="Advanced" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Movie #1</th>
<th><img src="image" alt="Movie #1" /></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Movie #1" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Movie #2</th>
<th><img src="image" alt="Movie #2" /></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Movie #2" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Movie #3</th>
<th><img src="image" alt="Movie #3" /></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Movie #3" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Movie #4</th>
<th><img src="image" alt="Movie #4" /></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Movie #4" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Movie #5</th>
<th><img src="image" alt="Movie #5" /></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Movie #5" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Featured Movie</th>
<th>Most Popular</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Featured Movie" /></td>
<td><img src="image" alt="Most Popular" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vote for your favorite movie</th>
<th><img src="image" alt="Vote for your favorite movie" /></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Vote for your favorite movie" /></td>
<td></td>
</tr>
</tbody>
</table>

![Figure 1. Example of a moderately complex web page](image)

These broad categories of application and user interface semantic information have characteristics that differ somewhat and suggest that different approaches to their representation may be appropriate. Application level semantics are essentially unconstrained. After all, a web site might deal with virtually any subject. Representation of application level semantics effectively requires the full power of semantic web approaches [14]. These representations need to be associated, in some way, with the page markup. We’ll look at some examples in the section on Semantic Enrichment.

In contrast, user interface semantics form a far more restricted set. Annotation may still be appropriate for some representations. However, it seems possible that at least some of these semantics might be represented directly in the markup languages that express the interfaces themselves. Explicit representations tend to be simpler for authors than annotation. However, they also tend to require more comprehensive support in markup languages. We’ll return to this topic in the section on Semantically Rich Languages.

Providing additional semantics in authored materials is one thing. Using them, of course, quite another. However, once again, the needs of assistive technologies, which help those with disabilities access the Web, mirror those of systems which adapt content for mobile devices. In both cases, the additional semantics help the systems involved interpret the available materials in order to provide a more appropriate user experience. We’ll look further at this topic in the section on Semantics and Adaptation.

The W3C is actively involved in standards development related to web access for the users with disabilities and web access from a wide variety of different types of device. We’ll look at the work of the W3C Mobile Web Initiative (MWI) [13] and of the W3C Device Independence Working Group (DIWG) [12] in the section on Device Independence and the Mobile Web.

2. TRADITIONAL APPROACHES

The traditional approach to solving the problems of web access for users with disabilities and users with mobile devices is essentially the same. It involves processing the markup sent to the device to try and provide a more appropriate representation. In the case of users with disabilities, the function is usually carried out by some assistive technology that runs in the computer being used to access the Web. In the case of mobile devices, it usually runs as a server side process that transforms the material between its origin and the user’s device. This transformation process is normally termed ‘transcoding’ and the processor which performs it is usually termed a ‘transcoder’. The work of an assistive technology can also be viewed as transcoding.

Transcoding relies on the interpretation of markup created for a page by an author who almost certainly assumed that it would be used on a typical desktop or notebook computer by an able bodied person. As we’ve already noted, a significant portion of the semantics of most web pages is conveyed in the visual arrangement of elements. It is not explicitly present in the markup that a transcoder processes. In addition, lively, dynamic, interactive sites that provide compelling user experiences tend to make heavy use of
client-side scripting. This can mean that some of the semantics of the site are embodied within program code that executes within the browser. Not only might a transcoder or assistive technology need to interpret the markup of a page, it may also need to try and interpret the program code within the scripts that are used.

Figure 1 shows the wire frame layout for a moderately complex web page. This hypothetical example is taken from the movie web site mentioned earlier. We’ll use it to illustrate some of the challenges that transcoders and assistive technologies can face.

The page is laid out as a series of sections on a two dimensional grid. Within each section, there are further subdivisions. The relationship between sections and subsections is not necessarily explicit. For example, consider the section ‘New this Week’. The layout shown in the figure could be achieved by using a table in which each movie has its own row. The title of the section simply occupies the first row of this table. The movies and the title are at the same level of nesting. In this case there is no explicit containment relationship that can be used, by a transcoder or assistive technology, to label this part of the page as ‘New this Week’. The grouping intended by the author is achieved implicitly by visual juxtaposition. This same issue might afflict any of the sections shown in Figure 1, of course.

The forms in Figure 1 use a variety of different layouts. Once again, the relationships between labels and fields are achieved visually, making interpretation of the markup difficult. In the case of the ticket booking section, the labels and fields are not even in the same row of the table that the author uses for layout. They are simply text items that happen to appear within the same table as some form fields.

One other relatively common approach to form input can also be illustrated by Figure 1. The author has decided use just a single form processing URL. Consequently, the ‘Vote’ button, the ‘Go’ button and the scripts associated with the drop down ‘City’ and ‘Movie’ selection lists all submit to the same URL. All of the input controls are associated with the same form. When the form is submitted, the processing code examines the fields that are present to determine the function to be performed.

While this may be convenient for the author, it makes life much more difficult for interpretation. Not only is the relationship between input fields and labels implicit, so is the function of the submission buttons. And, of course, one of the submission methods is hidden within the scripts that process the ‘City’ and ‘Movie’ drop down selection lists.

Now of course, there are authoring approaches that are less unfriendly to transcoders and assistive technologies than those described here. However, even really well written pages require some level of intelligent guesswork, often called heuristics in polite circles, to interpret their semantics.

Although this is a contrived example, it should be clear by now that even relatively simple web pages can hold real challenges for automatic interpretation. It should also be clear that at the heart of these challenges is the lack of semantic information in the page.

3. SEMANTIC ENRICHMENT

If the issue of interpretation is caused by a lack of semantic information, then one obvious strategy is to make up the deficit. This approach goes under the general title of semantic enrichment. The approach concentrates on giving the authors the tools needed to make their intentions clearer by adding semantics explicitly. In the last year or so, considerable progress has been made in the definition of the kinds of semantics that can aid interpretation of web pages.

Late in 2004, the W3C DIWG [12] held a workshop on Metadata for Content Adaptation [4]. Mechanisms for describing additional semantics in web pages were discussed and some general principles were defined. The notion of the ‘role’ of a particular part of a web page was identified as a key item of semantic information. The relationship between parts of a web page was also identified as a key item.

The W3C Web Accessibility Initiative Protocols and Formats Working Group (WAI-PF) [16] is currently developing a taxonomy [9] for the roles associated with particular parts of a web page. The taxonomy identifies roles associated with user interface components, such as those on forms. It also identifies various structural roles, such as menus, toolbars and lists.

Recent versions of the Mozilla Firefox browser incorporate support for additional semantic information, such as the roles defined by WAI-PF [16]. Information about the support in Firefox for accessible dynamic HTML (DHTML) is available [17].

Support for the addition of semantic information is based on a new attribute used in the markup of the page. This attribute, named ‘role’ was first proposed for inclusion in a major new version XHTML [1]. Subsequently it has been proposed for and included in implementations of other markup languages [17]. The approach has very little impact on languages into which it is introduced. For example, the following XHTML markup fragment identifies a table whose role is to behave as a spreadsheet.

<table id="table1" x2:role="wairole:spreadsheet" >
</table>

The prefix values ‘x2’ and ‘wairole’ identify XML namespaces that remove problems associated with duplicate names being used for different purposes.

have been omitted. These would apply to the entire page in which such markup appears. The semantic information is conveyed by the value of the role attribute. The meanings...
of values like ‘spreadsheet’ are defined in [9]. Using them, authors and those responsible for interpreting markup can gain a shared understanding of the semantics of the user interface.

Although the current work is being driven by the needs of assistive technologies, the roles that are being defined have general utility. They are capable of labeling parts of a page with the author’s intended user interface semantics. As such, they are as applicable and useful for assisting transcoding as they are for supporting assistive technologies.

The ability to annotate sections of web pages with semantic information does not, of course, remove the need to structure the page appropriately. In particular, there must be markup that represents the section to be annotated. As we saw earlier in the discussion of the section ‘New this Week’ in Figure 1, this may not always be the case with existing web pages. Nevertheless, the definition of a taxonomy of user interface roles and the appearance of support for it within a leading browser represents considerable progress.

4. SEMANTICALLY RICH MARKUP LANGUAGES

Semantic enrichment provides a powerful way to extend markup languages by retaining additional information about the author’s intent. As we’ve already noted, a complementary mechanism is the use of semantically rich markup languages. Where a facility can be provided directly by a language, rather than requiring semantic enrichment, it is arguably simpler for authors to use and less liable to error.

4.1 User Interface Semantics

Commonly used markup languages do not, as yet, contain such as rich a set of facilities as are described in [9]. However, newer W3C markup languages are introducing additional abstractions which improve the level of semantic information available within web pages. A good example, which overcomes a number of the issues associated with Figure 1, is XForms [2]. This replaces the traditional HTML forms support with a semantically richer and more capable set of facilities. In addition, XForms provides much more explicit linkage between the various components of a form. Control of the way in which form components are rendered is removed from the markup and made a concern only of styling. For example, the operation of selecting of one option from many is defined by its semantics, ‘single selection’, and not by its representation, ‘radio button’ or ‘drop down list’.

In addition to its interest in semantic enrichment, the W3C DIWG [12] has also been pursuing the notion of semantically rich markup languages to support use of the web on a wide range of devices with different characteristics. Rather than defining entirely new languages, DIWG has based its work on other W3C specifications. In particular, work has focused on XHTML 2.0 [1] and XForms [2] as the basis for a Device Independent Authoring Language (DIAL) profile [10]. To this base, DIAL adds facilities that are particularly useful for authors that must support a variety of different kinds of device.

Some aspects of support for different devices can be automated, but some may require additional work by authors. For example, consider the task of helping a potential customer travel to a shop to collect some goods. This is usually accomplished by displaying a map. Often this is supplied by one of the on-line services readily available in many countries. The map is composed of one or more images showing locations, roads and other features. Such a representation is appropriate for an able bodied person using a typical desktop or notebook computer. For someone with a visual impairment, or who is accessing the site using a mobile phone, the map may be of little or no use. For example, automatic transformation of the map to a smaller size appropriate for the user’s mobile phone may render it illegible.

One solution, in this particular case, is for the author to provide an entirely different form of information in addition to the map. An alternate, textual description of how to find the shop may be more appropriate for the phone user than another visual representation. For a visually impaired user, spoken material may be of more appropriate. In either case, an improved user experience depends on the author providing alternative versions of the materials. The ability to create such alternate representations, and to have them delivered when appropriate, is one of the extensions on which DIWG is currently working [6]. This capability also forms part of the DIAL profile.

DIWG is also working on the provision of explicit mechanisms for authors to define page layout separately from page content. Current practice for page layout often involves the subversion of the XHTML table mechanism or the use of advanced CSS. Using tables means that the layout of a page is embedded in the markup and cannot easily be changed for use on different devices. CSS currently does not include sufficiently subtle means of associating styling with different devices. It also has limited support for coarse-grained definitions of page layout. These are the aspects of particular interest to DIWG.

Often, the kinds of layout change required in order to make a page render well on a different device are rather simple. It may only be necessary to move sections of the page in relation to one another. For example, one way to rearrange the page in Figure 1 might be to convert it to a single column layout with the sections following one another in a particular order. This kind of layout change can be achieved if each section is associated with a specific area in a layout representation held externally to the page markup. Sections can be moved simply by changing the associated layout without affecting the page markup itself. Different layouts can be used to support devices with different...
capabilities. Commercial implementations, such as those provided by Volantis, have been available for a number of years. The approach has been found to be versatile and very effective.

4.1.1 Richer User Interfaces

Recently, W3C has initiated new work in the area of richer user interfaces with more capability. The Web Application Formats Working Group (WAF WG) [19] is looking at ways to enhance existing W3C specifications using the results of work by specific browser manufacturers. The aim is to provide more capable platforms for web application development. Part of this work is likely to lead to richer user interface abstractions with higher semantic content.

4.2 Application Semantics

As we noted earlier, application level semantics relate to the concepts embodied in some application, rather than those of the user interface by which it is represented. In the hypothetical movie web site, such concepts include reviews, tickets, prices, movie theatres and times of shows. It is possible to construct markup languages that deal explicitly with such concepts. One common approach is to create a language based on XML by defining an appropriate schema [3]. Such a language would include markup that explicitly represents the key concepts. For a movie, for example, there might be explicit representations of the title, director, leading actors, genre, audience suitability and so on. The important characteristic of such markup languages is that they represent only the semantics and not the associated user experience. Such representations can be adapted to create a user experience, but do not explicitly define it. An adaptation step is used to convert the semantic representation to one that can be used in a web page. This step can create different markup for different classes of device if necessary. However, whereas adaptation of languages that represent user interface semantics can be generalized, adaptation of application level semantics is tightly coupled to the application itself. Applications are too numerous and too varied for there to be much likelihood of general agreement about the form of the semantics employed.

Over the last few years, a number of mobile operators have created languages that include some application level semantics. In their desire to provide data services over their networks, mobile operators have, historically, provided systems that distribute specific web pages to their customers. These pages are accessed via a home page provided by the operator. Applications and pages are provided by companies and organizations that are in partnership with the operator. This sort of arrangement is often known as an operator ‘portal’. The operator provides the language used to create pages within this portal environment and is responsible for adaptation of those pages to allow them to work on any device that is supported. Such languages usually provide some form of user interface abstraction, but also provide abstractions that relate to the portal itself, or to applications that run within it. However, every operator language is different and encapsulates different aspects of the application semantics. This makes it very difficult for application and content partners who wish to provide materials for multiple operators. They may have to rewrite their application for use on different operator’s networks.

5. SEMANTICS AND ADAPTATION

In the traditional Web, the concept of a ‘page’ is fundamental. Users access pages. Authors create pages. A page is the unit returned in response to a request from a user. Actually, that last statement is not entirely true, since images and other media are delivered separately. However, it’s broadly true to say that what the user perceives, what the author creates and what the Web delivers in a single request-response cycle are essentially the same.

In its efforts to describe a system that could deliver materials to a wide range of different devices with very different capabilities, DIWG generalized the definitions of the concepts associated with web pages to allow a more precise description of the associated architecture. We’ve already noted that under certain circumstances, authors may need or indeed want to provide alternative representations of specific materials. This immediately suggests that a page is not an indivisible item. Rather, it is composed of one or more ‘authored units’ [7], sets of materials from which the actual user experience will be constructed.

Even in the traditional Web, the delivery of materials to a browser usually occurs as several distinct steps.

Figure 2. The page from Figure 1 after adaptation

Pages may refer to style sheets, script functions, images and other media, that are requested separately. DIWG has defined the term ‘delivery unit’ [7] to describe a set of material transferred in a single request-response cycle.

Using these definitions, we can describe the process of adaptation in the context of the request for a web page from a particular device. Look again at Figure 1. Let’s suppose that the author constructed this page knowing that it would need to be adapted for use on some particular mobile devices. In particular, the author created an authored unit for each of the major sections in the page, such as ‘New this Week’ and ‘Book Tickets’. When the URL is accessed from a traditional web client, all of the authored units are aggregated to form the delivery unit, which is returned to the client. The user sees the entire page as in Figure 1.
In contrast with these selection mechanisms based on the rules about whether sections with specific roles are used for the mobile device in question. Of course, at some point the author would need to create the rules about whether sections with specific roles are delivered to specific types of device.

Alternative mechanisms, based on the notion of the priority or importance of a section were discussed in the workshop described in [4]. In this approach, sections are given numerical priorities. Authors also define some threshold of priority for each type of device to control which sections are delivered. Though less precise than role-based labeling, this mechanism adds semantic information to the sections that can be used by the adaptation process.

In contrast with these selection mechanisms based on labeling, the alteration to the layout within the search section could be viewed as purely a user interface adaptation. The material in the section has not changed. However, its representation has been altered. In this case, approaches based purely on styling and layout are possible. One approach might be to specify a style sheet for use on the mobile device which differs from the one used for access from desktop and notebook systems.

One additional layout needs to be considered when comparing Figure 1 and Figure 2. Not only does each section have its own internal layout, but in addition, the sections are laid out within the page. This layout differs between the two versions of the page. Some mechanism is needed that allows authors to specify this sort of coarse-grained layout separately from the authored units that define each section. As with the style sheets, different versions of such layouts could be used with different devices. Unlike the case with style sheets, there are no agreed standards for such definitions yet. However, a number of commercial implementations exist and DIWG is working on this topic with other groups in W3C.

In addition to the kinds of technique described here, there are many other ways in which authors can specify materials that are used in adaptation. DIWG is currently working on extensions to W3C markup languages that allow such information to be expressed by authors.

5.1 Delivery Context

As we have seen, adaptation uses the materials provided by authors to create versions of a page, appropriate for use on particular devices. To enable the adaptation to be appropriate, certain information about the target device must be available. DIWG has chosen to call this sort of information the ‘delivery context’ [7]. This name reflects the fact that information that influences adaptation could be more than just the characteristics of the device itself. In particular, characteristics related to intervening networks might also influence adaptation, as might the personal preferences of the user.

Device-related information in the delivery context might include items such as:

- The physical size of the device’s display screen in some linear measure
- The size of the device’s display in pixels
- The number of colors that the device can represent
- Markup languages supported by the device
- Image formats supported by the device
- Audio formats supported by the device
- etc.

Information in the delivery context is used during adaptation to select or create suitable representations from the materials provided by the author. For example, if an author has created several versions of a company logo, an adaptation processor might use display size information, color capability and supported image formats, to select the most appropriate version from those available. If none of
the available versions were appropriate, the processor
might be able automatically to generate a new image from
those available.

Likewise, by knowing that a device requires cHTML or
WML, for example, an adaptation processor can
automatically transform the markup used to create authored
units into one appropriate for the delivery units.

The category of personal preferences in the delivery
context has been a topic of discussion within the
accessibility community. It’s relatively common for
disabled users to influence the user experience they receive
explicitly through settings associated with their browser.
This might be as simple as altering text sizes, or as complex
as creating alternate style sheets.

The ability for an adaptation process to be influenced by
personal preferences offers at least the possibility that, in
future, much more sophisticated control of the user
experience might be available to users with disabilities.
However, there is a challenge. By its nature, adaptation is
distributed, and might take place anywhere in the chain of
processing between the user’s device and the origin server.
The basic architecture proposed by DIWG shows delivery
context flowing throughout the processing chain. If user
preferences are part of the delivery context, this implies
that some level of personal information will be transmitted
from the device into the network. This raises questions of
security and even of personal safety. Already, research has
shown that it is possible to reason about a user from the sort
of information that would be available in such a context
(see for example position paper number 26 in [18]).
Clearly, users making this sort of information available
need to be assured that it will not be used inappropriately.
This leads to the need for trust relationships between users
and the systems that they use.

6. DEVICE INDEPENDENCE AND THE
MOBILE WEB

In recent years, improvements in the capability of mobile
devices and in the networks they use have led to renewed
interest in the provision of general web access. In 2005, the
W3C formed the Mobile Web Initiative (MWI) [13] to
provide a focus for standardization work associated with
web access for mobile devices. The initiative has received
support from a broad range of organizations.

The overall goal of the initiative is to enable the greater use
of the Web on non-traditional devices, such as mobile
phones and other handheld systems. It will achieve its goals
in a number of ways that are complementary to those of
other groups within the W3C. Some of its work builds on
W3C specifications by providing help and guidance to
authors in the form of best practices (see for example [8])
and other outreach programs. Some of its work will result
in additional requirements and clarifications that will be
used by other W3C groups. Indeed, DIWG is already being
assisted in its work by information provided by groups
within the MWI. The method of operation of the initiative
is similar in nature to that of the Web Accessibility
Initiative (WAI) [15] at W3C. Groups within both
initiatives provide a focal point for expertise in their
respective fields. They influence developers of W3C
specifications and provide specific help and guidance.

Although the ultimate objectives of these two initiatives are
different, it is recognized that there are common solutions
that can benefit both. There is a close working relationship
between their members, as there is with other, relevant
working groups at W3C.

7. CONCLUSION

There are many tensions on the Web. At one end of the
spectrum are social and political tensions, such as the
conflicts between free speech and censorship. At the other,
are tensions between different organizations who would
like to exercise control at the technical level. Somewhere
between these extremes is a tension that has existed as long
as the Web itself. It is the tension between authors and end
users.

In many cases, authors need to control the precise look and
feel of a web site to meet specific requirements of the
organizations for which they work. Many organizations
have very strong views on how their sites must appear and
how they must behave. Style specifications are common,
and are often strictly applied. This desire for controlled
look and feel is also common in the kinds of operator portal
provided for mobile users and discussed earlier.

Users of web sites, on the other hand, may wish to alter the
way in which a site renders. Simple changes, such as a
modification in overall font size to improve readability or
for use during a presentation, are common. As we have
seen, users with disabilities may need to alter many aspects
of the rendering in order to use a site effectively. Such
changes are anathema to an author who has spent a great
deal of time and effort in meeting the requirements of a
style guide. They may also impair overall usability when
compared with the original look and feel. However, they
may be crucial for the end user with a disability.

Part of the problem is that those who specify style guides
and those who implement them tend to consider only able-
bodied users with the same kind of web access as they have
themselves. In addition, the technologies that underpin
the Web and the tools that have been built on them, have not
really given authors the tools they need to cater for a wide
range of different kinds of access.

The increasing interest in support for users who are mobile
and who use devices very different from those used by
authors, is forcing a rethink. There are commercial
pressures on organizations to provide a mobile presence,
and this is compelling them to devise and use solutions that
can support a much wider diversity of access. Adaptation
and the provision of alternative content, styling and layout
plays a key role in such solutions.

Interestingly, the same kinds of approach to authoring that
can support access from diverse devices can also be used to
assist in supporting some types of disability. After all, the
display on a small mobile device behaves in a very similar way to the display on a desktop system on which a web page has been zoomed up by a user with low visual acuity. Work amongst vendors and standards bodies is progressing to the point where the technologies needed to support access from diverse environments will be widely available. If authors adopt such technologies, the very same materials needed to support mobile access may also play an important role for users with disabilities.

At present, support for disabilities tends to place a burden on the end user. Often, the user is required to control web page rendering explicitly to achieve acceptable results. The changes that a user needs to make may affect the page adversely, causing it to be more difficult to understand and use.

An intriguing possibility for the future is that it might be possible to serve appropriate content automatically to users with at least some classes of disability, using the very same techniques used to support mobile devices. Sites would effectively tailor themselves to users via the same techniques by which they tailor themselves to different devices.

Much work remains to be completed. However, the renewed focus on addressing these problems, which has resulted from the need to support small mobile devices, is driving progress in the development of standards that may be applicable in addressing at least some of the needs of disabled users.

8. REFERENCES


[13] The Mobile Web Initiative (MWI) at W3C http://www.w3.org/Mobile/


[16] The WAI Protocols and Formats Working Group (WAI-PF) at W3C http://www.w3.org/WAI/PF/


Evaluating Interfaces for Intelligent Mobile Search

Karen Church, Barry Smyth and Mark. T. Keane
Adaptive Information Cluster,
Smart Media Institute, Department of Computer Science,
University College Dublin, Belfield, Dublin 4, Ireland
{Karen.Church, Barry.Smyth, Mark.Keane}@ucd.ie

ABSTRACT
Recent developments in the mobile phone market have led to a significant increase in the number of users accessing the Mobile Internet. Handsets have been improved to support a diverse range of content types (text, graphics, audio, video etc.), infrastructure investments have delivered improved bandwidth, and changes to billing models offer users much greater value for content. Today large numbers of users are moving away from browsing operator portals and towards off-portal search, leading to a growing need for mobile specific search engine technologies. In this paper we argue that existing mobile search engines are unlikely to offer an adequate service for mobile searching. Most borrow traditional query-based search and list-based result presentation formats from Web search and as such are not well optimised for the input and display features of mobile devices. For example, many simply attempt to translate Web content for the mobile space which is not appropriate. In this paper we evaluate an alternative strategy which replaces the usual result snippet with a more economic alternative that is composed of the keywords used in related queries. We argue that this alternative is better suited to the display characteristics of mobile devices, without compromising the informativeness of result snippets.

Categories and Subject Descriptors
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; H.5.2 [Information Interfaces and Presentation (e.g.,HCI)]: User Interfaces

General Terms
Design, Experimentation, Human Factors

Keywords
Mobile Search, Mobile Web, Mobile Internet, Mobile Interfaces, Search Interfaces, User Evaluation

1. INTRODUCTION
New life has been breathed into the Mobile Internet as a result of a combination of significant device, content, infrastructure and billing improvements. The result is that Mobile Internet usage is growing at a significant rate. According to a recent report published by Strategic Analytics, the total number of mobile phone subscribers worldwide approached 2.2 billion at the end of 2005 and looks set to reach 2.5 billion by the end of 2006 [11]. Over 800 million mobile phones were sold in 2005 and as the prices of cellular handsets continue to drop, this figure looks set to rise to 930 million [7] in 2006. The latest statistics also indicate that there has been a significant increase in the number of users accessing the Mobile Web. Ipsos Insight, a market analysis company, have recently published a web study showing that 28% of mobile phone subscribers worldwide have used their phones to browse the Internet, an increase of approx. 3% on the figures released in 2004. Interestingly, this pattern of growth was driven primarily by more mature users (age 35+) indicating that the traditional early adopter group, i.e. young males, no longer dominate Wireless Internet access [17].

Recent trends suggest that users are beginning to move away from the traditional walled garden of the operator portal as they begin to explore the burgeoning off-portal content. A similar effect can be traced back to the early growth of the World-Wide Web as users who had previously been content to browse early portals such as Yahoo, quickly began to explore the greater Web with the help of the latest search engines. And so we might expect mobile search to quickly come to dominate as the primary mode of information access for users, as it has in the World-Wide Web. Certainly there has been significant industry activity in the mobile search space with major players within the search engine industry venturing into the mobile sector. Google and Yahoo have released a number of mobile search solutions including a local search service and an SMS-based search service. Ask Jeeves is currently developing a new wireless search application and America Online (AOL) have recently added enhancements to their mobile search solution. A number of new mobile-specific search services have also come onto the market including Moolbl1, 4info2, UpSnap3 and Technorati Mobile4.

However, despite this flurry of activity it is our contention that the state of mobile search is not a healthy one. In particular, we believe that the current strategy of simply retrofitting traditional search engine technologies borrowed from the Web for the mobile space is ill conceived. In this paper we argue that such an approach is unlikely to succeed because of the significant and unique challenges presented by the Mobile Internet. We begin with a summary evaluation of 7 existing mobile search engines. The results point to

1http://www.moolbl.com/
2http://www.4info.net/
3http://www.upsnap.com/
4http://m.technorati.com/
some significant limitations of current approaches and in particular highlight some serious problems when it comes to presenting search results on small-screen devices. In the remaining sections we focus on the example of result presentation and argue that the traditional approach of providing snippet text alongside results is inappropriate in a mobile context. Instead we propose a more effective solution based on the reuse of past queries as the basis for a more economic approach for gisting the meaning of results. We argue that this alternative is better suited to the display characteristics of mobile devices without compromising the informativeness of result snippets.

2. RELATED WORK

In the context of this paper, there are two separate strands of related work that are especially relevant when it comes to delivering information content to, or adapting information content for, mobile devices. The first strand concerns the general area of web page adaptation which involves automatically transforming or restructuring Web pages so they may be displayed more effectively on mobile devices. The second strand of related work is more focused on the presentation of search results on mobile devices and is obviously closely related to the work presented in this paper.

2.1 Web Page Adaptation for Mobile Devices

Many mobile search engines (including Google) seek to provide mobile users with access to normal Web content. But of course to do this, the content must be adapted so that it is compatible with mobile device displays. For example, the Digestor system [2] uses a re-authoring approach to transform a Web document using a range of design heuristics as well as text summarization, page categorization, the removal of irrelevant content and image reduction techniques. For example, Digestor’s design heuristics tell it that keeping at least some images is important (usually the first and last image), that header tags (H1-H6) cannot be trusted as proper semantic headers for use as proxies for a block of text (instead text blocks are better summarised using the first sentence or phrase). Preliminary results suggest that while Digestor does a good job of preserving key content during re-authoring, the results are often not aesthetically pleasing when viewed on a small-screen device.

The WEST Browser (WEB browser for Small Terminals) [3], uses a technique called flip zooming which is a tile based focus + context visualization technique for displaying web pages on handheld devices. A comparative evaluation carried out by the authors showed that the WEST browser provided users with a better overview and an easier search mechanism when compared to the HotJava browser. However, users also thought that the flip zooming interaction technique was quite difficult to use and took some time to get used to. The PowerBrowser system [4] provides a set of tools for searching, navigating, browsing, and input entry on small devices. Results of an evaluation carried out by the authors demonstrated that users experienced significant time-savings while using PowerBrowser for directed tasks.

In WebThumb, [18], the layout of pages is left untouched. Instead, graphical thumbnails are used to display whole pages and the browsing experience is improved by enhancing the normal interaction techniques available to the user. For example, a pick up tool enables users to extract elements from a page and display them in a separate window and zooming and panning tools allow users to take a closer look at content of interest.

The thumbnail concept is extended by [13]. The authors present a prototype application called SearchMobil which is able to partition a document into a number of different regions by examining the underlying structure of the page and provide a thumbnail overview of the document using these regions. These thumbnails are then annotated to show the location of query terms within the document with the aim of directing users to the most promising sections of the page. A user study carried out by the authors, showed that this approach was well-suited to fact-finding tasks. In [19], importance values are assigned to different segments of a web page in order to present mobile users with more compact search results and hopefully point users in the direction of more relevant results. There are obvious challenges when it comes to adapting graphically-rich Web content for screen-poor mobile devices and in our opinion it is difficult to envisage an automated solution that will be capable of competing with the experience associated with mobile-specific content that has been designed for mobile devices.

2.2 Displaying Search Results on Mobile Devices

More directly relevant to the focus of this paper is the very specific challenge of how best to present search results for a mobile device. The standard Web strategy of presenting page titles, URLs and (hopefully) informative snippets of text is often adopted but comes with its problems, not the least of which is the high screen “real-estate” demands that snippet text imposes on mobile devices. One alternative that is also common place sees the elimination of snippet text altogether, leaving users at the mercy of often uninformative result titles. Clearly there is a need for a solution that can provide information regarding the relevance of specific results, but without consuming the screen resources of full snippet texts.

Work in this area has been rather limited to date. For example, [9] have looked at the general issue of the performance of Web searchers and their mobile counterparts on a range of different search tasks. The overall aim of the evaluation was to identify the impact of screen size on search performance and the results, unsurprisingly, pointed to a significant drop in search performance for mobile searchers. In addition the study also highlighted significant challenges for mobile searchers when it came to interpreting the usefulness of individual results according to their search needs. This issue of how to help users to understand the value of recommended results is addressed by [10]. Instead of using standard snippet text approaches (which involve the extraction of a block of document text, usually related to the query) they use a set of key phrases, automatically extracted from result pages. The resulting key phrases provide for a more economic use of screen space and are at least as effective and informative as using long result titles.

3. THE STATE OF MOBILE SEARCH

Before continuing it is worth reflecting on the current state of the Mobile Internet and mobile search. And so in this section we will briefly review a number of important Mobile Internet developments over the past few years, developments that are responsible for increased levels of mobile search, in addition to the summary results from a recent review of some of the leading mobile search engines.

3.1 The Mobile Internet and Handsets

The late 1990’s saw the advent of WAP (Wireless Application Protocol), the first generation of the Mobile Internet, promising users a new era of Mobile Internet Services. Unfortunately these promises rarely stood up to scrutiny and the content-light text-based services that were available did little to excite users about the potential of the Mobile Internet, especially when combined with low-bandwidth connections and expensive billing models.

In recent times however, the Mobile Internet has experienced something of a rebirth. Content has improved dramatically, offering users a wide range of rich-media services including colourful...
information pages, polyphonic ringtones and video on the go. In addition, the slow connections associated with the early Mobile Internet have been replaced by a much faster GPRS and 3G infrastructure providing users will almost instant access to content. Operators have also overhauled their billing practices to offer subscribers a much more cost-effective Mobile Internet service; instead of charging users for their time online, they are now only charged for the content they consume.

While all of these changes have led to significant improvements in the state of the Mobile Internet, perhaps the single most significant development has been the mobile handsets themselves. Gone are the tiny text-based, monochrome displays of the original WAP phones and in their place we have high-resolution, colour handsets with built-in browsing features and enhanced data input capabilities. In general, today, there are three types of mobile handset on the market: (1) standard WAP phones that offer high-resolution colour screens, albeit small screens, with predictive text input; (2) 3G smart phones with larger displays, enhanced browsing support (e.g. xHTML), and miniature keyboards; (3) PDAs with large colour displays, stylus/pen input, full HTML and Flash 6 support, and enhanced interaction features (e.g. full QWERTY keyboards).

3.2 Evaluating Mobile Search

The usage increases that have been concomitant with these improvements in handsets, bandwidth and content have led to an increase in mobile search, as users venture beyond operator portals to explore the growing content of the Mobile Internet. However, when it comes to helping users to locate information in the growing information space, improvements have been slow to come. For example, mobile search engines remain limited and fail to offer users a high level of user experience. To qualify these limitations, in the remainder of this section we outline the results of a recent evaluation of the state of mobile search.

3.2.1 Methodology

To begin with, a representative sample of 7 mobile search engines were chosen as evaluation targets. Click4WAP, Google, Ithaki, Mooobl, Seek4Wap, WAPAll and WAPly were chosen for a number of reasons. First they represent a mixture of older search engines as well as the latest offerings; for example, Mooobl was launched as recently as June 2005. These search engines were also chosen because each focuses on the retrieval of mobile specific content, rather than attempting to retrofit standard HTML pages for mobile handsets. That said, it is worth noting that Google is distinguished by its indexing of xHTML pages as well as WML pages whereas the other engines primarily index WML pages. Furthermore, the Ithaki search engine is a meta search engine that combines results from WAPAll, FreeWAP, IndexCell, Click4WAP and WAPitOut.

A sample of 20 queries (e.g.: cheap flights, tour de france, weather forecast, ringtones) were submitted to each of the 7 search engines and their results were retrieved and recorded (See Appendix A for a full list of the 20 queries). Each result was manually assessed for its relevance to the target query and a range of relevance statistics were calculated for each engine including:

1. (NRR) The average number of results returned per query;
2. (1stRR) The average position of the 1st relevant result in the result-lists for each query;
3. (%QwR) The percentage of queries for which the search engine return results;
4. (%QwRR) The percentage of queries for which a relevant result could be found in result-lists;

To illustrate the type of interfaces each of our sampled search engines present to their users we have included screen shots of the initial search results screen generated by each of the 7 search engines to the query news on a Nokia series 60 WAP phone; Figure 1 shows each of these screen shots. In addition to the above relevance considerations we will also assess the different ways in which these search engines present their result lists with a view to better understanding the presentation trade-offs they have come to adopt in light of the significant screen limitations of mobile devices, compared to their large-screen desktop and laptop relations.

3.2.2 Relevance and Coverage

The results corresponding to the above are presented in Table 1. The first thing to notice is the difference between Google and the other engines when it comes to the number of results retrieved per query on average. Google retrieves 126,000 results per query on average compared to a much lower average for the remaining engines; WAPly retrieves the least with 7 results per query, with Click4WAP the best of the rest at 38 results per query. These differences point to significant variations in the index size of the different search engines.

![Figure 1: Illustration of the Search Results Displayed on each of the 7 Search Engines in Response to the Query ‘News’. Figure (h) in this Group Illustrates our Related Query Interface Approach.](image-url)
different engines, although it would be misleading to claim a simple
direct correspondence between the average number of results re-
trieved per query and index size; we do not propose to consider this
issue in detail here but the interested is referred to [12] for a related
Web study. That said it is clear that Google’s index size is signif-
ically larger than the competing engines most likely owing to its
coverage of xHTML content as well as WML content.

Google’s indexing of xHTML pages as well as WML allows it to
perform best of all. It has results to offer for all of the queries and
returns relevant results for 85% of queries, with the top relevant re-
sult occurring at position 7 on average. However it is clear that the
state of the pure WML search engines is somewhat less healthy, and
given that a large number of mobile handsets are not yet equipped
to handle xHTML content, this highlights an important problem for
most Mobile Internet users. For example, when we look at the av-

dge position of the 1 st relevant result in result lists we see that
Seek4Wap performs best with an average position of 1. However,
we also see that while this search engine retrieves results for 90% of
the queries it only offers a relevant result in 15% of queries. To put
this another way, for the vast majority of queries (85%) Seek4Wap
delivers an average of 10 irrelevant results (and no relevant ones),
although when it does locate a relevant result it returns it in position
1. Compare this to Mooobl, which also retrieves a relevant result
for 15% of the queries, but returns these relevant results at position
4 on average. However, since Mooobl only retrieves results for
20% of queries, it can at least claim to avoid returning lists of irre-

dent results when relevant pages do not exist in its index. Overall
Ithaki, with its meta-search strategy, performs best of the WML
search engines, retrieving relevant results for 40% of queries (with
an average position of 5 for the top relevant result) and retrieving
irrelevant results for only 10% of queries.

### Table 1: General Properties of the Search Engine Results

<table>
<thead>
<tr>
<th>Search Engine</th>
<th>RR</th>
<th>IstRR</th>
<th>%QwR</th>
<th>%QwRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>126,000</td>
<td>7</td>
<td>100</td>
<td>85</td>
</tr>
<tr>
<td>Mooobl</td>
<td>21</td>
<td>4</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>Click4WAP</td>
<td>38</td>
<td>10</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>Seek4Wap</td>
<td>10</td>
<td>1</td>
<td>90</td>
<td>15</td>
</tr>
<tr>
<td>WAAPII</td>
<td>18</td>
<td>6</td>
<td>35</td>
<td>15</td>
</tr>
<tr>
<td>WAAPly</td>
<td>7</td>
<td>7</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>Ithaki</td>
<td>9</td>
<td>5</td>
<td>50</td>
<td>40</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td>17.2</td>
<td>6</td>
<td>43.3</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 2: How Each Search Engine Presents the Search Results

To consider the screen-space economics of search results in more
detail, in Table 3 we present information about the average length
(in characters) of each result feature for the different search en-
gines. As expected the average Google result is the most space
hungry of the engines, consuming an average of 138 characters per
result. In contrast, the majority of WML engines require less than
half of this. Interestingly, two of the WML engines, Seek4Wap and
WAAPII, use greatly truncated title text, requiring only 7 characters
per title on average compared to an average of about 42 characters
per title for the other engines. Overall Seek4WAP is the least space
hungry of the engines tested. Its use of truncated title and truncated
snippet text means that it requires an average of only 26 characters
per result, which is even less that the average title text space needed
by many of the competing engines.

In general then it should be clear that there is a major presenta-
tion issue when it comes to how best to present search result infor-
mation in a way that is informative to users while at the same time
sensitive to the screen limitations of most mobile phones. Consid-
ering that a standard Nokia series 40 mobile phone only fits approx.
6 lines or about 130/140 characters of text, while a Nokia series 60

### Table 2: How Each Search Engine Presents the Search Results

<table>
<thead>
<tr>
<th>Search Engine</th>
<th>Num/Rank</th>
<th>Title</th>
<th>Snippet</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Mooobl</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Click4WAP</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Seek4Wap</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>WAAPII</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>WAAPly</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Ithaki</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>
mobile phone (smartphone) fits approx. 7 lines of text or approximately 200 characters of text per screen, most of the search engines examined will allow for only one or two results per screen.

3.3 Challenges for Mobile Search

The challenges for mobile search should be becoming clear. Existing search engines suffer from significant coverage and relevance issues with many queries either going unanswered or being answered by misleading result-lists containing irrelevant results. Coverage is likely to improve as search engines improve their ability to crawl the mobile Web. However, there are significant crawling issues to resolve going forward, issues that are quite different from those addressed in traditional Web search. For example, even though the mobile Web is many orders of magnitude smaller than the traditional Web, crawling remains a challenge because of the transient, short-lived nature of mobile content. Furthermore, mobile pages are much smaller than their Web counterparts and thus there is less information available as a source of indexing. Careful authorship is likely to be much more important than on the traditional Web if search engine indices are to be accurate. Finally, user queries are often vague in Web search (average query size tends to be between 2-3 terms [8]) but this is likely to be exacerbated by the limited input capabilities of mobile devices. So mobile queries are likely to be even shorter and more ambiguous that their Web counterparts. These issues probably explain, at least in part, the poor relevance reported in our preliminary evaluation above.

In addition to these coverage and relevance issues, presentation and interface design becomes much more critical in mobile search than in traditional Web search. Our study indicates that the interface priorities of most mobile search engines facilitate the display of only 1 or 2 results per screen on a typical handset, and those that can display more results do so by sacrificing important contextual information that a user will likely need to make a judgement regarding result relevance. Our focus in this paper is on the mobile phone (smartphone) which fits approx. 7 lines of text or approx. 200 characters of text per screen, most of the search engines examined will allow for only one or two results per screen.

<table>
<thead>
<tr>
<th>Search Engine</th>
<th>Title</th>
<th>Snippet</th>
<th>URL</th>
<th>Whole Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>17</td>
<td>62</td>
<td>31</td>
<td>138</td>
</tr>
<tr>
<td>Mooobl</td>
<td>59</td>
<td>-</td>
<td>-</td>
<td>59</td>
</tr>
<tr>
<td>Click4WAP</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>64</td>
</tr>
<tr>
<td>Seek4Wap</td>
<td>7</td>
<td>18</td>
<td>-</td>
<td>26</td>
</tr>
<tr>
<td>WAPAll</td>
<td>7</td>
<td>42</td>
<td>-</td>
<td>50</td>
</tr>
<tr>
<td>WAPly</td>
<td>19</td>
<td>65</td>
<td>-</td>
<td>84</td>
</tr>
<tr>
<td>Ithaki</td>
<td>50</td>
<td>-</td>
<td>-</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 3: Average Length in Characters of each Search Result

4. INTELLIGENT RESULT GISTING

The objective of this work is to present and evaluate an alternative approach to search result gisting that enjoys the informativeness of snippet text while providing for a more economic use of limited screen real-estate. The core idea behind this approach is to replace result snippets with a much shorter text representation that is made up of the terms of related queries that have led to the selection of a particular result in the past. This approach has been made possible as a direct consequence of a community-based personal-ized meta-search engine called I-SPY, which records the queries and search results of different communities of users.

4.1 I-SPY & Collaborative Web Search

I-SPY is a community-based meta-search engine that provides its users with search results that are informed by the past search behaviour of a community of like-minded users. In essence I-SPY selectively re-ranks search results according to the learned preferences of a community of users, promoting those results that are likely to be relevant to the current query using a record of search behaviors carried out by the community. Specifically, I-SPY monitors users selections or hits and maintains a record of queries and result selections [15, 16]. Each time a user selects a result, p, for a query q, I-SPY updates a community profile to reflect this new selection. This community profile forms the basis of I-SPY’s relevance metric. It records all of the queries submitted and results selected by a particular community and the relevance of a page p to a query q for a community c is calculated as the probability that page p will be selected for query q. This probability is estimated as the proportion of times that p has been selected for q in c in the past. Further details can be found in [15].

I-SPY maintains a separate profile for each community of users; e.g., searches that originate on a motoring web site are kept separate from searches that originate from a wildlife web site. This separation of communities, coupled with I-SPY’s approach to relevance, allows I-SPY to predict that users of the motoring web site are more likely to be looking for sports car sites and users of the wildlife site are most likely to be looking for information on large cats for the same query, for example. In this way I-SPY can disambiguate effectively between vague queries and a range of user trials have shown how I-SPY can generate superior result rankings than leading search engines such as Google. For example, a recent study has shown how I-SPY, working with Google as its underlying search engine, can reduce the percentage of search failures and improve the positioning of relevant results when compared to Google [14].

4.2 Query Reuse for Result Gisting

One of I-SPY’s distinguishing features is its storage of past search session information such as the queries that have been submitted and the results that have been selected for these queries. The work of [1] proposed how this information could be leveraged for a novel form of query recommendation in which related queries are recommended alongside certain search results that have previously been selected. For each search result p_k, selected for some target query q_T, I-SPY can generate a set of related queries, q_1,...,q_m, which have previously led to the selection of p_k. Our idea for result gisting is based on a modification of this query recommendation technique. Specifically, we combine the terms used in these queries to provide a compact yet meaningful summary of the corresponding search result.

Figure 2 illustrates this idea in action. If we take, for example, the query Java we can see that the first search result, Java Sun Technology is associated with queries such as j2sdk1.5 and java tutorials. These ‘related queries’ help to inform the user about other contexts in which this result was selected and at the very least tell the user that the result was found to be relevant for users looking for the latest SDK and Java tutorials.

In previous preliminary evaluations [5, 6], we have considered whether these related queries can be used as an alternative to snippet-text to gist search results. The empirical evidence from both of these previous evaluations demonstrated that the related queries were as informative as snippet text and offered the potential for...
a significant space saving. However these evaluations were limited because they did not involve the judgments of real users in realistic search scenarios. Thus in this section we describe the results of a new live-user evaluation that focused on how informative users perceived these related queries to be as an alternative to search result snippets. But first, we will briefly review the previous offline results presented in [6] to provide a context for our latest study.

4.3 An Offline Evaluation

We evaluated the usefulness of related queries as an alternative to snippet-text for result-gisting using data from I-SPY search logs. The data for the evaluation consisted of 684 result-pages that were selected by searchers in response to more than 2,600 queries. Each of these pages had at least 2 related queries associated with it as well as a unit of snippet text (generated by Google). The primary goal of our offline evaluation was to determine how well these related queries represented the page in question, relative to its snippet text. To do this we supposed that the representativeness of a set of terms relative to some page could be measured by the position of the page in the result-list generated by some search engine when using these terms as a query. Hence, in our evaluation, for each target page we transformed its set of related queries and piece of snippet text into two new search engine queries; one based on the related query terms and one based on an equivalent number of snippet text terms. We then submitted the queries to HotBot and compared each query according to the rank of the target page in the corresponding result-list.

4.3.1 Query Generation

The most crucial part of the evaluation was the generation of the test queries. In all, six query generation strategies were tested: two that produced queries from the terms contained in the related queries for a page and four that used terms from the page’s snippet text. Strategy RQ1 produces a test query by concatenating the related query terms into a single query. Strategy RQ2 uses a similar approach but duplicate terms are removed.

The snippet text conversion was slightly more complicated. To generate the test queries we parsed the snippet text to remove stop-words and special characters and then selected terms from the remaining snippet text using four different strategies. In strategy S1 we select a random set of k terms, where k is the number of terms in the test query produced by RQ1. Strategy S2 selects the top k most common terms in the snippet text, where k is the number of terms in the test query produced by RQ2. For strategy S3 we again select a random set of k terms but this time k is the number of terms produced by RQ2. Finally, strategy S4 selects the top k most common terms in the snippet text, where k is the number of terms produced by RQ1.

4.3.2 Relevance Assessment

After submitting each test query to HotBot, we examined the top 500 HotBot results only and compared the position of p, the target result-page, in the result-lists produced for each test query. The higher p is in the result-list the more representative the test query must be as an indicator of p’s content and hence the more representative the related queries or snippet text. Along with this positional information we also examined the percentage of results matched by each test query strategy as well as the average length in number of terms of the related queries vs. snippet-based strategies.

4.3.3 Results

First we looked at the average position of each target page p in the result-lists produced by HotBot. Table 4 shows that both related query strategies, RQ1 and RQ2, perform very well. RQ2 locates p at an average position of 39 in the result-list compared to the best performing snippet-based strategy which locates p at an average position of 103. Note that if the target result cannot be found, p is given a default position value of 501 (because we are only examining the first 500 HotBot results).

<table>
<thead>
<tr>
<th>Test Query</th>
<th>RQ1</th>
<th>RQ2</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Position</td>
<td>46</td>
<td>39</td>
<td>144</td>
<td>118</td>
<td>180</td>
<td>103</td>
</tr>
</tbody>
</table>

Table 4: Average Position of p in Result-Lists

The poor performance of the snippet text strategies could have been due to frequent penalties being incurred when p was not present in the top 500 results. To understand this we examined the percentage of results found for each test query strategy in the top 500 HotBot results, see Table 5. The related query strategies succeed in producing result-lists that contain p for between 92% and 94% of queries, a significant improvement when compared to the snippet text strategies which return p for between 66% and 80% of the test queries.

<table>
<thead>
<tr>
<th>Test Query</th>
<th>RQ1</th>
<th>RQ2</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage (%</td>
<td>92</td>
<td>94</td>
<td>73</td>
<td>78</td>
<td>66</td>
<td>80</td>
</tr>
</tbody>
</table>

Table 5: Percentage Found in Top 500 HotBot Results

These results suggest that the terms contained in the related queries are more representative of the pages they refer to than an equivalent number of terms taken from the snippet text associated with these pages. They also suggest that by using related queries instead of snippet text we can achieve a significant saving in display-space. For example, as shown in Table 6, our related queries contained only 4 unique terms on average, compared to snippet texts with an average of 35 terms or 21 terms with stop-words removed.

<table>
<thead>
<tr>
<th>Test Query</th>
<th>RQ1</th>
<th>RQ2</th>
<th>Snippet</th>
<th>Parsed Snippet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Num Terms</td>
<td>6</td>
<td>4</td>
<td>35</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 6: Average Number of Terms in Test Queries

The results of the above evaluation suggest the use of related queries as an economical alternative to snippet text for result gisting. The terms contained within related queries appear to have the potential to better capture the essence of their associated pages than
the terms in the snippet texts, and so may serve to be a more informative gisting approach. These good terms have the advantage that they were used in situations where a given page was ultimately selected, the same is not true for snippet text terms. Moreover related query terms have been generated (in I-SPY) by a community of like-minded searchers, which should help to constrain the possible interpretations of possible query terms. Moreover, related query terms take up a small fraction of the screen space associated with the display of snippet text, which is a major advantage for the provision of mobile search.

Of course the limitation of the above study is obvious. It does not involve live-users during relevancy assessment and so it is not clear as to whether real users would likely behave in a similar way. In the following section we describe our live user study designed to make some progress in this regard. In particular it is designed to determine if real users accepted our related query interface as an effective interface for displaying search results on a mobile phone.

4.4 A Live User Evaluation

In this new study we asked users to evaluate three different interfaces for displaying search results on a mobile phone. The goal of the evaluation was to understand how users judged the informativeness space trade-off between snippet text and our related queries approach to result gisting. In brief, the evaluation asked a set of test users to evaluate three different result-list interfaces for a particular set of result pages and their related queries and snippet texts.

4.5 Phase 1 - Related Query Generation

The first part of this evaluation involved the generation of 3 different result-lists, for a set of 18 result pages, but each focusing on a different presentation style. The first style presented result title information only (Interface 1). The second presented title plus snippet text (Interface 2). And the third, used our new approach to gisting, presenting title plus a related query string (Interface 3).

For the third presentation style we needed a source of realistic related queries as the basis of our query string. To generate these related queries we asked 5 users to view a set of result pages and formulate a set of queries they would enter in a search engine if they were looking for the page in question. Each participant examined 18 pages relating to 6 different AI & Computer Science conferences including WWW, SIGIR and IJCAI; that is we presented 3 different web pages per conference. We asked the participants to open and view each web page and devise 5 different queries they would enter in a search engine if they were looking for the web page in question. These queries constituted the related query database and in this evaluation scenario corresponded to the sort of queries that a community of computer science students might enter when looking for conference information.

To generate the related query string for each result page, we extracted the top 3 most frequent queries entered for that page and then computed the union of these terms to produce a single related query string. For example, the top three terms entered by the participants for the IJCAI 2005 General Conference web page were, IJCAI 05, IJCAI Edinburgh and IJCAI 2005 Info. The combined query string was IJCAI 05 IJCAI Edinburgh IJCAI 2005 Info and the related query string we were left with after removing duplicate terms was IJCAI 05 Edinburgh 2005 Info.

4.6 Phase 2 - Interface Evaluation

During this part of the evaluation we presented our three different styles of result-list to 120 users from a computer science / IT background.

4.6.1 Methodology

Each participant was presented with 6 sets of our 3 different mobile phone interfaces with each of the 6 sets presenting the results of a different search for a particular computer science / AI conference; each set of results was presented using an interface designed for a Nokia series 60 WAP phone. And within each set we presented the result listing using title information only, title and snippet information, and title and related query string. The titles we used were extracted directly from each of the 18 result pages described earlier and the snippet text we used was generated by Google for each of the 18 result pages.

Each user was then asked to answer two questions for each of the 6 sets of results bearing in mind that as a searcher they might be interested in different types of information that was not declared as part of their original query; for example, the query WWW 2005 might be used by a searcher looking for the call for papers or registration information. The two questions presented to users were:

1. Which interface provides the most useful information about the search results returned?
2. Which style of presentation strikes the best balance between use of screen space and information conveyed?

Figures 3, 4 and 5 show some of the different interface sets presented to users of the study. The curly bracket to the left of the interface sets illustrates the approximate screen size of a typical XHTML-enabled mobile phone. We included this bracket to help users visualize how much information actually fits within a single screen on the mobile device. By examining the information within each of the different interface sets you can see that the information displayed to users is of varying quality. For example, figure 3 shows the three different interfaces for the WWW 2005 conference pages. In this case, each of the titles, snippet text and related queries are quite informative and easily distinguishable. However if we look at figure 4, IJCAI 2005, we see that all the title text is the same and in figure 5, ICCBR 2005, all of the snippet text is the same even though the results point to different pages. This is a common problem on both the World Wide Web and the Mobile Internet and is primarily caused by improper web authoring. It is especially important in these cases to provide some additional contextual information to help users understand and distinguish between relevant and irrelevant search results. Related queries can provide this additional context.

In total our questionnaire resulted in 112 completed users sessions. Each completed session involved the user selecting one in-
interface in answer to each of the above questions for the 6 sets of result-lists; thus, each session produced 12 user interface selections. These formed the basis of the results presented in the following section.

4.6.2 Results

Figure 6 shows the overall performance of each of the three interfaces. The overall performance of each interface refers to the percentage of user selections each interface received, averaged across both questions. We can see from the graph that interface 3, the related query interface, performs best overall, with 54% of the selections received for this interface across both questions. Interface 2 received 34% of the selections, while interface 1 performs worst with just 11% of user selections. This suggests that interface 3 might provide a suitable interface type in mobile search environments, striking a good balance between use of screen space and the quality of information displayed. Interface 1 performs worst overall thus confirming our prediction that providing just the title of each search result does not provide the user with enough information about the context of the result in question.

Figure 7 illustrates the performance of each of the interfaces for each of the two questions asked. On the issue of which interface provides the most useful information (question 1) there is a clear preference for interface 2 (title and snippet text), which received 60% of user votes. Interface 3 (title and related queries) attracted only 33% of the votes and interface 1 only 7% of the votes. This is not surprising and the results correlate precisely with the quantity of information provided with each search result (see Table 7). The content-rich snippets of text used by interface 2 certainly provide more information to the user about the result than the more economical related query strings used by interface 3. However, this evaluation question purposefully ignores the issue of the informativeness-space trade-off.

Question 2 focuses on this trade-off explicitly by asking users to evaluate which interface presents the best balance between information and space usage. And on this issue there is a strong preference for interface 3 (title and related query strings), which attracted over 75% of user preferences. In fact on this issue the traditional result-list presentation approach epitomised by interface 2 (title and snippet text) performs worst of all, attracting only 8% of user preferences. Indeed we find that interface 1 (title only) performs better, attracting 16% of votes, which is consistent with a preference (over title plus snippet text) for this style of result presentation format among the current set of WML search engines.

Table 7: Average Length of the Search Results in Characters

<table>
<thead>
<tr>
<th>Interface</th>
<th>Presentation Style</th>
<th>Result Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Title Only</td>
<td>28</td>
</tr>
<tr>
<td>2</td>
<td>Title + Snippet Text</td>
<td>167</td>
</tr>
<tr>
<td>3</td>
<td>Title + Related Queries</td>
<td>73</td>
</tr>
</tbody>
</table>
We carried out a two-way analysis of variance (ANOVA) to determine whether our results were statistically significant. The design was a 2 x 3 one with question (question 1 or question 2) and interface type (interface 1, interface 2 or interface 3) being the within-subject variables. The ANOVA test revealed a significant main effect of interface type \( F(2, 66) = 109.5, p < 0.001 \), and a highly significant interaction between the question and interface type \( F(2, 66) = 218.2, p < 0.001 \). We carried out Tukey’s post-hoc comparisons to determine if the interaction effects found were reliable, the results of which showed that there were reliable differences between all of the questions and all of the interfaces.

In summary, these results suggest that providing title information alone is not optimal; titles on their own lack sufficient detail to be truly informative for the average searcher. At the same time, title plus snippet information, while much more informative is not appropriate for mobile handsets because of its high space demands. The evidence points to interface 3, which combines title information with our novel related query strings, as providing a better balance between information and space usage that is suitable to unique characteristics of mobile devices; interface 3 requires less than half of the screen space of interface 2, for example (see Table 7).

5. CONCLUSIONS & FUTURE WORK

As the Mobile Internet continues to develop at a pace, mobile search is likely to become a more important way for users to access information, just as Web search is the primary mode of information access on the Web today. However, the limitations of modern mobile handsets introduces a number of crucial challenges when it comes to delivering useful and usable search engine services. For example, one of the main issues concerns the manner in which search results are displayed. In this paper we have argued that traditional presentation styles are not optimal through an extensive study of 7 existing mobile search engines. We have proposed using related queries as a more economical alternative to the use of snippet text for displaying search results and as a more informative alternative to displaying result titles alone. We have included the results of two separate evaluations including one live-user trial. These suggest two important conclusions: first, they indicate that related queries do provide an informative alternative to snippet text; second they also suggest that users judge the use of related queries to provide a better balance between informativeness and screen-space on mobile handsets.

The findings of our mobile search engine study and the results of our related query evaluations point to some very interesting avenues for future research. At present, we are pursuing a number of different areas relating to both mobile search and the Mobile Web in general. Regarding our work in the area of search result presentation on mobile devices, our evaluations to date have yielded very positive results but we are aware that our studies are limited in some regards. This is a work in progress and we understand that more quantitative evaluations are needed in order for us to obtain a more objective evaluation. Therefore, as our next step, we plan to carry out a quantitative user evaluation where we ask users to perform real search tasks in a more dynamic and interactive setting.

We also plan to explore our related query technique in more detail. In particular we want to add more intelligence into both the generation and display of the related query strings. We also want to investigate the use of other non-textual interface types for mobile search result presentation. Finally, in a separate but related area, we are exploring the dynamics of the Mobile Web, in particular the evolution of content in the mobile space. Early results indicate that the Mobile Web is highly dynamic, presenting a whole new set of challenges for the Mobile Internet community.

6. ACKNOWLEDGEMENTS

The authors would like to thank all of the volunteers who took part in our user evaluation. This material is based on works supported by the Science Foundation Ireland under Grant No. 03/I3/1361.

7. REFERENCES


**APPENDIX**

**A. LIST OF THE 20 QUERIES SUBMITTED TO THE MOBILE SEARCH ENGINES**

1. Jobs Ireland
2. Cheap Flights
3. Boston
4. Learn Italian
5. Premiership Results
6. Tour de France
7. BBC TV Listings
8. News
9. Weather Forecast
10. GI Diet
11. Britney Spears
12. Buy iPod
13. Music Downloads
14. Harry Potter
15. Java Tutorial
16. David Beckham
17. Big Brother
18. Banking 365
19. Recipes
20. Ringtones
Use of RSS feeds for Content Adaptation in Mobile Web Browsing

Alexander Blekas
University of Patras
Computer Engineering & Informatics
Department
26500 Patras, Greece
Computer Technology Institute
26500 Patras, Greece
+306932321212
mplekas@ceid.upatras.gr

John Garofalakis
University of Patras
Computer Engineering & Informatics
Department
26500 Patras, Greece
Computer Technology Institute
26500 Patras, Greece
+302610997866
garofala@ceid.upatras.gr

Vasilios Stefanis
University of Patras
Computer Engineering & Informatics
Department
26500 Patras, Greece
Computer Technology Institute
26500 Patras, Greece
+306973281930
stefanis@ceid.upatras.gr

ABSTRACT

While mobile phones are becoming more popular, wireless communication vendors and device manufacturers are seeking new applications for their products. Access to the large corpus of Internet information is a very prominent field, however the technical limitations of mobile devices pose many challenges. Browsing the Internet using a mobile phone is a large scientific and cultural challenge. Web content must be adapted before it can be accessed by a mobile browser. In this work we build on the proxy server solution to present a new technique that uses Really Simple Syndication (RSS) feeds for the adaptation of web content for use in mobile phones. This technique is based in concrete design guidelines and supports different viewing modes. Experimentation shows a significant decrease in the transformed content of about 80% in size facilitating cost-effective web browsing.

Categories and Subject Descriptors
H.4.3 [Communications Applications]: Communications Applications - Information browsers; H.5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces

General Terms

Keywords
Mobile devices, RSS feeds, content adaptation, mobile browsing, web browsing.

1. INTRODUCTION

During the past decade the world has witnessed a revolution in the field of wireless networks and mobile devices. The term ‘mobile device’ refers to a device specially designed for synchronous and asynchronous communication while the user is on the move. It includes a wide variety of appliances such as PDAs and mobile/smart phones. Tablet PCs are also included in the category of mobile devices, however they are bigger in size and more costly to acquire. Mobile phones are by far the most popular mobile devices. Among the many facilities they provide is access to the Internet, however browsing using a mobile phone is not as easy as browsing using a common desktop PC. Screen size and resolution, number of supported colors, entering method, computation power, memory size, rate of data transfer and energy required for proper functionality are the main limitations for using a mobile device to browse through the Internet [7]. On the other hand, users demand access to information anytime, anywhere. Since within a few years, most of the devices accessing the Web will be mobile, there is a need for developing methods and techniques that will allow satisfactory web browsing using mobile devices.

The problem of web browsing through a mobile phone has attracted much attention by the research community and the industry alike. One of the proposed solutions is to use a proxy server which adapts the web content for mobile devices on-the-fly and returns the result to the mobile user. In this paper we build on the proxy server idea to propose a novel solution that makes use of feeds in a web site in order to achieve better content adaptation. RSS, an XML-like notation, is used mainly in large information-centric web sites to summarize web site information. By taking advantage of this information, it is possible to transform the content of the regular web site so that it can be accessed by a mobile phone. The new technique is able to scan any web site that includes RSS, removes ‘unwanted’ information and presents a set of different packed versions which are both comprehensive to the user and have a small size. The technique is available as an online system that works even for web sites without RSS feeds. Experimental results show a significant reduction of about 80% in the size of the web page processed using this method. Although this technique also removes multimedia information (images, video, animation) its value relies on the very small size of the adapted content, making it accessible even by mobile phones of limited capacity through heavy-loaded or slow wireless networks.

This work is structured as follows. In section 2, a review of content adaptation methods for mobile devices is presented. In section 3, RSS feeds are described while in section 4, the design principles for mobile applications are discussed. Finally, section...
5 provides a complete presentation of the application that we have developed and in section 6 research suggestions and future work are presented.

2. CONTENT ADAPTATION FOR USE IN MOBILE DEVICES

There are four general approaches for adapting web content for small screen devices: device-specific authoring, multi-device authoring, automatic re-authoring, and client-side navigation [1]. The first two approaches obtain high quality results by authoring device-specific web content. Having two versions of the same web content (one for regular and one for mobile users) is costly and the vast majority of web sites does not use it. This makes most of the Internet actually inaccessible for mobile phone users. On the other hand, automatic re-authoring and client-side navigation techniques do not require the collaboration of page authors and are therefore more widely applicable.

Research prototypes that use automatic re-authoring fall into two main categories: page reformattting and page scaling. An example of techniques based on page reformatting is the Power Browser [3], where images and white space are removed, and the WEST browser [2], which uses flip zooming, a visualization technique that breaks pages into screen-sized tiles and presents them as a stack. Difficulties with recognizing layout and leveraging the desktop browsing experience are common to all these approaches, since they all have an impact on the page layout.

Other techniques based on page reformatting include scaling the page and return the result to mobile device as a thumbnail. An example of this approach is the SmartView system [8]. SmartView creates an image map thumbnail of the page based on its semantic content. The user is able to zoom to the preferred block of the thumbnail and read the content of the web page. The main problem with this approach is the size of the thumbnail; the screen size of mobile devices does not allow the text to be readable in many cases. Another approach tries to solve this problem by combining the previous method with text summarization techniques [6]. The main idea behind this approach is for the first level of the thumbnail to contain only some keywords from every text unit and not the full text of the web pages. When the user zooms to a specific area she/he is able to read the full text contained there. These methods are mainly intended for PDA users and generally for mobile devices with relatively large screens. Small mobile phone screens are not able to present a satisfactory view for the original thumbnail and as such they are not recommended to be used with the techniques mentioned above. The typical screen resolution of a mobile phone is between 128x128 and 176x220 pixels. A web page, with rich content, in a 1024X768 pixels resolution usually requires scrolling even for a 17 inch monitor. It is obvious that the thumbnail for the mobile phone will be so small that the user will not be able to read text, especially if he/she is a first time visitor (and thus unfamiliar with the web site content organization). Finally, using images obviously increases user satisfaction but requires significant computational and power resources from the mobile device.

In client side browsing, the whole content is delivered to the browser and then the browser decides how to present it. For example, the Opera Browser (opera.com) applies the Small Screen Rendering™ technique where the content is presented in a single column, so horizontal scrolling is no more necessary. All client side browsing methods require the full code of a web page to be downloaded to the mobile device. This presupposes the existence of enough main memory to the mobile devices for storing and running the page code. Furthermore, the user is usually billed according the amount of downloaded data, so downloading large content is costly.

3. RSS FEEDS

Our approach is associated with the automatic re-authoring combined with RSS feeds with the purpose to improve web browsing from mobile devices. RSS is a family of XML file formats which summarizes web site information. It is mainly used by news sites and, generally, sites that their content changes very often, for example a weblog (a web-based publication consisting primarily of periodic articles, normally in reverse chronological order). Currently, most web sites use one of the three different RSS versions; version 0.91 [9], 1.0 [10] and 2.0 [11]. The number of web sites that are RSS-enabled is increasing geometrically. According to feedster.com (a search and indexing engine for RSS feeds) more than sixteen million RSS feeds are available in the web including more than 75,000 professionally published sources such as the BBC, CNN and The New York Times [5].

<table>
<thead>
<tr>
<th>Table 1. RSS feeds tags</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RSS tag</strong></td>
</tr>
<tr>
<td>&lt;rss&gt;</td>
</tr>
<tr>
<td>&lt;item&gt;</td>
</tr>
<tr>
<td>&lt;ttl&gt;</td>
</tr>
<tr>
<td>&lt;title&gt;</td>
</tr>
<tr>
<td>&lt;description&gt;</td>
</tr>
<tr>
<td>&lt;link&gt;</td>
</tr>
</tbody>
</table>

Table 1 summarizes the functionality of the main RSS tags. Every RSS file has the root element `<rss>` where the version of the RSS file is defined. The only child of the `<rss>` element is the element `<channel>`. The element `<channel>` may contain any number of elements `<item>`, the main element in a RSS file. Every element `<item>` always contains the tags `<title>`, `<link>` and `<description>`. Beside the above tags, there are other tags supporting more functionalities, depending on the RSS version.

4. DESIGN PRINCIPLES FOR MOBILE APPLICATIONS

The limitations of mobile devices obligate developers to be very careful when designing applications. In order to design the RSS-enabled content adaptation technique, we had to follow certain guidelines [13], [14].

The interface of the application should be based on a consistent and easy to learn navigation model. Mobile users’ requirements are different from PC users’. Mobile users do not browse the Web in the classic way but they demand quick access to specific information. Therefore developers of mobile applications should avoid content with a lot of graphics, animations and different colors. In any case they should present the right information at the right time.
Another point that we have to take care is text inputting. For the majority of mobile users, using a keyboard of a mobile phone, even with T9 [12] support, is a slow process. Interaction with the application requires the use of checkboxes, lists and drop down menus. Furthermore, if a page contains forms it should present the forms in one screen so that the user is able to confirm data input without returning to previous screens.

Developers also have to pay attention to navigation hierarchy. Finding services should require only a few clicks to hyperlinks. If the navigation model is too complex (e.g. a deep tree of hyperlinks), the user may get lost and frustrated. Developers also need to avoid creating buttons for functions already implemented by the system. For example, if we create hyperlinks for going to the previous page in a mobile web application, a function already implemented by the browser, valuable space is misused in an already small screen; on the other hand, special buttons are needed so that the user can browse back to the main page or other important pages in a click.

For mobile web applications, grouping the hyperlinks effects the usability of the application. For example, if we have to present twenty relative hyperlinks, it is preferred to present them sorted in the same long page and not to four small pages. In every page the important content and the navigation bar if it exists, has to be located at the top of the page. So, if the user browses to a page which has no important content according to his/her interests, she/he can continue to the next page without vertical scrolling. Finally, pages that welcome the user to the application are unnecessary (and some times frustrating) for mobile users.

A designer of mobile web applications must have in mind that the applications refer to devices with small screen but at the same time these applications should be also user-centric; reaching a consensus there is difficult. Every page has to contain the main content starting from the beginning. The need for vertical scrolling should be kept to a minimum; horizontal scrolling is generally not recommended at all. Every modified web page should contain the tag <title> so that the user may read a short description of the page at a glance. The browsers usually put the content of the <title> tag to a visible place. The title should have a length of fourteen characters, at the most.

The designer has to use the different types of text alignment (right, center, left) in a way that facilitates information grouping and makes the text more readable. Hyperbolic formation of the text like bold or underline letters may bring the opposite results; finally, it is proposed to replace complex and big words because they make sentences longer and the user may have to scroll in order to understand the meaning.

Color usage is also important. Using colors obviously gives a pleasant and friendly interface, but a too colored screen confuses. All the pages of the application must have the same colors so the user can feel that he/she is navigating in the same environment. Furthermore, it is proposed to avoid sentences that refer to colors by their name, such as “Click to the purple link to continue”, because some users may have devices with screens that support few colors.

The use of images in Internet applications is common. Nevertheless, using images in mobile web applications significantly increases download and response time and thus, usage cost. If the browser supports incremental page rendering, a technique where the browser incrementally presents a web page before the whole content has been downloaded, the loading of images in a later time needs more computational power and confuses the users. For the above mentioned reasons we argue that the use of images should be kept to a minimum.

5. AN RSS – BASED CONTENT ADAPTATION BROWSING SYSTEM

5.1 General Architecture

A standard way of processing web pages for viewing on small screen devices is through a proxy server that transforms pages on-the-fly. A proxy server is a program that receives web page requests (here from mobile devices), loads the respective pages, converts them, and serves them to the devices that requested them. In this way the proxy server, which usually runs on powerful servers, unleashes mobile devices from computational needs.

Figure 1 depicts the functional architecture of the system. If the system detects RSS feeds in a web page, it is proposed to the user to use the RSS feeds for the browsing. If the user agrees and chooses one of the available RSS feeds, the system reads every <title>, <link> and <description> element from the feed, adds the application’s menu and returns the XHTML – MP content to the user. If the page has no RSS feeds or the user does not want to use them, the proxy servers adapts the web content for mobile devices, adds the application’s menu and returns the XHTML – MP content to the user as described previously. The technique used to detect and exploit the RSS feeds is presented in the following section.

![Figure 1. The functionality of the general RSS adaptation mechanism.](image-url)
The RSS-feed adaptation technique

There is no need for configuring the device prior to the use of the application. The user has to start the browser of the device and provide the url of the server that hosts the proxy server application. The content that will be returned to the user is the main XHTML – MP page of our application, total size 0.65 KB (figure 2). In the text box of the main page, the user provides the url of the web site he/she desires to browse. This is the only part where the application needs text inputting. After this step, the user uses his device browser’s interface. When the user hits the “GO” button, the proxy server creates a connection with the given url, gets the html code, transforms it and returns the new code to the mobile device. Details like the “http://” string in the url does not concern the user, since we want to achieve the minimum keyboard usage. Also, if the web server redirects the original url (code 302 at HTTP response) the application finds and follows the new url without disturbing the user.

![Figure 2. Main page of the application.](image)

The proxy server has now saved the html code of the web site the user had requested. In this point, a parsing algorithm remove tags that are not consistent with the design guidelines of section 4. Those tags are <script>, <noscript>, <style>, <link>, <iframe>, <object> and <embed>. The algorithm also removes the comments from the html code. Comments do not affect the final result because browsers ignore them. But if the final code includes comments, the mobile user pays for useless data. Forms are also removed. This action decreases the usability of the application in many sites since it suspends the use of searching, login, on line transactions and generally web applications that involve interaction of this type with the user. However, the purpose of the system is focused on cost effective browsing and not facilitating more complicated on-line transactions so, in our view, this is acceptable. Furthermore, the majority of mobile users browse the Web for news and entertainment. Therefore, if one user is interested in mobile transactions for example, he will use the special application that the bank may offer and not a proxy server of this type. Furthermore, our application focuses on delivering to mobile users the content of a web page as good as possible and not to implement the functionalities of a random web site from mobile devices.

The use of tables in the web is a common practice because they summarise content elegantly. However, in mobile devices the use of tables is a problem because of the lack of space on the screen. The application solves this problem by removing tables and presenting their content in a new line for every cell. If nested tables in a cell are used, the procedure is applied recursively.

![Figure 3. A web page with nested tables.](image)

Figure 3 shows a 2X2 html table as presented by a web browser. A 2X2 table is nested in one cell of another table. If the user requests the above page through the proxy server, the result is presented in figure 4.

![Figure 4. Nested tables as showed from the application.](image)

For further processing of the code, we need some knowledge about its structure (for example if one tag is nested in another). For this purpose, we use the functions of the DOM library of PHP. With DOM functions we create the DOM tree of the web page. The DOM interface of PHP follows the DOM Level 2 standard [4].

The next step is to remove the images from the web page. From the DOM tree that has been created, we locate all <img> tags by using the getElementsByTagName PHP function. If the image is at the same time a hyperlink to another page, removal of the image must be done in such a way that hyperlink information is not lost. To ensure this, we check the DOM tree. If node <a> has a node <img> as a child, in the original place of the image, a text hyperlink is created. The text of the hyperlink is the string of the alt attribute of the original <img> tag. If the alt attribute is missing or is empty, the new hyperlink contains the text “IMG:” followed by the url of the hyperlink. To read the value of the alt attribute
we use the getAttribute attribute of the class DOMElement. When the new hyperlink has been created, the <img> tag is removed from the code using the removeChild function.

One of the design goals of our approach is also to ensure transparency. When a page has been served to the user, she/he can browse to new pages by following the hyperlinks or by giving a new url to the main page of the application. So, the destination of each hyperlink in the transformed web page must change in a way that requests are redirected through the proxy. For example, if a hyperlink points to http://www.acm.org the new hyperlink must be changed to the following format:


Of course, a hyperlink may point to things other that web sites. For example, a hyperlink may point to a file that is hosted at the same server using a related path such as <a href="../test.html">Test page</a>. In this case, the hyperlink points to the file test.html which is located at the directory which is a level higher from the current one. The proxy server that we have developed checks all those cases.

When the parsing algorithm has finished with the content adaptation process, it checks the size of the new page. If the size is bigger than 10KB then it sends, through the proxy, the content in sub-pages with size of about 10KB. If the point where the page has to split is in the middle of a paragraph, the splitting point is moved so that the whole paragraph appears in the same sub-page. The 10KB limit may seem small for new mobile devices however, with this limit we are sure that there will be no memory problems with any device that supports WAP 2.0. One other reason for creating sub-pages is that larger pages may lead to longer response times. Furthermore, a page may contain content which is useless to the user or may be an intermediate page for another destination. In this case, the user may ignore this page having paid only for the 10 KB of the first sub-page. The 10 KB limit can be easily changed or cancelled by the administrator of the proxy server.

At the end of each page or sub-page a menu is added that allows access to the other sub-pages that may exist or returns to the main page of the application to view only the links or only the text of a page. These functionalities are described in detail later in this section. The menu is at the end of each page but is quickly accessible from the user by pressing the 0 key in the keyboard. This function is implemented using the attribute accesskey of the tag <a>.

When the user is navigating in a site she/he usually browses through hyperlinks ignoring most of the content, that is until the proper information is found. To speed up this process the application provides the user with the option to view only the hyperlinks of a page. Similarly, the user has the option to view only the text of a web page. This functionality may be useful if a page contains for example, an article. Therefore, text is the most important part in this case. Of course, the user may switch from one mode to another or return to the original viewing mode.

If the user requests a page that includes RSS feeds then the application does not present the content of the page immediately but recommends the user to browse the site through the information that the RSS feed contains. By default, the RSS feed includes all the latest and interesting information that the site contains. In order to discover if a page contains RSS feeds, the parser checks all the <link> tags before they are removed from the content adaptation process that we described previously. If the attribute type of the tag <link> has the value "application/rss+xml", then the attribute href of the same tag contains the location of the RSS feed’s XML file. The parser returns the title of the feed as it is defined by the attribute title of the tag <link>. If a page contains more than one RSS feeds, the parser returns all the feeds in the order that they are detected in the web page code. The application recommends the user to continue browsing through the available RSS feeds. However, the user can choose to browse the site through the process that we have described previously. Table 2 contains the PHP code that locates the RSS feeds to the HTML code.

### Table 2. PHP code for RSS feeds detection

```php
$rss_check = new DOMDocument();
$rss_check->loadHTML($code);
$tags = $rss_check->getElementsByTagName('link');
$rss_count = 0;
foreach ($tags as $tag)
{
    $type = $tag->getAttribute('type');
    if ($type == 'application/rss+xml')
    {
        $rss_urls[$rss_count] = $tag->getAttribute('href');
        $rss_titles[$rss_count] = $tag->getAttribute('title');
        if (strpos($rss_urls[$rss_count], 'http') !== FALSE)
            $rss_urls[$rss_count] = $url.$rss_urls[$rss_count];
        $rss_count++;
    }
}
```

If the user selects to browse based on RSS feeds as recommended, an XML parser is used to read the RSS feed. For the implementation of the RSS parser we use the xml_parse_create function from the XML library of PHP. The XML parser works with all three versions of RSS feeds, which are versions 0.91, 1.0 and 2.0. From every RSS feed, the application returns the title, the description and the link for every element <item> of the feed. Just like in every page, users may only view the links or the text of the page. Table 3 presents the code that implements the RSS feed reader.

### Table 3. PHP code for the RSS reader

```php
$xml_parser = xml_parse_create();
$xml_set_element_handler($xml_parser, "startElement", "startElement");
$xml_set_character_data_handler($xml_parser, "characterData");

function startElement($parser, $name, $attrs) {
    global $insideitem, $tag, $title, $description, $link;
    if ($insideitem)
    {  
        $tag = $name;
    }
    elseif ($name == "ITEM")
    {
        $insideitem = true;
    }
    function endElement($parser, $name) 
```
We have developed our server using PHP (php.net). The only requirements for the application is a web server with PHP 5 support or newer. All the functions that have been used are included to the default installation of PHP. For our tests we have used the Apache 1.3.33 as a web server and the PHP 5.0.4.

From the client side, the only requirement is a WAP 2.0 compatible browser, a browser that most of the devices today have. The pages of the application are XHTML-MP valid, so we are sure that the content the proxy server sends is presented correct in every category of mobile devices.

5.3 Examples of usage

In the next two paragraphs we present two examples of how our system works with two popular sites. The first site contains RSS feeds and the second does not.

5.3.1 The site cnn.com

In figure 5, we can see the main page of cnn.com as viewed from a web browser. The main page contains RSS feeds, that we have already loaded at the “Bookmarks” menu. The total size of the page is 313 KB. If we try to browse the same site from our application the result is shown in figure 6. At first, the application locates the existence of two RSS feeds, “CNN – Top Stories” and “CNN – Recent Stories”. The user may ignore the RSS feeds by choosing “Normal Page”. In our example we have selected the first RSS feed which is loaded to a new page. As we can see, the user has the most important content of cnn.com in about 12 KB.

5.3.2 The site acm.org

As previously, figure 7 shows the main page of acm.org as presented to a web browser.
The acm.org does not contain RSS feeds. Therefore the application follows the content adaptation procedure. The original page is 100 KB. The same page as generated from the application, which is at most 20 KB, is shown in figure 8. The mobile user may access the same content with a minimum benefit of 80% concerning downloaded data. The last observation was confirmed by other experiments we made on different sites.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented a method that allows cost-effective web browsing for users of mobile devices. This method exploits mainly the existence of RSS feeds, metadata that summarise information. We concluded that the existence of RSS feeds improves significantly the content presented to the mobile user decreasing at the same time the size and access cost.

The final result in a random site, through our proxy server, depends entirely on its content. The technique is not suitable for accessing on-line transaction systems since forms are omitted. Finally, the fact that the application removes all the images, may not take full advantage of the capabilities of newest mobile devices. All the above matters will be included in our future work.

7. REFERENCES


[14] XHTML Mobile Profile and CSS Reference, 2003, Openwave Systems
A Web Browsing System based on Adaptive Presentation of Web Contents for Cellular Phones

Yuki ARASE† Takuya MAEKAWA† Takahiro HARA† Toshiaki UEMUKAI‡ Shojiro NISHIO†
†Dept. of Multimedia Engineering, Grad. Sch. of Information Science and Tech., Osaka Univ.
‡KDDI R&D Laboratories Inc.
{arase.yuki,tmaekawa,hara,nishio}@ist.osaka-u.ac.jp, to-uemukai@kddilabs.jp

ABSTRACT
Cellular phones have already been widely used to access the Web. However, most existing Web pages are designed for desktop PCs, and thus, it is inconvenient to browse these large Web pages on a cellular phone with a small screen and poor interfaces. Users who browse a Web page on a cellular phone have to scroll the whole page to find an objective content, and then, have to scroll within the content in detail to get useful information. In this paper, we propose a novel browsing system to break off these burdensome operations by adaptively presenting Web contents according to their characteristics.

Categories and Subject Descriptors
H.4.3 [Communications Applications]: Information browsers; H.5.2 [User Interfaces]: User-centered design—Screen design

General Terms
Human Factors

Keywords
Cellular phone, Web browsing, Overview, Adaptive presentation

1. INTRODUCTION
Cellular phones have been widely used to access the Web due to their progress in processing and communication facilities. However, most existing Web pages are designed for desktop PCs, and thus, it is inconvenient to browse such large pages by using cellular phones. Since cellular phones only have a small display and poor interfaces, users have to perform numerous operations for scrolling a whole page.

Generally, a Web page is composed of a large number of different components[1, 3, 5], each of which is an information block such as a site directory, news, and a search form in the top page of a portal site. Figure 1 shows an example, where each block enclosed with frame is a component. Due to such a structure in Web page, users usually browse a Web page in the following two steps. First, a user looks over a page to find an objective component, then reads information within the component in detail. Because of the small display of a cellular phone, the user has to scroll the whole Web page for a long time to find the objective component. Furthermore, the user has to scroll within the component to read the information in detail. To provide comfortable Web browsing using cellular phones, these burdensome scroll operations have to be reduced. In [2], Chen et al. proposed a system that provides users an overview, which is a scale-down image of a Web page. When a user selects a content which he/she wants to read in an overview, the area around the selected content is zoomed in. By doing so, the user can find an objective component from the page with fewer scroll operations. However, the system proposed in [2] did not focus on reducing complicated scroll operations to read within components. Besides, users’ behaviors in reading Web components differ depending on the components’ characteristics. Thus, it is effective to present each component adapted to the users’ reading behaviors.

In this paper, we describe design and implementation of a Web browsing system which adaptively presents Web pages according to their characteristics to reduce users’ scroll operations. Our system displays the overview of a page so as to reduce time-consuming scroll operations to read an objective component from the page. In addition, this system presents the component adapted to its characteristics so as to reduce burdensome scroll operations to read the component.

The reminder of the paper is organized as follows: In section 2, we describe the components’ classification and a preliminary experiment. Section 3 describes the design and implementation of our system. In section 4, we further discuss our proposed system. Finally, we give conclusion and remarks about the future work in section 5.
2. COMPONENT CLASSIFICATION AND PRELIMINARY EXPERIMENT

As described in section 1, Web pages are composed of various components. In this section, we explain components’ classification and a preliminary experiment to determine an appropriate presentation for each component class.

2.1 Components classification

We consider that components have common characteristics according to their contents. We have checked fifteen typical Web sites such as portal and news sites and found that components are classified into the following six categories according to their contents:

- **Text**: contains only texts.
- **Text&Image**: contains both texts and images associated with the texts. Figure 2(a) shows an example.
- **Image**: contains only images.
- **Link&Image**: contains both a set of links and images. Figure 2(b) shows an example.
- **Vertical link set**: contains a set of vertically listed links. Figure 2(c) shows an example.
- **Vertical&horizontal link set**: contains a set of both vertically and horizontally listed links. Figure 2(d) shows an example.

2.2 Reading style of each component class

Users’ reading behaviors change according to components’ class, which we described in subsection 2.1. For example, a user thoroughly reads each sentence in a “Text” component, while he/she looks at images as well as reading texts in a “Text&Image” component. Besides, a user searches and selects an objective link or a link of some interest in a “Vertical link set” or “Vertical&horizontal link set” component.

By adaptively presenting a component according to the users’ reading behavior for each component class, burdensome scroll operations can be reduced.

2.3 Preliminary experiment

We conducted a preliminary experiment to determine how to present a component according to its belonging class and characteristics. We consider auto-scrolling as an effective approach because it can reduce users’ burdensome scroll operations within a component. In this experiment, we examined the effectiveness of auto-scrolling and its appropriate setting.

2.3.1 Experimental setting

Sixteen subjects were asked to read six components using auto-scrolling, where each component belongs to each of the six component classes described in subsection 2.1. Our experimental system determines a scroll path for each component based on the component’s shape. Specifically, when the component’s height is higher than that of the cellular phone’s display and the component’s width is narrower than that of the display, the scroll path is set in vertical direction as shown in Figure 3(a). On the contrary, when the component’s height is lower than that of the display and the component’s width is broader than that of the display, the scroll path is set in horizontal direction as shown in Figure 3(b). When both the component’s height and width are larger than those of the display, the scroll path is set in zigzag as shown in Figure 3(c). Here in Figure 3, a rectangle with solid line represents a component’s shape, a rectangle with dotted line represents a cellular phone’s display, and an arrowhead represents a scroll path with direction. In this experiment, we examined nine patterns for auto-scrolling with different speed and zoom ratio, where the subjects selected the best pattern for auto-scrolling for each component class.

2.3.2 Experimental result

We show the result of the preliminary experiment. Due to the limitation of space, we describe only a short summary. From the result, it is shown that auto-scrolling is not suitable for “Text” and “Text&Image” components, because the speed to read sentences differs among individuals. It is also shown that subjects who are reading a “Text&Image” component by auto-scrolling could not look at images within the component when they wanted. On the other hand, we confirmed that auto-scrolling is suitable for “Image”, “Link&Image”, “Vertical link set”, and “Vertical&horizontal link set” components. However, some subjects said that it was hard to read these components if they were presented in zigzag. For “Image” components, most subjects evaluated that pattern 4 is the best, while for “Link&Image”, “Vertical link set”, and “Vertical&horizontal link set” components, pattern 8 is the best.
link set” components (in the following, we call them “Link set” components), subjects’ evaluations are split across, some thought pattern 1 the best and the others thought pattern 4 the best. This seems due to differences in the subjects’ style for searching links. Subjects who liked pattern 1 thoroughly read each text associated with a link, and the others skimmed through the component.

3. SYSTEM DESIGN AND IMPLEMENTATION

Based on the result of the preliminary experiment, we designed and implemented a system that offers adaptive component presentation in a Server/Client architecture. The server was implemented on a PC with Windows XP, coded with Visual C# and PHP (Hypertext preprocessor). The client was implemented on a cellular phone (SH900i), which is equipped with i-appli developed by NTT Docomo, coded with Java. Within the 240 \( ^{\times} 320 \) [pix] display of the cellular phone, i-appli can only use a 240 \( ^{\times} 240 \) [pix] area.

Figure 4 shows the design of our system. In response to the client’s request, the server receives the corresponding Web page from a WWW server, and makes a screenshot of the page. Then, the server extracts components from the page, determines the components’ classes, and then, sends the screenshot, information on the components’ classes, and information on the components (the location, height, and width) to the client. The client displays the overview of the requested page, which is created from the screenshot and the information on the components received from the server. When the user selects a component on the overview, the client performs an adaptive presentation of the component. In the following, we describe the details of our implemented system.

3.1 Overview

A user can select a component which he/she wants to read on the overview of a Web page. An overview is a scale-down page that fits the display size of the user’s cellular phone (Figure 5(a)). The component that the user is watching with attention, i.e., that the pointer is currently focused on, is automatically zoomed in. The user can further zoom in the component by pressing a certain key of his/her cellular phone (Figure 5(b)). In addition, the system offers users “Jump” function. The user can jump to the top of the next sentence by pressing a key of the cellular phone when he/she finishes reading the current sentence.

3.2 Adaptive presentation

3.2.1 “Text” component

In our system, users read “Text” components by scrolling manually, because it has been shown from the preliminary experiment that the speed to read sentences differs among individuals. Figure 6(a) shows a screenshot of a cellular phone displaying a “Text” component, where the arrowhead in the center is a pointer. The shape of the pointer changes when the pointer is on a letter string of a link as shown in Figure 6(b), where the user can select the link. In addition, the system offers users “Jump” function. The user can jump to the top of the next sentence by pressing a key of the cellular phone when he/she finishes reading the current sentence.

3.2.2 “Image” component

An “Image” component is presented by auto-scrolling in speed of 50 [pix/sec] which is determined by the preliminary experiment. The preliminary experiment also showed that auto-scrolling in zigzag is hard for users to follow. Therefore, in our system, if the size of a component is larger than that of the cellular phone’s display, the component is zoomed out until it’s height or width fits the display’s height or width. In addition, users can stop auto-scrolling, then he/she scrolls manually, and can select a link in the same way as “Text” components.

3.2.3 “Text&Image” component

The preliminary experiment showed that users cannot read “Text&Image” components comfortably by auto-scrolling because they cannot watch images within the components at
the time they wish. Therefore, in our system, users read sentences within “Text&Image” components by scrolling manually, and they can switch over images within the components by pressing a key when he/she wants to watch them. That is, the users can switch over the images and sentences in rotation by pressing the key consecutively. By doing so, the users can watch images associated with sentences when they wish, and then can go back to the sentences. In addition, the system offers “Jump” function, and users can select a link in the same way as “Text” components.

3.2.4 “Link set” component

“Link set (Link&Image, Vertical link set, and Vertical&horizontal link set)” components are presented by auto-scrolling because the effectiveness was confirmed by the preliminary experiment. In doing so, the system determines the speed and zoom ratio of auto-scrolling according to contents of the component. First, the speed is determined by using the following formula:

\[ \text{Speed}^\prime \text{[pix/sec]} = \frac{\alpha}{\text{ID} \cdot \text{Breadth}} \]  

(1)

Here, \( \alpha \) [characters/sec] is the number of characters which human can recognize per one second in auto-scrolling, \( \text{ID} \) (information density) [characters/pix\(^2\)] is the number of characters per unit area, i.e., 1 [pix\(^2\)], supposing that all the characters within the component are uniformly distributed. \( \text{Breadth} \) [pix] is the width (height) of the component in the orthogonal direction with that of auto-scrolling, represented by the following formula:

\[ \text{Breadth} = \begin{cases} \text{component’s width} & \text{(vertical scrolling)} \\ \text{component’s height} & \text{(horizontal scrolling)}. \end{cases} \]

Consequently, \( \alpha \) is given by the following formula:

\[ \alpha = \text{Speed} \cdot \text{ID} \cdot \text{Breadth}. \]

We determined the value of \( \alpha \) respectively for each of “Link set” components, using \( \text{ID}, \text{Breadth}, \) and auto-scrolling speed (20 [pix/sec] or 50 [pix/sec], which the users liked in the preliminary experiment) which were obtained by the preliminary experiment. Then, by using these parameters’ values, the auto-scrolling speed is calculated by using formula (1). The preliminary experiment also showed that there are two different styles for searching an objective link; thoroughly reading link texts and skimming through the component, and these styles require different auto-scrolling speeds. Accordingly, a user should select an appropriate value of \( \alpha \) adapting to his/her reading style so that he/she can set an appropriate scrolling speed.

Next, the path and zoom ratio of auto-scrolling are determined as follows. In our system, if the width (height) of the component is larger than the height (width), the path is set in horizontal (vertical) direction. If the size of component is larger than the cellular phone’s display, the component is zoomed out so as to avoid auto-scrolling in zigzag.

In addition, the system offers functions of rewind and fast-forward. Moreover, users can stop auto-scrolling and scroll manually anytime they want, and then can select a link in the same way as “Text” components.

4. DISCUSSION

4.1 Path selection for auto-scrolling

In our current implementation, the path of auto-scrolling is determined by comparing the width with the height of the component. This is effective when auto-scrolling is performed in one direction, vertical or horizontal direction, while there is a serious problem for auto-scrolling in zigzag. Specifically, the information within the component is sometimes presented by snatches in auto-scrolling in zigzag. For example, let us suppose a component that includes a large number of links with letter strings enumerated in vertical direction and the length of the strings are longer than the width of the cellular phone’s display. This component is presented by auto-scrolling in vertical direction, however, the letter strings of the links are presented by snatches. Thus, our system should be extended to determine a path of auto-scrolling considering the direction and lengths of letter strings of links, as well as the existence of images and the width and height of a component.

4.2 Decision of component’s class

Our system sometimes misjudged the component’s class, for a “Vertical&horizontal link set” component as “Link&Image”, because it deals equally with large images and small images, e.g., icon images. To solve this problem, it should be effective to classify images within a Web page according to their roles [4] and deal with them in different ways.

5. CONCLUSION

In this paper, we proposed an adaptive presentation system of Web contents for Web browsing using cellular phones. This system reduces successive scroll operations to find an objective component by providing the overview of the page. In addition, the system presents the component adaptively according to it’s component class, and thus, users can read it comfortably. We confirmed the effectiveness of our system by using a simple user experiment. Due to the limitation of space, we omit the details in this paper.

As part of our future work, we plan to address an issue to determine a better path of auto-scrolling. Moreover, we plan to conduct detailed user experiments and verify an appropriate speed for auto-scrolling.

Acknowledgments

The authors wish to thank Dr. Shigeyuki Akiba, President & CEO of KDDI R&D Laboratories Inc. for his continuous support for this study. This research was partially supported by The 21st Century Center of Excellence Program “New Information Technologies for Building a Networked Symbiotic Environment” of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

6. REFERENCES


ABSTRACT
While many of the issues that are being raised in relation to mobile web accessibility are similar or the same to those that have been promoted over the past few years in relation to mainstream web accessibility, that doesn’t necessarily mean that we’re simply going over old material. Rather, it provides a jumping off point for mobile web accessibility. In turn, the differences in emphasis which result from the specific constraints of mobile devices could be crucial in highlighting some aspects of accessibility which, until now, have been neglected.

Categories and Subject Descriptors
H.1.2 [Models and Principles]: User/ Machine Systems—human factors, human information processing
; H.5.4 [Information Interfaces and Presentation]: Hypermedia—user issues, navigation
; K.4.2 [Computers and Society]: Social Issues—assistive technologies for persons with disabilities

General Terms
Human Factors, Standardisation, Design

Keywords
Accessibility, Usability, Universal Usability, Universal Access.

1. WEB ACCESSIBILITY: IS IT JUST A “MERRY-GO-ROUND”?
The issues¹ that are being raised in relation to making the mobile web accessible and easily usable for all users do tend to create a feeling of deja vu for those who have been involved in campaigning for and promoting accessible web design for some years. There are many areas where “traditional” accessibility (designed to facilitate access to the web for those with a disability) and mobile web accessibility overlap.

However, I don’t believe we’re simply treading old ground here. For one thing, the development of best practice recommendations and techniques for improving the user’s experience of the Web via a mobile device is able to build on the extensive work already carried out over the past few years in developing principles for accessible and usable web design. And the technologies available to implement these principles have improved hugely too, so the “starting point” options available to content designers are both more diverse and more powerful than was the case a few years ago.

It is also interesting to see that, while the best practice recommendations that are emerging for mobile web accessibility encompass many of the basic established web accessibility principles, they place a new focus and emphasis on issues which, because they sit in that slightly uneasy “grey area” between accessibility and usability or because they haven’t been seen as having a significant impact on more than a very few users, have been overshadowed and somewhat ignored. This bodes well for those users whose needs have, in general, been overlooked until now. There is a very good likelihood that the explosive growth in the use of mobile devices, and the economic incentives that provides (particularly in the commercial sector which, with a few notable exceptions, has tended to drag its heels in this area) to encourage the implementation of more accessible and usable website design, will push accessibility forward faster than ever before.

One of the key areas where there is a risk of the mobile web rehashing old problems is that of consistency across different devices and user agents. There was a period some years ago when the web was at serious risk of fragmenting because of the massive differences in how different browsers interpreted HTML, and one could argue that there is a risk of the same thing happening with regard to the mobile web, because of the even greater range and variety of devices and software agents in general use, with more being developed every year. But the very fact that the web went through that period of risk means that these risks are known and are much more clearly understood. That means that they can be minimised from a much earlier stage in the technological development of mobile devices and software for accessing the web than was the case first time round.

General awareness of the need for accessibility is another
area where, although there might be some overlap in the work that needs to be done, it is less than would be required if so much work had not already gone into publicising and raising awareness of accessibility from a disability perspective. Many web designers are already aware of accessibility and usability as important aspects of web design, and a growing number of designers are becoming skilled in implementing accessibility without compromising on visual interest or functionality. Incorporating additional techniques to further enhance the accessibility of websites should not present the same steep learning curve that these designers faced when first trying to learn or develop the accessibility techniques they now use all the time. That’s an important difference, since the web design world very much progresses as a result of many learning from a few, and as those with the knowledge and skills to develop new design techniques increase in number, there will be more examples of good, innovative design for other designers to copy and adapt.

In short - I don’t think we’re on a merry-go-round at all. Rather, it feels to me like a spiral staircase. There are times when one has a feeling of “haven’t we been here before?” But each time that comes round, we’re actually at a higher level than we were when we last tackled that particular issue. And while, like revising for exams, we might go back over some of the fundamentals, we also add new insights and thoughts and techniques to what was there before.
ABSTRACT
Accessible e-learning is becoming a key issue in ensuring a complete inclusion of people with disabilities within the knowledge society. Many efforts have been done to include accessibility information in e-learning metadata and the major result consists in the IMS AccessForAll Metadata definition. Unfortunately the complex behavior managed by this standard could be perceived by authors as a new boring and difficult activity enforcing the idea that the production of accessible Learning Objects (LOs) is too complex to be accomplished. This paper presents a novel component of an authoring and producing software architecture, designed and implemented to automatically create the IMS AccessForAll Metadata description of an accessible LO.

Categories and Subject Descriptors

General Terms
Design, Experimentation, Human Factors, Standardization.

Keywords
Accessibility, Metadata, Learning Objects.

1. INTRODUCTION
A large use of multimedia components is today widely utilized to improve efficacy of Web-based instruction, ranging from Flash animations to prercorded video-lectures [11]. These rich media are often a welcome addition to a Web-based course and are used to improve the quality and the effectiveness of learning contents in several situations, e.g. by describing a complex concept or by showing a process flow. Sometimes audio and video elements are exploited just to add a human touch to courses taught at distance [2]. Finally real lectures are frequently recorded to integrate or enrich on line materials.

In all the above mentioned situations, the provision of fully accessible and portable multimedia contents is usually considered a complex and expensive charge. Working with multimedia resources, major difficulties originate from the basic need to provide a textual equivalent for each multimedia resource and from the final accessibility evaluation that requires different tests to be accomplished. Both these tasks could become truly time-consuming, depending on the type of content that the teacher is writing and on the authoring tool he/she is using [10]. Frequently, the resulting LO is partially accessible or totally inaccessible. This could happen whether the structure of objects/alternatives is incomplete or it is related to a specific content or activity that is itself a source of barriers.

In this context, the IMS Global Learning Consortium [5], has developed several guidelines and specifications to address accessibility issues in e-learning. The IMS is a non-profit organization. Its members come from every sector of the global e-learning community and they include hardware and software vendors, educational institutions, publishers, government agencies, systems integrators, multimedia content providers, and other consortia. IMS have a fundamental role in defining specifications to ensure interoperability across e-learning systems. Main IMS guidelines and specifications on e-learning accessibility are:

- The IMS Guidelines for Developing Accessible Learning Applications [6], used to support the production of accessible learning materials.
- The IMS Accessibility for Learner Information Package (LIP) [7], to describe Personal profile of preferences (visual, aural, device) for tailored presentation of learning content (e.g. preferred /required input devices or preferred content alternatives). Profiles described by
using this specification are usually referred as ACCLIP profiles.

- The IMS AccessForAll Meta-data (ACCMD) [8] that describes accessible learning content by specifying, for example, what kind of content is being presented and if there is an equivalent or alternative form of the content.

The AccessForAll Meta-data specification is intended to make it possible to identify resources that match user's stated preferences or needs (expressed as an ACCLIP profile). ACCMD fully describe accessibility of learning content by specifying textual alternatives for each element that contains visual or auditory information. Finally, AccessForAll Metadata provides support to functional interoperability by allowing the substitution or augmentation of any resource with an equivalent one whenever it is required for accessibility purposes. To describe the alternative content structure, IMS uses a description of one primary resource, and the description of the features of zero or more additional resources that are somehow equivalents for that specific primary resource [10].

While these metadata represent a truly enabling option, implementing an ACCMD description of each LO could turn into a new tiresome and protracted task for authors. Reducing the distance between users' needs and authors' efforts is now a crucial aspect to ensure accessibility of e-learning materials. The solution relies on authoring tools for creating LO that have to accomplish two main goals:

1. Offering support to author in creating fully inclusive materials by suggesting correct behaviors and sometimes imposing the completion of all additional information needed to ensure accessibility (e.g., once the image is inserted, the authoring tool ask for a description that is required for blind users).

2. Automatically structuring the media alternatives, both inserting correct markup inside the (X)HTML pages and describing the whole structure with ACCMD.

This paper presents the ACCMDBuilder, a component added to a preexistent authoring tool called ISA-LOB in order to obtain the ACCMD description for each created LO. In particular, the accessibility of LOs was ensured by this tool since it has been designed and implemented as described in [3]. The new component adds the automatic production of ACCMD and includes it in the SCORM [1] LO package. The latter is provided through the ISA-LOB tool that has been used in an e-learning project of the University of Bologna, (A³ Project) to create about 40 accessible LOs.

The reminder of the paper is structured as follows. Section 2 describes design issues of our system, focusing on accessibility of rich media elements. Section 3 presents the architecture of the whole system, detail on the new component that supports IMS AccessForAll Metadata. Section 4 presents a complete example of automatically created and packaged resource, with the ACCMDBuilder. Finally Section 5 concludes the paper.

2. PROVIDING ACCESSIBLE RICH MEDIA

As mentioned above multimedia contents are frequently used to enrich efficacy of e-learning materials. In particular, each module provided to students through A³ Project may include several rich media and specifically:

- Static images, animations (based on GIF and Flash objects) and simple simulations, all contained in HTML pages. GIF and Flash are just convenient formats to illustrate our approach, which anyway equally works well with other raster formats and vector ones.
- Video lectures, used by the teacher to support the learning activity and implemented by using SMIL which synchronize video, audio and a sequence of slides.

To ensure that all these multimedia elements can be offered to the user in an accessible form, different mechanisms are provided in order to force the creation of all the required equivalents (textual and not-textual) [4] [9]. In this sequel, two different scenarios for use of rich (continuous) media will be presented:

1. a Flash animation, and
2. a SMIL video lecture.

These examples concretely exemplify the two main categories of multimedia contents used in A³ without the ambition to exhaustively cover all available media and formats. In both cases, we will detail the work needed to make the content accessible, respectively in the following Section 2.1 and 2.2.

2.1 Flash Animation

Flash animations are used in A³ Project LOs to introduce concepts (mainly related to Computer Science education) that result better explained by using a graphical representation. Examples of such animations are dynamics in a streaming-based communication and client-server interaction. The chief issues to address complete accessibility are:

- Providing a similar content for people that have not or cannot use a Flash player, e.g. a sequence of images that represent the evolution of the buffer content.
- Giving a textual description of the whole animation for people that cannot see or cannot display images.

2.2 Video Lecture

From a technical point of view, in Scenario 2, we have to synchronize the three main information flows that compose the video lecture:

1. The video track, filming the instructor while teaching.
2. The audio track, playing the instructor talking.
3. The slide sequence, simply considered as a sequence of images reproducing text, schemas or other graphical elements.

We need to add two additional flows that ensure accessibility, adding the textual equivalent for people with auditory and visual impairments respectively:

4. The caption sequence; captioning is essential for deaf and hard of hearing students, for foreign students and for students with an auditory processing disability.
5. The slide contents descriptions sequence. These descriptions could be simply obtained reproducing the text contained on each slide and adding further textual descriptions of images and animations, if any.

The SMIL multimedia obtained by composing these 5 flows is compliant to Accessibility Guidelines for SMIL but is still not enough accessible, especially for students that use assistive
technologies, such as screen readers or magnification systems. Two main problems arise from this rich media:

- SMIL uses strict layouts structures that are not adaptable to low resolutions needed to magnify (for students with low vision).
- The above defined presentation produces a cognitive overload to blind users. Two (synchronous) audio tracks are technically available: the main audio track reproducing the teacher talk and the audio track produced by voice synthesis by reading slide contents. Usually assistive technologies don’t read textual contents that change dynamically, so the second track and its provided information are lost.

User testing phase of SMIL video-lectures has confirmed the necessity to provide alternative versions for the online lecture. A first alternative consists of an HTML hypertext with textual slide contents (alternative for image-based slides) linked to the corresponding audio explanation. Blind students can use this version of the video lecture by reading text of slides (with voice synthesis or Braille display) and, after or before that, listening to the teacher explaining that slide. By compounding speech synthesis and human voice the video lecture entire content is provided as auditory flow.

Another alternative is needed to meet requirements of students who need captioning and have difficulties reading text while it is synchronously changing. Such a situation could happen, for example, to deaf or hard of hearing students who consider teacher speech too fast to be completely understood.

An automatic transcoding process transforms the original SMIL by deriving from the synchronous (SMIL) source, different discrete or semi-discrete versions of the lecture (fully accessible XHTML lectures and printable PDF documents).

3. ARCHITECTURE

Both the above mentioned rich media are normally included in A’ LOs that are created by using a set of software designed and developed to support the implementation of accessible Learning Objects. In specific, we initially have two post-production chains:

1. Flash animations are embedded in the (X)HTML pages, like the most part of hypertextual materials. In order to support this task our system provides the user with a word processor (MSWord or OpenOffice Writer) used to create formatted documents and a post-producing tool (called ISA [12]) that derives structural and semantic information. The result of this two steps activity is (i) a set of XML files containing textual contents and the SCORM metadata together with (ii) a set of media files for images, audio, video and all the non-textual media.

   In both cases an XML-based intermediate format is produced. It contains a structured representation of all the alternatives. The presence of additional information which are necessary to guarantee accessibility, is ensured by the specific tool used in each of the two cases forcing the author to specify all required data. The complete process is more exhaustively described in [3]. Once obtained all the intermediate files another post-production component creates a LO and specifically:

   A. It assembles data stored in a intermediary XML files creating XHTML pages.

   B. It structures all contents in appropriate directories and describes the didactical structure as a SCORM manifest by calling a component called ACCMBBuilder.

   C. It creates the IMS ACCMD description of the resources.

   D. It produces a single .ZIP file that represents the LO.

The Learning Object Builder (LOB) production process is based on a set of templates and configuration files that are used to define structural elements as well as layout and graphical aspects of the automatically produced LO. The whole process is depicted in Figure 1.

We will now focus on step C, showing the most innovative result presented in this paper.

4. PRODUCING IMS ACCMD

The ACCMBBuilder reconstructs a global XML representation of all the alternative information (The IMS manifest). An XSLT stylesheet, generates final XML metadata from data stored in the XML intermediate files. The whole amount of necessary information comes from the requirements fulfilled by authors as illustrated above. All alternative resources are identified and represented in the IMS ACCMD description. We have used the ACCMBBuilder on a benchmark resource, taken from A’ Project which contains both the cases study described in Section 2.1.

The following code fragments (depicted in Figure 2, 3 and 4) represent the IMS ACCMD description of the animation introduced in Section 2.1 and managed by post-production chain 1 described in section 3. The code fragment of Figure 2 describes the primary resource and the first (unique) equivalent resource of the animation: the alternative image, that is useful for people that have not or cannot use a Flash player. The primary resource (the Flash animation) is defined as a visual and textual resource.
Figure 2. IMS ACCMD for the case study 1 (alternative image)

The `<accessibility>` element is the root element that groups the accessibility information about the resource. The `<resourceDescription>` is the element that describes the features of the resource. The `<primary>` element defines the features of the primary resource. Its attributes respectively indicate:
- `hasAuditory`: Boolean value that indicates whether or not the resource contains auditory information;
- `hasVisual`: Boolean value that indicates whether or not the resource contains visual information;
- `hasText`: Boolean value that indicates whether or not the resource contains text;
- `hasTactile`: Boolean value that indicates whether or not the resource contains tactile information.

The `<equivalentResource>` element is a pointer to an equivalent resource (meta-data) of the described resource or parts of there.

Figure 3. IMS ACCMD for the case study 1 (alternative animation)

Figure 3 shows another code fragment of the IMS ACCMD description of the animation and its equivalent image. It is worth noticing that it is mandatory to consider and declare the primary resource as an equivalent one of itself. A textual detailed description is declared as an alternative resource for the whole animation for people that cannot see or cannot display animations. The `<equivalent>` element is the description of the features of resources that are equivalents for a particular primary resource. Its attribute `supplementary` accepts Boolean value and indicates whether or not the resource is supplementary (as opposed to alternative) to the primary resource. The `<primaryResource>` element is a reference to the primary resource (meta-data) which the described resource is equivalent to. The `<content>` defines the features of the content of the equivalent resource that parallel the ACCLIP specification.

The textual description is, it its turn, intended as an alternative for the image resource too. In particular, in the code fragment shown in Figure 4, this textual description is declared to be an alternative for visual content, in order to provide the description of the image to people that cannot see or cannot display images.

Figure 4. IMS ACCMD for the case study 1 (alternative textual description)

It is worth noticing that some particular users conditions are not considered in IMS ACCMD standard. In fact, in this case it could be significant to provide alternative resources in order to meet epileptic users needs, but the standard does not consider any appropriate equivalent alternative.

The `<alternativesToVisual>` element defines visual content presented in a different modality. The `<longDescriptionLang>` element indicates that the described resource contains long description text in the specified language (through the attribute `xml:lang`) for the referenced primary resource.

Furthermore, notice that a new namespace is declared as the default one in order to define content that is alternative for visual resources. This namespace is relative to the ACCLIP profile [7]. This namespace is used whenever it is necessary to declare alternative contents to visual, auditory, textual and tactile resources.

The code fragments in Figure 5, 6 and 7 represent a more complex IMS ACCMD description, related to the video lecture introduced in Section 2.2 and then managed by post-production chain 2 (illustrated in section 3). In particular the following Figure 5 shows the description of the primary resource (the SMIL multimedia presentation) and its directly equivalents. It is worth noticing that the primary resource is declared as its equivalent resource (we’ve already detected that this is a mandatory
The primary resource is defined as a visual, auditory and textual resource. Equivalent resources are the above illustrated HTML hypertexts. Captions (defined in Italian language) are declared as alternative for related audio synchronized streams within the SMIL multimedia presentation.

The <alternativesToAuditory> element defines auditory content presented in a different modality. The <captionType> element indicates that the described resource contains a text caption for the referenced primary resource. The <verbatim> element indicates that the caption is a verbatim caption. The <reducedSpeed> element indicates that the caption is a reduced rate level caption (or not, if its attribute value is set "false"). The <enhancedCaption> element indicates that the caption is an enhanced caption.

The fragment code shown in Figure 6 describes the first alternative HTML hypertext and its equivalent resources. This HTML page is created by composing textual slide contents (alternative for image-based slides) linked to the corresponding audio explanation.

The <audioDescription> element indicates that the described resource is an audio description for the referenced primary resource. Its attribute xml:lang defines the language of the audio description.

Besides the primary resource, the following code declares the audio description of the whole presentation to be its equivalent, in order to provide alternative for visual resources.

The fragment code shown in Figure 7 defines the second alternative HTML hypertext and its related equivalents. This HTML page composes textual slide contents (alternative for image-based slides) and captions (alternative for audio explanations).

Finally, the last fragment code (depicted in Figure 7) defines the third alternative HTML hypertext and its related equivalents.

In this case, the primary resource is declared to be an equivalent resource too. Furthermore the alternatives for visual resources are declared. In particular, in this version, textual descriptions are
provided as slides alternative. In both these two last cases the primary SMIL multimedia presentation is intended as an equivalent resource.

It is worth noticing that IMS ACCMD standard allows the description of each single file alternative (every single image, text, audio, etc. file), but in our case we decide to consider alternative HTML hypertexts as atomic resources, in order to avoid an extreme heaviness of the IMS ACCMD code.

5. CONCLUSIONS AND FUTURE WORK

The creation of fully accessible LOs is a difficult work that needs to be backed by specific tools. In particular, these tools have a strong impact if they can improve easiness of the whole process by sustaining authoring and automatizing post-production. This work presents components, which are embedded in an existing authoring/producing tool and automatically creates the IMS AccessForAll Metadata description of a LO, starting from the natural structure of multimedia contents. Some considerations arise from experimental uses of the tool, pointed out by the two cases study. First of all, some items can be fully described just by considering it as atomic elements. This is particularly true for continuous media whose description includes timing and synchronization elements (e.g. SMIL resources). Another aspect that impacts on the granularity of the association with ACC Meta-Data refers to the set of condition considered by this standard. As a final result we have declared that the video-lecture accessible to blind people is the one created in HTML. This is true (this version is fully accessible to blind students) but at the same time unsighted people could also appreciate the continuous version, for example for a first overview of issues introduced by the teacher. The defined structure, associated with an appropriate management of ACCLIP and ACCMD, would probably provide to blind user just the XHTML-based materials.

Such a tool is now integrated in a complex process used inside the University of Bologna to create accessible LOs. Accessibility of e-learning materials produced has been widely tested by involving a group of people with disability in verifying on-line contents and services. Universality of materials has been tested by using different browser running on different platforms (specifically MS Internet Explorer 5.0 and later, Mozilla Firefox 1.0 and later, Netscape Communicator 7.0 and later, Lynx 2.8.4 rel. 1, IBM Home Page Reader 3.0, Apple Safari 1.0). Finally, LOs produced by our process are compliant to all the constraints considered by the Italian Law on Web Accessibility, meeting also WCAG 1.0 AA level and Section 508 requirements.

Unfortunately, the IMS description is ignored by the LCMS (Learning Content Management System) in use. Generally this new technology is not fully supported and there are just few solutions that use ACCMD and ACCLIP to provide adaptive accessible contents. We assume that a growing availability of IMS ACCMD tagged LOs will drive the development of adaptive modules for the more diffuse LCMS and will definitively diffuse the use of the whole IMS specification on accessibility. Thus, tools that support an automatic production of accessibility meta-data are a strategic issue for the complete success of such technology. The decision about granularity of the description reveals as a crucial aspect also by considering dimensional aspects. Testing the system on a set of LOs that require 32 images, 4 animations and a video-lecture we add more than 1.000 lines of ACCMD code. This is a strong reason to assert that such a meta-data have to be automatically produce by the authoring tool that creates the LO. Future work will be mainly devoted to extend the proposed approach by managing a wider set of rich media. We are also working on the harmonization of ACCMD whith requirements imposed by the Italian Accessibility Law.

6. ACKNOWLEDGMENTS

Authors want to thank for their precious suggestions all the colleagues who worked on A³ Project and particularly Fabio Vitali, Angelo di Iorio and Antonio Feliziani.

7. REFERENCES


A Semantic-Web based Framework for Developing Applications to Improve Accessibility in the WWW

Christos Kouroupetroglou
Dept. of Applied Informatics,
University of Macedonia
Thessaloniki, Greece
+30 2310 791604
kourou@teithe.gr

Michail Salampasis
Department of Informatics,
T.E.I. of Thessaloniki
Thessaloniki, 57 400, Greece
+30 2310 791284
cs1msa@it.teithe.gr

Athanasios Manitsaris
Dept. of Applied Informatics,
University of Macedonia
Thessaloniki, Greece
+30 2310 891898
mantis@uom.gr

ABSTRACT
One of the biggest issues the World Wide Web (WWW) community has to overcome is accessibility for all. The rapid expansion of the WWW using problematic web authoring practices, together with the dominance of the desktop metaphor in web page design has raised many WWW accessibility problems for people with disabilities. In this paper we present a what may be termed as a “Semantic Web application framework” which allows different applications to be designed and developed for improving accessibility of the WWW. Apart from the architecture, the tools and the technologies that compose the framework, the key idea of the framework is that it aims at promoting the idea of creating a community of people federating into groups each playing a specific role: ontology creators creating concepts using an ontological approach to describe various elements of the WWW, annotators using concepts to annotate specific pages, user-agent developers creating tools based on the framework, and finally end-users (people with disabilities) that use these tools for their benefit. Within the proposed framework, these groups cooperate and interact with each other, having as their ultimate goal the improvement of WWW accessibility.

Categories and Subject Descriptors
H.4.3 [Communications Applications]: Information browsers.

General Terms
Design, Measurement, Management.

Keywords
Semantic web, metadata, RDF, voice browser, accessibility, information seeking.

1 INTRODUCTION
This paper presents a generalised application framework, which enables applications for improving accessibility in the WWW to be designed and developed within an open and extensible underlying framework. The framework is based on the idea of the Semantic Web and it could be utilised to provide the basis for developing tools and applications (user agents in more formal terms) for augmenting mobility and information seeking performance in the WWW for people with disabilities. One key objective of the proposed framework is to promote the idea of forming communities of collaborating people each playing specific roles in the framework. In other words the proposed application framework, like any other metadata scheme, signifies community membership [16].

There are many different accessibility and information seeking problems in the WWW depending on the type of user disability, the information need that must be satisfied, the level of expertise of the user, how the web content was created, the information task at hand etc [2][4][21]. In the following paragraphs of this introductory section we will describe accessibility problems when people with disabilities use the WWW. However, it is not our goal to make a thorough discussion of accessibility and other mobility and information access problems in the WWW. Although the framework could be utilised in any situation in which some form of annotation could be beneficially applied to facilitate mobility and information access problems in general (e.g. visually impaired people, elderly), in the rest of this paper we will focus on accessibility problems of visually impaired (VI) people for two reasons. First because we have implemented a voice web browser based on the suggested application framework [22], therefore we possess experience about this type of disability and associated accessibility problems. This facilitates our need to give some specific examples of how the generalised framework could help the development of applications addressing specific accessibility problems. But also, because visually impaired are probably the best example of web users that their disability (regarding navigating and accessing information from the WWW) may be compensated, if support based on our framework is provided.

After the discussion of accessibility problems, in this introduction we will also shortly describe the framework and how we envisage that it could act as a foundation and an underlying platform for creating tools and applications that address the accessibility issue in the WWW.

Nowadays, web page design is eminently dominated by the desktop metaphor and generally web page authoring uses unreliable, variable and inconsistent authoring practices. The layout of a web page, the use of various fonts, colors, images, and other visual cues convey navigational and semantic information to sighted users. Unfortunately these visual cues are not accessible by people with disabilities such as blind users [6][9]. For example, a list of links appearing in a table in the left side of a web page with different font and background color is immediately associated by a sighted user with the concept of a navigational...
The latter exposes another problem that originates from the misuse of HTML tags from web developers. One of the basic principles in developing accessible web pages is to use HTML tags correctly to make functional mark-up. For example, using an H1 tag to make a certain text appear in bigger fonts without being a level one heading is a wrong but often used technique. Or a H2 tag may appear without H1 for example. Another common use of HTML tags which causes problems is the use of the TABLE tag for layout and lining reasons without having any tabular data within. The visually conveyed information in these cases is missed and sometimes it confuses even more blind users who use screen readers or voice browsers to read them.

Generally this lack of accessibility leads to poor navigation, mobility and consequently inefficient information access for VI users. Harper [9] has introduced the notion of travel and travel objects to draw an analogy between web navigation and travel in the real world, in an effort to improve web accessibility for VI. Although Goble’s contribution served the need for a mobility analysis framework, Yesiada et. al. [28] have suggested that a more systematic foundation is required for engineering tools in a more systematic way that will support mobility.

Most of the blind users today use programs such as generalised screen readers and specialised voice browsers for their web browsing. A basic functionality of these tools is that they serialise a web page into a simple text. This serialization often brings uninteresting parts of a web page (e.g. advertisement banners and other peripheral material) in front of the main content of a web page. This fact disappoints, upsets and disorientates users. To avoid this “noise” of useless information, voice browsers and screen readers provide additional features such as listening to the links or the headings of a web page. These features provide a quicker access to some parts of a web page that are difficult to reach using the simple serialization. However, the misuse of HTML tags sometimes disables their usability and creates additional problems to browsing within a page.

In addition to problems related to within page browsing, blind users do not have the ability to scan quickly a web page in order to attain a digest and a general conception of it. Scanning is one of the most crucial sub-processes of sighted users when seeking for information in the web. Programs may provide some kind of scanning simulation using various techniques of summarization. However, these again depend on how well structured and authored are the web pages.

The discussion above illustrates that the problem of web accessibility for all has many facets, and such as, is the focus of many emerging areas of study. Each area may contribute a little bit (depending on the specific problem) to produce together with other techniques an overall efficient web navigation and effective information access for people with disabilities. In that context, one contribution of our application framework is that it portrays the backdrop for the work of different user agents to complete activities that will enhance web accessibility for people with disabilities. Our research effort aims at designing what may be termed as a “Semantic Web application framework” to support the development of accessible WWW applications for all. The framework suggests an architecture that can be generalized and applied in developing WWW applications for many types of accessibility problems. However, the framework as it is presented and discussed in the rest of this paper is focused particularly in the accessibility problems related to the information seeking process of blind users in the web.

A variation of techniques and strategies can be used for information seeking in the WWW. Browsing is one of them and it is the specific strategy that the actual implementation of the SeeBrowser tool that is based on the framework tries to improve. Browsing is separated in across document and within document browsing. The tools that were developed based on the framework, for this specific project, aim in improving and solving problems for both types of browsing.

The proposed application framework delineates an architecture which in our case is instantiated by a set of software tools that we will describe in the following sections. But, it also presupposes a community of people separated into groups, each playing a different role. The first group is ontology creators responsible for creating concepts using an ontology editor. The second role is played by annotators. Annotators use available concepts to annotate specific web pages, aiming at increasing their accessibility. Third, user-agent developers that create tools based on and exploiting the framework. The last group is end-users (people with disabilities) that use user-agents such as the voice web browser presented later in this paper. The roles of the groups of people envisioned in our proposed framework will be presented in parallel with the basic tools that compose the application framework. These groups need to interact and cooperate with each other. This cooperation and interaction is another crucial part of the framework and it is also discussed in the paper.

2 A SEMANTIC WEB APPLICATION FRAMEWORK

Before we describe the application framework we review some earlier work that partially inspired us. As already discussed earlier, one of main problems in within document browsing, is the reconstruction of web pages by serialization. This was realized quite early and several solutions for this problem were suggested. Some of these solutions presented by Tagaki & Asawaka [3][25] and Huang & Sundaresan [11] was based on a transcoding server which transforms web pages according to specific annotation and patterns. The transformation process reconstructs the web pages in a way that makes navigation for blind people easier. The transcoding server approach though, had some drawbacks presented by Hanson & Richards [8]. In addition an interesting approach in storing and retrieving annotations is presented in the Annotea project. The concepts of using annotations, client-side architecture and having the annotations stored on a server are fundamental in our framework too.

One of the latest advances in web technologies, the Semantic Web, came as the ideal background to support all of them. The Semantic Web raised a great deal of discussion and many expectations. Marshall & Shipmann [15] present, categorize and discuss the various views and expectations raised by the Semantic Web. Some people see the Semantic web as library cataloguing system for the web. Others hope that it will increase machine awareness of web content, improving this way searching facilities in the current web. One other category uses Semantic Web as a
way for metadata syndication, enabling the communication among various information sources and agents. This latter approach is the one that our framework is closer to. In our research work we envisage the Semantic Web as a metadata layer upon the current WWW, through which user agents will syndicate, interact and collaborate in order to improve accessibility. These metadata can be produced by various sources and can be used by many users and agents for a variety of goals.

At the cornerstone of our application framework are metadata and their manipulation. Our metadata are stored and retrieved in a storage server using RDF/XML formatted files. RDF is the standard used for metadata in the Semantic Web and is a language for describing resources. We use this language for describing web pages in a way that will help to improve their accessibility. Storing and retrieving metadata from a storage server, allows different users to contribute to the development of metadata at the same time, thus forming a community of users contributing in developing the Semantic Web.

A key element in describing resources in the semantic web is the vocabulary used for this description. The vocabulary can vary depending on the goal of the description. There are already many vocabularies such as DC, Foaf, etc, that can describe various resources in various ways. For the description of a web page in order to improve its accessibility by blind users there are specific needs that were not covered by any of the existing vocabularies. To satisfy these needs we use an ontological approach of OWL (Web Ontology Language) to create a vocabulary especially for them. This solution was preferred among other because is one of the most commonly followed in the Semantic Web nowadays.

Our voice web browser finally utilizes the two previous characteristics, vocabulary and metadata in RDF. All three of them together are the key points of our framework. Figure 1 illustrates an overview of the framework and the relationship between the three key tools used in our framework. SeEBrowser finally uses the outcome from both the annotation tool, which are the metadata, and from ONAR, which is the vocabulary used for the annotation.

Based on the annotation mechanism using metadata combined with the vocabulary, the browser provides to the blind users with a set of browsing shortcuts to the previously annotated elements. This mechanism according to the findings of our preliminary experimental tests with a set of blind users using our voice web browser could be quite useful for them [22].

![Figure 1: The suggested framework scheme](image1)

Of course, it should be said that the framework could be generalized, as many different uses of metadata could be applied. In our research so far, we have investigated just one of them (i.e. improve information seeking for blind people in the WWW). The feedback provided by end-users apart from the SeEBrowser development group was also useful for the annotators’ group, because it suggested changes in annotations, better descriptions for the annotations etc. Annotators are one of the groups in the community model suggested by the application framework (see Figure 2).

Their role is to annotate web pages for blind users. People in this group can be either related to web authoring process such as web developers, designers etc. or related to the blind users group such as teachers in special schools for blind people. All of them together could contribute with their annotations to web pages and create an extensible layer of annotations over the existing web.

![Figure 2: Framework community diagram](image2)

During the annotation process annotators realize shortcomings in the vocabulary they are using and suggest changes in it. The group taking this feedback is the vocabulary developers and they are responsible for developing vocabularies according to users and annotators needs. This group of people can consist of
knowledge professionals, annotators, and generally people related to knowledge engineering. The process and the connections described in the previous paragraphs are shown in Figure 3 which shows the data flow for the specific instantiation of the framework in our project.

3 OWL ONTOLOGY

One of the key points in our application framework is the vocabulary used for producing the annotations. The vocabulary is described as a set of entities, properties and relationships between ontologies in an OWL file. OWL is a language used for declaring and describing ontologies. The ontologies can depict information systems using classes, properties and relations between classes. Our information system which is modelled is the WWW and the web pages that exist in it together with their key visual and other elements that are often used by sighted users. A menu for example can be a class that describes the entity of a menu in a web page and could have a property named “number of items”. In the following section we will demonstrate how the concepts described as classes in one ontology are instantiated in web pages.

In the specific instantiation of the proposed application framework we have developed, an ontology editor called ONAR [27]. ONAR provides a GUI where the knowledge engineer-vocabulary creator can easily create classes, relations and assign properties. All of them are presented in a graphical way as seen in Figure 4. The main advantage of ONAR is that allows users who don’t know the OWL language to create ontologies easily. In addition, it is easier to understand an ontology developed by another person when you see it in a diagram, such as this shown in Figure 4, than in an OWL file. This encourages the collaboration amongst vocabulary developers having as a common goal the production of a final ontological vocabulary.

ONAR in its initial form was designed to process ontologies locally on a user’s computer. This would mean an additional cost of workload and communication between users when needed to update or review an ontology developed by someone else. Needless to say that there would be also synchronization problems if someone was developing an ontology in parallel with someone else. To avoid all this confusion we enhanced ONAR with an additional feature of downloading and uploading ontologies in the web using a web service. Every user has a username and uses this in order to upload his or her ontologies. In addition he can assign a group of users that can update the current ontology. This encourages the collaboration amongst vocabulary creators having as a common goal the production of a final ontological vocabulary. With this scheme the network of vocabulary creators can either develop an ontology on their own or cooperate with other users in the development. However, when using the ontologies for the annotation process there was the need to have a certain user whose ontologies would be the result of all the suggestions and discussions in the development process. To achieve this we created a specific user with name * whose ontologies are final products from the various collaborations and suggestions. This doesn’t exclude ontologies developed by various users or groups to be used too but ontologies by user * are more official than any other.

Figure 4 : Screenshot of ONAR representing an ontology graph
3.1 OWL
The vocabulary issue is in general a knowledge representation issue. The choice of OWL in our framework was based first of all on the fact that it is a standard which is used widely in the Semantic Web society. This makes it easier for anyone who wants to contribute in the development of a vocabulary to do so having a common language for communication. In addition ontologies in OWL can serve many different purposes and apart from the ontologies developed for our purpose, anyone could develop a similar ontology for other purposes such as improving accessibility of elderly or people with dyslexia. So, the use of OWL allows extensibility of our framework.

In addition, OWL leaves the knowledge engineer free to construct many kinds of relationships apart from the standard types of relationships. This freedom though, comes with the cost of producing an ontology that might be perceived differently by different annotators. These problems of misinterpretation weren’t strong enough to prevent us from using this freedom to create our kind of relationships. For example, in our ontology there is a class called “menu” and another one called “menu item”. These two are connected with a relationship named “menu contains”. Similar relationship are the “form contains” and the “result list contains”. Using “contains” as a part of the relationship’s name is a naming convention in our relationships. This way, we could create several different relationships that all have the same meaning and can be used in a certain way by the agent. In our browser for example when the user reaches an annotated web page and listens to the annotated elements found there, elements that are contained within other container type elements are excluded from this initial list. When later the user reaches a container element he can listen the annotated elements found within this container element.

The vocabulary we developed in our SeeBrowser project aims in describing elements in a web page that help blind users to move faster and more efficient within a page and also across pages especially of a certain site. Yesilada et. al. [28] have already presented such a set of elements on web pages. In our vocabulary there is a set of classes with the appropriate properties and relationships that describe many of these elements. This means that we have classes that describe various way points such as menus, headings, sections, banners, advertisements, links to specific places (i.e. site map page, home page, index page) etc. In addition there are classes that describe elements that are widely used in reference pages when searching for information such as subject list, alphabetic list of items, short descriptions of items, elaborate descriptions of items, navigation links within multiple result pages such next, previous, first and last page etc. Another part which is under development describes specific elements widely used in portal sites such as, search box, web directory, login form, weather box, news section, etc. Finally we plan to develop the vocabulary with even more classes especially for educational sites.

4 ANNOTATIONS
Annotations are the second key element of our framework. They are produced in the form of RDF/XML files by an annotation tool. Annotations are stored on an annotation storage server (see Figure 1). In this section we will discuss issues related to the structure of annotation files, architecture of the annotation storage server, and the role of annotators. Annotators play the role of the middleware group between vocabulary creators and end-users and user agent developers. They are also closer to end users and therefore they have a very important and crucial role in our framework.

4.1 Structure of annotation files
Before explaining the structure of the annotation files we should make a brief introduction to RDF. RDF is based on statements that are formed in triples (Subject, predicate, object). Having this in mind, we can say that a subject is a resource a predicate is a property name for this resource and the object can be either another resource or a literal as a value of the property. This means that an object of a statement can be a subject in another statement so that we can have a series of statements in a chain. A usual presentation of an RDF file is a directed graph where resources are presented with ellipses, predicates with arrows and literals with rectangles. A typical graph can be seen in Figure 5.

![Figure 5: RDF Graph of an annotation file](image)

For the particular application we investigated as a case study of our application framework (i.e. develop a voice web browser for blind people), we use an xml namespace called “SeESyntax” that includes the schema according to which our files are produced. The second namespace called “SeeBrowser" points to a vocabulary for describing a web page which is an OWL file as seen in the previous section. Having defined these two namespaces, the typical structure of an annotation file in our application is as follows:

```
<rdf:RDF
xmlns:SeEBrowser="[SeEBrowser namespace URL]"
xmlns:SeESyntax="[SeESyntax namespace URL]"
xml:base="[URL of annotated page]">
  <SeESyntax:Annotation rdf:about="[URL of annotated page]">
    <SeEBrowser:[OWL Class] rdf:about="[URL + XPATH of the annotated DOM element]">
      <SeEBrowser:[OWL Class property]=[value]>...
      <SeEBrowser:[OWL Class property]=[value]>...
      <SeEBrowser:[OWL Class property]=[value]>...
    </SeEBrowser:[OWL Class]>
  </SeESyntax:Annotation>
</rdf:RDF>
```

A file structured like this can describe various elements of a web page using the XPATH of the HTML element one wants to...
annotate. For example, an annotator assigns the concept of a menu in a certain TABLE element on a web page and similarly assigns values to its properties defined in the OWL file. In a similar way another part within a DIV element in an HTML file, can instantiate the concept of “main content” and so on. This is done using the “SeESyntax:Contents” nodes.

The node “SeESyntax:Template” is a property of the SeESyntax:Annotation class and is used in order to declare whether the URL of the annotated page is a template for other URL’s too or not. This is done in order to reduce the workload and size of annotations produced when a certain page is similar to others. Pages within a site are quite common to follow a certain layout as a template for their design. When annotating one of them we can use the Template property and in combination with a regular expression instead of the actual URL we describe not just one but a set of web pages described by this expression. This mechanism provides a way of semi-automatic annotation of web pages, which is crucial when annotating a large amount of pages.

As already discussed, annotations could be produced by many different annotators using different vocabularies. The capability of using different vocabularies is addressed in our framework by the capability of defining different vocabularies. This is done by the XML namespace declaration where every annotator can choose and use any preferred vocabulary.

4.2 The annotation tool

Annotation files are produced using specially designed software. Figure 7 illustrates the user interface of the annotation program that is separated into three main areas. The main area in the center presents to the annotator the web page to be annotated. On the left hand side a tree view represents the DOM tree of the web page. When the annotator selects an element from the DOM tree the corresponding area of the element is highlighted in the page view. On the other side there is a list of the OWL classes available in the vocabulary. When a user right-click on an element from the DOM tree, this list appears in a popup menu and the annotator can select the concept that s/he wants to assign to the element. After the selection, a series of dialog boxes appear asking the user to input values to the properties of the class. Repeating these steps annotators produce the annotation file for a web page. Before saving or uploading the file to the storage server the tool asks them to input their username so that the reification statements can be produced. Finally it also asks them whether the page they annotated is a template for other pages or not. If yes then the user has to input the regular expression that will describe the set URL’s for the annotated pages.

4.3 Storage Server

The storage server is based on the idea of Annotea project [1] and exploits some of its advantages. First of all the RDF annotations are stored on a relational database using mySQL as RDBMS. The Jena API [13] is used for managing the database. Jena API provides the developer with the ability to work with an RDBMS as an RDF database using RDQL commands instead of SQL. RDQL is a language similar to SQL but especially developed for selecting and presenting RDF graphs stored in RDF databases.

One of the advantages of the Annotea project is that the communication and management of annotations in a server can be done through simple HTTP POST and GET commands. We also used a similar protocol for our communication with the annotation server.

In particular a user requests the annotations for a URL with an HTTP POST command. If there are many annotations that uploaded annotations for the specific URL, then the response is a list of them and there must be a second HTTP POST command accompanied by an annotator’s name in order to get the annotations from a specific annotator. Otherwise the annotations are sent directly as a response to the first POST command. When the server searches for a URL matching the requested it also checks if the requested URL matches with regular expressions existing in template descriptions.

The upload process is also similar. The server keeps a table of annotators that can upload annotation to the server and when a user uploads a file the username and the password kept in the server must be also sent through an HTTP GET command to be checked in the server. This requires annotators to be inserted in the annotator's table beforehand by the annotation storage server administrator. This is done for authorization purposes so that there is a control on who uploads annotations.

For the purpose of our project we set up a specific storage sever that we are using. The architecture though of this scheme is extensible so that we could also have a network of storage servers. This means that this network could become a second semantic layer upon the current web. Agent developers can then take advantage of this layer and use their metadata in any way they like.

Figure 6: A sample of an annotation file

This scheme however is problematic when annotations from different annotators will be uploaded to the server. Annotations are stored all together using a specific API that manages them using an RDBMS. This means that statements of various users about the same resource might create conflicts. To avoid this confusion we use a mechanism provided by RDF and is called reification. Reification is used when someone wants to make a statement about a statement. This is done by using a certain set of classes and attributes that convert a statement to a resource itself so it can be used as subject in other statements. To achieve this we use the abbreviated syntax for reification. This means that we assign an rdf:ID property to each statement’s predicate and later using this rdf:ID we can assert the reification statements. Consequently, the structure of the file changes to what is shown in Figure 6. This way not only the problem is solved using RDF mechanisms, but there is also the possibility to use annotations synthetically by many users.
4.4 The role of an annotator

The authorization process is required because it is essential to control the annotations uploaded in the server. We need to know if an annotator produces invalid or misleading annotations and isolate or delete them. End users depend on good annotations, so annotators need to have certain level of commitment and reliability.

The latter shows that this group is very important in the framework. An annotator should have some primitive knowledge of HTML in order to understand the DOM tree structure and use it correctly. The other important part of the annotation process is the vocabulary. Names of the classes, their description, role and relationships should be well understood before the annotation process begins. Misinterpretation of concepts in the vocabulary could lead to false or incomplete annotations. Finally annotators should have a clear and complete overview of the web pages that annotate in order to know their structure, layout and navigational aids that they provide.

Another very important aspect of the annotation process is the purpose of the annotation. The annotator should know where and how these annotations would be used in order to achieve a better description of the page. Knowing the needs of end users the annotations may respond better to their needs. It is similar to the situation when one must describe a building to an architect and also to a friend without particular knowledge of the subject. First of all, the vocabulary in the first case should be more technical and specific where in the other case simpler. In addition the architect will need more details in the description such as exact sizes and places where the other person will be satisfied even with a general description of the building.

In an analogous way, annotators in our example should know what problems a blind user faces when browsing in the WWW, in order to provide usable solutions through the annotations. Consequently, interaction between these two groups is necessary. Certainly the closer an annotator is to the end users group, it is more likely he can produce a more effective annotation. Annotators also provide feedback to vocabulary developers in order to transfer to them needs of blind users that are not satisfied by the current vocabulary or possible misinterpretations of the concepts defined in a vocabulary.

To sum up, annotators as an intermediate group, between vocabulary developers and end-users take feedback from end-users and provide feedback to vocabulary developers regarding the expressiveness, correctness and appropriateness of the vocabulary. Sometimes it is even better if an annotator plays also the role of vocabulary creator because he can solve annotation problems related to vocabulary. There is also the possibility for end-users to be annotators. However, in our case blind users that want to annotate pages need to have an even better understanding of HTML and they also need to have a sufficient browsing experience with the pages to annotate.

5 SEEBrowser

SeeBrowser is the final part of the framework and is the tool that utilizes the result of all other tools and groups of people for end-users benefit. It should be clear-cut that in our framework the two other components that have been already discussed, i.e. the ONAR ontology editor and the annotation tool can be used in any condition and for developing any type of Semantic-Web application. In this section we will shortly describe the basic
functions of SeEBrowser and we will discuss the findings from a preliminary usability test.

5.1 Basic features of SeEBrowser

SeEBrowser uses SAPI5 compatible TTS engines and voices. Especially for the Greek language it uses the “Demosthenes” TTS engine [30]. Using SAPI5 compatible voices means that users should be able to change their preferred voice. This is done using a voice profile control panel where users can configure the preferred voices and other configurable aspects of speech (e.g. adjusting voice rate).

Similar to other web browsers, SeEBrowser users can insert a URL to browse, follow a link within a page, go back and forth in visited pages and go to the home page. There is also a search text feature allowing users to move directly to instances of a specific text within a page. Finally there is a bookmarks feature that allows users to save favourite URL addresses in a list. This feature however has been adapted slightly to blind users’ needs. Apart from the URL and the title it can also store the current position of the reading cursor. This is later used to transfer the user directly to the specific position when opening the page from the bookmarks list.

5.2 Browsing within a web page

How the reading cursor moves, depends on how the user browses within a page. Users listen to the web page fragmented depending on the combination of which browsing and speaking mode is selected. Based on these two modes a web page is decomposed in two levels in order to be separated into the fragments to be browsed.

The first level is the browsing mode selected. The browsing mode defines whether the user will browse either the whole text of a web page or parts of it (e.g. links only). In particular the user can select either to browse the whole page or its links, headings or forms. In each of the later cases a list of the HTML elements to be browsed is formed. It is also important to say that when returning from a specific collection of elements to browsing the whole page, the “cursor” automatically moves to the corresponding place next to the last element browsed by the previous mode.

The second level is the speaking mode selected. Here the text of each element is further separated either in paragraphs, sentences or words. This way the user pressing the Up and Down arrow keys can listen the selected fragment word by word, sentence by sentence or paragraph by paragraph. The combination of browsing and speaking modes provides the user with a variety of possible ways to browse a page according to his/her needs.

5.3 The use of annotations

As already pointed out a distinctive feature of SeEBrowser is the use of the annotations produced by the process described in the previous sections. There can be various uses of annotations but the one implemented in SeEBrowser in this phase of the research aims in improving browsing as an information seeking strategy. This means that we aim in improving both browsing across pages and within a page in order to make the information seeking process more efficient and effective. The feature provided by our browser for this purpose is the shortcuts to the annotated elements.

This feature aims basically in improving the browsing within a web page by simulating the layout scanning process of a sighted user. When a user browses an annotated web page he can listen to an overview of the page based on the annotated elements that exist in it, by pressing Alt+1. Then using the Alt+Up or Down arrows can move to each of them and start browsing its content. When for example he visits the page show in Figure 7 he listens that there are a main content area, a search box, a login form and various other elements. Then by pressing Alt+Down arrow he can move to the element he wants. If for example he wants to reach the main content area he has to browse through the elements and move to it. Once he hears the message “You are now in the main content area” he can navigate and listen to it using the up and down arrows. This reduces the time needed to reach the specific point if he was using the simple navigation within the page. In this case he should have “travelled” through every single bit of peripheral uninteresting information in the page and then reach the main content area. This way the overview presented in the beginning offers a set of choices of starting points to reading the content of a page similarly to what a sighted user does when visiting a web page.

SeEBrowser using browsing shortcuts provides also faster navigation through various elements of a page. Consider for example an end-user who starts reading the main content and judges it as not relevant s/he might need to move directly and use a navigational aid such as a menu in the page. Using SeEBrowser’s browsing shortcuts feature s/he can move to the desired element faster by simply navigating the annotated elements list. Without this feature the end-user would have used the simple navigation within the text in order to find a specific point, possibly a phrase that would signal the existence of a menu. Many blind users memorize distances in paragraphs or links for these elements in order to find them later using the start of the page as a landmark. Both ways are more time and effort consuming than SeEBrowser’s browsing shortcuts utility.

The mechanism hidden behind this feature is the annotations and their properties. Every class in the ontology has two standard properties. The first, named “ID”, identifies uniquely each annotated element from any other in the page. The second named “Description” contains a short description about the annotated element and is heard when the user reaches the specific element.

Furthermore, the groups of relationships presented in the OWL Ontology section allow some special management for some of the annotated elements. In particular the “contains” group of relationships indicates to the browser the existence of a hierarchy of classes. In our case, when a user listens to annotated elements found in a page, some of the annotated elements are excluded from the initial list. These are the elements that are contained within other container elements (e.g. items within a navigational menu). Users can find and hear a list of them only if they reach the corresponding container element. For example the blind user hears that there is a menu in the web page, but only after reaching the menu element and pressing Alt+1 again s/he listens that there are 7 menu items within the certain menu element. This allows the annotator to create a quite detailed description of a web page with a controlled level of granularity. It allows also blind users not to be overloaded by hearing elements that are not useful at a specific browsing moment. These elements will be hidden until discovered by the blind user while browsing within the page.

In the current phase of the research the first aim of the vocabulary and the annotations in pages is to improve browsing within a page. However, this improvement of browsing within a page sometimes leads to improvement of across document browsing. For example, pointing to a menu and describing the destination
web page of menu items, makes browsing within a site quite easier. In addition, when examining a search engine's result list the annotations of links to next, previous, first and last page improves a lot the browsing in it. The main benefit from these annotations is that they provide a quicker way of reaching important parts of a web page instead of having to listen to useless information to reach it.

5.4 Preliminary evaluation and experts testing
SeEBrowser was tested by blind users in an experiment presented in [22]. The usability evaluation indicated that the browser was found quite usable, easy to learn and especially the shortcuts feature rated as very helpful by all users.

Further examination of the log files led to some more findings. As seen in Figure 8 the percentage of keystrokes used when using annotations shows that most of the keystrokes are for movement within pages. Excluding these keystrokes (Up and Down arrows) we can have an analysis on the rest of the keystrokes. For the case of using the annotation that we see in Figure 9 we can say that most of the movement across pages are done by following links within pages and rarely going back to already visited pages. This might be come as a result of the structure of the experimental site which was quite simple. The use though, of annotation related keystrokes, shows that they were used almost in every page visited since the Alt+I and Alt+Down percentages are similar to those of EnterLink.

Another analysis on the speaking and browsing modes showed that most users had selected the combination of the whole page as browsing mode and paragraphs as speaking mode. There are very few cases of use of links browsing mode and sentences speaking mode. This could be caused by either the fact that the users weren’t too experienced with the application or because of the site’s construction that encouraged this combination. It is also important to say that at the time of the experiment the browsing modes for headings and forms were not implemented.

After further development of the application, we gave the tool for experimental use to a number of experienced blind users that would provide feedback through interviews. They were asked to use the tool both in a not annotated site and in an encyclopaedia site annotated for the next experiment. Summing up the feedback from the interviews there are some very important conclusions.

5.4.1 Percentages of moves using annotations

First of all, the modification in the bookmarks feature was welcomed and judged as very helpful. There was also a suggestion to implement the “history browsing” (going back and for in pages already visited) in a similar way. This means that when they move back to the previous page visited they are also transferred directly to the place (paragraph, sentence or word) they followed the link from.

Second the use of annotations in a more realistic environment such as the encyclopaedia was found much more helpful than in the first experiments environment. The use of relationships and annotated elements within other using the “contains” group of relationships was also rated positively.

A disadvantage of SeEBrowser was the lack of particular handling for tables containing tabular data. In our solution a table is read row by row and each cell in a row is presented as a paragraph. This leads to disorientation when navigating large tables. Currently, our solution deals with tables when used for layout because they can be annotated and convey their visual information to blind users too. The solution provided for tables containing tabular data was thought to be sufficient but the feedback we got is now suggesting that it’s not. This problem has been investigated by Filepp et. al. [4] and Yesilada et. al. [28] and there are already solutions suggested and could be adapted to our application. Information seeking often has to do with reading tables so this is a crucial part that needs improvement on our browser and it is rated as high priority in our further development.

Finally there was a suggestion for a notepad like feature. In particular they requested a mechanism that would help them in gathering information while seeking. The idea is that when the user finds an interesting part in a page he could mark it and move it directly to a notepad. After gathering an amount of information there, he could save it in a separate text file and further process it later.

6 FURTHER RESEARCH

6.1 Domain specific vocabularies
The next stages of our research include development of domain specific vocabularies and annotation of pages using them in order to improve even more the across document browsing. This can be done by using these annotations for providing functions similar to the scanning function of sighted user which is one of the basic functions in information seeking.
6.2 Automatic annotations

Our application framework depends heavily on the annotations produced by annotators. These annotations up to now are produced mainly by users that contribute them to our framework. It is easily understood that the large amount of information existing in the web and its daily update makes this job quite difficult even for a large network of human annotators.

To deal with this problem there are already thoughts and research in the field of automatic and semi-automatic annotation. The field can be separated in web content and web structure mining. In the field of web content mining there is already research from Huang & Sundaresan [12] and Mukherjee et. al. [17]. In the part of web structure mining there are interesting approaches by Pontelli & Son [19] and Kottapally et. al. [14]. We are currently investigating solutions in structure mining since it seems that in web page design there are specific patterns that are repeated numerous. One of them, that is widely used, is the desktop metaphor. If there can be a set of rules that could find patterns that usually present elements described in our vocabulary then the annotation could be done almost automatically. There will still be human interference in the annotation process to confirm the suggested annotations or correct them but a great load of work would be done automatically.

Up to now none of the suggested solutions for automatic web content or web structure mining is 100% successful to all pages and there are doubts if there will ever be such a solution. There are however, a number of solutions that present quite important percentages of success in specific domain web pages such as in [7] and [30]. This is a good indication that there can be solutions in other domains too.

6.3 Other uses of annotations

Another area of research to be followed is other possible uses of annotations apart the shortcuts functionality. An approach to this could be the formation of profiles that would reconstruct a web page according user needs. These profiles could either be constructed by users themselves, by annotators or even better by machines through semantic web usage mining.

The latter one is a field of research that is now starting to be exploited. For our project there is already a logging function enhanced in the browser that captures movement of users within and across document. These log files could provide valuable information especially for annotated web pages. They could provide users with navigational guidance based on previous visits and browsing in pages. A similar approach in web usage mining is presented by Spiropoulou [24]. In our research similar techniques could be used on log files from navigation within annotated pages. In general the layer of annotations created by annotators could provide valuable feedback when used by blind users.

6.4 Discussion

Summing up, the paper presented a framework that is based on the Semantic Web idea and may contribute to improving accessibility in the WWW. The framework utilizes various existing standards of Semantic Web such as OWL and RDF. But also it goes beyond the simple utilization of standards, by suggesting an architecture and an application framework that could be generalized to virtually any (Semantic Web) application.

We have presented a particular implementation of the framework that uses an ontology editor and a graphical annotation tool. These two components can be used in any condition and for developing any type of Semantic-Web application. As an example of such application we have presented a specialized voice web browser called SeEBrowser which particularly aims to make browsing blind users more efficient using browsing shortcuts.

However, the most important idea of the proposed framework is that it promotes and encourages the creation of a community that will work together having as their goal the improvement of accessibility. The framework also provides all the necessary tools to facilitate collaboration. This community consists of groups of people each having a specific role in it. Anyone willing to help can contribute from his or her own part, as an ontology creator, annotator, application developer or even end user. The power of this community is its independency and freedom from the current web authoring community. Our community is not tightly connected to the web authoring society, which is quite large and difficult to educate in accessibility issues. However, it can work independently upon the products of the web authoring society.

7 ACKNOWLEDGEMENTS

The research project is funded by the E.U. and the Greek Ministry of Education under the research program “Archimedes”[23]. We would also like to thank Dimitris Tektonisidis for his contribution to the project with the development of ONAR and Marios Chatzidimitriou for the development of the annotation storage server software.

8 REFERENCES

[1] Annotea project http://www.w3.org/2001/Annotea/


[23] SeEBrowser Project (Archimedes) http://erodios.it.teithe.gr/archimedes/English/Index.htm


Physical Usability and the Mobile Web

Shari Trewin
IBM T.J. Watson Research Center
P.O. Box 704, Yorktown,
NY 10598, USA
+1 914 784 7616
trewin@us.ibm.com

ABSTRACT
This paper examines the degree of overlap between good design for physical ease of access on the Web in general, and design for physical ease of use on the mobile Web. There are marked differences in the basic interaction techniques used and usability issues experienced. As a group, people with physical impairments tend to have a broader range of needs. These differences impact Web page design in various ways. Problems can be addressed in a unified way by designing for device independence. At least for physical ease of access, a unified set of mobile/accessibility best practice guidelines would be mutually beneficial. This approach may be helpful in preventing fragmentation of the Web.

Categories and Subject Descriptors
H.5.2 [Information Interfaces and Presentation]: User Interfaces – input devices and strategies, interaction styles

General Terms
Design, Human Factors.

Keywords
Mobile web, accessibility, usability, text entry, selection, navigation, physical disability, input devices, Web browsing, WCAG, UAAG.

1. INTRODUCTION
The challenges of designing for the mobile Web are strikingly similar to those of designing for accessibility. On the face of it, the mobile device’s small screen, different operating modalities and low communication bandwidth seem to mirror the access requirements of people with sensory, physical and cognitive impairments.

This paper focuses specifically on physical usability. It asks: How similar are physical usability issues for desktop users with physical impairments and for mobile users, when accessing the Web? Are similar solutions appropriate?

In the physical domain, accessibility issues are rarely clear cut or easily quantifiable. Keystroke navigation is a notable exception – if it is not provided, then people are excluded. Users with physical disabilities have very different input methods, but once they have a method that is best for them, usability issues center around the time and effort required to perform a physical task, and the accuracy with which it can be done. Making a task physically easier for existing users will also bring it within the capabilities of others who were previously excluded. So much depends on the individual’s patience and perseverance that it is impossible to say whether a given user with a given device, on a given day, will complete a Web based task successfully. As a result, this paper focuses on issues that make Web pages more or less usable, rather than accessible or not accessible.

First, this paper gives a brief overview of interaction techniques, issues, guidelines and solutions developed in the domain of physical Web usability (with the assumption of a desktop environment). Interaction techniques, issues, guidelines and solutions for mobile Web access are then presented and the two domains are compared.

2. PHYSICAL USABILITY
2.1 Interaction Techniques
Although the vast majority of desktop users provide input via a standard keyboard and mouse, there is an enormous range of other options available to people who find one or both of these devices difficult to use [7]. These options include alternative keyboards and pointing devices, speech input, keyboard-based pointing methods, and pointing-based typing methods (e.g. use of eye gaze tracking with an on-screen keyboard and selection by dwelling on an item). Users with very limited motion can use one or two binary signals generated by a physical switch, tongue switch, EMG sensor or other device, and can control a computer by scanning through the available options (manually or automatically) and selecting the items of interest.

Because all of these input techniques can provide text input, the standard way to accommodate this diversity is to ensure that browsers and Web sites can be accessed and controlled via keystrokes. However, providing keyboard navigation does not solve all physical usability problems.

2.2 Issues
One significant issue is speed of input. For switch users, typing rates may be just a few words per minute. Selecting on-screen items by dwelling on them typically adds up to 3 seconds to the time required for each selection. Some disabling conditions, such as Parkinson’s Disease, cause slowness of movement, so that typing rates on a physical keyboard can also be greatly reduced.

Accuracy is another major issue. People with motor impairments may have high error rates when using keyboards and mice [14], but often need or prefer to use these standard devices. Common errors include unwanted extra characters, and unintentional or wrongly positioned mouse clicks. Alternative input mechanisms such as speech, eye gaze pointing or EMG can also be inherently error prone or difficult to control. People with severe motor
impairments are often pioneers in the use of such technologies, at a stage when they are not robust or reliable enough to become more generally popular.

These broad issues give rise to some specific challenges when using the Web. Sites that give users a limited time to fill in and submit a form may be unusable. Some sites even present moving targets! Target size is an important consideration – the larger it is, the easier it will be to select. Typing passwords accurately can be a challenge, so sites that lock users out of their accounts after a number of failed login attempts also pose barriers.

It is also crucial that users are able to find information within a Web site as easily as possible. If it takes sixty seconds of hard work to select a link or scroll down the page, users need to know that they are navigating in the right direction. One major study of Web accessibility found that the two main usability problems reported by people with motor impairments were lack of clarity in site navigation mechanisms, and confusing page layout [4]. The third most commonly reported problem was small text and graphics, which would be difficult to select.

2.3 Solutions and Guidelines

Physical usability issues are significant even when the user has chosen the most accessible input mechanism available. Usability can be improved with support provided by the operating system, the browser, transcoding software, or by Web sites themselves.

Operating systems provide useful solutions such as the ability to control the delay before a key starts to repeat when held down. Assistive software such as word prediction can improve the user’s input rate and accuracy. Although there has been much research into mechanisms for making pointing and clicking easier [5,15], there is little operating system or browser support currently available. In particular, users cannot ask to have all targets increased in the way that text size can be increased. Desktop browsers render check boxes and radio buttons at a fixed, small size.

Desktop browsers do provide other physical usability supports, including the ability to increase the size of menus and scrollbars. The W3C’s User Agent Accessibility Guidelines (UAAG) [8] provide recommendations including

- design for device independence (e.g. allow keystroke-based navigation, which can be generated by many different assistive technologies);
- ensuring user access to all content (e.g. automatically pause a time-sensitive presentation to allow user input);
- and provision of direct and sequential navigation mechanisms (keyboard shortcuts and tabbing through controls).

Mankoff et al [11] experimented with a Web browser specifically designed to support Web access by users interacting via four keys or switches. It had an explicit control area for accessing browser functions and the links on the current Web page. This provided efficient navigation. It also showed an image preview of the link with focus, helping users to decide whether to follow that link.

Users are often unaware of browser and operating system features that may be helpful [4]. An alternative approach is to transcode individual pages before they are presented to users. Mankoff et al [11] identified several useful transcoding techniques for supporting low bandwidth input, including the addition of links to support backwards and forwards navigation at each paragraph break, other links for skipping unwanted sections, information about link targets, and specialized form widgets with built-in scanning capabilities. IBM’s Web Adaptation Technology [6], a client-side transformation system, includes Web page transformations that could improve physical usability such as changing the text size to make links bigger and linearization of pages to reduce horizontal scrolling. It also provides easy access to little known operating system features such as the Mouse Keys feature for keyboard-based pointing.

The W3C’s Web Content Accessibility Guidelines (WCAG) also cover physical usability. Relevant guidelines from the WCAG 1.0 recommendation [2] include

- providing a logical order for tabbing through page links;
- making event handlers device-independent (e.g. avoid actions that can only be triggered by a mouseover event, since some users will not have a pointing device);
- ensuring that users can control time-sensitive content changes (e.g. change the length of time allowed for filling in a form);
- using link text that is meaningful when taken out of context in a list of page links;
- providing shortcut keys to access important features;
- and providing navigation bars.

WCAG 2.0 [1] is a W3C working draft. It makes similar provisions. The guidelines focus on input device independence, and the ability to do keyboard navigation. The WCAG and UAAG guidelines do not address the significant problems of users with motor impairments who are using direct selection to select small non-text objects.

3. MOBILE USABILITY

3.1 Interaction Techniques

Interaction techniques for mobile users are also very varied. There are many different text entry mechanisms based on the phone keypad and novel input devices [10]. Some people use a stylus to make selections or gestures on a small screen. Some use speech commands. The most striking difference between mobile input mechanisms and those used by people with disabilities is that the very limited bandwidth input mechanisms are not represented.

Today’s typical mobile device is relatively limited. It may have no means for direct selection, relying on key-based navigation and keypad text entry. Future devices using different forms of display (e.g. head mounted projections onto eyeglasses) may take a very different approach, replacing the use of keys with direct selection, gestures, EMG input and/or greater use of speech.

3.2 Issues

Input speed and accuracy are both impacted in the mobile context. Keypad-based text entry mechanisms are slower than those for a full keyboard. Key based navigation takes longer than direct selection. It may also take longer (or be impossible) to follow a link, because of reduced communication bandwidth and limited memory and processing power on the mobile device itself. Many users are unaware of, or do not have, the ability to go back from a link they have followed [12]. This increases the cost of selecting a link in a similar way to the cost experienced by desktop users with motor impairments.
It seems likely that working in a mobile environment, especially on moving vehicles, will introduce errors in typing and navigation, but little has been published in this area. Techniques for error rate measurement appropriate for mobile devices are emerging [13] but have yet to be widely applied. Speech and gesture input mechanisms may be especially error prone in mobile environments with background noise and movement.

These general issues give rise to specific challenges when using the Web that echo some of those experienced by people with motor impairments. Sites that give users a limited time to fill in and submit a form may be unusable. Scrolling around a large page on a small screen is tedious and disorientating.

Other features of the mobile environment also generate issues that are similar to physical usability issues. The cost (in time and money and the difficulty of going back) of downloading pages to mobile devices makes it crucial that users are able to find information within a Web site as easily as possible, with a minimum of ‘dead ends’. Since many devices provide key-based navigation only, efficiency of key-based access becomes important.

3.3 Solutions and Guidelines

The W3C’s Mobile Web Initiative aims to make Web browsing from mobile devices a reality. It has drafted a set of guidelines for Web pages intended for mobile access [12]. These guidelines are in fact derived from the WCAG guidelines, and extend them by adding considerations that are specific to the mobile environment. The guidelines most relevant to input are:

- Keep the number of keystrokes to a minimum.
- Avoid free text entry where possible.
- Provide pre-selected default values where possible. (e.g. pre-fill a form field labeled ‘email’ with the user’s preferred email address when this is known)
- Specify a default text entry mode, language and/or input format, if the target device is known to support it.
- Create a logical tab order through links, form controls and objects.
- Use navigation mechanisms in a consistent manner (with the suggestion of using a “drill down” method and means to skip entire sections of content)
- Assign access keys to links in navigational menus and frequently accessed functionality

Mobile Web pages are recommended to conform to both the WCAG and Mobile Web guidelines.

While text input rate does remain an issue in the mobile environment, techniques developed to support people with low bandwidth input are already proving useful (e.g. word prediction, reduced key typing with automatic disambiguation).

Researchers have also investigated the potential of transcoding technologies to dynamically adapt Web pages for serving on mobile devices [9]. Although transformations focus on page reformatting to fit small screen devices, those providing a compact efficient navigation structure also make physical access easier. For example, consider a keyboard user looking for information on Collie dogs on a pet shop web page. She can get there in far fewer keystrokes if the navigation structure allows her to jump straight to the section on dogs without having to tab through every kind of cat along the way.

In addition, pages that are able to scale down to a small display screen with small elements will also be capable of adapting to a large display screen with large elements that are easy to select. This is an example of a transformation that is intended to support visual display but may also benefit physical usability.

4. COMPARISON

There are strong similarities between physical usability issues in mobile and accessible desktop Web browsing scenarios:

- They share the need to support a great variety of different input devices and interaction techniques. Both require Web pages that make no assumptions about the input mechanisms that will be used, and that support efficient key-based navigation and selection.
- Both communities also benefit from pages that are flexibly authored so that the layout can be adjusted, for example to fit a small screen, larger text size, or to minimize scrolling.
- Text input may be slower and more error prone than is normally assumed for a desktop environment, so user control of timeouts, techniques that reduce the amount of typing required, and opportunities to correct errors are all useful. For example, pre-filling form fields with previously used entries will benefit both communities.
- Navigation is likely to be slower, and the cost of following wrong links is high. Pages that provide clear and efficient navigation mechanisms with the ability to skip sections will benefit both communities.

However, physical usability in a mobile environment is not just a rehash of desktop physical usability issues. There are some significant differences:

- The very low bandwidth physical input mechanisms used by people with disabilities are not appropriate for mobile users, who already have higher bandwidth mechanisms available. For example the use of gestures made with a pen allows text input at rates far higher than scanning an on-screen keyboard. In the future, the mobile community may be less concerned about reducing the amount of typed input, because more efficient text entry mechanisms will be available, while it will remain an issue for people with severe physical impairments.
- The basic assumptions about mobile text input are very different to those of physical usability on a desktop. In the mobile scenario, typing is generally slow but little is known about error rates. The slower desktop typists are much slower than mobile users, and error rates can be very high. Three attempts to enter a password may be sufficient for mobile users, but not for desktop users with typing difficulties.
- Difficulty with direct selection is a major issue in physical usability, while direct selection is not often used on today’s mobile devices. Where it is used, a high level of precision can be assumed. Users with physical impairments are more heterogeneous. Whether using key-based or direct selection, long selection times
and high error rates must be accommodated. Techniques used by people with disabilities to make it easier to point to targets may also be helpful to people with no impairments when pointing to very small targets. So developers of devices with small, high resolution screens and direct selection facilities may find techniques of interest in the accessibility field.

In general, where similar needs are identified, physical usability issues represent a broader range of requirements. Enhancements to physical usability will often benefit mobile users.

Designers building pages specifically for mobile delivery may be tempted to use techniques that actually introduce accessibility problems in other contexts. For example, it might be useful for mobile Web pages to use small images for links that are to be displayed on a small screen, but extra large links are preferable to make desktop access physically easier. A more appropriate general requirement would be to allow target size to be adjusted depending on the delivery context. Future browsers that support such scaling would then adjust target sizes to suit the current user and device capabilities.

Many of the requirements of the two communities lead to very similar general Web page design guidelines, the overriding principles being to avoid making assumptions about input mechanisms, to support a variety of navigation mechanisms, and to support users in providing input wherever possible. Combining the W3C WCAG and current Mobile Web best practice guidelines will actually enhance physical accessibility for all users, not just those in mobile contexts.

5. CONCLUSIONS

This paper set out to explore the similarity between physical usability issues for desktop users with physical impairments and for mobile users, when accessing the Web.

The physical interaction techniques used by the two groups are very different, but often give rise to similar input rate and accuracy issues. Some text entry techniques pioneered by people with disabilities have already been adopted for mobile access, and this trend is likely to continue in the future.

However, it would be wrong to conclude that the issues are basically the same, and that the same solutions are always appropriate in both domains. People with physical impairments tend to have a more extreme range of needs and requirements. Extremely low bandwidth input using a single switch and scanning, for example, produces typing that is much slower than that of mobile text entry mechanisms. People with mild to moderate motor impairments rely heavily on direct selection, which is not currently common for mobile Web users.

These differences do impact Web page design. Designs that are highly optimized for the mobile context will be less physically usable where the needs of the two communities pull in opposite directions. Guidelines and approaches that emphasize flexibility typically benefit both communities.

It seems feasible that these two communities could develop a single set of mutually beneficial guidelines and techniques for maximizing physical usability. This may encourage designers to produce pages that can be used in multiple delivery contexts. If this happened, users with physical disabilities would find the Mobile Web a welcoming place.

6. REFERENCES


How People use Presentation to Search for a Link: Expanding the Understanding of Accessibility on the Web

Caroline Jay†, Robert Stevens‡, Mashhuda Glencross† and Alan Chalmers†

†School of Computer Science
University of Manchester
‡Department of Computer Science
The University of Bristol
E-mail: caroline.jay@manchester.ac.uk

ABSTRACT

It is well known that many Web pages are difficult to use by both visually disabled people and those who use small screen devices. In both cases there exists a problem of viewing a great deal of information with presentation capabilities reduced from the intended formatted large-screen colour display. It is pertinent, however, to ask how the presentation of Web pages on a standard display makes them easier for sighted people to use. To begin to answer this question, we report on an exploratory eye-tracking study that investigates how sighted readers use the presentation of the BBC News Web page to search for a link. We compare the standard page presentation with a “text-only” version and observe both qualitatively and quantitatively that the removal of the intended presentation alters “reading” behaviours. The demonstration that the presentation of information assists task completion suggests that it should be re-introduced to non-visual presentations if the Web is to become more accessible. Finally, we propose that models derived from studies that reveal how presentation is used to aid task completion can form the basis for annotation and transcoding of Web pages to present pages in a more usable non-visual form.

Categories and Subject descriptors

K.4.2 [Computers and Society]: Social Issues Assistive Technologies for Persons with Disabilities; H.1.2 [Information Systems]: User/Machine Systems; Software Psychology; H.5.2 [Information Interfaces and Presentation]: User Interfaces; Evaluation/methodology; H.5.2 [Information Interfaces and Presentation]: User Interfaces; Screen Design; I.7.5 [Document and Text Processing]: Document Capture; Document Analysis

1. INTRODUCTION

In this paper we explore the problems encountered by people with profound visual disabilities when browsing Web pages in a manner that replicates interaction with small screen devices. We ask two questions: firstly, what is it about audio screenreader presentation of a Web page that really makes it difficult to use; secondly, what is it about the standard presentation that is missing and needs to be replaced? The obvious and unattainable answer is that the external memory provided by paper or a screen, together with the speed and accurate control of information flow afforded by the human visual system is required [17]. All assistive technology for visually disabled people attempts, in some way, to provide some kind of replacements for these fundamental facilities.

The difficulties with access to Web content by visually disabled people is well documented [6, 5], however what is less well documented is the exact nature of the problems encountered by those users. Profoundly Visually disabled people usually use a screenreader† (such as Jaws [1, 8]), which speaks screen content under the direction of the user, in order to “read” what is on the screen. When a Web page is loaded into a browser, by default the page contents are typically spoken from top-left to bottom-right. Naturally, such a spoken presentation is difficult to use—memory cannot handle such quantity and review is not really possible. Of course, the screenreader allows finer control over what is spoken than this cumbersome mechanism. A user can move around a page using cursor keys. This allows movement at the level of lines, characters, words, paragraphs, etc. It would seem that all parts of a page can be accessed; so, what is the problem?

The screenreader access described above is slow and cumbersome, but still presented as difficult. What makes such access difficult? Much effort in Web accessibility has focused on access to content. This includes access to graphics; tables; figures; etc. Much has been achieved, but access to Web pages continues to be a problem; why is this?

In a text-only presentation (such as that shown in Figure 1), the majority of the formatting and layout is removed [19]. The user is given what amounts to a traditional linear document, but with little differentiation between text serving different purposes, such as headings, plain text and emphasised text that usually supports the processing of the printed information [20]. Through eye-tracking studies, we have examined how readers use a Web page’s presentation to accomplish their task. We have compared this with how the presentation of the text-only version of the same page is used

†In this article we restrict ourselves to those users who are profoundly visually disabled. Assistive technology for users with impaired vision cover a broad range of tools including magnification, colour or contrast enhancement, etc.
to accomplish the same task. The latter simulates, to an extent, both the presentation on small screen and the experience of a visually disabled person using a Web page through reduced visual effects and reduced information content on the screen.

The presentation of a page generally facilitates a sighted user in achieving their goal, be it searching, browsing, following links etc. It is well known from the principles of graphic design and information presentation that how content is laid out can influence how a task is performed and what interpretation is made of data [20]. If we are to support visually disabled and small-screen users to use Web pages it is important to characterise the situation beyond “difficult”. We would like to have insight into how Web presentation is used by sighted users. It is not the presentional qualities themselves that need to be re-introduced, but what users do with those formatting and layout features. We attempt to do this by examining eye-tracking data during the exercise of basic Web based tasks on standard and text-only versions of a page—which features are looked at and for how long in each presentional style.

Once we know the difficulties a lack of presentation causes, it will be easier to know what to replace or re-introduce into audio presentations. Our eye-tracking studies have produced a vast quantity of data, of which we examine only a portion here. From these studies, we can see how a human uses the page’s presentation during navigation and orientation. From our observations of how readers use presentional information to orientate and navigate in order to accomplish tasks, we propose that eye-tracking studies can be used to provide models of page usage that can be applied to different styles of presentation in order to transcend those pages. It is known that such transformations can, to some extent, replace some of the facilities of a standard Web page presentation [2]. Our investigations should provide a basis for driving such transformations.

In Section 2 we describe the context in which this work has been performed. We then describe our experiment: its design, procedure and equipment (see Section 3). Our results are presented in Section 4. A discussion of the results is given in Section 5.

2. RELATED WORK

Profoundly visually impaired users usually access the Web using screen readers [8] or specialised browsers [1]. There are many sources that state the difficulty that such users have using the Web via tools such as screenreaders. The Disability Rights Commission (DRC) report of 2004 [6] concludes that most UK Web sites (81%) fail to satisfy even basic accessibility requirements. A similar situation exists in the USA [5]. Coyne and Nielsen [5] concluded that “the Web is about three times more difficult to use for people who are visually impaired than it is for sighted users”. The question still remains, however, as to exactly what makes it so difficult.

One part of the answer is in simply extracting information encoded within the HTML of a Web page. Thatcher et al. [19] give practical advice on constructing accessible Web pages by elucidating guidelines [4] and using HTML to best advantage. Text-only alternatives to standardly formatted Web pages are offered as “accessible” options. However, [19, Chapter 1] states the issue of text only versions crosses into the idea of separate versus inclusive design. Why should visually disabled people use different pages to sighted users? As well as problems of maintenance etc., there is a resistance to the development of Ghettos. [15] observe that, “designers look at sites that are meant to serve as models of accessibility and are appalled by the aesthetics. For most designers, accessibility equates with boring, uninteresting designs. The state of accessibility on the Web today represents a failure of the imagination”.

Even when pages are built according to guidelines that are meant to increase accessibility, there seem to be “difficulties”. Takagi et al. [18] classified the problems into three categories:

1. Adherence to guidelines, not usability issues;
2. Over-reliance on syntactic checking of Web pages;
3. No attention on the time component in the operations provided to aid access.

These observations usefully describe some of the sources of problems encountered by visually disabled users without really describing the nature of those problems.

The problem is summed up by Hanson [7], who says, “specifications for accessibility of Web pages do not necessarily guarantee a usable or satisfying Web experience for persons with disabilities. It is not uncommon to have pages that meet standards but are still difficult to use by persons who have difficulties.”

Harper et al. [3] and Yesilada et al. [22] have used the metaphor of travel to raise the notion of using a Web page above that of dealing with mere ‘sensory translation’. Just as people use travel objects in the environment (signs, landmarks and other cues) to help them orientate and navigate, so a Web user can use travel objects on a Web page to aid mobility—the ease of travel in a Web page. The layout and presentation of a Web page provides these travel objects. A well designed page eases travel and a badly designed page hinders travel and decreases mobility. A text-only page will tend to lack more of these travel objects and thus the reader is again hindered in their task.

Screen readers, unlike sighted users, cannot see the implicit structural and mobility knowledge encoded within the visual presentation of Web pages [21]. “It is impossible for blind users to distinguish visually fragmented groupings only from the sequence of tags read to them [2].” So, we begin to see that basic sensory translation of what is encoded within HTML still lacks what is necessary to support usable access to the Web. We can identify the Web correlates of what in printed material enables and supports effective information processing [20], as something lacking in current non-visual renderings of Web based material.

We have used eye-tracking in this study because we wish to find out how sighted users exploit the visual presentation of a Web page. Once we have an insight into what it is that visually disabled people can no longer do without this information then it is possible to begin to plan to replace those facilities. Whilst eye tracking has been used to investigate cognitive processes for over 100 years [14], recording the pattern of fixations on a Web page is a powerful tool, enabling us to determine those areas that are most salient (attract the most fixations), and those that receive little attention.

Its most obvious applications are in improving the standard design and layout of Web pages, and evaluating their usability [16]. Studies have also examined the saliency of items on a page under varying conditions (e.g., [9, 12, 10]), how eye movements vary according to information scent [13] and how looking for a menu is influenced by page complexity and prior expectations [11]. To our knowledge, no one has investigated how gaze patterns differ when searching on either the standard or text-only versions of the same page. Here, we do just that, by tracking participants’ eye movements as they search for specific links on the standard and text-only versions of the BBC News Web site.

3. METHODOLOGY

Eye-trackers provide an effective means of recording where a user’s gaze falls on a particular part of a computer screen, and hence
a good indication of the location of his or her attention. The num-
ber of fixations an area receives indicates the amount of attention
it attracts; the average duration of these fixations corresponds to its
complexity (more complex stimuli require longer fixations to pro-
tect) [14].

In the current study, participants’ eye movements were tracked
while they were searching for a link on either the standard or text-
only versions of the BBC News Website (see Figure 1). The
time to locate the link provided a measurement of task difficulty. The
number of fixations indicated the amount of attention required to
find the link, and the average duration of fixations indicated the
relative complexity of the information presented on each page.

3.1 Equipment

The equipment used for the study is displayed in Figure 2. The
experiment was run on a SONY VAIO VGN-FS315S laptop. Stim-
uli were displayed on a SONY VAIO SDM-HS75P monitor and
a Tobii x50 Eye Tracker, positioned at the base of the monitor,
tracked the participant’s gaze. The Tobii ClearView Analysis soft-
ware was used to record and analyse eye-movement and event data.

3.2 Design

The study used a between-subjects design in which participants
searched for links on the BBC News Manchester Web page (see
Figure 1). This site was chosen as it is widely known in the UK, but
the actual content on any one day will change. BBC pages are also
automatically provided with a text-only page. Sighted users, de-
spite having the facilities of the visual system outlined in Section 1,
will experience much of the effect of an audio screenreader render-
ing through experiencing a text-only version. The text-only version
will also reflect the effects of a small-screen rendering, irrespec-
tive of the BBC’s provision of a version for small-screen devices.
This mimicry arises from the text-only version having little of the
visual formatting of the standard presentation and the consequent
reduction of information content. Of course, the correspondence

http://aig.cs.man.ac.uk/people/jayc/bbc_study.htm
between sighted users reading a text-only page and the reality of profoundly visually disabled people’s use of an audio screenreader is only partial at best. It does, however, provide some flavour of the interaction.

Half of the participants searched for a link to the BBC Manchester Website on the standard version of the page, and a link to a story about the Chinese community on the text-only version of the page; the other half searched for the link to the BBC Manchester Website on the text-only page, and the link to the story about the Chinese community on the standard page. The links were chosen because they were next to each other on the page, and positioned on the right in the standard version and approximately two-thirds of the way down in the text-only version, so participants would not see them immediately on entering the page. The presentation order of the two site versions alternated between users. At the end of the experiment, users were asked to indicate whether they used the BBC website more than once a week, and whether they found it easier to perform the search on the standard or text-only version of the page.

3.3 Participants and Procedure

Eighteen participants between the ages of 17 and 50 with normal or corrected vision took part. The study was carried out as part of visit days by prospective students to the School of Computer Science. Both prospective students and their parents were asked to volunteer to take part in the study. The procedure was explained and the study performed, which took about five minutes. Participants were shown the recording of their eye-movements after the experiment.

Each participant sat 50 cm from the monitor and went through the calibration process. The participant was told to look for either the BBC Manchester or Chinese community link. The participant entered the appropriate version of the page and started searching, indicating that he or she had located the link by hovering over it with the mouse. The participant then searched for the other link on the alternative version of the page.

4. RESULTS

The eye tracker recorded the position of the participant’s gaze on the Web page throughout the experiment. We used the ClearView analysis software to calculate the number of fixations that occurred (a fixation is where the eyes rest on part of the stimulus for more than 100 msec, in order to process information), their position, their order, and their average duration. Figures 3 and 4 show the areas of the page that received the most fixations in the standard and text-only versions of the page respectively (red = 9 or more fixations, green = 4 or more, grey = 0). Participants tended to fixate on the salient areas when searching the standard page: headlines, some images, and prominent words in the text. The areas with the most fixations also appeared to vary according to the link participants were searching for - when looking for the Manchester link, they looked more at the menus, as this may seem a logical place from which to navigate to a BBC regional Website. On the text-only page, participants fixated in a uniform manner in nearly all of the text on the left-hand side of the page, indicating that it was allocated equal importance as they read down from the top until they found the link.

Qualitative analysis of the gaze replay and fixation order data confirms that participants simply read down the page in a serial fashion when looking for the link on the text-only page. On the standard page, however, participants’ eyes dart around as they attempt to locate what appears visually to be the next most likely location for the link. The gaze plots in Figure 5 illustrate this process for participants searching for each link on the standard page and those in Figure 6 show the same two participants searching for the other link on the text-only page. When looking at the standard page, participants will make large saccades to move from one salient area to the next; on the text page however, saccades are much smaller, as the participants read through the text from top to bottom.

A two-way ANOVA (page × link) showed that participants made significantly more fixations on the text-only site, regardless of the link they were searching for, than they did for the standard site ($F_{1,32} = 2.183, p<0.005$). This suggests that the text-only version of the page required more of their attention to process (see Fig-
The duration of a fixation, however, lasted significantly longer on the standard page ($F_{1,32} = 0.208, p<0.005$), indicating that a greater amount of cognitive processing was occurring in a fixation on the standard page than the text-only page (see Figure 9). It may be that the information obtained in a single fixation on the standard page was more complex, or that some of the extra processing time was due to participants orientating themselves and planning their navigation to the next part of the page (fixations during visual search are known to be longer than fixations during silent reading [14]).

The majority of participants found it easier to search for the link on the standard version of the page (see Figure 8). It is important to acknowledge that familiarity with the page may have had a strong influence - all but two of the participants used the BBC website more than once a week, and of those who did not, one rated the text-only page as easier and the other did not express a preference. However, the possibility that participants found it easier to search the information in the standard page because of the way it was laid out cannot be dismissed.

Figure 4: Hotspot analysis of text-only presentation of the BBC Manchester Web page.

Figure 5: Gazeplots for participants searching for the Manchester and China links on the standard BBC Manchester Web page.

Completion times (shown in Figure 10) did not vary significantly as a result of page type. Closer analysis shows that this result occurred due to the serious difficulty two participants had locating the Manchester link (taking more than 50 seconds to find it, in contrast to the 13 seconds it took the other participants). This may have arisen due to the prior expectations of the participants. Both spent a long time looking at menus, convinced that the link should be located on one, rather than positioned on the right of the page. A T-test considering the China link searches alone shows that participants locate it significantly more quickly on the standard page ($t_{16} = 3.696, p<0.005$). It is also worth considering the fact that the two pages varied not only in layout, but also in colour. Sighted users, used to seeing white text on a black background, may have have had more difficulty processing yellow text on a black background.
5. DISCUSSION

An Audio rendering lacks the very facilities that make the visual system the predominant human information gathering system, that is, the ability to control information flow from the outside with speed and accuracy, because there is an external memory holding the information [17]. In the case of Web pages, there is the additional factor of even being able to access what is being presented—the focus of much of the work reviewed in Section 2. Current audio renderings lack the formatting and presentation available in the visual mode: Spacing; typefaces; font-size; colour; etc. Non-speech and audio cues can potentially add some of this information, but this has not so far been greatly exploited in the rendering of Web pages.

In this exploratory experiment we have attempted to further characterise the problem of Web accessibility. In this study qualitatively we see a stark difference in eye-movement behaviour in the
two conditions. In the standard, formatted presentation (preferred by the majority of participants), saccades are greater and fixations longer. The link menu is being used, along with the distinct representational fragmentation of the page into areas. In contrast, the text-only page provoked eye-movements more akin to those seen in reading of ordinary printed text. The page is being “read” rather than being used to navigate towards the link goal. The speculation is that a sighted user can use the formatting of the page to achieve the task and that the longer gaze duration is a consequence of decision making on orientation and navigation. The text-only page is in effect similar to that which a blind user encounters in an audio presentation. The eye-movements seen in this condition are similar to those seen in cursor movements during reading. The lack of opportunity to “dart” around the page to orientate and navigate are removed.

Text-only versions of Web pages have been advocated as options for presentation on small-screen devices. The observations of this investigatory study would suggest that this is not sufficient for a usable interaction with a page on a mobile device. Similarly, for visually disabled users, a text-only version of a Web page does make the information available, but using that information is seen to be difficult without the facilities that a formatted presentation affords a sighted Web user.

So the question is how to replace the facilities afforded by presentation to both visually disabled users and those sighted people who use small screen devices? Layout provides borders for portions of information. Ideally, the objects on a page guide a reader through the information such that he or she can accomplish their task [20]. This is the basis of the use of the travel metaphor [3] in increasing Web accessibility. Here, the travel objects that facilitate movement through a Web page are identified and semantically marked up [22]. This markup is then used to help transcode a page into either an order that facilitates use or a fragmentation that facilitates use.

Eye-tracking studies have confirmed our hypothesis about the utility of a standard and a text-only presentation. One way of bringing this information back into a presentation is via transcoding. Does this eye-tracking data provide any mechanism by which transcoding can be driven? Obviously, markup and transcoding cannot occur by hand alone. Our aim is to see whether we can create models of how presentation is used for navigation and orientation about a page such that we can look at a page’s presentation and infer how the presentation will be used. Such models will be able to drive the transcoding of a page and begin to re-introduce into an audio rendering the same support for tasks that a sighted user has in representational cues for task completion.

Acknowledgements: Caroline Jay is funded by EPSRC grant EP/D036518/1.

6. REFERENCES

[22] Yeliz Yesilada, Robert Stevens, and Carole Goble. A
Contextual Web Accessibility -
Maximizing the benefit of Accessibility Guidelines

David Sloan
Digital Media Access Group
University of Dundee
Dundee, UK
+44 1382 385928
dsloan@computing.dundee.ac.uk

Andy Heath
Axelrod Research and Consulting
51 Paterson Close
Stocksbridge, Sheffield, UK
+44 7881 955997
AndyHeath@axelrod.plus.com

Fraser Hamilton
Designed for All Ltd
109-111 Farringdon Road
London, UK
+44 2078 417417
fraser@designedforall.com

Brian Kelly
UKOLN
University of Bath
Bath, UK
+44 1225 383943
b.kelly@ukoln.ac.uk

Helen Petrie
Department of Computer Science
University of York
York, UK
+44 1904 434336
helen.petrie@cs.york.ac.uk

Lawrie Phipps
JISC Executive
Beacon House, Queens Road
Bristol, UK
+44 117 954 5078
l.phipps@jisc.ac.uk

ABSTRACT
We argue that while work to optimize the accessibility of the World Wide Web through the publication and dissemination of a range of guidelines is of great importance, there is also the need for a more holistic approach to maximizing the role of the Web in enabling disabled people to access information, services and experiences. The persistently disappointingly low levels of usability of Web content for disabled people indicates that focusing on the adoption of accessibility guidelines by content authors, tool developers and policy makers is not sufficient for a truly inclusive Web. This approach fails to acknowledge the role of the Web as an enabler in a broader context and may stifle creative use of Web content and experiences to enhance social inclusion.

Using e-learning as an example, and describing current metadata developments, we present a framework that will guide Web authors and policy makers in addressing accessibility at a higher level, by defining the context in which a Web resource will be used and considering how best existing or new alternatives may be combined to enhance the accessibility of the information and services provided by the site in question. We demonstrate how guidelines such as those produced by the W3C’s Web Accessibility Initiative have a role to play within this wider context, along with metadata and user profiling initiatives.

Categories and Subject Descriptors
H.5.2 [User Interfaces – Evaluation/methodology]; K.4.2 [Social Issues - Assistive technologies for persons with disabilities]

General Terms
Measurement, Documentation, Human Factors, Standardization, Legal Aspects, Verification.

Keywords
Web accessibility, people with disabilities, WAI, WCAG, guidelines, methodologies, metadata, contextual design.

1. INTRODUCTION
Accessibility of the World Wide Web (the Web) to end users, regardless of ability or browsing environment, is widely accepted as a fundamental requirement if the Web is to reach its true potential as an enabler for the widest possible audience. Indeed, the rights of Web users with disabilities are becoming ever more defined in anti-discrimination policy and legislation around the world [49].

Unfortunately studies have regularly shown that the accessibility of Web sites falls short of an acceptable level. This is despite an ever-increasing quantity of work that has gone into:
- developing tools to support Web content providers in authoring accessible material and evaluation of the accessibility of content [46].
- developing assistive technologies for end users, whether hardware or software, adaptations or extensions to browsers, or server-side transformation tools to allow disabled people to overcome or reduce the impact of their impairment when interacting with Web content.
- disseminating the importance of, and best practices in, accessible Web design, e.g. [34], [44].

The reasons behind the continuing disappointing levels of Web content accessibility have been widely discussed (e.g. [13]). What seems clear is that, while still a factor, a lack of awareness of the importance of accessibility amongst Web developers and site commissioners is no longer the predominant issue. A key challenge is effective and appropriate implementation of accessible Web design techniques. Other challenges include the perceived complexity and cost of the task of making a Web site accessible [28], and also the need to unambiguously define what is actually meant by ‘accessible’, understand what is required to develop a Web site to be considered to have met that definition, and to evaluate it such that one can judge whether it has met the specified level of accessibility [3].

From the disabled Web user’s perspective, complicating factors include the browsing and assistive technologies available, the
user’s ability to use these technologies, and the difference between the technologies available and those most appropriate for the user’s needs.

2. W3C, WAI AND WEB ACCESSIBILITY

2.1 The WAI Model of Accessibility

As the body responsible for the coordination of developments to Web standards, the World Wide Web Consortium (W3C) has taken a lead in promoting accessibility of the Web for disabled people, not only as Web users, but also as Web authors. Since 1997, the W3C’s Web Accessibility Initiative (WAI) has been extremely active and very successful both in raising awareness of the importance of Web accessibility and in developing a model which can help organizations to develop accessible Web resources. This model provides guidelines which can be used to ensure that Web resources are optimally accessible, as well as influencing the development of other W3C standards for Web technologies.

The WAI promotes a tripartite model of accessibility, with the goal of universal Web accessibility in theory provided by full conformance with each of three components [7]. Of particular relevance to developers of Web resources is the Web Content Accessibility Guidelines (WCAG) [52]. WAI has been successful in promoting the WCAG around the world, the guidelines having been adopted by many organizations, embraced by the Web Standards movement [55], and are increasingly being adopted at a national level [49].

In the WAI model, the WCAG is complemented by accessibility guidelines for browsing and access technologies (the User Agent Accessibility Guidelines, UAAG [51]) and for tools to support creation of Web content (Authoring Tools Accessibility Guidelines, ATAG [46]). This approach acknowledges that in addition to providers of Web content, developers of authoring tools and of browsers, media players and access technologies also have responsibility towards the provision of accessible Web content.

2.2 Shortcomings of the WAI Model

Although WAI has been very successful at a political level, and to a large extent at a technical level, the authors feel that the model of Web accessibility adopted by WAI is flawed. This is due partly to the nature of the WCAG and also to the overwhelming domination of the role of WCAG with respect to the other guidelines in the accessibility of online information and services.

2.2.1 Limitations of WCAG

Shortcomings of the WCAG have been noted by a number of commentators (e.g. [8], [12]), and documented by the authors [25]. In brief, these include:

- **Theoretical nature of the guidelines**: current guidelines promote the use of open, W3C standards, and ignore widely used proprietary technologies, many of which have made significant improvements in accessibility support in recent years.
- **Dependencies on other WAI guidelines**: conformance of a site to WCAG may not be enough to ensure optimal accessibility if this conformance results in a site that requires use of a UAAG-conformant browser.
- **Ambiguity of the guidelines**: It is well documented that there is a significant degree of subjectivity in applying many WCAG checkpoints; and some are conditional on levels of support for a feature across browsing technologies.
- **Complexity of the guidelines**: The organization of the guidelines (each with constituent checkpoints of varying priority) means that the WCAG can be difficult to understand and apply to a particular situation. We expect this to be remedied in future versions of WCAG, but interpretation difficulties may be hard to avoid.
- **Logical flaws of the guidelines**: The content of some WCAG checkpoints is such that they are open to quite extreme interpretations.
- **Level of understanding of accessibility issues required**: Given the range in technical abilities of the intended audience (anyone who creates Web content), there may be significant cognitive demand on understanding the principle behind a specific checkpoint, and applying that in a particular situation. Indeed, the format of the WCAG was found several years ago to present developers with problems of interpretation [10].

At this point we must acknowledge the ongoing development of WCAG 2.0 [53]. We have confidence that, when published, WCAG 2.0 will address many of the problems highlighted above. We are less confident that the release of a revised set of guidelines will overcome the wider issues addressed in this paper.

2.2.2 The dominance of WCAG

The WAI model also places significant responsibility on end-users to be aware of the technologies that most suit their needs. The need for improved user awareness, through better training and support, was a point noted by the UK Disability Rights Commission’s Formal Investigation into Web site accessibility [13]. It has also been echoed by many developers promoting Web standards (e.g. [29]) who express frustration over the lack of user awareness and uptake of standards-conformant browsers and assistive technologies, of accessibility features of browsers and operating systems, and the resulting impact on the effectiveness of moves to promote adoption of Web standards.

Nevertheless, Web developers remain comparatively powerless to persuade end-users to adopt conformant browsing technologies. The lack of awareness about assistive technology and more capable browsing technology amongst the wider Web-using population has been found to be a significant issue [31]. Developers are limited to attempting to provide bespoke accessibility features such as large-print style sheets [9] or audio-enabled versions of their pages, for example through Readspeaker [37].

Since usage of UAAG-conformant browsing technologies cannot be relied upon, and usage of ATAG-conformant authoring tools can neither be relied upon nor guaranteed to produce WCAG-conformant content, WCAG remains the ultimate standard which developers must meet. Even then, accessibility problems may remain, as discussed further in Section 3.

2.3 Alternatives to WCAG

The W3C process of developing its “Recommendations”, in the form of specifications is a lengthy, but thorough and robust process, involving domain experts and public consultation. Yet evidence used to support decisions made, for example in formulating the WCAG, is not made explicitly available as part of the guidelines or the supporting documentation, and thus the
guidelines have been criticized as lacking empirical evidence [32].

Some sets of guidelines have applied the WCAG to a particular environment, for example the IMS Guidelines for accessible e-learning [18]. In addition, research-based guidelines for Web accessibility and usability for groups who would appear to be catered for by the WCAG, including disabled and elderly people, have emerged over recent years, (e.g. [11], [27], [32] and [45]).

Some of these guidelines are broadly in agreement with the WCAG; some are more prescriptive, while there are also situations whereby comparison of these guidelines may result in potential conflict between the needs of specific user groups. Even the fact that the research teams producing the above guidelines felt the need to carry out the work indicates a level of dissatisfaction with the nature and content of the WCAG, in particular with respect to lack of published supporting evidence for specific checkpoints.

This situation reinforces the need for guidelines to be perceived to be useful and usable by developers - in terms of being credible, proving sufficient information on the consequences of following (or ignoring) particular checkpoints, and clearly defining the scope of applicability of a particular checkpoint.

3. UNIVERSAL USABILITY OR INCLUSIVE DESIGN?

3.1 Accessibility? Or Usability for Disabled People?

The relationship between accessibility and usability has long been a source of discussion, and as yet no definitive model exists [4]. However, it may be an unnecessary and artificial distinction in practice [1], [44]. This is illustrated by a discussion of the findings of the DRC’S Formal Investigation into Web Site Accessibility [13]. The results produced two telling conclusions:

1. Some problems encountered by disabled people could not directly be matched to any single checkpoint of the WCAG. A WAI response [50] argued that many of these problems could be caused by non-conformant user agents and authoring tools. However, an expectation that all disabled people use conformant user agents is, we would argue, hopelessly idealistic at the present time, not least given that if a UAAG-conformant user agent exists, the likelihood that many disabled people would be aware of it, and have the means to obtain it, install it and use it, would be low.

2. Some Web sites were found to perform extremely well in usability evaluations with disabled people, yet did not meet certain WCAG checkpoints. When these sites were announced some commentators (e.g. [54]) criticized the DRC for hailing as examples of best practice sites that did not meet basic conformance levels of the WCAG. Thus, the somewhat strange situation emerged whereby sites that were found to be usable by disabled people were nevertheless rejected by Web developers as being inaccessible!

The WCAG may imply that full conformance will result in a universally accessible Web site, yet the very goal of universal accessibility has been questioned. For example, Newell and Gregor [34] acknowledge the limitations of universal accessibility by proposing a new approach of “user-sensitive inclusive design”, while Nevile [33] describes the problems that can face an attempt to create and provide universally accessible resources, noting that:

“...it is not conceivable that even the new guidelines (WCAG 2.0)...will solve all the problems. Perhaps it is just not sensible to expect such guidelines to anticipate every solution and provide for it.”

3.2 One Site for All – Always?

The one-site-for-all approach to accessibility has, correctly in the authors’ opinion, largely conquered the perception that accessibility can be achieved through a separate stand-alone text-only Web site. Yet advocates of the single-site approach have shown a remarkable intolerance of attempts to use the Web to reach out to neglected groups, in particular people with severe learning difficulties, and who may have requirements quite distinct from most other Web users.

For example, a recent request was made to a Web accessibility email discussion list for feedback on a site designed for children with severe learning disabilities. This request produced feedback that was extremely critical of the appearance and content of the site. Yet while the distinct lack of evidence-based guidelines for Web site design for people with learning disabilities has been pointed out [8], [39] the reaction to the email request indicated an expectation that the resultant site should be equally understandable and usable by the members of the list as well as by the target audience.

We argue that this appears to be a prime example where the Web can be used to enhance the quality of life for a specific group of disabled people through providing information and entertainment, and facilitating communication, yet in a way that may present other people with significant usability problems. Here, the Web is enhancing accessibility for disabled people – but not through a universally accessible Web site.

In fact, there appears to be increasing use of solutions such as text-transcoders, text-to-speech features and alternative style sheets to provide on-the-fly alternative formats of one single site. This could be seen as a realization that designing one WCAG-conformant site for all, and leaving users to adopt UAAG-conformant browsers, is not a realistic approach at the present time.

3.3 Context of Use

Designing digital systems to meet the requirements of the people who will be using it is the classic approach to usability. One can group requirements into several categories, including:

- User characteristics: the abilities (and disabilities) of the target users including perceptual, cognitive, motor, and linguistic abilities.
- Domain requirements: the tasks that need to be supported, group, social and cultural dynamics, communication patterns, environmental factors, and so on.
- Technological requirements: such as availability of hardware and software and the availability of plug-ins.
- Performance requirements: for example, task success rates, task-completion times, satisfaction ratings, and quality of task output (e.g. comprehension outcomes in an e-learning environment).

Taken together these categories of requirements are often called the “context of use” [2]. Ultimately the stakeholders associated with a particular digital system want that system to be
suggest that the UK’s DDA is an example of legislation that

Ultimately, by not referring to the WCAG or any other technical
definition of an ‘accessible’ – and thus lawful – Web site, we suggest that the UK’s DDA is an example of legislation that

4. LEGISLATION AND POLICY

We argue that only by taking this approach – by considering the context of use – can meaningful and productive discussions be conducted about the accessibility or usability of a system. Using context of use as a benchmark for success also eliminates the illogical situation in which some commentators reject sites that are usable by disabled people (i.e. fit the context of use) but do not meet specific WCAG conformance levels. Appropriate “fit to context of use” should be the goal of developers, with a recognition that guidelines can serve as means to this end, but that conformance to guidelines is not itself the end.

The terms of the UK’s Disability Discrimination Act 1995 focus on the rights of employees (Part II), the provision of goods, facilities and services (Part III of the Act) and of post-16 education (Part IV). No mention is made of the Web in the terms of the legislation. Commentators have generally agreed that the Web is likely to be covered by the Act, especially given case law that exists in Australia, under similar legislation [39] and we have no reason to question this position. What is increasingly apparent, though, is that while the provision of a service may equally be achieved through the Web or through other means, the key requirement is to ensure that access to that service is not made unreasonably difficult or impossible for someone on account of their disability.

In many cases, an accessible Web site would indeed be the single most appropriate way of providing that service, and under the terms of the UK’s DDA, a “reasonable adjustment” for a service provider would be to make their service available through an optimally accessible Web site. But for those who may still have trouble using the site, there would seem to be scope to provide alternative access to the same service, either through an alternative Web interface, or by non-Web means.

While there are accusations within industry that the DDA does not provide Web site developers and commissioners with enough information on what constitutes lawful practice, we suspect that what is needed is a better explanation of what is meant by “reasonable adjustments” – examples, or, in the absence of case law, hypothetical case studies may help.

4.1 A Holistic Model for E-learning Accessibility

In our promotion of a contextual approach to accessibility, we now look to the e-learning field for an example of how a holistic approach to accessibility can be effectively applied.

The development of accessible e-learning resources has additional complexities over the development of conventional informational Web resources. As described by Kelly et al. [24] e-learning developers are faced not only with the difficulties of understanding and interpreting WAI guidelines, but also ensuring that the resources implement an appropriate pedagogical approach. With informational resources, the aim is to ensure that the relevant information can be easily accessed. With learning resources, however, this approach could mean learners are provided with answers rather than having to think carefully in order to come up with answers for themselves (and it is a legitimate part of the learning process for students to initially arrive at incorrect answers).

The approach to e-learning accessibility developed by Kelly et al. argues that the focus should be on the learner’s needs and that, rather than mandating that individual learning resources must be universally accessible, it is the learning outcomes that need to be accessible. In this blended approach, it can be perfectly legitimate for inaccessible digital learning resources to be replaced by learning which makes use of physical resources. This approach is based on regarding the learner with disabilities as an individual with an alternative learning style, rather than the medical model implicit in the WAI approach, which focuses on the disabilities themselves. Within many educational institutions, especially in the UK, blended learning, which exploits both digital and physical learning resources to provide a rich educational experience, is growing in popularity. The holistic model for e-learning accessibility reflects a wider approach to learning and therefore is more likely to be embedded within the development of learning resources.

The holistic model for e-learning accessibility also recognizes that other contextual aspects are important. As well as pedagogical issues, factors such as available resources, organization culture, etc. (and usability, as discussed previously) are also relevant factors, as illustrated in Figure 1.
5.2 Application of the Holistic Model

The benefit of the holistic approach is demonstrated when considering the importance of multimedia in accessible learning and teaching. While it is important that rich media is provided with appropriate accessibility features to compensate for sensory or physical impairments, it is equally important to take advantage of the potential of the very use of rich media in enhancing the accessibility of the learning environment to people with specific learning difficulties [39].

Figure 1: Holistic Model for E-Learning Accessibility

As a specific example of an application of the holistic model for e-learning accessibility and a comparison with the WAI approach, we will consider the provision of slide materials supporting a lecture or similar, such as those created by Microsoft PowerPoint.

Microsoft PowerPoint is widely used for presentations in many learning situations. Providing access to PowerPoint presentations after a lecture can help to maximize access to the resource and can provide a useful aid for revision. In many organizations the approach taken may be to simply provide a link to the PowerPoint file - but with this approach, the slides cannot be viewed on platforms which do not support the Microsoft viewer. A better approach is thus to provide access to a HTML version of the slides. As saving as HTML from within PowerPoint creates non-WCAG conformant HTML resources, the WAI approach would encourage use of alternative conversion software to create conformant HTML pages, or alternative presentation software, such as S5 [30]. Although such approaches can enable a better WAI conformance rating to be claimed, it is questionable whether this will always lead to enhanced accessibility benefits for the end user.

A danger with this mechanistic application of WCAG is that it can lose sight of the aims of the presentation. Presentation software such as PowerPoint could be used for a variety of reasons such as:

- an aide memoiri for the speaker – and therefore of limited use to anyone who was not at the presentation; or
- a content-rich resource possibly containing information not provided in the spoken presentation (such as multimedia content, URLs, etc).

We argue that in the first instance, the presentation file may be of little use to anyone not at the presentation perhaps as a limited reference list. Its presence online in HTML format cannot be assumed to provide an equivalent of the information given by the lecturer, and therefore in no way makes the experience provided by the lecture ‘accessible’.

It should also be noted that although presentational software such as S5 make use of conformant XHTML and CSS, it is questionable as to whether such presentations are as usable as standard PowerPoint presentations. PowerPoint, for example, has an advantage that text can automatically resize to fit the screen display whereas with S5 the viewer needs to manually resize the text (and is not made aware that materials may not be displayed). It should also be noted that handouts from PowerPoint presentations can be designed to provide useful notes when printed, whereas this option in S5 is poor in comparison. Although both of these issues affect the usability or the learning resource, they are not addressed in WCAG guidelines.

Technological innovations, such as the use of Podcasting to support learning, may have a role to play in enhancing access to such presentations. With the growth in the provision of wireless networks at academic conferences, and the simplicity of creating Podcasts, we are seeing an increase in the numbers of Podcasts of lectures. Such recordings can clearly enhance the accessibility of talks, providing benefits not only for visually impaired users, but also enhancing access to others. Ironically, however the lack of clarity and the dogmatism of the WAI approach can lead to Podcasts being regarded as breaking WAI guidelines: transcripts of Podcasts are not normally provided. If one wishes to pursue the “use W3C technologies” directive to an extreme, there is an additional issue whereby Podcasts typically make use of the proprietary MP3 format rather than, say, an open format such as Ogg.

An application of the holistic model would recognize that enhancing the accessibility of the presentation resources will be dependent on the role they play; that the non-compliance of HTML versions of such resources does not necessarily lead to accessibility problems; that the usability of the presentation is a relevant factor; that the usability of hardcopy printouts is a relevant factor and that it can be beneficial to users if alternative multimedia versions of presentations are provided, even if such enhancements fail to comply fully with WCAG guidelines themselves.

The main point to be made is that it is the experience of listening/seeing the lecturer which should be accessible rather than the emphasis which can be placed on addressing the accessibility of HTML versions of the slides which results from a mechanistic implementation of WCAG guidelines. In such a case, a more appropriate alternative might be an audio or video recording of the lecture plus text transcript or captions and audio descriptions.

Of course, the resources required to create multimedia alternatives and corresponding accessible alternatives may impact on the ability to provide the multimedia. The primary role of this example, though, is to show how the effectiveness of the accessibility solution depends on the context of use of the presentation.

5.3 The Role of Metadata - Personal Needs Profiles, Resource Definitions and Other Standards

A promising approach to the problem shows in the IMS AccessForAll work [19], [20]. This takes the approach that an accessible outcome needs to be provided not by a single universal resource but by a collection of related resources selected and adapted to match the requirements and preferences of that user in
that particular context. The basic notion is that each user has a Personal Needs Profile that can contain a number of different personal profiles, each for a different identified context, such as "AtWork", "InTheGym" and "LateAtNight". Each profile consists of a set of "functional" preferences, which contain standardized descriptions of requirements for particular hardware, content or control mechanism that the user requires or prefers in that context.

We stress that the information contained is not medical but is functional. For example a profile may contain information specifying that a user needs to have content where no essential information is encoded with colour alone or the user needs to use a screen reader such as JAWS with specific parameters for speech rate or needs "Sticky Keys" to operate the keyboard. Matching metadata is associated with each piece of content (Web page or Web page component). With this information, and content appropriately labelled it is possible to collect and aggregate resources with alternatives authored to make those resources accessible in the specific context matched to the user’s functional profile. The basic components of this approach are shown in Figure 2.

**Figure 2: Using Personal Profiles and Resources**

<table>
<thead>
<tr>
<th>Personal Needs and Preferences Statement</th>
<th>Content (eLearning and otherwise)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Description Metadata</td>
<td>IEEE LOM: Accessibility Application profile</td>
</tr>
<tr>
<td>Dublin Core Application Profile</td>
<td></td>
</tr>
</tbody>
</table>

The "Application Profiles" in Figure 2 are under development and show how Resource Description Metadata can be used in interoperable ways with the two most commonly used e-learning resource metadata schemes. The application profile of the IEEE Learning Objects Metadata standard (LOM) [17] is being developed by CEN-ISSS Learning Technologies Workshop [5] and a Dublin Core Metadata Initiative (DCMI) [14] profile is under development by the Dublin Core Accessibility Working Group [15]. These pieces of work are being developed in close collaboration so that they will interoperate.

Inherent in the Resource Description Metadata model is the concept that an original resource could have adaptations authored separately so as to support the very common use case where resources such as Web pages are authored by a person without specialist accessibility knowledge and are then made accessible for some real context by the provision of alternatives and services by a person or organization with specific accessibility expertise. The metadata provides standard ways to describe the access modalities of a resource (such as "visual", "auditory", "tactile"), adaptations for those (such as specific caption kinds for visual modalities) and ways to refer to and relate the physically separated parts. With this model the requirement that all authors have detailed accessibility knowledge is minimized and accessibility expertise can be drawn from specialists available.

A possible instance of latest versions of the AccessForAll metadata work showing a resource and some alternatives is given in Figure 3. In the diagram, the resource "knows" about some of the alternative adaptations that are available because it has URL pointers to them ("hasAlternative"). Sometimes an alternative is known only in its own local context as in the case of the captions alternative for the auditory component. Alternatives can be used as supplementary or replacements and not shown in the figure is the less common case where a resource can contain alternatives within itself (for example as with a video that has captions that can be switched on or off). Alternatives would be selected from those available to match the preferences and requirements in the Personal Needs profile.

**Figure 3: A Possible AccessForAll instance**

With accessible outcomes provided by collections of resources and services adapted and matched to the user at or close to the time of use, then some of those resources could be online, some offline and some could be (and will need to be) services, such as the provision of a human interpreter.

Despite the deficiencies in the universal accessibility model and the WAI WCAG guidelines some of the guidelines and checkpoints are of some value in this context in ensuring that online content such as Web pages are transformable to meet some Personal Needs Requirements and in providing tests of some properties that can aid in the matching and authoring of digital alternatives. More work is needed here amongst the WCAG tests and tools to determine and document what does and does not work in systems with distributed resources.

Though the work began in IMS, at time of writing it is being developed as a freely available ISO standard [21]. So far the context of this work has extended to digital resource preferences, requirements and descriptions but it is notable that the Framework Document for the ISO SC36 work [21] describes the concept of "blended learning". It is to be hoped that this signals intention to extend the work to provide for description of non-digital preferences and resources, such as railway stations, and non-digital alternatives (including resources and services) to digital resources and services. In the authors’ view this is work that needs doing.

Closely related work is underway in IEEE Learning Technology Standards Committee in devising a Resource Aggregation Model for Learning Education and Training (RAMLET) [16]. The aim of this work is to show how different media and aggregation formats can be mapped together so as to technically facilitate disaggregation, authoring and re-aggregation of disparate resources and formats.
In the opinion of the authors, it is true that to date a great deal of work in the domain of technical standards for e-learning has had a "hard" edge so that someone is either "doing e-learning" or "not doing e-learning" or they are "using the Web" or "not using the Web". Real practice of learning is much messier and there is a general need within the standards to soften the edge and show how the standards can be used in holistic contexts where some resources and services are digital and some are not. Accessibility is chief amongst the domains that needs that broadening of context.

6. DEVELOPING THE HOLISTIC FRAMEWORK FOR ACCESSIBILITY

6.1 An Inclusive Approach to Web Accessibility: the Tangram Model

We have argued the practical difficulty of applying the WAI model for Web accessibility. We have described the holistic approach for e-learning and how it can be applied within the context of cultural resources. There can also be a cultural or experiential context to informational resources [22] that is essential to preserve for all users.

We have described our holistic model for e-learning accessibility. This model reflects the wider approaches to learning which underpin current pedagogical approaches. Although this model was developed within an educational context, the approach is being developed to address more general provision of Web resources.

For example, Kelly et al. [26] describe the application of the holistic model within the context of cultural resources. The i-Map service [43] developed by the Tate Gallery provided resources on the paintings of Picasso and Matisse aimed specifically at visually impaired users. The approach taken was to focus on the requirements of a specific user community, rather than aiming to provide a universal resource. The i-Map service also makes use of a blended approach, providing access to raised images of the paintings.

Another example of an application of our hybrid approach to accessibility is the recommendation given to a public library in the UK. A presentation on a variety of IT developments within the library was given including an example of a popular Macromedia Flash-based game aimed at children who visited the library. The response to a question of the accessibility of the game was that, although the game had proved popular, it would probably have to be removed, as it did not comply with the organization’s commitment to WCAG guidelines. Subsequent discussion explored the purpose of the game. It was found that the game was aimed at parents who visited the library with young children, with the game providing distraction for the children while their parents browsed the library. It was pointed out that, rather than attempting to build an accessible version of the Flash game, an alternative approach could be the provision of building blocks or other types of games.

Our approach therefore emphasizes the importance of the user and on satisfying the user’s requirements. Such requirements will have a cultural context to them. We therefore advocate an approach to accessibility which is based on social inclusion rather than on the notion of universal accessibility.

The inclusive approach we advocate should be open and capable of integrating with complementary approaches developed by others, and is neutral on format issues. In addition to technical issues, we argue that the inclusive approach should recognize that alternative ways of providing accessibility can be provided, either through use of accessibility metadata (which can be used to provide resources which are accessible to a user’s individual requirements) or through a hybrid approach.

Our initial idea was to develop a jigsaw metaphor. However this approach implies that there is a single, correct solution. We have rejected this notion as we argue the need to recognize the contextual aspects as accessibility and the need for solutions which are appropriate to the particular context. Our proposed model makes use of a Tangram metaphor as an approach for Web developers, based on the Tangram - a seven piece puzzle where the object is to form a shape using all pieces.

Using the WAI model the Web developer has very little flexibility. A series of rules are provided for the developer to follow. Although some of the rules are undoubtedly useful, others are flawed and, as we have seen, alternative approaches, not covered in the guidelines, may also be useful. We feel there is a need to provide a wider set of guidelines, but that the developer needs flexibility in selecting guidelines which are applicable. For example, a Web developer may choose a subset of WCAG guidelines in conjunction with usability guidelines and relevant style guidelines.

The Tangram metaphor makes it clear that there is no single universal solution. Instead the developer can select relevant guidelines in order to implement a solution which is usable to the target audience, and taking into account any access requirements. Web developers will then be expected to make use of a range of guidelines covering best practices in areas of accessibility, usability and interoperability. So for example a simple Web site may make use of well-established guidelines such as WCAG and Nielsen’s usability heuristics to develop a simple solution.

Figure 4: The Tangram Model for Web Development

In Figure 4, the 7 different pieces of the Tangram puzzle each refer to a set of guidelines. The metaphor is meant to be extensible: as well as WAI guidelines, other guidelines may also be used, such as Nielsen’s 10 usability heuristics [35], guidelines for design for specific user groups, such as older people [32], or even guidelines on clarity of written content, for example [37].

A developer of a more sophisticated Web service may require a richer set of guidelines to call upon. This could include organizations guidelines such as use of corporate logos and navigational features, guidelines on writing style. The developer will have flexibility in the approach taken, and may choose a solution most appropriate to the context of the application.

The aim of this approach is to provide a solution which maximizes the usefulness to the end user, as opposed to the
current WAI approach which encourages mandatory application of a limited set of guidelines. The metaphor is meant to clarify that the most appropriate solutions can be obtained by engaging with the users rather than simply applying a set of rules.

We can see several advantages with the application of this model:
- The model can be extensible (we can make use of additional ‘pieces’). This allows the approach to be extended as, for example, new technologies become available (for example, guidelines for use of accessible Macromedia Flash or PDF can be incorporated).
- The model can cover general IT accessibility and is not limited to Web accessibility.
- The model can be extended to include real world solutions.
- The model can be extended to include Web accessibility issues which are not covered in WCAG (e.g. the accessibility of hard copy output of Web pages).
- The model is well-suited for use with Web resources which are personalized though use of accessibility metadata (the model emphasizes the service provided to the end user rather than individual components).
- The model can be deployed across a range of different legal systems.
- The model is neutral regarding technologies.

7. APPLYING THE FRAMEWORK
We have described a richer underlying framework for accessibility which is based on the Tangram metaphor. Application of this approach will require a wider framework of activities, including further advice and support for both Web developers and policy makers. Accessibility researchers should also look to how their work can support a contextual approach to accessibility.

7.1 For Web Developers
Using the components of context of use introduced in Section 3.3, Web developers can establish context by providing answers to the following:
- **User characteristics:** Who is your target audience? What assumptions can you make as to the level of expertise the target audience have in the subject area of the Web site?
- **Domain requirements:** What is the purpose of the Web site? What sort of tasks do you expect users to be able to perform using the site?
- **Technology requirements:** What assumptions can you make as to the browsing and assistive technology available to the target audience and their knowledge of that technology? What would be the tradeoff in using a non-(X)HTML format to deliver information or experiences along with or instead of (X)HTML?
- **Pre-existing alternatives:** What other ways already exist to provide access to the information or services provided by the Web site in question? What pre-existing assets (e.g. multimedia clips, telephone information line) can you take advantage of to widen access?
- **Quality of alternatives:** What other ways could be created to provide alternative routes to the same goals to which the site is intended to provide access? What are the accessibility (or other) barriers to accessing these alternative means?

Defining this context allows the development team to adopt an appropriate accessibility strategy - one that has the needs of the target users of the site and the purpose of the site at heart. This can then be used to define technical requirements that will allow the most appropriate solution(s) to be created, using appropriate guidelines to inform development.

7.2 For Policy Makers
Using the proposed framework, policymakers can refer to promotion of a holistic approach to accessibility rather than mandate a strict conformance to a technical standard that may not in fact guarantee equality. There is a danger that if policy mandates Single-A WCAG conformance, then this will be all that organizations will consider, to the detriment of the true accessibility and usability of their services.

Policy should encourage information and services provided through a Web site to be made as accessible as possible, and at this point may make reference to the WCAG. However it should also promote the use of other digital content, or even non-Web means along side the Web site, if doing so would enhance the accessibility and usability of the service in question to specific groups.

Policy should also encourage, rather than discourage, the use of multiple routes to achieving the same end goal – whether information, a service or specific experience – where multiple routes might better serve different groups within the target audience.

Perhaps most importantly, policy should require documentation of the process of development – including those attributes of context outlined in Section 3.3, and the association of alternative routes to achieving the same end goals. This provides for transparency in decision-making and allows justification of decisions that, without the availability of contextual information, might appear to be discriminatory.

There does remain a role for legislation that more directly refers to minimum technical standards, but we argue this should refer to the production of software and hardware to support creation of and access to Web content. To be more specific, given the blurring of the concepts of Web sites and Web applications, we would qualify this as covering applications under the domain of the UAAG and ATAG. Keates and Clarkson [23] note the positive impact that the amended Section 508 of the Rehabilitation Act has had, on major players in the Web technology sector, in terms of improvements they have made to accessibility of Web authoring software and proprietary Web formats.

7.3 Using the Framework in Evaluation and Benchmarking
One objective of the European Commission-funded Support-EAM project [42] is to develop a ‘quality mark’ – a pan-Europe accreditation that can be applied to Web sites that have reached a recognized level of accessibility. Given our arguments about the importance of context, and the principle that it is the service or experience that should be the focus of attention, not the Web site, we have serious reservations about the implications of such a scheme on Web content that has been developed as part of a wider strategy to improve the accessibility of information and services.

The framework we propose, on the other hand, provides a more realistic way of assessing accessibility, by considering the Web site in context. By taking a contextual approach to establishing accessibility, it also supports a more task-oriented approach to accessibility evaluation, for example the heuristic walkthrough for
accessibility as proposed by Brajnik [4], and is in line with the AccessForAll strategy discussed in Section 5.3.

The contextual approach also lends itself to documenting the approach taken by the site providers to accessibility. Documenting decisions taken from an accessibility perspective is a transparent demonstration that accessibility has been considered throughout the design lifecycle. This may also be an important step towards demonstrating that developers have been proactive in taking steps to ensure that unjustified discrimination does not occur.

8. CHALLENGES FOR WAI

In this paper we have outlined an alternative approach for addressing Web accessibility to that provided by WAI. We feel our approach reflects the working practices used by Web developers who do not aim to address usability issues independently of other factors, and who are acutely aware of factors such as resources implications and timescales which they will invariably face. We feel the approach also addresses the needs of the user who wants access to online information, services and experiences, and not simply a WCAG AAA conformant Web site.

It is important to note that our model is not intended to replace the work carried out by WAI, but to provide a context for WAI’s activities. There will still be a need for WAI to continue the work of the UAAG and ATAG working groups in order to improve the quality of authoring tools and user agents. Similarly, given WAI’s high profile, there is a need for WAI to continue its outreach and educational activities.

We feel, however, that WAI should recognize the limitations of its model for Web accessibility. We feel that WAI should similarly recognize that there can be many approaches to addressing accessibility issues, and that the guidelines developed by WAI groups should be capable of being integrated with other approaches. Acknowledging the diversity of solutions which can benefit end users should also lead to recognition that this diversity needs to be reflected within legal frameworks.

It should be noted that such an approach has been taken by other activities within W3C. When the US government threatened the introduction of the US Communications Decency Act (which addressed the concerns over content which may be inappropriate for children) the response of W3C - an international membership organization - was not to engage in political debate with the US government, but to develop a technical solution (PICS - the Platform for Internet Content Selection) which could be deployed at an appropriate level (its use could be mandated within legislation or could be applied across particular sectors).

At this point, we acknowledge the recent emergence of the ‘baseline’ concept as part of the work by WAI towards the publication of version 2.0 of the WCAG [46], an encouraging development in this area. It is a welcome acknowledgement by WAI that accessibility can, and should, be measured with contextual factors in mind, such as anticipated browsing technology available to the target audience.

In summary, we feel that the underlying principle to WAI’s work should be to widen participation rather than to seek universal accessibility. Both phrases reflect similar goals, however universal accessibility implies there is a single universal solution whereas widening participation acknowledges that there can be a variety of solutions.

9. CONCLUSIONS

We have argued that a guideline-based approach to accessibility for disabled people has a role to play in the drive towards harnessing the Web as a means of reducing inequality, but it can be harmful if guidelines promote the assumption by both Web site developers and policymakers that, in every case, one fully conformant Web site will be accessible and usable to absolutely everyone. Accessibility guidelines become essential only when the role of a Web site has been defined in a wider context of provision of services, information and experiences.

Taking the approach of the e-learning community as an example, we believe that the goal of universal accessibility on the Web is inappropriate, and instead we should not be afraid to explore multiple routes to equivalent experiences, whether those routes are multimedia alternatives, or tactile alternatives, or whatever might be appropriate. The Tangram Model provides a metaphor for this approach, while the AccessForAll metadata work will give a solid framework for applying contextual accessibility.

We believe that this approach will lead to more effective policy and legislation, will support evaluation and benchmarking of accessibility, will give developers greater freedom to meet the needs of their audience, and will allow disabled people to access information and services in a way that most suits their requirements.

A more holistic approach to Web accessibility – to using the Web as an access technology - is one that will allow the Web to reach its true potential, overcoming the frustrations of conflicting user needs, competing Web technologies, sub-standard browsing technology and users who may never be fully aware of the technologies that can help them most effectively browse the Web.

10. REFERENCES


[6] CEN-ISSS (Comite Europeen de Normalisation) Learning Technologies Workshop. CEN Workshop Agreements (standards) are available for download from a link on this


2005 Accessibility Diagnosis on the Government Web Sites in Taiwan, R.O.C.

Yui-Liang Chen  
Shih-Hsin University, Taiwan  
#1, Lane 17, Mu-Cha Road, Sec. 1, Taipei, Taiwan 116.  
Tel.: +886-2-2236-8225 ext. 2301  
ychen@cc.shu.edu.tw

Yen-Yu Chen  
Shih-Hsin University, Taiwan  
#1, Lane 17, Mu-Cha Road, Sec. 1, Taipei, Taiwan 116.  
Tel.: +886-2-2236-8225  
eponine@cybertutor.com.tw

Monica Shao  
Research, Development, and Evaluation Commission, Executive Yuan, Taiwan  
6F, No.2-2, Sec.1, Jinan Road, Taipei, Taiwan 100, R.O.C.  
Tel.: +886-02-2341-9066  
monica@mail.dra.rdec.gov.tw

ABSTRACT

Improvement in web technology and services alone with diversity development has caused a high demand of Internet usage. New web technologies and equipment have opened infinite possibilities for global communication, but these possibilities are limited by various factors such as setting the browser version too high, causing limitations to lower version holders, or making faster speed hard-drives producing delays in lower speed hard-drives. However, the most severe factor limiting web communication’s performing at full potential is accessibility for the both physically and mentally disabled.

The Executive Yuan of the Taiwanese Government has recently pushed forward the idea of Web accessibility in Governments’ websites. Assessment of 35 websites has shown to pass Priority 1 Level Validation (machine recognition/machine review), of which 28 reached the Conformance Level “A+.” Apart from the checkpoint numbered 1.8 of machine recognition/machine review that had an increase in failed website percentage, the rest presented a decline in the number of failed websites, which suggested improvements in Web accessibility development in the year 2005. The most commonly seen checkpoint errors were similar in 2004 and 2005, and included checkpoint error numbered 5.5 (Provide summaries for tables), 10.6 (Do not use space to separate adjacent links), 4.3 (Identify the language of the text), 3.5 (Use relative sizing and positioning (% values) rather than absolute (pixels)), 3.3 (Use a public text identifier in a image), and 9.3 (Make sure that event handlers do not require use of a mouse).

Comparison between Freego and Bobby validation tools using the 58 checkpoints listed in the Web Accessibility Regulations have shown six checkpoints need to be revised. Five checkpoints were different in Priority Level setup, and one checkpoint numbered 9.3 (Make sure that event handlers do not require use of a mouse) was different in the calculation of number of errors. Apart from that, the 90 checkpoints listed in the Web Accessibility Regulations in Freego, none can be compared with checkpoint number 13.1 (Create link phrases that make sense when read out of context) in Bobby. With these results, it was clear that the Freego Validation Tool needs to be improved, and that Web Accessibility Regulations needs to be discussed further.

Keywords

Accessibility, Accessible Examination, Accessible Website, Accessible Checkpoint, Accessible Examining System

1. INTRODUCTION

1.1 Background

Due to the advancement of web technology, online users increase day by day. Global internet has not only stretched into diversity, it has also advanced to provide various services. As a result, web technology is something we cannot live without in this century.

Web technologies provided information communication with infinite possibilities; however, it is people that make web communication impossible to reach its full potential. For example, some Web pages indicate that their websites are best browsed by a resolution of 800 x 600, and are recommended to use at least an IE4.0 version of browser. Setting a certain speed of resolution or asking for a specific version of browser has limited Web accessibility for those who have lower grade hard drives or slower dial-up internet connections, not to mention Web accessibility for people who are physically or mentally disabled. The power to search for information and information comprehension ability determines a person’s ability to survive in this “burst of technology” society. This over-advanced technology century will not benefit people with disabilities at all.

At the end of 2004, the Minister of the Interior obtained the number of 911,640 people in Taiwan who are registered as disabled, that is, 4% of the total population. Because technology information is so freely accessible, those who have abundant resources are far more advanced in their economic and information growth than those whose limited resources and access to friendly technology have caused an imbalance in this information society.

1.2 Web Accessibility

The definition of accessibility formulated by C. James Huang and Mei Chao [8] was to emphasize convenience and direct access for each one who may need different assistance in his/her life, such as...
in transportation, learning, working, etc. The concept can be expanded to the Internet. Webmasters have to consider different users, especially people with disabilities, and how they can obtain the information through the Internet by relative equipment, techniques and resources. Paddison [15] defined accessibility as “barrier-free,” which means that the information provider or service must be available to everyone, regardless of software, platform, environment, and user ability. For software, it includes accessing the Internet using desktop browser or voice browser; for platform, it includes using desktop, mobile phone or PDA (Personal Digital Assistant); for environment, it includes working in noisy or under-illuminated surroundings; and for user ability, it includes visual impairment or dyslexia. In addition, accessibility indicates that people with disabilities can surf the Internet through assistant technology to gain the complete information, and have full opportunities to interact with the Internet [12] [14].

In general, accessible websites are able to give everyone equal opportunities to access the complete Web content regardless of software, hardware and user ability.

**Interested groups**

With regard to Web accessibility, several domestic and foreign interested groups are introduced in the following subsections.

**The World Wide Web Consortium:** The World Wide Web Consortium (W3C) is a nonprofit group founded in 1994 to develop standards for the Web and also to play a major role in promoting Web accessibility. Three major long-term goals of W3C include universal access, semantic Web and Web of trust. Regarding the limitations of users in different cultures, languages, education, abilities, resources, technology for surfing the Web, the developed techniques should satisfy all limitations around the world. The development directions of W3C are: Architecture Domain, Interactive Domain, Technology and Society Domain, and Web Accessibility Initiative.

Web Accessibility Initiative (WAI) provides Web Content Accessibility Guidelines 1.0 (WCAG1.0) [5] [21], Authoring Tool Accessibility Guidelines 1.0 (ATAG 1.0) [23], and User Agent Accessibility Guidelines 1.0 (UAAG 1.0) [22] for different aspects to contribute to Web accessibility. WCAG 1.0 makes the Web content directly usable for individuals with disability. Many checkpoints following these guidelines are being developed for practical validation. All checkpoints are classified into three levels: Priority1, Priority 2 and Priority 3. Priority 1 indicates the Web content MUST satisfy its checkpoints. Priority 2 indicates the Web content SHOULD satisfy its checkpoints, including those in Priority 1. Priority 3 indicates the Web content MAY satisfy its checkpoints, including those both in Priority 1 and Priority 2. All checkpoints of these levels provide specific and detailed explanation to webmasters [13]. Most software tools providing Web accessible validation capability are compliant with the guidelines developed by WAI.

**The Center for Applied Special Technology:** The Center for Applied Special Technology (CAST) is also a nonprofit group founded in October 1984 to address the problems of expanding learning opportunities for individuals with disabilities by providing or developing assistive technology. However, CAST has recognized that the more focus on individuals the more burdens and cost of adaptation, and has not conquered all barriers that individuals with disabilities encounter. Therefore, over the past several years, CAST has undergone a major shift in its approach, now starting that: “We now believe that the most effective strategy for expanding opportunities for individuals with disabilities is through ‘Universal Design for Learning’.” The slogan “Universal Design for Learning” refers to the contribution of software tools and learning models that are feasible for use by everyone, regardless of age or whether they are typical learners, or have special needs. A Web accessible desktop validation tool, Bobby, helps to expose barriers to accessibility and encourages compliance with existing accessible web guidelines, including Section 508 of the US Rehabilitation Act and the Web Content Accessibility Guidelines (WCAG) of W3C [1]. In 2002, Bobby was adopted as part of Watchfire Corporation to provide two types of services. One service was free validation for single pages and was renamed WEBXACT (http://webxact.watchfire.com/) in 2005, and the other was paid validation. So far, Bobby can examine two types of Web accessibility designs (WAI WCAG 1.0 and section 508 standards of United States), and when websites pass validation, the standards and conditions are posted on the Web as evidence.

**The RDEC, Executive Yuan, Taiwan:** Taiwan is actively pushing the policy of filling in digital gaps and developing a computerized society. The Research, Development and Evaluation Commission (RDEC) of the Executive Yuan is the governmental department in charge of making the regulations and supervising the websites of all governmental agencies to stratify Web accessibility in Taiwan. In fact, providing accessibility service has been a major policy of the Taiwanese Government for some time now. In 2002, The RDEC has established “the guidelines of accessible websites.” In 2003, The RDEC has provided “on-line validation service,” and has delivered “the claim of conformance” to webmasters with perfect validation results. In 2004, the RDEC released the stand-alone validation tool called “Frego,” and has supplied more convenient and efficient validation services in order to improve Web accessibility.

The global trend shows that governments make many efforts to compensate for digital gaps and to provide individuals with disability with fairness and equal access to information. Although, some researchers have found that designer’s perspectives are substantial and long-term matters [7] [18], policies, laws and guidelines play the most important roles.

**1.3 Current Status**

When e-government is being promoted world-wide, the rights of those with disabilities are something that cannot be ignored, and that is why Web accessibility to those with disabilities is being emphasized today. The United States of America, Great Britain, Canada, Portugal, and Australia have noticed the problem and have started resolving the issue by making websites accessible to people with disabilities. In 1998, America re-wrote section 508 of the Rehabilitation Act of 1973, asking all states to ensure that access to websites is available to people with disabilities. The Global E-Government (table 1-1-1, 1-1-2) issued by Brown University of America and the Global E-Government Indices provided by the United Nations will include government websites as one of the assessments for Web accessibility for the disabled. In year 2000, Japan clearly stated in their Information and Communications Technology Booklet (ICT) that Web accessibility must be provided for the disabled, and in the E-Japan2002 proposal issued in 2001 also emphasized Web accessibility for the minority and the elderly populations. A ubiquitous oriented society proposed by u-Japan was granted in year 2004, that wishes to construct a web environment be
leveraged by anytime, anywhere, any device, anyone, by the year 2010.

Table 1-1- 1 Table of article of 2004 Global E-Government

<table>
<thead>
<tr>
<th>2004</th>
<th>Taiwan</th>
<th>Singapore</th>
<th>United States</th>
<th>Canada</th>
<th>Monaco</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rankings/</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Ratings</td>
<td>44.3</td>
<td>43.8</td>
<td>41.9</td>
<td>40.3</td>
<td>39.0</td>
</tr>
<tr>
<td>Online</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service</td>
<td>46</td>
<td>87</td>
<td>77</td>
<td>29</td>
<td>50</td>
</tr>
<tr>
<td>Publican</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>97</td>
<td>100</td>
</tr>
<tr>
<td>Data bases</td>
<td>100</td>
<td>63</td>
<td>95</td>
<td>87</td>
<td>100</td>
</tr>
<tr>
<td>Privacy</td>
<td>50</td>
<td>97</td>
<td>82</td>
<td>90</td>
<td>50</td>
</tr>
<tr>
<td>Security</td>
<td>54</td>
<td>93</td>
<td>67</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>W3C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accessibility</td>
<td>0</td>
<td>3</td>
<td>42</td>
<td>81</td>
<td>0</td>
</tr>
</tbody>
</table>

Global E-Government, 2004

Table 1-1- 2 Table of article of 2005 Global E-Government

<table>
<thead>
<tr>
<th>2005</th>
<th>Taiwan</th>
<th>Singapore</th>
<th>United States</th>
<th>Hong Kong</th>
<th>China</th>
<th>Canada</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rankings/</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Ratings</td>
<td>57.2</td>
<td>54.5</td>
<td>50.5</td>
<td>46.2</td>
<td>44.3</td>
<td>43.3</td>
</tr>
<tr>
<td>Online</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service</td>
<td>77</td>
<td>77</td>
<td>75</td>
<td>65</td>
<td>70</td>
<td>27</td>
</tr>
<tr>
<td>Publican</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>93</td>
</tr>
<tr>
<td>Data bases</td>
<td>100</td>
<td>100</td>
<td>95</td>
<td>100</td>
<td>100</td>
<td>60</td>
</tr>
<tr>
<td>Privacy</td>
<td>77</td>
<td>100</td>
<td>82</td>
<td>85</td>
<td>78</td>
<td>100</td>
</tr>
<tr>
<td>Security</td>
<td>85</td>
<td>100</td>
<td>64</td>
<td>65</td>
<td>61</td>
<td>100</td>
</tr>
<tr>
<td>W3C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accessibility</td>
<td>92</td>
<td>13</td>
<td>44</td>
<td>15</td>
<td>0</td>
<td>70</td>
</tr>
</tbody>
</table>

Global E-Government, 2005

According to the Global E-Government 2004 and Global E-Government 2005, out of the first five countries developing Web accessibility for the disabled, only Canada has achieved the desired level, while the other four have shown improvement. What is noteworthy is that, in 2004, Taiwan accumulated a total of 92 points in the rating indices, and is the fastest growing country of all in the area of developing Web accessibility for the disabled. Taiwan has reduced the imbalance of information growth within its society, and has pushed for Web accessibility for the disabled as one of the primary goals that need to be actualized by E-Government. Facing the rapid development of Internet technologies, obstacles encountered by the disabled are increasing; therefore, it is crucial to resolve each problem as it arises.

The current review adopted the Freego Stand-Alone Validation Tool, developed by the Research, Development, and Evaluation Commission (RDEC), Executive Yuan, R.O.C. The aim in so doing was to stratify Web accessibility for the disabled in Taiwan, and to propose future developments for better services.

2. DEVELOPMENT FOR GOVERNMENT WEB ACCESSIBILITY

2.1 Web Accessibility Inspection and Implement Target

The Executive Yuan, in its Web Accessibility Regulations, clearly states the direction of movement including central governments, associated organizations, local governments, social welfare, and academia. By June 2004, the RDEC has achieved four levels of accessibility: Conformance Level “A,” Conformance Level “A+,” Conformance Level “double-A,” and Conformance Level “triple-A” (see Figure 2-1-1).

The purpose for the RDEC to start working on Web accessibility within government systems is to provide a guideline for other organizations to follow. Therefore, this research is targeted at government websites. By using automated accessibility tools, it is possible to examine the effectiveness of the guidelines and the level of conformity to them in society. A total of 117 websites were examined in this research.

2.2 Web Accessibility Inspection and Instruction for Implementation

Web Accessibility Services provided by the RDEC, and the On-Line Validation Service and the Stand-Alone Validation Tool provided by Freego have offered Single-page Diagnosis (specific for Web page Diagnosis) and Website Diagnosis (specific for all the links related to the website). The difference between On-Line Validation Service and the Stand-Alone Validation Tool lies in the ability to inspect individual website separately, allowing high efficiency by the latter.

The passing standard for Web accessibility is usually set at Website Diagnosis; therefore, most overseas research selects the homepage or various levels of websites for their assessments. The present review conducted Website Diagnosis on 117 governments by using 24 accessible checkpoints (machine recognition/machine review) (see figure 2-2-1) listed in the Accessible Web Development Guidelines (AWDG).
Table 2-2-1 Guideline and 24 accessible checkpoints (machine recognition/machine review)

<table>
<thead>
<tr>
<th>Guideline number / Checkpoint code</th>
<th>Priority level / Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H101000 (1.1) 1</td>
<td>Provide a text equivalent for every image.</td>
</tr>
<tr>
<td>H101001 (1.2) 1</td>
<td>Provide a text equivalent for every applet.</td>
</tr>
<tr>
<td>H101002 (1.3) 1</td>
<td>Provide a text equivalent for every object.</td>
</tr>
<tr>
<td>H101003 (1.4) 1</td>
<td>Provide alternative text for all image-type buttons in forms.</td>
</tr>
<tr>
<td>H101004 (1.5) 1</td>
<td>Provide alternative text for all image map hot-spots (AREAS).</td>
</tr>
<tr>
<td>H101007 (1.8) 1</td>
<td>Provide other descriptive links (e.g. D link) to describe the content of LONGDESC.</td>
</tr>
<tr>
<td>H301015 (1.16) 3</td>
<td>Contain a link client-side image map not presented elsewhere on the page.</td>
</tr>
<tr>
<td>H203002 (3.3) 2</td>
<td>Use a public text identifier in a DOCTYPE statement.</td>
</tr>
<tr>
<td>H203004 (3.5) 2</td>
<td>Use relative sizing and positioning (% values) rather than absolute (pixels).</td>
</tr>
<tr>
<td>H203005 (3.6) 2</td>
<td>Nest headings properly.</td>
</tr>
<tr>
<td>H304202 (4.3) 3</td>
<td>Identify the language of the text.</td>
</tr>
<tr>
<td>H305004 (5.5) 3</td>
<td>Provide summaries for tables.</td>
</tr>
<tr>
<td>H106001 (6.2) 1</td>
<td>Refer a HTML file to each FRAME source.</td>
</tr>
<tr>
<td>H206005 (6.6) 2</td>
<td>Provide a NOFRAMES section when using FRAMES.</td>
</tr>
<tr>
<td>H207001 (7.2) 2</td>
<td>Avoid blinking text created by the BLINK element.</td>
</tr>
<tr>
<td>H207002 (7.3) 2</td>
<td>Avoid scrolling text created by the MARQUEE element.</td>
</tr>
<tr>
<td>H207004 (7.5) 2</td>
<td>Do not cause a page to refresh automatically.</td>
</tr>
<tr>
<td>H207005 (7.6) 2</td>
<td>Do not cause a page to redirect to a new URL.</td>
</tr>
<tr>
<td>H109000 (9.1) 1</td>
<td>If possible, use a client-side image map instead of a server-side image map.</td>
</tr>
<tr>
<td>H209002 (9.3) 2</td>
<td>Make sure that event handlers do not require use of a mouse.</td>
</tr>
<tr>
<td>H310004 (10.5) 3</td>
<td>Provide a default in the text area.</td>
</tr>
</tbody>
</table>

2.3 Accessible Validation Procedure

Having accessed 117 websites using the Freego Validation Tool during July and August of 2005, the accumulated results were categorized into three main Priority groups (see Table 2-3-1).

Table 2-3-1 Distribution of the Three Priority Levels Checkpoint Error

<table>
<thead>
<tr>
<th>Mistake checkpoint</th>
<th>Number of websites</th>
<th>Priority level 1</th>
<th>Priority level 2</th>
<th>Priority level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>35</td>
<td>13</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>10</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>12</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>20</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>21</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>16</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>9</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>6</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>4</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>4</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>117</td>
<td>117</td>
<td>117</td>
<td></td>
</tr>
</tbody>
</table>

From Table 2-3-1, the validation for Priority Level 1 (machine recognition/machine review) was shown to certain thirty-five websites (29.91%) with the Conformance Level “A.” Of all the 35 websites, 28 also passed validation for Priority Level 2 (ensuring that event handlers do not require use of a mouse) thus reaching to Conformance Level “A+.” A total of 13 websites (11.11%) had passed the Priority Level 2 (machine recognition/machine review) as well as Priority Level 1 (machine recognition/machine review), reaching to Conformance Level of “double-A.” Of those four websites (3.42%) that passed Priority Level 3 (machine recognition/machine review), only 3 had also passed Priority Level 1 & 2 (machine recognition/machine review), reaching to Conformance Level “triple-A.” These three websites are http://www.cpa.gov.tw/, http://www.moea.gov.tw/ and http://w2kdmz1.moea.gov.tw/index.asp.

2.4 Comparison of the Accessible Validation Procedure between year 2004 and 2005

The developmental trend for the three Priority Level Validations can be obtained by comparing results found in 2004 and 2005 assessments (see Table 2-4-1, 2-4-2, and 2-4-3).
Table 2-4-1 Priority Level 1 Validation Comparison between 2004 and 2005

<table>
<thead>
<tr>
<th>Website</th>
<th>Number of Checkpoint Errors</th>
<th>2004</th>
<th></th>
<th>2005</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number</td>
<td>%</td>
<td>Number</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>4.05</td>
<td>35</td>
<td>29.91</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>14.86</td>
<td>21</td>
<td>21.37</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>13.51</td>
<td>17</td>
<td>14.53</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>16.22</td>
<td>11</td>
<td>9.40</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>21.62</td>
<td>14</td>
<td>11.97</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>13.51</td>
<td>7</td>
<td>5.98</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>13.51</td>
<td>7</td>
<td>5.98</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2.70</td>
<td>1</td>
<td>0.85</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>74</td>
<td>100</td>
<td>117</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Average number of Checkpoint Errors</td>
<td>3.45</td>
<td>1.98</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From both Table 2-4-1, comparing the data between 2004 and 2005, it was evident that website percentage was higher when checkpoint errors were between 0~2 for year 2005. However, a decrement of website percentage was seen when checkpoint errors were between 3~7. The change of trend was further supported by the decrease in average checkpoint error from 3.45 in 2004 to 1.98 in 2005. In addition, the distribution of checkpoint errors presented for 2005 was evidently more consistent than the distribution for 2004.

Table 2-4-2 Priority Level 2 Validation Comparison between 2004 and 2005

<table>
<thead>
<tr>
<th>Website</th>
<th>Number of Checkpoint Errors</th>
<th>2004</th>
<th></th>
<th>2005</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number</td>
<td>%</td>
<td>Number</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>2.70</td>
<td>4</td>
<td>3.42</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.35</td>
<td>3</td>
<td>2.56</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1.35</td>
<td>7</td>
<td>5.98</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>9.46</td>
<td>19</td>
<td>16.24</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>28.38</td>
<td>39</td>
<td>33.33</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>42</td>
<td>56.76</td>
<td>45</td>
<td>38.46</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>74</td>
<td>100</td>
<td>117</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Average number of Checkpoint Errors</td>
<td>4.30</td>
<td>3.89</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From both Table 2-4-2, it is apparent that in 2005 website percentage increased from zero checkpoint error to four checkpoint errors. However, it decreased dramatically after five checkpoint errors. A decrease in averaged checkpoint errors from 5.05 in 2004 to 3.78 in 2005 further supported the trend. Furthermore, the overall distribution trend in Priority Level 2 Validation did not fluctuate as much as in Priority Level 1 Validation.

Table 2-4-3 Priority Level 3 Validation Comparison between 2004 and 2005

<table>
<thead>
<tr>
<th>Website</th>
<th>Number of Checkpoint Errors</th>
<th>2004</th>
<th></th>
<th>2005</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number</td>
<td>%</td>
<td>Number</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>2.70</td>
<td>4</td>
<td>3.42</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.35</td>
<td>3</td>
<td>2.56</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1.35</td>
<td>7</td>
<td>5.98</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>9.46</td>
<td>19</td>
<td>16.24</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>28.38</td>
<td>39</td>
<td>33.33</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>42</td>
<td>56.76</td>
<td>45</td>
<td>38.46</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>74</td>
<td>100</td>
<td>117</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Average number of Checkpoint Errors</td>
<td>4.30</td>
<td>3.89</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From both Table 2-4-3, the website percentage in 2005 increased from zero checkpoint error to four checkpoint errors, but decreased slightly at five checkpoint errors. The decreased average checkpoint error (4.30 in 2004 to 3.89 in 2005) also supported the claim. Moreover, the overall website percentage distribution trend in Priority Level 3 Validation assessment appeared in an opposite direction to that of Priority Level 1 Validation.

Synthesizing the above comparisons for year 2005, the overall website percentage that passed the three validations was raised, with more evidence shown in Priority Level 1 Validation. The number of websites passing the validations as checkpoint errors increased was on the decrease. The average checkpoint error increased from Priority Level 1 Validation (1.98) to Priority Level 3 Validation (3.89).

2.5 Comparison of Failed Website Percentage between 2004 and 2005

The percentage of failed websites was calculated by dividing the number of failed websites by the total websites. There are 24 checkpoints assessing websites, and these are categorized into three Priority Levels. The first Priority Level consisted of 9
accessible checkpoints (see Table 2-5-1), the second Priority Level consisted of 10 accessible checkpoints (see Table 2-5-2), and the last Priority Level consisted of 5 accessible checkpoints (see Table 2-5-3).

Table 2-5- 1 Priority 1 Level Failed Website Percentage comparisons between 2004 and 2005

<table>
<thead>
<tr>
<th>Item</th>
<th>The Number of Failed Websites</th>
<th>Website Mistake Percentage (%)</th>
<th>Average Decline Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>2004</td>
<td>2005</td>
<td>2004</td>
</tr>
<tr>
<td>Sample</td>
<td>74</td>
<td>117</td>
<td>74</td>
</tr>
<tr>
<td>Priority 1 level</td>
<td>1.1</td>
<td>67</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>1.2</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>1.3</td>
<td>53</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>1.4</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>47</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>1.8</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>6.2</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>9.1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>12.1</td>
<td>47</td>
<td>43</td>
</tr>
</tbody>
</table>

Table 2-5-1 clearly stated the difference in website error percentage for each of the nine accessible checkpoints. Eight of the 9 checkpoints have a decline rate of 42.90% to 24.57% in their website mistake percentage from 2004 to 2005. Only the checkpoint numbered 1.8, which stated it provides other descriptive links (e.g. D link) to describe the content of LONGDESC, showed an increase in the number of failed websites from 1 in 2004 to 2 in 2005, while the other checkpoints showed a decrease in the number of failed websites. The averaged percentage decline of website error was 38.33%.

Table 2-5- 2 Priority 2 Level Failed Website Percentage comparisons between 2004 and 2005

<table>
<thead>
<tr>
<th>Item</th>
<th>The Number of Failed Websites</th>
<th>Website Mistake Percentage (%)</th>
<th>Average Decline Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>2004</td>
<td>2005</td>
<td>2004</td>
</tr>
<tr>
<td>Sample</td>
<td>74</td>
<td>117</td>
<td>74</td>
</tr>
<tr>
<td>Priority 2 level</td>
<td>3.3</td>
<td>68</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>3.5</td>
<td>68</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>3.6</td>
<td>15</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>6.6</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>7.2</td>
<td>21</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>7.3</td>
<td>37</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>7.5</td>
<td>14</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 2-5- 3 Priority 3 Level Failed Website Percentage comparisons between 2004 and 2005

<table>
<thead>
<tr>
<th>Item</th>
<th>The Number of Failed Websites</th>
<th>Website Mistake Percentage (%)</th>
<th>Average Decline Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>2004</td>
<td>2005</td>
<td>2004</td>
</tr>
<tr>
<td>Sample</td>
<td>74</td>
<td>117</td>
<td>74</td>
</tr>
<tr>
<td>Priority 3 level</td>
<td>1.16</td>
<td>51</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>4.3</td>
<td>68</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>5.5</td>
<td>70</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>10.5</td>
<td>59</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>10.6</td>
<td>70</td>
<td>105</td>
</tr>
</tbody>
</table>

Table 2-5-2 presented the Priority Level 2 Validation with a decrease in the number of failed websites in all ten checkpoints in 2005, with an average decline percentage of 27.52%. Likewise, Priority Level 3 Validation showed a similar trend of 11.73% decline in the five checkpoints (see Table 2-5-3). In addition, by comparing the most often seen checkpoint errors, a similarity in percentage rate was evident in both 2004 and 2005 assessments (see Table 2-5-4)

Table 2-5- 4 The most often seen checkpoint errors

<table>
<thead>
<tr>
<th>rank</th>
<th>Checkpoints</th>
<th>Priority level</th>
<th>2004</th>
<th>2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.5 Provide summaries for tables.</td>
<td>3</td>
<td>94.59 %</td>
<td>92.31 %</td>
</tr>
<tr>
<td>2</td>
<td>10.6 Do not use space to separate adjacent links.</td>
<td>3</td>
<td>94.59 %</td>
<td>89.74 %</td>
</tr>
<tr>
<td>3</td>
<td>4.3 Identify the language of the text.</td>
<td>3</td>
<td>91.89 %</td>
<td>88.89 %</td>
</tr>
<tr>
<td>4</td>
<td>3.5 Use relative sizing and positioning (% values) rather than absolute (pixels).</td>
<td>3</td>
<td>91.89 %</td>
<td>86.32 %</td>
</tr>
<tr>
<td>5</td>
<td>3.3 Use a public text identifier in a DOCTYPE statement.</td>
<td>2</td>
<td>91.89 %</td>
<td>76.92 %</td>
</tr>
<tr>
<td>6</td>
<td>1.1 Provide a text equivalent for every image.</td>
<td>1</td>
<td>90.54 %</td>
<td>67.52 %</td>
</tr>
<tr>
<td>7</td>
<td>9.3 Make sure that event handlers do not require use of a mouse.</td>
<td>2</td>
<td>86.49 %</td>
<td>63.25 %</td>
</tr>
</tbody>
</table>
The proposal was made by the Executive Yuan at the end of 2005 promoting government websites’ achievement of Conformance Level “A+.” Its results can be seen in Table 2-5-4. The checkpoint errors numbered 3.3, 1.1, and 9.3 were significantly decreased in their website mistake percentage, and all of them belonged to the Priority 1 and 2 Levels. The checkpoint error number 9.3 in particular had achieved Conformance Level “A+,” suggesting the positive result came from the influence of promoting Conformance Level “A+.”

2.6 Comparison of average number of checkpoint errors between 2004 and 2005

The average number of checkpoint errors was calculated by dividing the total number of checkpoint errors by the total sample of websites. There are 24 checkpoints assessing websites, and they are categorized into three Priority Levels. The first Priority Level consisted of the average number of checkpoint errors of 9 accessible checkpoints (see Table 2-6-1), the second Priority Level consisted of the average number of checkpoint errors of 10 accessible checkpoints (see Table 2-6-2), and the last Priority Level consisted the average number of checkpoint errors of 5 accessible checkpoints (see Table 2-6-3).

| Table 2-6- 1 Comparison of Priority 1 Level Validation of the averaged number of checkpoint errors between 2004 and 2005 |
|---|---|---|
| Year | 2004 | 2005 |
| Samples | 74 | 117 |
| 1.1 | 5051.14 | 2109.00 |
| 1.2 | 1.07 | 0.32 |
| 1.3 | 23.81 | 7.32 |
| 1.4 | **10.91** | **51.62** |
| 1.5 | 699.24 | 533.20 |
| 1.8 | 0.26 | 0.26 |
| 6.2 | 0.65 | 0.00 |
| 9.1 | 0.01 | 0.00 |
| 12.1 | 34.38 | 5.05 |

Table 2-6- 2 Comparison of Priority 2 Level Validation of the averaged number of checkpoint errors between 2004 and 2005

<table>
<thead>
<tr>
<th>Year</th>
<th>2004</th>
<th>2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samples</td>
<td>74</td>
<td>117</td>
</tr>
<tr>
<td>3.3</td>
<td>428.47</td>
<td>131.44</td>
</tr>
<tr>
<td>3.5</td>
<td>20878.39</td>
<td>4686.53</td>
</tr>
<tr>
<td>3.6</td>
<td>1.18</td>
<td>0.06</td>
</tr>
<tr>
<td>6.6</td>
<td>2.31</td>
<td>0.07</td>
</tr>
<tr>
<td>7.2</td>
<td>0.64</td>
<td>0.07</td>
</tr>
<tr>
<td>7.3</td>
<td><strong>5.27</strong></td>
<td><strong>48.65</strong></td>
</tr>
<tr>
<td>7.5</td>
<td>0.64</td>
<td>0.58</td>
</tr>
<tr>
<td>7.6</td>
<td><strong>0.58</strong></td>
<td><strong>0.91</strong></td>
</tr>
</tbody>
</table>

In Table 2-6-1, eight of the 9 checkpoints showed a decrease in average number of errors in 2005, with only the checkpoint numbered 1.4, which stated that it provides alternative text for all image-type buttons in forms, showing an increase in checkpoint error (an average of 10.91in 2004 to 51.62 in 2005).

In Table 2-6-2, eight of the 10 checkpoints in Priority 2 Level Validation showed a decrease in average number of errors. Two checkpoints numbered 7.3 (Avoid scrolling text created by the MARQUEE element) and 7.6 (Do not cause a page to redirect to a new URL) showed an increase in their average number of checkpoint errors. Likewise in Table 2-6-3, four of the 5 checkpoints in Priority 3 Level Validation showed a decrease in average number of errors. The checkpoint numbered 1.16, which stated that it contains a link to a client-side image map not presented elsewhere on the page, showed an increase in its average number of checkpoint errors.

3. COMPARISON OF WEB ACCESSIBILITY EQUIPMENT (FREEGO VERSUS BOBBY)

3.1 Comparison of Web Accessibility Design Inspection Regulation

WAI WCAG 1.0, proposed by W3C, is one of the relatively completed Web Accessibility Regulations there is. It contained the requirements/necessities by the Web Content Accessibility Guideline (Nielsen, 2000). Bobby, from Watchfire Corporation, adopted WAI WCAG 1.0 as its standard for providing Web accessibility diagnosis services. In the same way, the Executive Yuan of Taiwan, R.O.C., adopted the regulations of WAI WCAG 1.0 and proposed the “Web Accessibility Regulations” which also consisted of 14 regulations and 3 Priority Levels. This Web Accessibility Regulations was the regulation guide when developing the Freego Web accessibility equipment. When comparing Web Accessibility Regulations and WAI WCAG 1.0, it is apparent the checkpoints in each of the 14 regulations are somehow different. In the Web Accessibility Regulations, a total of 90 checkpoints were found to be different, and in the WAI WCAG 1.0, sixty-five checkpoints were found to be different (see Table 3-1-1).
Table 3-1- 1 Accessibility Regulations and the WAIWCAG 1.0

<table>
<thead>
<tr>
<th>WAI WCAG1.0 Number of Checkpoints</th>
<th>Number of Checkpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research, Development, and Evaluation Commission, Executive Yuan. (Web Accessibility Regulations)</td>
<td></td>
</tr>
<tr>
<td>Web Content Accessibility Guidelines</td>
<td></td>
</tr>
<tr>
<td>1: provide equivalent alternatives to auditory and visual content.</td>
<td>1</td>
</tr>
<tr>
<td>2: Don’t rely on color alone.</td>
<td>2</td>
</tr>
<tr>
<td>3: Use markup and style sheets and do so properly.</td>
<td>1</td>
</tr>
<tr>
<td>4: Clarify natural language usage</td>
<td>3</td>
</tr>
<tr>
<td>5: Create tables that transform gracefully.</td>
<td>7</td>
</tr>
<tr>
<td>6: Ensure that pages featuring new technologies transform gracefully.</td>
<td>6</td>
</tr>
<tr>
<td>7: Ensure user control of time-sensitive content changes.</td>
<td>6</td>
</tr>
<tr>
<td>8: Ensure direct accessibility of embedded user interfaces.</td>
<td>1</td>
</tr>
<tr>
<td>9: Design for device-independence.</td>
<td>6</td>
</tr>
<tr>
<td>10: Use interim solutions.</td>
<td>6</td>
</tr>
<tr>
<td>11: Use W3C technologies and guidelines.</td>
<td>4</td>
</tr>
<tr>
<td>12: Provide context and orientation information.</td>
<td>6</td>
</tr>
<tr>
<td>13: Provide clear navigation mechanisms.</td>
<td>1</td>
</tr>
<tr>
<td>14: Ensure that documents are clear and simple.</td>
<td>3</td>
</tr>
<tr>
<td>Total number of Checkpoints</td>
<td>9</td>
</tr>
</tbody>
</table>

Overseas research literature often uses Bobby for inspecting Web accessibility, whereas Taiwan always adopts Freego for inspection. Whether there are differences between the two tools, or how significant the differences imposed on the inspection are questions that need to be examined and compared in order to improve the quality of Freego and, hence, Web accessibility inspection.

3.2 Results from the administration of Freego versus Bobby

During the months of July and August 2005, one Web page was selected and underwent analysis using both Freego and Bobby equipment in order to find the conditions of the 58 checkpoints listed in the machine recognition/machine review and machine recognition/human review sections of the Development for Web Accessibility Regulations (originally there are 59 checkpoints, however the one numbered 3.2 H203001, which assures that Web page design documents can be used in HTML, was amended for human recognition). Table 3-2-1 below provides a detailed conclusion for the 58 checkpoint differences, and a suggestion for Freego is future development.

<table>
<thead>
<tr>
<th>Checkpoint</th>
<th>Revision Required</th>
<th>No Change Required</th>
<th>Further Discussion Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 H101000</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2 H101001</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3:H101002</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4 H101003</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>1.5 H101004</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.6:H101105</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.7 H101106</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.8 H101007</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.9 H101108</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.10 H101109</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.12 H101111</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.13 H101112</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.16 H301015</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1 H102100</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2 H202101</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3:H203002</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.5:H203004</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.6 H203005</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.7 H203106</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.8 H203107</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.9 H203108</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.3 H304002</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.1 H105100</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.2 H105101</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.3 H205102</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.4 H205103</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5 H305004</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.6 H305105</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.7 H305106</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.1 H106100</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.2 H106001</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.3 H106102</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.4 H106103</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.5 H206104</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.6 H206005</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

139
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2 H207001</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>7.3 H207002</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>7.4 H207103</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>7.5 H207004</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>7.6 H207005</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>8.1 H208100</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>9.1 H109000</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>9.3 H209002</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>9.4 H309103</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>9.6 H309105</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>10.1 H210000</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>10.2 H210101</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>10.3 H210102</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>10.4 H310103</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>10.5 H310004</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>10.6 H310005</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>12.1 H112000</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>12.2 H212101</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>12.3 H212102</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>12.4 H212103</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>12.6 H212105</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>13.2 H213101</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>13.5 H213004</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Of a total of 58 checkpoints, between Freego and Bobby were compared, the two systems were found to have 49 checkpoints that are alike. As shown in the Table 3-2-1, 49 checkpoints were selected for “No Change Required,” and have been tested, and been found to have 85% consistency. However, six checkpoints were selected for “Revision required,” and they were:

3.3 「Use a public text identifier in a DOCTYPE statement.」
3.5 「Use relative sizing and positioning, rather than absolute.」
3.9 「Make sure BLOCKQUOTE is used only for quotations, not indentation.」
6.1 「If style sheets are ignored or unsupported; ensure that pages are still readable and usable.」
10.5 「Provide a default in the text area.」
10.6 「Do not use space to separate adjacent links.」

Another five checkpoints (shown below) were different in Priority Level setup, and one checkpoint, numbered 9.3 (Make sure that event handlers do not require use of a mouse), was different in the calculation of number of errors.

10.1 H210000
10.2 H210101
10.3 H210102
10.4 H310103
10.5 H310004

Comparison of Website Priority Level Validation between 2004 and 2005 has shown an overall increase in websites passing all three Priority Levels, with more evidence shown in Priority 1 Level Validation. The number of websites passing the validations as checkpoint errors increase was on a decrease. The average checkpoint error increased from Priority Level 1 Validation (1.98) to Priority Level 2 Validation (3.78), and finally, again, to Priority 3 Level Validation (3.89).

1.4 「Provide alternative text for all image-type buttons in forms.」
5.6 「Provide a summary for tables.」
8.1 「Provide accessible alternatives to the information in scripts, applets, or objects.」
9.1 「If possible, use a client-side image map instead of a server-side image map.」
9.3 「Make sure that event handlers do not require use of a mouse.」
12.6 「Explicitly associate form controls and their labels with the LABEL element.」

In addition, none of the 90 checkpoints listed in the Web Accessibility Regulation in Freego can be compared with checkpoint number 13.1 (Create link phrases that make sense when read out of context) in Bobby. With these results, it is clear that the Freego Validation Tool needs to be improved, and that the Web Accessibility Regulations needs to be discussed further.

4. CONCLUSION AND DISCUSSION

Web accessibility Development has recently placed much emphasis on the Web Technology Industry. The current research has shed light onto the necessities for different organizations to unite and develop Web accessibility. The results of the research have gained International recognition.

The current research has found that, out of the 35 websites that passed the Priority 1 Level Validation (machine recognition/machine review), 28 websites also passed the checkpoint numbered 9.3 (Make sure that event handlers do not require use of a mouse) Priority 2 Level Validation, reaching the Conformance Level “double-A.” Thirteen websites that passed Priority 2 Level Validation (machine recognition/machine review) had reached Conformance Level “triple-A.”

Comparison of Website Priority Level Validation between 2004 and 2005 has shown an overall increase in websites passing all three Priority Levels, with more evidence shown in Priority 1 Level Validation. The number of websites passing the validations as checkpoint errors increase was on a decrease. The average checkpoint error increased from Priority Level 1 Validation (1.98) to Priority Level 2 Validation (3.78), and finally, again, to Priority 3 Level Validation (3.89).

Results from the failed website percentage assessed by the 24 checkpoint errors has shown a decrease in overall percentage, with the one exception of the checkpoint numbered 1.8 (Provide other descriptive links (e.g. D link) to describe the content of LONGDESC), which showed an increase of one failed website in 2004 to 2 failed websites in 2005. Further comparison of commonly seen checkpoint errors was found to be similar between 2004 and 2005, and they were as follows:

5.5 「Provide summaries for tables.」
10.6 「Do not use space to separate adjacent links.」
4.3 「Identify the language of the text.」
3.5 「Use relative sizing and positioning (% values) rather than absolute (pixels).」
3.3 「Use a public text identifier in a DOCTYPE statement.」
1.1 「Provide a text equivalent for every image.」
9.3 「Make sure that event handlers do not require use of a mouse.」

The checkpoint errors numbered 3.3, 1.1, and 9.3 significantly decreased their website mistake percentage, and all of them belonged to the Priority 1 and 2 Levels. The checkpoint error number 9.3 in particular had achieved the Conformance Level “A+,” suggesting that this positive result came from the influence of promoting Conformance Level “A+.” In addition, twenty of the 24 checkpoints were shown to have decreased their average numbers of checkpoint errors, which was consistent with the increment of websites passing the Priority Level Validation percentages.

With the comparison of Freego and Bobby equipment, forty-nine of the 58 checkpoints listed in the Web Accessibility Regulations were shown to have 85% consistency. Of all the checkpoints, six of the Freego equipment checkpoints were suggested to be revised, and they were:

3.3 「Use a public text identifier in a DOCTYPE statement.」
3.5 「Use relative sizing and positioning (% values) rather than absolute (pixels).」
3.9 「Make sure BLOCKQUOTE is used only for quotations, not indentation.」
6.1 「If style sheets are ignored or unsupported, ensure that pages are still readable and usable.」
10.5 「Provide a default in the text area.」
10.6 「Do not use space to separate adjacent links.」

Another five checkpoints (shown below) were different in Priority Level setup, and one checkpoint, numbered 9.3 (Make sure that event handlers do not require use of a mouse), was different in the calculation of number of errors.
1.4 「Provide alternative text for all image-type buttons in forms.」
5.6 「Provide a summary for tables.」
8.1 「Provide accessible alternatives to the information in scripts, applets, or objects.」
9.1 「If possible, use a client-side image map instead of a server-side image map.」
9.3 「Make sure that event handlers do not require use of a mouse.」

12.6 「Explicitly associate form controls and their labels with the LABEL element.」

In addition, none of the 90 checkpoints listed in the Web Accessibility Regulation in Freego can be compared with checkpoint number 13.1 (Create link phrases that make sense when read out of context) in Bobby. With these results, it is clear that Freego Validation Tool needs to be improved, and that the Web Accessibility Regulations needs to be discussed further.

It is believed that after revision and discussion of the checkpoints, the Web Accessibility Regulation and the function of the Freego equipment will be more accurate in improving Web accessibility. It is also hoped that the results of this research can bring about more services to assist governments in pushing forward Web accessibility services.

5. ACKNOWLEDGMENTS

Special thanks to RDEC, Executive Yuan, for supporting the grant for this research.

6. REFERENCES


Position Paper: Mobile Phones may be the Right Devices for Supporting Developing World Accessibility, but is the WWW the Right Service Delivery Model?

Tapan S. Parikh
Dept. of Comp. Sci. and Engineering
University of Washington
tapan@cs.washington.edu

ABSTRACT
In this paper we detail the synergies we have observed between the features and limitations of mobile phones, and the usability and accessibility requirements of rural developing world users. This includes support for sequential interaction, multimedia input and output, asynchronous messaging and a universally familiar numeric keypad. However, we argue that the WWW as currently conceived may be an inappropriate model for delivering mobile information services in this context. We highlight a number of tensions we have observed between the traditional web model, and the design synergies that we have uncovered. To demonstrate an alternative framework, we describe CAM — a platform for delivering mobile information services in the rural developing world. Supporting scripted execution, media-driven, tangible interaction as well as an offline usage model, CAM is uniquely adapted both to rural accessibility requirements and the inherent capabilities of mobile phones. By learning from the CAM design, we can either improve the design of existing mobile web standards and services, or implement a more appropriate framework altogether.

Keywords
mobile computing, mobile phones, novice users, rural development, ICT

1. INTRODUCTION
Over the past five years, we have been investigating the design of interfaces for a variety of rural Indian users - ranging from uneducated, semi-literate farm laborers to high-school and college-educated youth [10, 11, 12].

In this paper we detail the synergies we have observed between the features and limitations of mobile phones, and the usability and accessibility requirements of rural developing world users. These include a small screen — limiting decision-making requirements for novice users; audio feedback — found to be important for the subjective satisfaction of both literate and semi-literate rural users; a numeric keypad — familiar to billions of users and uniform across languages and cultures; and asynchronous messaging services, which are already immensely popular. Moreover, the very mobile nature of a mobile phone allows it to reach regions that are difficult for traditional computing devices.

However, thus far the user of mobile web services is dwarfed by voice calls and text messaging in both the developed and developing world. Part of the reason could be the poor usability of mobile web applications [2, 8]. In this paper, we argue that the WWW as currently conceived may be an inappropriate model for delivering mobile data services to rural developing world users. We highlight a number of tensions we have observed between the traditional web model, and the design synergies that we have uncovered.

To demonstrate an alternative framework, we describe CAM — a platform for delivering information services to the rural developing world[12]. Supporting scripted execution, media-driven, tangible interaction as well as an offline usage model, CAM is uniquely adapted both to rural accessibility requirements and the inherent capabilities of mobile phones. By learning from the CAM design, we can either improve the design of existing mobile web standards and services, or implement a more appropriate framework altogether.

2. MOBILE PHONES ARE THE RIGHT DEVICE FOR RURAL ACCESSIBILITY
In this section we detail the synergies we have uncovered between mobile phone user interface and device affordances and the accessibility requirements of rural Indian users.

2.1 Small Screen
In an early design experiment we observed that a user interface with small and discrete task spaces was more comprehensible to semi-literate rural users [10]. Later, we found that a scripted sequence of mobile data entry tasks was learned and used efficiently by both educated and uneducated rural users [11]. Other researchers have noted the suitability of sequential execution for small-screened mobile devices [9]. Presenting one task at a time reduces the potential for confusion or indecision on the part of novice users.

2.2 Audio Feedback
We have repeatedly observed local language audio feedback as being the most important factor for the subjective satisfaction of rural Indian users. Initially, audio in the local language served as the bridge between poor rural users and an expensive foreign device [10]. Later, we found that the audio phrases uttered by our application were becoming local colloquialisms [11]. Voice-based input can also improve the accessibility of the system for semi-literate and illiterate rural users. A microphone and speakers are integral parts of any mobile phone.

2.3 Camera

Most medium to high-end mobile phones now come equipped with some kind of camera. Some may have several cameras, with high resolution and video capabilities. During our research in rural India we observed the importance of paper forms and ledgers in local information practices [10]. More recently, we have demonstrated the use of printed barcodes on paper forms as a way of navigating mobile form-filling applications [11]. A camera can also be used to capture rich information and experiences without writing or typing.

2.4 Numeric Keypad

The numeric phone keypad is immediately familiar to billions of users. When you include its inverted and cheaper cousin — the calculator, the number increases further. Exposure to the Qwerty keyboard pales in comparison. A numeric keypad also obviates the need for hardware localization or of using a foreign keyboard mapped to your native language. Moreover, significantly more people are numerically literate than textually. We have observed that numeric input is accessible even for illiterate and semi-literate rural users [10].

2.5 Mobility

For the foreseeable future, most of the world’s people will not be able to afford their own digital device. Like other resources, technology will be shared by the family or community. Currently, there are two common alternatives for shared rural computing deployments. In one model, an Internet-connected PC kiosk is installed in some percentage of towns and villages [3, 7]. Villagers from other locations must travel there to access computing resources. In the other model, agents with handheld devices travel to collect information from and deliver information to villagers. This model has already been implemented in microfinance [6], and for health data collection [5], just to cite two examples. The advantage of this model is that people can access information services at their doorstep. This is definitely the more accessible approach, given the time-consuming nature of travel in the developing world. The potentially lower cost of mobile handsets when compared to PCs contributes to the affordability of this approach.

3. BUT IS THE WEB THE RIGHT SERVICE MODEL?

In this section we describe some tensions we have observed between these synergies, and the traditional WWW model.

3.1 Spatial vs. Temporal Layout

While HTML was originally intended for defining the structure of web content, in practice it has been largely used to specify the spatial layout of web pages. The WAP (and more recently, XHTML) standard continues to be steeped in this tradition. In contrast, sequential presentation of tasks and content may be more appropriate for the limited screen space of mobile devices, and also for the limited interaction vocabulary of novice rural Indian users.

3.2 Textual vs. Multimedia Interaction

While audio and video are part of many web sites, and digital media is available for sale or distribution over the web, presentation and interaction on the web is still largely a textual affair. Screen readers allow the disabled to browse the web, but the underlying markup is designed for graphical presentation, making it cumbersome to browse aurally. In contrast, capturing, transmitting and emitting audio (and now, video) is fundamental to the design of a mobile phone. Current mobile web services do not take advantage of these features.

3.3 Direct Manipulation vs. Numeric Selection vs. Tangible Interaction

The user interface of the web is based on the point and click world of the desktop WIMP GUI. Due to the small screen and limited input options, it has been difficult to adapt this model for mobile devices [1]. Similarly, novice rural users have found it difficult to understand and accept the many abstractions inherent in the WIMP model [10]. In our research, we have explored several alternatives. These range from the simplicity of numeric selection, to the potential for tangible, paper-based interaction. We discuss some of these in the next section.
3.4 Online vs. Offline Access

While there have been some attempts to implement local, searchable web caches that can be accessed via a proxy [13], the web is still mostly intended to be used as an online medium. Web sites are not designed with the intention of providing offline, disconnected access. Many rural villages in the developing world are only weakly connected to the telecommunications infrastructure, if at all. Internet connections in these locations can be expensive, unreliable, or both, at least using current technologies. While mobile phones are intended to provide real-time voice communications, asynchronous messaging-based services are also very popular. These include message-based data services (for example, see [4]). Users do not need to continuously be online to access these services. Using asynchronous protocols, messages are automatically cached and delivered when the phone has an active wireless connection. In the next section we describe how we have developed a robust application layer on top of these lower level messaging protocols.

4. CAM APPLICATION FRAMEWORK

CAM is an application framework for developing and delivering mobile information services in the rural developing world. The CAM architecture has been described in detail in a prior publication [12]. Here we discuss the advantages of this system in relation to the previous section.

- **Scripted** - CAM programs are built using scripted actions and functions rather than spatial layout primitives. The display limitations of mobile phones, and the interaction limitations of mobile users, both dovetail nicely with this approach.

- **Support for Rich Media** - CAM user interaction is driven by a scripted sequence of prompts and actions. Each of these can be associated with arbitrary audio and graphics. In early usability tests, the audio was so essential that some users didn’t even refer to the screen [11]. CAM also provides functions for capturing audio clips and images. This dramatically increases the possibilities of user input, partially compensating for the difficulty of mobile text input, especially for native languages.

- **Tangible and Numeric Interaction** - CAM applications can be accessed and navigated using barcodes or numeric strings printed on paper forms and other artifacts. In this way the process is more familiar to users accustomed to paper-based tasks, while overcoming the limited screen navigation area of mobile devices. Barcodes and numeric strings also serve as convenient persistent references to applications and data.

- **Can be used Offline** - Using asynchronous messaging services like SMS and MMS, CAM applications can be accessed without an active Internet connection. CAM provides an interactive, multimedia client on top of these lower level transport protocols. In a different sense, CAM is like online Interactive Voice Response (IVR) services, except that CAM applications work offline and also take advantage of the mobile phone’s screen and other UI features.

5. CONCLUSION

One of the reasons that we have been able to design and develop a new framework for delivering mobile information services to the rural developing world is that we do not have to support a large, existing application and content base such as the WWW. If the creators of the mobile web did not need to support access to these resources, clearly they would have designed different protocols and systems.

In this paper we have presented how a mobile services framework might look if it were designed from the ground up to be accessible to rural Indian users, while taking full advantage of the mobile phone’s inherent features and capabilities. Even we have been surprised by the synergies that have emerged between these requirements.

We believe there are lessons to be learned from our experience. Several of the approaches and techniques we have described could be applicable to other novice users around the World. In the future, we plan to conduct more experiments with diverse user groups to assess this potential. Moreover, given the fundamental differences between a PC and a mobile phone as a hardware device, we hope that our paper encourages others to envision entirely new models of mobile information service delivery. By learning from these experiments, we can either improve and extend existing web standards to incorporate new and successful techniques, or we may reach a point where we have something better altogether.

6. REFERENCES

http://mobile.google.com/.
