Generating User Interfaces for Users with Disabilities Using Libraries of XSLT, UIML, and Stylesheet Files

Lawrence Henschen[1(✉)], Julia Lee[1], Ning Li[2], and Xia Hou[2]

[1] Northwestern University, Evanston, IL, USA
henschen@eecs.northwestern.edu, julialee@agep.northwestern.edu
[2] Beijing Information Science and Technology University, Beijing, China
{lining,houxia}@bistu.edu.cn

Abstract. We describe a method for reconfiguring and reformatting documents, in particular web pages, to meet the needs of users with different abilities. The method merges our previous work on semantic markup [1] and presentation of intelligent documents [2] with a new approach to interoperability of document processing [3]. Semantic markup provides information about the purpose of elements in a document, in the spirit of HTML5 [4]. The work on intelligent documents provides means for dynamically adding functionality to a presentation system. The first new concept in this paper is to use XSLT [5] to reformat and reconfigure the material in a document to better meet the needs of a user. The second new concept is to create public libraries of XSLT, UIML, and stylesheet files for classes of users with different needs. A user then configures his or her browser for that user's abilities. When the browser opens a document, it retrieves an appropriate publicly accessible library to use in transforming and presenting the document.

Keywords: Universal access · XSLT translation · UIML · Semantic mark-up · Document presentation

1 Introduction

In this work we merge two lines of research presented in previous HCII conferences. In [1] we described an approach to universal access based on the idea of using semantic markup in web pages to describe the purposes of the various elements on the page - functionality, author intent, etc. We proposed that these could be used by more sophisticated browsers of the future to reorganize the presentation of the material to users with different disabilities, such as visually impaired users or deaf users. In [2] we showed how UIML could facilitate easy extension of browser display features and functionality to match the needs of a particular application area. In [2] the application area was extension of capabilities of document processing systems. Now we want to apply the techniques from [2] and a new idea in document processing [3] to our approach to universal access presented in [1], specifically to solve the problem of actually implementing the new meta-level markup in browsers without the need for the browser companies to modify their software.

M. Antona and C. Stephanidis (Eds.): UAHCI 2015, Part I, LNCS 9175, pp. 174–182, 2015.
DOI: 10.1007/978-3-319-20678-3_17

The use of semantic tags and attributes allows the author of a document to specify his or her purpose as well as the content. This can be used to adjust the presentation based on the medium, as in the use of HTML5 [4] to achieve platform independence. It can also be used to adjust the presentation to the needs of users with different abilities, as proposed in [1]. While HTML5 emphasizes the semantic aspects of its elements and attributes, we believe it is not sufficiently rich for universal access. Some elements, such as "article" and "section", convey semantic intent of the author and do have semantics that is the same for all users, and HTML5 encourages the content of these to be presented in ways that are appropriate for both the medium and the user. On the other hand, the attributes and elements are not rich enough to distinguish some kinds of semantic intent. For example, the "em" element (indicating emphasis) in HTML5 is more semantically meaningful than "font_size = 40" and allows different platforms to display the corresponding information in a way most relevant for the medium. However, there could still be different types of information whose emphasis is strong, such as warning or section heading or even just ordinary emphasis, and these would need to be presented differently to visually impaired or hearing impaired users. The "role" attribute in ARIA [6] approaches this idea, but the role attributes in ARIA are not sufficient to distinguish content to the degree we feel is needed for impaired users. Therefore, a much richer set of semantic tags and attributes is needed to address the needs of users who are visually impaired, hearing impaired, manually impaired, combinations of these, etc. The need for special mark-up for special classes of users has been used in many different contexts, among these for example the DAISY system [7] for reading text out loud to visually impaired users in a more naturally sounding way.

This need for a rich set of tags and attributes places stronger requirements on the presentation systems and precludes simply adding a few new capabilities through modest additional programming. We propose in this paper to combine the techniques of [2, 3] to solve this problem. XSLT is a well-known technology for reformatting information in XML documents. Through XSLT translation the information can be restructured and reorganized, and new tags and attributes can be added. An XSLT library could use the semantic information in a document to restructure the document for presentation to and interaction with a user with special needs. The techniques of [2] can then be used to map the elements, attributes, and functionality of the restructured document to appropriate presentation and interaction implementations. In [2] it was shown that the applications that implement the functionality could be anywhere that is accessible on the internet and need not be tied to an individual browser or document system. This achieves a very high degree of platform independence.

There are many different kinds of users with special needs. Therefore, a single XSLT library will not suffice. Furthermore, corresponding libraries of style sheets will be needed to present the reorganized content properly. Therefore, a set of libraries is needed that can be accessed every time a document is loaded. Reference [3] proposes a technique for maintaining a dynamically expanding set of such libraries. It also proposes that a document system determine a suitable library to retrieve each time a document is opened. In the case of word processors, the document processor examines the document type, for example OOXML or UOF. For the purpose of universal access, we suggest that the processor be configured for the intended user by specifying that user's special needs.

After that, the system can retrieve a suitable XSLT and style sheet library based on both the user's configuration and the document type.

In Sect. 2 we review the three references that form the foundation of this proposal. In Sect. 3 we give a few examples of how documents might be restructured to be more presentable to users with some disabilities. In Sect. 4 we make concluding remarks.

2 Motivation and Review of the Three Main References

In [1] we proposed the use of semantic information representing author intent as a means to help reorganize web page presentation for users with different impairments. For example, simply reading the textual content of a web page is not sufficient for a visually impaired user. Some portions of the content, for example major headings or warnings, on a page are more important than others, and some important material is not presented in text form. Inclusion of semantic information is consistent with the philosophy of HTML5 in moving away from visually oriented elements and attributes. However, HTML5 does not include a sufficient set of elements and attributes to address the needs of universal access. In [1] we showed examples of additional elements and attributes and how they could be used to present web pages to users with different abilities. Examples included:

- An attribute "purpose" with a value like "site_running_title". With this information a browser might read the corresponding content to a blind user once but not every time the user navigated to a different page.
- An element "navigation". This element would include the navigation information for the page, and a browser could read the choices to a blind user as soon as the page was opened.
- An attribute value "warning" for the attribute "purpose". This informs the browser to render the content in special circumstances and in a way suitable for the kind of user.

As noted, the role, property, and state attributes in ARIA [6] help, but they are not rich enough in our opinion.

While browsers may implement HTML5, they would not be expected to handle much richer sets of elements and attributes as proposed in [1]. Moreover, HTML5 is still oriented towards browser display rather than general document display and would therefore not be sufficient for general document-processing systems; for example, the common word-processing systems use different XML schemas such as OOXML and UOF. A more general means of translating a marked-up document to a target processing system (browser, cell phone, word processing system, etc.) and in a way that accounts for users with differing abilities is needed. XSLT [5] is designed to accomplish such transformations. Improved algorithms for performing such transformations, such as [8], make on-load or dynamic translation of even large documents feasible without introducing delays for the user.

The techniques of [2], particularly the use of UIML, were proposed to allow document processing systems to dynamically generate user interfaces with expanded capabilities and functionality that could be accessed anywhere on the web. We propose to adapt

these techniques to universal access. In [2] we showed a specific example of how UIML, and in particular the "interface" section of a UIML document, can be used to map the vocabulary used in a document to a new vocabulary in preparation for presentation to a user. When combined with semantic information this could allow transformation of original document elements into forms suitable for users with various disabilities. For example, an HTML5 "input" element marked with role "command" could be mapped to an ordinary button with on_click function for regular users but mapped to a voice command input system for blind users. Note that it is important to know both that it is an input element and also that its purpose is a command. Other input elements, such as basic information entry, could still be mapped to ordinary keyboard input rather than voice input, while the command inputs might be collected together and handled by a separate part of the system that monitors for any voice input. In [2] we also showed how operations mentioned in the document can be mapped to services anywhere on the web through the "peers" section of the UIML document. For example, a blind user who is also hearing impaired may need a special voice output service, not just the one built into the computer's browser. An element like

```
<uiml:d-component id="TextToSpeechService"
    location ="http://MyPersonalServer/TextToSpeechService/Speaker.svc">
      <uiml:d-method id="speaker" maps-to="speak">
            <uiml:d-param id="speakertext" type="string" />
      </uiml:d-method>
</uiml:d-component>
```

in the "peers" section of a UIML document would map the voice-to-text function to the user's own speech software on the user's own server, thus allowing that user to access the web using his or her special speech software from a laptop from anywhere in the world.

There are many audiences with special needs because there are many disabilities and combinations of disabilities. Therefore, a set of translations is needed that can map, say, a semantically marked-up HTML5 document to a particular kind of user (e.g., blind, hearing impaired, manually impaired, combination, etc.) and perhaps even a platform (PC web browser, cell phone, etc.). While there would be a large number of such transformations, generating XSLT libraries for each situation is much more feasible than reprogramming browsers and document processing systems to meet all the variety of needs. Moreover, the XSLT transformations can be refined as more experience with real users is gained and as HTML itself expands and progresses toward richer sets of semantic mark-ups, whereas hard-coded implementations are not likely to adapt to such learning.

Once such a library of transformations is developed, the work in [3] can be applied. In [3] the goal is to obtain interoperability among document processing systems. The idea is that a set of libraries be developed that provide stylesheets suitable for rendering documents written in one of the major document processing formats – for example, OOXML, UOF, ODF, etc. - in a document system designed for a different one of those formats. A document processing system would check the type of a document being opened and retrieve the appropriate stylesheet library before actually rendering the

document. Style sheets for various element types would be grouped into sub-libraries to provide better access and higher reusability. For example, paragraph styles for different document formatting systems might be stored as

http://public.styles.lib/public/paragraph/normal/1/uof
http://public.styles.lib/public/paragraph/normal/1/ooxml
…
http://public.styles.lib/public/paragraph/table/uof
http://public.styles.lib/public/paragraph/table/ooxml…

The first of these, for example, would contain information about styling the first paragraph in a section in a uof document. Document systems could retrieve appropriate ones of these and incorporate into larger stylesheets for a given document in a given system.

In [3] the focus was on office documents, and the libraries contained style sheets only. We propose to extend this concept by adding XSLT translation files and UIML files to the libraries. A user would configure his or her system, for example browser, with parameters specifying that user's situation - hearing impaired, visually impaired, etc. Then, when the system opens a new document it uses the configuration to determine a suitable transformation and style library to retrieve, translates the original document into a format suitable for that user, and then uses the style sheets to actually present the content and manage the interaction. We note that for universal access styling alone is not enough. As we illustrate in the next section, it will likely be necessary to a restructure a document, sometimes drastically, before presenting it to users with a disability. Therefore, the libraries we are proposing would contain both XSLT files and regular style sheet files. Moreover, functions used by regular users, such as mouse motions or clicks, may also need to be replaced, perhaps by special software not available on the user's machine. Therefore, our libraries will also include UIML with information like that indicated in a previous paragraph.

3 Examples

In this section we present a number of examples of how content and interaction might be presented to different users. Space precludes showing complete XSLT and UIML files. Therefore, our presentation here focuses on how the information in a semantically marked up document should be reorganized and presented. These are only suggestions; detailed studies of actual users should be made to determine how best to reorganize and present web content and handle interaction. These examples are only meant to illustrate the kinds of reorganization that are possible and the wide variety of presentation styles that will be needed to achieve universal access.

Much web page content is generated dynamically, for example by executing Javascript either on page load or in response to an event. Thus, all of the transformations that would be applied to a static page must also be applicable to content generated on the fly. This might be accomplished by modifying the Javascript as well as the static page content through XSLT when the page is first loaded. It might also be accomplished by allowing the Javascript to operate on the original form of the web page and then pass

the Javascript output through the translation process before rendering. The point is that any techniques that are proposed for transforming content must be applicable to both static and dynamically generated files.

We consider a small set of disabilities - blind, color blind, deaf, manually impaired (cannot use a mouse or keyboard) - and how content and interaction might be changed to accommodate users with those disabilities.

3.1 Blind Users

Presentation to blind users will likely require the most reorganization and the most changes in the interaction methods. Here are some ideas on changes that would help blind users.

Sighted users can easily comprehend the organization of most web pages and can scroll to areas of interest. Blind users need to be told what kinds of information are on the page and must have a different means to access, or "scroll to", the desired content. HTML5 elements, like article and section, can be used to inform the bind user about the content, especially if the author has added attributes in these elements that give descriptive information about the content and purpose. When the page is retrieved, a system like the one proposed here can extract such information from the article, section, and possibly other elements and present that information by voice as the page is actually loaded into the browser. The user could then navigate the page by voice input and could request the organization information to be repeated, perhaps by a voice input such as "page organization", if the user needed to refresh his or her memory. Other voice command inputs, such as "next section" or "previous paragraph", would simulate ordinary navigation by mouse scrolling/clicking.

Many pages contain input elements representing commands, such as submit, clear-form, buttons that activate Javascript functions, etc. Sighted users can easily identify and locate these, but blind users need to be told that the page has such inputs. One approach is to notify the blind user by voice output that the page contains such inputs when the page is first loaded. The user can then request a list of the available commands by voice, say by speaking the word "commands", at which point the browser can present the list of available commands by voice output. The user could speak the desired command, thus simulating a mouse click, or say "done".

Many pages contain input boxes for normal text, for example first and last name in a form or feedback to the web author or server. Many blind users are proficient typists, and those users may prefer to use the keyboard for normal text entry. These users need help bringing such input boxes into focus. A sighted user would just mouse over the box or use the tab key to move to the next box. For blind users a method similar to the one in the preceding paragraph could be used. To accomplish this, the list of text input boxes would be identified when the html file was retrieved from the server, as suggested for command inputs. On page load the list of these input boxes could be read out to the user. The user could focus on a text input by speaking the name of the input box and could indicate when he or she had finished text entry by speaking a word like "done". Alternatively, the user could speak commands like "first input box" or "next input box". Of course, some blind users may prefer voice input for both normal text and command

inputs. Thus, even for a single class of impaired users there would be a need for multiple types of preprocessing to allow for the preferences and abilities of individuals within that class.

Visually-oriented elements, like img and video, are of little direct use to a blind user. If the purpose of the element is purely decorative, the element can simply be removed before the page is loaded into the browser. If the element has a non-decorative purpose and the page author has added additional information through suitable attributes on the element, the element can be changed into a different form, for example into an audio element that contains the alternate information from the "alt" attribute value of an img element.

These are but a few suggestions of how information can be presented to blind users and how a web page might interact with a blind user. But it is enough to illustrate several points. First, obviously not every browser manufacturer is going to build all of these features and many more into their browsers. Likely, in fact, none of them will. However, the above can be accomplished by a system as proposed here. Extracting content from an HTML5 file (or any XML file) can be accomplished through XSLT transformation. An XSLT transformation can find the command elements in a suitably marked-up HTML5 file and generate suitable Javascript, for example, to accomplish the voice read-out and voice input. UIML files can be used to map functionality, such as voice output and voice recognition for verbal command input, to appropriate software either in the browser itself or on the user's computer or anywhere on the web.

3.2 Color Blind Users

Section 3.1 illustrates the main ideas proposed in this paper. We now present a few additional examples to illustrate the need for a large collection of libraries and also to illustrate the flexibility and adaptability of the method we propose.

Users who are color blind do not need drastic reorganization and presentation of the material. These users are able to locate and mouse-over the elements on the page. They are also able to see the organization of the page and locate both textual and non-textual information as easily as sighted users. Therefore, the elements of the original HTML5 page can be retained with little if any modification. Uses of color on the page would need modification, and this could probably be accomplished by transforming the styles sheets for the page rather than the HTML itself. An obvious example is the use of color to highlight links. For a color blind user links could be highlighted by emphasis plus underlining rather than color plus underlining. Other uses of color specified by the author of the page could be replaced by combinations of font size and style. If the author had followed the guidelines of HTML5, all the color information would have been included in a stylesheet. XSLT could be used to transform that stylesheet before the page was loaded into the browser. Alternatively, the whole stylesheet could be replaced by one from the library for color-blind users.

3.3 Hearing Impaired Users

Users who were hearing impaired but had normal sight would only need to have the sound elements modified. For video or voice audio one approach would be to import voice-to-text software or software that provides closed-captioning from video. UIML could provide the connection between the web page that was loaded by the browser and such software, again either on the user's own machine or from anywhere on the web.

3.4 Manually Impaired Users

Users who cannot use a mouse and keyboard could interact with the page by voice input. Again, assuming the user had normal sight, the elements of the page would not need reorganization. However, voice input and output would be used, and UIML would provide the link between elements on the page and appropriate software.

4 Conclusion and Future Work

Achieving universal access is extremely difficult. It is not enough just to read the text to a blind user or just to replace sounds by text for a deaf user. In many cases the content needs to be completely reorganized, and often the interaction also needs to be drastically modified. And, because there are many diverse groups of users with a variety of combinations of disabilities, this reorganization needs to be done in many different ways. It is unrealistic to expect browsers to be implemented that could achieve such a diversity of functionality, much less to expect that to be done on the variety of platforms ranging from PC browsers to cell phones to book readers to…. Further, the variety of platforms will increase over time, and it is not likely that all these new platforms will implement the presentation and functionality required for the many kinds of users. We have proposed an approach, based on reformatting marked up documents, that is feasible to implement, expandable as future platforms and document systems are introduced, and that provides usable access to a much wider array of users.

We close by noting that an important aspect of universal access has yet to be addressed by the HCI community. Prior work, including our work in this paper, has focused on the technology. There is little work on understanding how different types of users can best interact with the web. We have hinted in Sect. 3 at some plausible ways a few kinds of users might want to receive the content and interact with it, but we make no claims that our suggestions there are exhaustive or are the best ways for those users. We only presented those ideas to illustrate the method. What is needed is an interdisciplinary effort involving HCI professionals and also psychologists, sociologists, medical professionals, and others to gain a deep understanding and model of different types of users and what are the best ways for each type to interact with our machines.

Acknowledgements. This work was supported in part by the Project of Construction of Innovative Teams and Teacher Career Development for Universities and Colleges under Beijing Municipality (IDHT20130519).

References

1. Henschen, L., Lee, J.: Using semantic-level tags in HTML/XML Documents. In: Proceedings of the 13th HCII International Conference, pp. 683–692 (2009)
2. Henschen, L., Li, N., Shi, Y., Zhang, Y., Lee, J.: Intelligent document user interface design using MVC and UIML. In: Proceedings of the 16th HCII International Conference, vol. 1, Part I, pp. 423–432 (2014)
3. Li, N., Hou, X., Fang, C.: Harmonization of office document format and web page format driven by separating presentation from content. Submitted to The 15th ACM Symposium on Document Engineering, Lausanne, Switzerland
4. W3C.HTML5. http://www.w3.org/TR/html5
5. W3C XSL Transformation (XSLT) Version 2.0. http://www.w3.org/TR/xslt20
6. World Wide Web Consortium. http://www.w3.org/TR/wai-aria
7. DAISY. http://www.daisy.org
8. Henschen, L., Lee, J., Li, N., Hou, X., Gao, X.: Parallelization of the XSLT transformation process by XSLT stylesheet partitioning. Submitted to ACM Transactions on the Web