Adapting Desktop Web Pages for Vocal Browsing

Fabio Paternò and Christian Sisti

CNR-ISTI, HIIS Laboratory,
Via Moruzzi 1, 56124 Pisa, Italy
{fabio.paterno,christian.sisti}@isti.cnr.it

Abstract. In this paper we describe a solution to make Web pages more suitable for vocal browsing by analyzing and modifying their logical structure. The solution exploits intermediate logical descriptions that are automatically created by reverse engineering techniques. The adaptation engine aims to identify the main logical structure of the Web page components and remove the aspects specific to the graphical modality. Then, a vocal implementation is generated to support browsing, which begins by the user's selecting from the main components.

Keywords: Adaptation, Web sites, Vocal Browsing, User Interface Models, Accessibility.

1 Introduction

Emerging ubiquitous environments call for supporting access through a variety of interactive devices. Vocal technologies and related applications are steadily improving and this makes possible services such as vocal searches and map navigation by Google or others.

Web applications are widely used and a number of approaches and techniques have been proposed to support their adaptation to mobile devices. Little attention, however, has been paid so far on how to adapt Web applications for vocal access. This seems useful and important, given the recent improvements in vocal technologies, which can allow users to access their applications when the visual channel is busy (e.g. when driving) or when they are visually-impaired. The assistive technology for blind people (mainly screen readers) have a number of usability limitations because they usually access the page implementation and require keyboard selection of the elements of interest, with little ability to filter content; only type filtering is currently available (e.g. listing links, headers, or other types). Thus, often users have to listen to a lot of useless content before reaching the point of interest or are provided with information that is difficult to interpret because they do not know the surrounding context in the Web page.

A work that has similar goals is DANTE [3], which provides semantic transcoding for visually-impaired users. For this purpose it uses ontology-based annotations that are manually created and included in the Web page. Such annotations are then used

P. Campos et al. (Eds.): INTERACT 2011, Part III, LNCS 6948, pp. 628–635, 2011.

for automatically transforming the original pages into pages more easily accessible for the visually-impaired. In our case we aim to obtain a tool that is able to automatically analyze the content of Web pages and transform them into vocal applications that are sufficiently usable, with the possibility to customize some adaptation parameters. For this purpose we consider intermediate model-based representations [1], which aim to represent the logical structure of the user interface and enable relevant reasoning to derive a more suitable structure for the vocal channel. While previous work (e.g. [5]) has considered the use of model-based techniques in the authoring of vocal or multimodal applications, we exploit them in the graphical-to-vocal adaptation obtained by transforming graphical Web pages in order to make their content browsable through (*menu-based*) vocal interaction.

2 The Proposed Approach to Vocal Adaptation

We exploit a model-based language [4] for performing a more semantically-oriented transformation. The MARIA framework provides abstract (independent of the interaction modality) and concrete (dependent on the interaction modality but independent of the implementation) languages. Such languages share the same structure with different levels of refinements. A user interface is composed of one or multiple *presentations*. While in graphical interfaces the concept of presentation can be easily mapped onto that of a set of user interface elements perceivable at a given time (e.g. a page in the Web context), in the case of a vocal interface we consider a presentation as a set of communications between the vocal device and the user that can be considered as a logical unit. One example is the case in which the dialogue between the user and the vocal application aims to provide some pieces of information to fill in a form regarding the user. The user interface elements are called *interactors.* Examples of interactors are *navigators* (allow users to move from one presentation to another) and *description* (allow provide output to the user). Each presentation is structured using *interactor compositions* such as *groupings* and *relations.* The grouping operators identify the main logical parts of the interface and contain both interactors and interactor compositions (such as groupings itself). The relation operator defines a kind of relation between two groups of elements, typical example is a form with one group for editing input values and one for controlling them (clearing, sending to the server, …).

Our solution is based on an **adaptation server**, which provides a number of functionalities (see Figure 1). Firstly the *Reverser* automatically parses the content of the Web page and the associated style sheets, and builds a corresponding concrete logical description. Secondly the *Graphical-to-Vocal Adapter* transform the graphical concrete description into a vocal one, which is optimized for vocal browsing. Lastly the *VoiceXML Generator* produce the final vocal interface so that the final result can be loaded onto a vocal browser for execution.

The approach has addressed various issues: *content*, some graphical content is badly rendered on vocal interfaces; *structure***,** the structure of a graphical page is

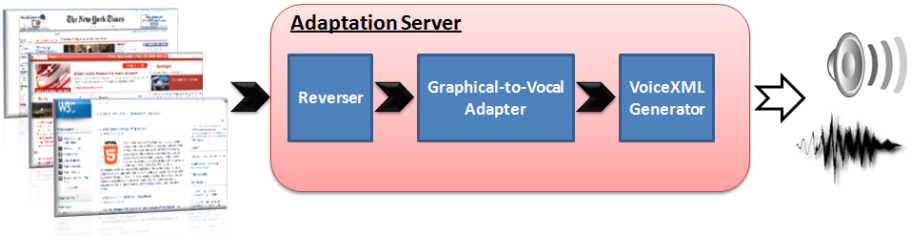

Fig. 1. The global adaptation architecture

typically deeply nested and not suitable for vocal menus; *menu items*, it is crucial to find names for the activation of the menu items that allow users to understand what the corresponding content will be if they are selected.

3 An Example

In order to illustrate how our approach works, we consider an example of a widely known Web site: the W3C. Figure 2 shows an example of the adaptation process, which generates two hierarchical levels. The screen dump (left) has been annotated with solid lines representing the higher sections while the dotted lines represent the nested sub-sections. In addition, we have added the text labels corresponding to the items of the corresponding vocal menus. Figure 2, right, shows the vocal menu structure of the example. Intermediate nodes are menus while leafs represent content.

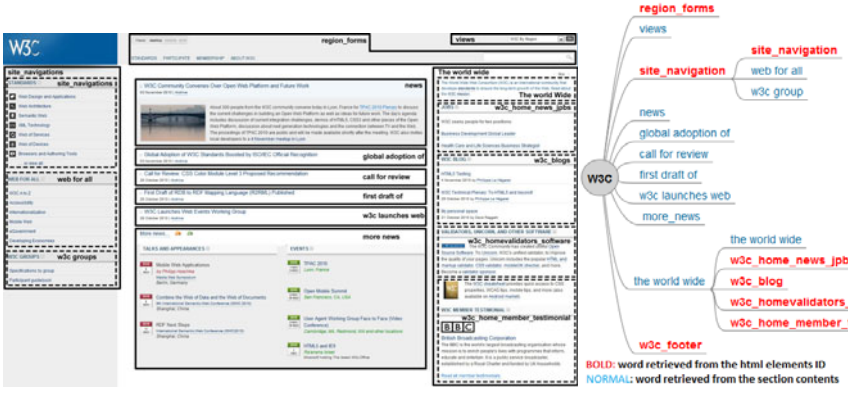

Fig. 2. W3C Interface Splitting and Vocal Menu Structure

Adjusting the adaptation process parameters (described in next sections) allows users to increase or reduce the level of nesting in the vocal menus navigation and, consequently, to control the fragmentation of the resulting vocal interfaces. The table below shows a possible resulting dialogue for the example in Figure 2.

Table 1. The Vocal Dialogue in the Example

	Input / Output	Action
System:	Welcome to Word Wide Web Consortium W3C Main Menu. Choose one of the following sections: region form, views, …	
User:	Call for review	*Menu Activation*
System:	You are in Call for review. Call for Review: CSS Color Module Level 3 Proposed …	
User:	Repeat	*Ask the system to repeat*
System:	Call for review. Call for Review: CSS Color Module Level 3 Proposed …	
User:	Next	*Go to the next part*
System:	Moving to part "The Cascading Style". The Cascading Style Sheet Working Group has published …	
User:	Previous	*Go back to the previous menu*
System:	Moving back to the main menu. Welcome to Word Wide Web Consortium W3C Main Menu. Choose one of the following sections: region form, views, …	
User:	Exit	*Close the interface.*
System:	Goodbye!	

4 The Graphical-to-Vocal Adaptation

This transformation is performed after the reverse engineering phase, in which an analysis of the implementation in terms of all the HTML tags and associated CSS files is performed on the Web page considered in order to build the corresponding logical description.

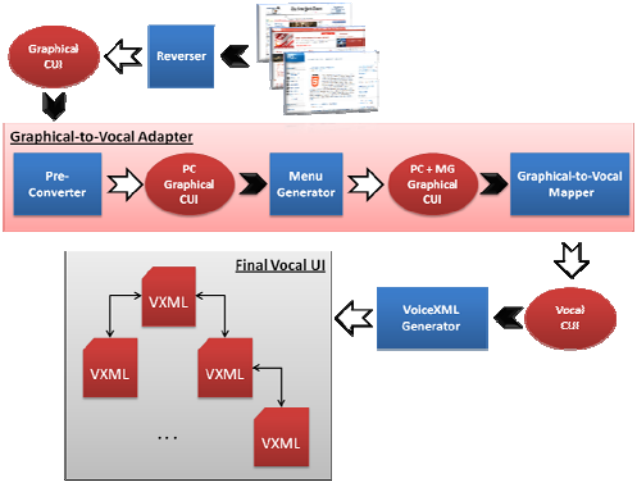

Fig. 3. The adaptation process in detail

The transformation is carried out through various phases (see Figure 3). Firstly the *Pre-Converter* processes the graphical concrete logical descriptions by removing elements that are considered useless for the overall conversion (e.g. images without alternative text description). Secondly, the *Menu Generator* produces a new hierarchical structure in order to allow users to navigate the interface through menus/submenus. Finally, the *Graphical-to-Vocal Mapper* transforms the graphical concrete elements onto concrete vocal ones having the same semantic effects but using different attributes.

4.1 Concrete Graphical Description Pre-conversion

A number of steps are necessary to transform the logical graphical description in input into a cleaner and more suitable one, which is then used for deriving the vocal logical description. In this work we have identified a number of parameters that can be modified in order to customize the adaptation process.

We perform four steps for the pre-conversion that should be accomplished in sequence. The first consists of removing the elements that are problematic for vocal interfaces. In particular we firstly remove the interactors that contain *special characters* (this option is user settable): the use of special symbols (e.g., chinese alphabet if not supported) by the voice browser may be problematic. Secondly, we remove all the image interactors (description, navigator) *without the ALT attribute*: without alternative description it is impossible to render an image vocally. Then we remove the description/image interactors that have *width or height less than a threshold* (this parameter is user settable): usually small images (but not image link) are used more for decoration rather than to provide content. We also remove *redundant navigator interactors*: if we find navigators with more than one label (image, text), then we maintain only one of them, preferably the textual one.

In the second step we simply remove any grouping that contains zero or one child. This step has to be performed recursively until the specification no longer requires further modifications.

In the third step we identify the words/sentences that are candidates as the title/subtitle for menu items. A *candidate word/sentence* is a description/text that has a *role* attribute of “heading1”, “heading2” or “heading3”, and that is placed just before a grouping. In this way we detect an association between title/subtitles and grouping that provides better results in the menu generation process. Candidate words/sentences are moved inside the grouping in order to facilitate their use in the corresponding menu item generation, which is performed afterwards.

The purpose of the fourth step is to avoid too heavily nested groupings, which would then be transformed into too many vocal menus. In this step after removing the groupings *with only one child*, which may be created by the previous step, the tool removes the groupings *whose children are only groupings,* only when the children are deeply nested (compared with the global structure). In this way we reduce the complexity of the structure, moving towards the root the information.

4.2 Menu Generation

The result of the previous phase makes it possible to generate menu sections that define a new logical structure of the interface itself. Moreover, through the choice of

appropriate parameters, it is possible to reduce the number of menu items, in case they are too many, in order to avoid short term memory overload.

We designed and implemented a recursive process that analyzes the presentations in the graphical logical description and decides whether to split them into connected multiple presentations. At the end of the process, there will be one presentation containing a menu and the others the actual content. The purpose of the menu generation phase is to identify when we should create menus that will allow the users to choose among various options. A presentation is split (and a new menu is generated) if at least one of the following conditions is verified: *it contains more than one relation*, we consider relations to be atomic and consistent interactions corresponding to form filling, so it is desirable to have a menu item for each of them. *The number of navigators is much greater than the number of description interactors*, this is the case of navigator bars, which should correspond to one single vocal presentation with the associated menu. The threshold ratio of navigators to description interactors can be defined by the designers. *The length of the overall text content exceeds a fixed threshold (user settable)*: this avoids the generation of unbalanced sections in terms of content. Parameterization can be performed through a user interface, which has on the left the modifiable parameters, and on the right the graphical representation of the structure of the corresponding vocal interface, useful for users to have an idea of the structure of the resulting vocal interface.

One of the main problems in the menu generation is how to find keywords (or short sentences) that describe the content of the destination. One possibility is to use the grouping ID originally defined by the designer of the graphical interface. In the case that this value is missing the tool performs an analysis of the grouping content and uses the string of the first description/text with the highest text *role* present in the content considered. The text roles indicated in the graphical concrete description are sorted from the most to the least important in this way: *Heading1, Heading2, Heading3, Heading4, Heading5, Heading6, Strong, Emphasis, List_element, Paragraph, Normal*. If the string is a sentence and not a simple word we use the first three words in the sentence. If no text role is specified, then we select the first string that we find (max 3 words) in the actual content.

4.3 Desktop to Vocal UI Mappings

This is a transformation that takes the elements of the graphical concrete presentations, as modified in the previous stages, and translates them into corresponding elements of the vocal concrete language that have similar effects.

The presentation *title* is used to build a suitable introduction sentence to use as the *welcome message* for the first time the user accesses the application.

Regarding the output-only interactors, we provide the following processing. The text elements are directly mapped onto speech elements. The audio element is mapped onto a *prerecorded message* vocal interactor for rendering. In the case of images, if the ALT attribute is specified in the source document, then we transform the element *image* into a *speech* element that renders the attribute. As in the case of an image, for videos we use the ALT attribute (when present) to vocally render its description, otherwise it is discarded. The *audio* elements are simply mapped onto the *prerecorded message* interactors. Data tables are mapped into *vocal tables*.

Regarding the control interactors we provide for the following processing. The text links are mapped onto the *vocal link* element. The name of the link is used as *activation vocal sentence*. The sentences that users must say to activate a link will be synthesized with higher pitch in order to let the user recognize it. For the image link currently we use the *navigator ID* as the corresponding vocal command. Graphical buttons are mapped onto *vocal links,* triggered by pronouncing their labels.

In the case of graphical interactors supporting selection, they are managed differently depending on whether they support *single* (radio button, list box, drop down list) or *multiple* selections (check box, list box), and consequently associated with the single or the multiple vocal selection interactors. In both single and multiple vocal selection, the labels of the graphical choice elements are used to build either the request or the list of accepted inputs.

The edit interactors allow users to enter pieces of information of various types. They raise the issue of the generation and use of grammars for vocal interfaces. For *numerical input* (spin boxes, track bars) we set a grammar able to recognize numbers. In case of *textual input,* we instead introduce two pre-generated grammars: a vocal grammar able to recognize short sentences composed of a well-defined list of words; and a DTMF grammar able to transform the keyboard inputs into words in a way similar to that used to write text messages in mobile phones. Lastly, once the vocal concrete description has been obtained it can be transformed into a VoiceXML implementation. The type of approach used for this purpose is described in [5].

4.4 Evaluation

We conducted a first user test of the results of our adaptation process on two Web sites: W3C[1] and BBC[2]. We gathered a number of positive user impressions and criticisms that gave us further motivation.

Accessibility was one of the strong points underlined by some users. This approach in fact expands the accessibility horizons not only for visually-impaired users, but also for people without Internet connection or, moreover, when the visual modality is busy (e.g. checking email when driving a car, follow a recipe while cooking, etc.). Another strong point is the simple navigation, also useful for people who are not used to interacting through vocal interfaces.

Some users criticized the TTS and the Speech Recognizer, but these features are not part of our architecture. Three users considered some of the sentences for menu activation too long, while another one asked for more attention in terms of re-prompting and help hints when the user seems to stop during navigation. Some users seemed confused by the terms selected to identify the structure of the generated vocal interface. The term “section part”, for example, misleads users to draw a parallel with book structures. A related problem was the name of some commands that seemed hard to memorize. For example using “next/previous part” instead of “next/back” for the navigation. Some users suggested expanding the command list.

[1] http://www.w3.org/
[2] http://www.bbc.co.uk/

5 Conclusions and Future Work

Our automatic adapting process has been applied to twenty Web pages including W3C and BBC Web sites, which have been used in the user test. The generated VoiceXML vocal interfaces have been tested with the VOXEO Voice Browser[3] and have passed the validation test integrated in it. The applications have been used through VoIP access to the vocal server. A configuration tool was developed in order to change the parameters of the adaptation process and have a preview of the structure of the generated vocal application. This allows the possibility for developers of vocal applications to control the adaptation process and better tailor it to their needs.
In general, the results depend on the original content. The solution is not able to manage content such as Flash or Java applets and the results are not as good with Web content that does not follow the standard W3C guidelines for Web applications.
Future work will be dedicated to improving the transformation rules and further performing empirical validation. We also plan to extend the adaptation support to HTML5 applications.

Acknowledgments. We gratefully thank the support from the EU ICT SERENOA Project (http://www.serenoa-fp7.eu/).

References

1. Fonseca, J.M.C. (ed.): W3C Model-Based UI XG Final Report (May 2010), http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui-20100504/
2. Franz, A., Milch, B.: Searching the web by Voice. In: Proceeding of the 19th International Conference on Computational Linguistic, Stroudsburg, PA, USA, vol. 2, pp. 1–5 (2002)
3. Yesilada, Y., Stevens, R., Harper, S., Goble, C.: Evaluating DANTE: Semantic transcoding for visually disabled users. ACM Transactions on Human-Computer Interaction 14(3), Article 14 (2007)
4. Paternò, F., Santoro, C., Spano, L.D.: MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. ACM Trans. Comput.-Hum. Interact. 16(4) (2009)
5. Paternò, F., Sisti, C.: Deriving Vocal Interfaces from Logical Descriptions in Multi-device Authoring Environments. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 204–217. Springer, Heidelberg (2010)

[3] http://www.voxeo.com/