Model-Based Customizable Adaptation of Web Applications for Vocal Browsing

Fabio Paternò, 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. It is possible to customize some parameters of the adaptation transformation in order to better control its results.

Categories and Subject Descriptors
H.5.2 **[Information Interfaces and Presentation]**: User Interfaces – D.2.2 **[Software Engineering]**: Design Tools and Techniques – User interfaces.

General Terms
Design, Human Factors.

Keywords
Adaptation, Vocal Interfaces, Web Applications.

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 (e.g. [3], [4]). 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. While solutions exist for supporting blind people in specific domains [5], the general 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 [10].

In this paper we present an original solution for transforming graphical Web pages in order to make their content browsable through vocal interaction. Our target is to generate VoiceXML vocal applications, that are semantically equivalent to the originals but accessible through menu-based vocal interaction. The solution exploits intermediate model-based logical descriptions in the MARIA framework [11], which provides one abstract (platform-independent) user interface language, and a number of concrete (platform-dependent) user interface languages. The implementation is based on a pipeline of XSL Transformations. In order to improve the flexibility of our adaptation tool, we also provide designers and developers with the possibility of customizing the adaptation rules, by specifying the values of some parameters that allow them to change the results accordingly. In the paper, after discussion of related work, we introduce our approach and provide an example application with an existing (widely known) Web site to better illustrate the possible results. The next section provides a more detailed description of how the adaptation transformations have been designed and implemented, along with the description of the customization tool. Lastly, some conclusions along with indication of future work are drawn.

SIGDOC'11, October 3–5, 2011, Pisa, Italy.

2. RELATED WORK
Recognition of natural language input is improving [7] and this opens up the possibility of obtaining vocal user interfaces able to recognize any user input.

A number of authoring environments for multimodal user interfaces has recently been proposed. The main goal of the MultiModal Web Approach's authoring environment [15] is enhancing the dissemination of knowledge used by proven solutions to recurring problems in the multimodal context. XFormsMM [8] is an attempt to extend XForms in order to derive both graphical and vocal interfaces. In this case the basic idea is to specify the abstract controls with XForms elements and then use aural and visual CSS for vocal and graphical rendering, respectively. The problem in this case is that aural CSS have limited possibilities in terms of vocal interaction and the solution proposed requires a specific ad hoc environment in order to work. We propose a more general solution able to derive implementations in the W3C standard Voice XML.

An approach to providing a portal engine with the capability to communicate and receive inputs using voice is presented in [1]. This solution indicates guidelines for developing multimodal Web sites exploiting combination of graphical and vocal modality but does not provide any specific tool to support automatic conversion from graphical to vocal or multimodal.

In contrast to [16], which aims to understand how cognitive

models can be useful to predict usability issues for screen reader users, we exploit models of user interfaces to build the logical descriptions and then adapt them for vocal browsing. A work that has similar goals is DANTE [9], 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 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 a Web page and transform it into a vocal application that is sufficiently usable, with the possibility to customize some adaptation parameters. For this purpose we consider intermediate model-based representations [6], 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. [14]) has considered the use of model-based techniques in the authoring of vocal or multimodal applications, we exploit them in the automatic adaptation from graphical-to-vocal content.

3. THE PROPOSED MODEL-BASED APPROACH TO VOCAL ADAPTATION

As mentioned, we exploit a model-based language [11] for performing a transformation that preserves the semantics of the interaction. The 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. An AUI (Abstract User Interface) is composed of a number of *presentations,* a *data model* and a set of *external functions*. Moreover, each presentation contains a number of user interface elements, called *interactors*, and a number of *interactor composition* operators. Examples of interactor composition operators are *grouping* and *relations* to group/relate different interactors. The interactors are first classified in abstract terms of *editing*, *selection*, *output* and *control* and may have a number of associated *event handlers*. While in graphical interfaces the concept of presentation can be easily defined as 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 application and the user that can be considered as a logical unit, e.g. a dialogue supporting the collection of information regarding a user. Examples of interactors are *navigators* (allow moving from one presentation to another) and *description* (allow TTS functionality). A grouping can contain both interactors and other composition interactors (such as groupings itself). Another composition concept used to structure the presentation is the relation operator, which 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):

- **Reverser**, parses the content of the Web page and the associated style sheets, and builds up a corresponding Desktop Concrete Logical Description;
- **Graphical-to-Vocal Adapter**, transforms the Desktop Concrete Logical Description into a Vocal Concrete Logical Description, which is optimized for vocal browsing;
- **VoiceXML Generator**, a VoiceXML implementation is generated from the vocal concrete description so that the final result can be loaded onto a vocal browser for execution.

The work described in this paper focuses principally on the Graphical-to-Vocal Adapter. In the next sections we first describe an adaptation example obtained with our approach, and then we describe our solution for the Graphical-to-Vocal Adaptation process.

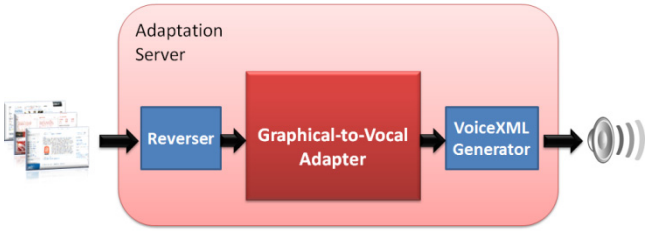

Figure 1: The global adaptation architecture.

4. AN EXAMPLE

In order to illustrate how our approach works, we consider an example of widely known Web site (BBC[2] , see Figure 2).

Figure 2: BBC User Interface Logical Structure.

We can thus see how we address one of the most complicated issue in the adaptation: recognizing the different logical parts that a graphical page may be subdivided into, information which is then used to structure the vocal navigation (as explained in Section 5.2). Figure 2 shows the components identified, which are structured into two hierarchical levels.
The screen dump has been annotated with solid lines representing the higher-level sections while the dotted lines represent the nested sub-sections. In addition, we have added the text labels of the corresponding items in the generated vocal menus.
Figure 3 shows an example of the vocal menu structure, in which intermediate nodes are menus while leafs represent content, for the Web page in Figure 2. Adjusting the adaptation process parameters (see next sections) permits increasing or reducing the level of splitting and, consequently, the fragmentation of the result. Another crucial point in the process is the identification of meaningful word/sentences that best describe the content of a logical part. In the next sections we describe the algorithm developed for determining the menu items and explain what the possible sources of information are.

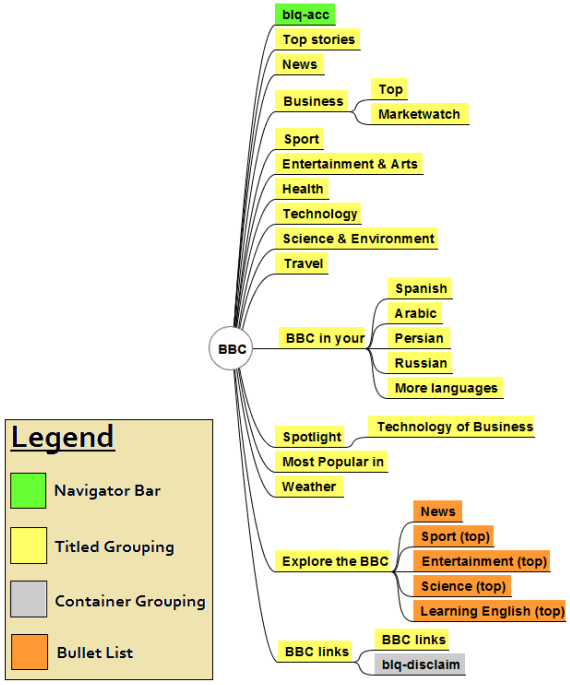

Figure 3: BBC Vocal Menu Structure.

The table below shows an example of a possible resulting dialogue.

	Input / Output	Description
System:	Welcome to BBC Main Menu. Choose one of the following sections: ..., Business, Sport, ...	*Introduction of a menu section*
User:	Sport	*Menu Activation*
System:	You are in Sport. Fulham appoint Jol as new Manager ...	
User:	Repeat	*Speak again the current dialogue*
System:	Sport. Fulham appoint Jol as new Manager ...	
User:	Next	*Skip to the next part*
System:	Moving to part "Mosley casts doubts". Mosley casts doubts over Bahrain ...	
User:	Previous	*Go back to the previous menu.*
System:	Moving back to the main menu. Welcome to BBC Main Menu. Choose one of the following sections: ..., Business, Sport, ...	
User:	Exit	*Close the interface.*
System:	Goodbye!	

Table 1: The Vocal Dialogue in the Example.

5. THE GRAPHICAL TO VOCAL ADAPTATION

Adapting a graphical Web page into a menu-based vocal one poses a number of specific problems. We summarize them into three categories:

- **Content:** some contents of a graphical page translates poorly into the vocal modality. For example, a Web page could contain text not supported by the target Voice Browsers (e.g. Chinese alphabet symbols).
- **Structure:** although the logical description of a graphical page is a tree (good for menu navigation), the depth and the width are typically too large for vocal browsing. So there is a risk of generating a large number of nested menus with little content on the leaves.
- **Menu Items Naming:** it is crucial to find simple meaningful names for activating the menu items that, in some way, anticipate the corresponding node content.

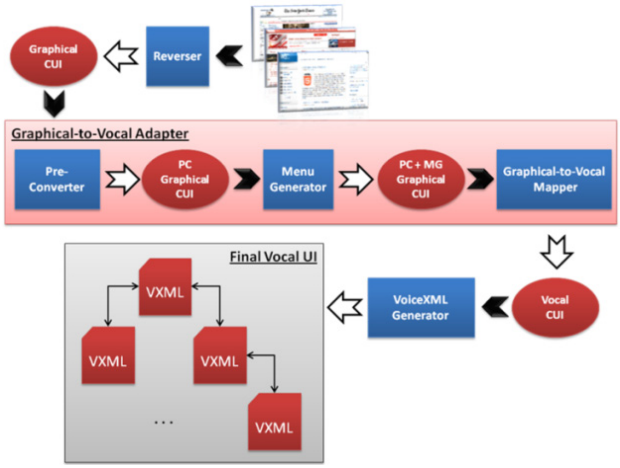

Figure 4: The adaptation process in details.

The adaptation 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. The transformation is carried out through various phases (see Figure 4):

- **Pre-Converter:** 1) performs the *Content Optimization* by removing from the graphical logical description the elements that badly reflect into vocal interfaces (e.g., images without alternative text); 2) performs the

Structure Optimization by recognizing the page components and removing the unnecessary one (e.g., grouping elements used for only formatting purposes); 3) performs the *Calculate Cost* computation (see next sections), whose results are used in the Menu Generator phase.

- **Menu Generator:** a new hierarchical structure is generated in order to allow navigating the interface through menus/submenus.
- **Graphical-to-Vocal Mapper:** the graphical interface elements are mapped onto vocal ones having the same semantic effect.

5.1 Concrete Graphical Description Pre-Conversion

Content Optimization

A logical specification automatically generated by the reverse engineering of graphical desktop Web pages contains elements that might have no sense in a vocal interpretation because they are used for graphical formatting purposes. For this reason, we have developed an algorithm to find and discard elements that are irrelevant. More in detail, this *Pre-Conversion* step performs the following:

- *Remove the interactors not graphically visible*: examples are interactors derived from hidden HTML elements, text with font size equals to zero or image with width, height equals to zero.
- *Remove the images that cannot be rendered vocally:* they are the images without the ALT attribute. In any case the "image links" are preserved and accessed by their destination path.
- *Correct grouping inconsistency:* due to particular graphical page layouts or previous processing, it is possible to have groupings without content or with only one child (so they are no longer definable as groupings). In these cases the groupings are removed and their (possible) contents are moved up in the tree structure.
- *Normalize words:* this feature transforms words containing accents or other diacritics into normalized forms without them (e.g., España is transformed into Espana) to avoid possible encoding errors. This feature can be disabled. More fine-grained normalization techniques will be implemented in future versions. For details about text normalization techniques see [17].
- *Remove unsupported words:* characters not supported by the Voice Browser (e.g., 中文 in English based applications) could cause crashes in the final vocal interface. To avoid this we permit choosing the supported character set. All the words that contain at least one character not in the set are removed. At the moment we provide support for three charsets: 1) Basic Latin 2) Basic Latin + Supplement 3) Arabic. We use regular expressions to recognize the words to delete: for example the following *regular expression* identifies all the words that contain non-Basic Latin characters:

```
\w*\P{IsBasicLatin}+\w*
```

Structure Optimization

The graphical logical description generated by the reverse tool contains hundreds of groupings used only for formatting purposes. This description, if not appropriately processed, can generate hundreds of vocal menus (and menu items) that will direct to sections with meaningless content. Thus, our solution aims to identify the different kinds of groupings and remove all those useless for vocal navigation.

For this purpose, we defined seven classes of composite interactors: 1) *Root;* 2) *Form:* in MARIA defined as a relation 3) *Bullet List*; 4) *Bullet,*; 5) *Navigator Bars,*; 6) *Titled,* a composition of user interface elements associated with a title by the "titling algorithm"; 7) *Container*, the remaining hybrid composition operators (containing at least one interactor). We then follow this class order to implement a *decision tree* to classify each composite operator in the graphical logical description. Each composition operator that does not belong to any of such classes is removed, and its content is moved up.

Since in HTML there is no strong semantic binding between page components and headings, connecting them is delegated to the user's perception of the actual presentation of the page. The same problem affects the graphical logical description, so the *titling algorithm* aims to perform this binding. After a study of a number of common Web pages we found four patterns. Figure 5 shows the patterns and the algorithm's behaviour.

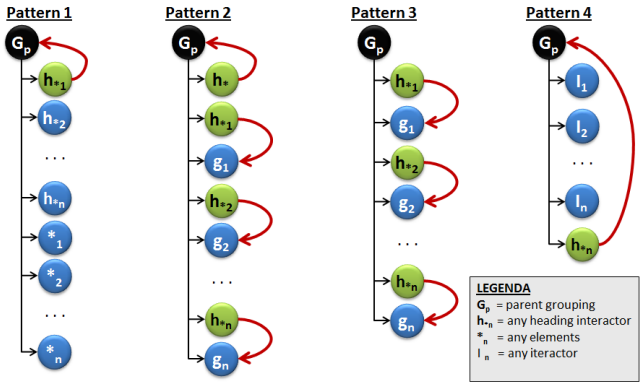

Figure 5: Heading Patterns.

$\mathbf{G_p}$ is a grouping containing some children; $\mathbf{h_{*n}}$ is a heading interactor (an interactor that has the attribute "role = heading1, heading2, …"); $\mathbf{*_n}$ is any possible element and $\mathbf{I_n}$ is a generic interactor. The curved arrow indicates the bindings between headings and groupings.

More in detail, *pattern 1* shows a grouping that contains a number of headings as top-child, followed by a series of other elements. In this case the first heading is used as title for the grouping parent. In *pattern 2* we have a heading as first child and then a series of couples [heading-grouping]. Now we use the top child as title for the overall grouping parent and the preceding headings as titles for the relative groupings. *Pattern 3* is similar to the previous one: we have the same couples [heading-grouping] but the top-child to be associated with the grouping parent is missing. Finally, *pattern 4* represents a situation with a number of interactors as top-child and then a heading that is used as title for the grouping parent.

Figure 6 shows the average grouping number and distribution (among the tree levels) before (dotted line) and after (solid line) the structure adaptation process was applied to five well-known Web sites (*w3.org*, *bbc.com*, *ebay.com*, *msn.com* and *yahoo.com)*.

On average the number of groupings was reduced from 455.2 to 90.2 and the number of levels from 20 to 9.

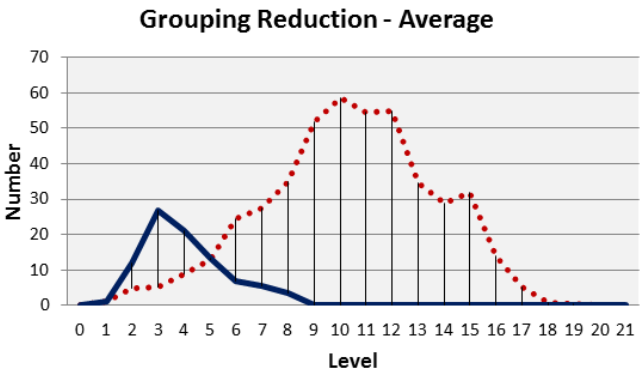

Figure 6: Grouping Reduction Average.

Calculate Cost

The next *Menu Generation* step performs a splitting of the original structure of the graphical logical description into different sections (which will be the new vocal presentations) connected by *menu presentations* (presentations containing the logic for the vocal menu browsing). Our goal is to avoid sections with too much content, so one simple solution would be to split the groupings that contain a large amount of text. This idea seems good but imagine having a grouping that contains two descriptions of ten words each and another grouping with two bullet lists of ten bullets each. Supposing that both groupings have the same overall number of characters, users will most likely find it logical to listen to the first grouping, since it is probably a sequential text that should be accessed as a single logical unit. On the other hand, the second one, which is mainly composed of two different lists of items should be separated. For this reason we introduce a heuristic to calculate the **cost** of each component class giving different **weights** to each of them. To calculate the costs we use the following recursive function:

$$c(n) = \begin{cases} l(n) & if\ n\ is\ interactor \\ w(n) * \sum_{i=0}^{k} c(child(n)_i) & otherwise \end{cases}$$

Where n is a node , $w(n)$ is a weight associated with the class, $child(n)_i$ is the i-th child of n (with k = number of n children) and $l(x)$ is the function that calculates the amount of synthesizable text in the node x. The stop condition is that the node n is an interactor (not classified).

In other terms, the cost quantifies the size of a well-defined page component depending on its class. The weights define the proportions (in terms of linguistic load) between the different page component classes.

The weights are parameters in the range [1..100]. Increasing the weight of a class means increasing the probability that the elements of that class will be associated with a specific section, and thus will be directly accessible through a separate vocal menu item (see next section for an example). The customization interface (see Section 6) supports the editing of the weights in order to change the structure of the resulting vocal application.

5.2 Menu Generation

The result of the previous phase generates a logical description, which has removed a number of elements deemed useless for the generation of the vocal description. The resulting description is based on the primary groupings, and makes it possible to generate menu sections (with a limited number of choice items) that define a new logical structure of the interface itself. This structure allows the user to better vocally navigate the content. We designed and implemented a recursive algorithm that analyses the presentations in the graphical logical description and recursively splits each of them into multiple presentations until the basic presentations for the vocal interface have been identified. Some of these will contain only one menu and others the actual content, which can be either information to deliver or some interactions to support user-generated input. 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. As mentioned each menu will be associated with one single presentation. A presentation is split (and a new menu is generated) if the cost (calculated in the *pre-processing* step) is above a *threshold* (which can be changed through the customization interface).

Figure 7 below shows the general navigation structure generated by the process. Both sections and menus are new presentations. The menus contain the logical description of the mechanism to navigate in the presentation children. The sections instead contain some content retrieved from the original graphical interface. The section itself contains the logical description of the mechanism to navigate through its own different parts.

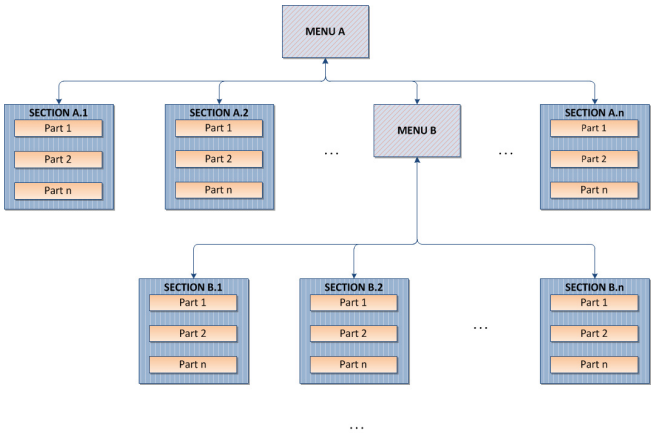

Figure 7: The General Navigation Structure.

The designer can customize the adaptation process choosing the value of the weights. Increasing the values increases the probability that a presentation is split. In the figure below we can see an example of how the weights may influence the splitting.

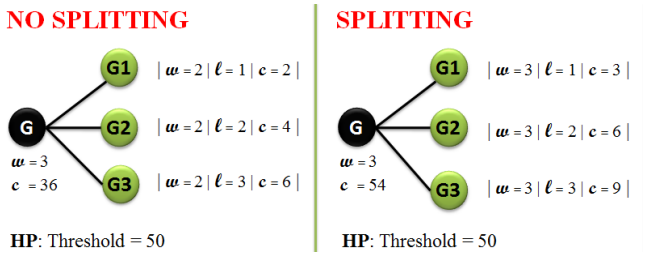

Figure 8: Weight influence in splitting

In the left side we can see that the cost (c) calculated in the node G is below the (predefined) threshold. In this case the child nodes become part of section G and will be evaluated (in the final vocal

interface) in the order G1, G2, G3. Increasing the weight (ω) from 2 to 3, the total cost is above the threshold and the presentation will be split into three new sections (presentations), accessible through a new menu presentation that allow direct access to each one.

One of the main problems in the menu generation is how to find keywords (or short sentences) that summarize the content of the destination. Our algorithm to retrieve the menu items works as described in the following:

1. **If** the node to be referenced has a "title" attribute (obtained in *pre-processing* step) use it.
2. **Else if** the same node has a descendant with a "title" attribute, use the first that is found.
3. **Else if** the node ID has been defined by the original user interface designer then use it.
4. **Otherwise** retrieve the main word from the node.

In order to retrieve the main word, the tool performs an analysis of the 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 no text role is specified, then we select the first string that we find in the actual content. In general for each menu items we limit the length at three words.

Testing our algorithm on various Web sites we found that the max number of menu items of the generated menu-based tree can be up to 20. This raises the issue of the user *short term memory effort*. We address this problem by adding a new feature to our Menu Generation process: when there are too many menu items (a limit can be set in the customization interface), the transformation splits the original menu into smaller sub-menus.

5.3 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. It was introduced in [12], each graphical presentation obtained is thus transformed into one vocal presentation.

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. A vocal presentation may have a number of associated vocal commands. The transformation enables the following commands: *previous*, to go back to the last menu heard; *main menu*, to go back (at any moment) to the main menu, *next/back*: allows the user to navigate back and forth through different parts of the same section (see Figure 6); *start/stop*, pauses the interface at any time and restarts it at the same dialogue position; *reprompt*, enable repeating the current dialogue; and *exit*, to close the vocal interface.

In the style to emphasis mappings we are constrained by the limited support of the vocal browsers. Regarding the output-only interactors, we provide the following processing. The text elements are directly mapped onto speech elements. If the text element defines a path to the content, we map this information directly onto the speech element. In the case of audio content, if the ALT attribute is set, we use it to give a description of the audio source. 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 mapped onto the *prerecorded message* interactors; if the ALT attribute is available we use it to provide a general description of the audio resource. Data tables are simply mapped into *vocal tables*. Two table browsing algorithms [13] have been implemented: *linearization,* if the table contains headers, *intelligent browsing,* if no header is present.

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. The image link differs because in this case we want to communicate to the user what to say to activate the link, but we do not have an explicit title for the link. Currently we use the *navigator ID* as the corresponding vocal command, where the ID is that defined by the developers of the original Web pages. Graphical buttons are mapped onto *vocal links,* triggered by pronouncing their labels or, if a button is an image, by pronouncing the corresponding ID. If the button is used to submit the values of a form then it is mapped onto a *vocal submit*. This interactor is automatically enabled when the all the form input fields have been filled.

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. To explain this kind of interaction to users, the vocal request contains the following sentence: "Single/Multiple Choice. Which would you like to choose: *list of admissible input* ?".

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* (which graphically can be provided by spin boxes, track bars and other similar techniques) 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.

6. CUSTOMIZATION TOOL

The adaptation process can be customized by the designers defining a number of parameters. Figure 9 shows the user interface that allows such parameterization: on the left the modifiable parameters, on the right the graphical representation of the structure of the corresponding vocal interface. In particular, the left panel is divided into three sub-panels: *input,* to choose the Web page that will be adapted and to start the process; *FTP Parameters,* to set the parameters of the FTP connection in order to send the generated VoiceXML files to the Voice Browser; *Transformation Parameters,* to set the parameters that influence the adaptation (e.g. the minimum size of an image in order to be considered relevant).

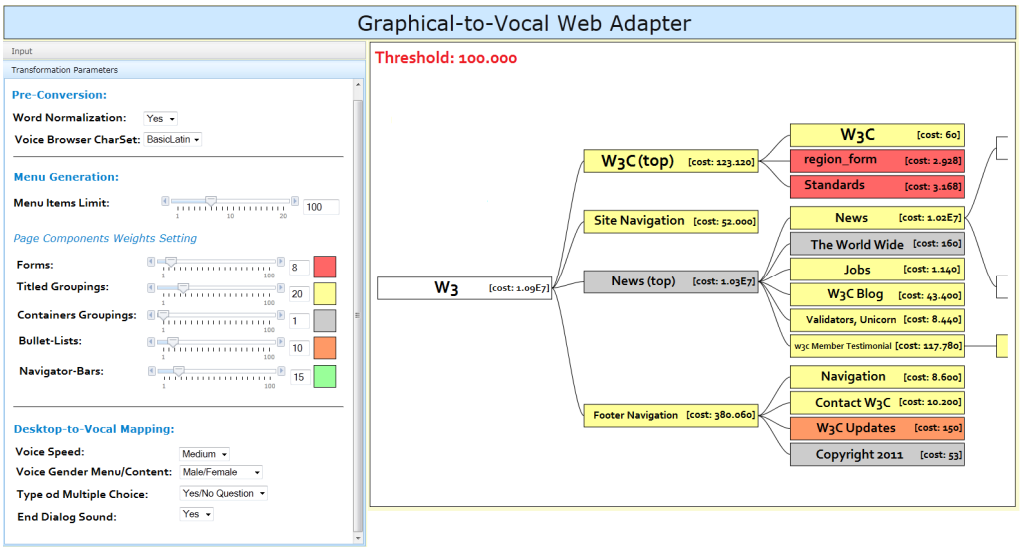

Figure 9: Customization of the Graphical-to-Vocal Web Adapter

The right panel shows the structure and the menu items of the generated vocal page. In this way the designer can decide whether to send the final vocal interface to the Voice Browser or, otherwise, change the transformation parameters in order to obtain a different structure.

7. EVALUATION

7.1 Evaluation Setting

We conducted a user test of our Customization Tool in order to collect feedback about the interface usability and the clarity of the results displayed. To evaluate the tool we recruited ten subjects (eight male and two female). The average user age was ~30.6 with a minimum of 23, a maximum of 42 and a standard deviation of ~6.20. Their education level was: two PhD, five Master Degree, one Bachelor Degree, and two High School.

The users' familiarity with the Web Interfaces was, in a range [1-5], on average ~4.4 with a minimum of 2, a maximum of 5 and a standard deviation of ~1.07. The users' familiarity with the vocal interfaces was, in a range [1-5], on average ~2.7 with a minimum of 1, a maximum of 4 and a standard deviation of ~1.42.

After a short introduction to the adaptation process we asked the users to focus on the first two labels of the *Pre-Conversion* sections (see Figure 9), which we defined as follows: *Word Normalization*, if true, normalize the words that contain accents (e.g. España→Espana); *Voice Browser CharSet*, to delete the words not contained in the selected charset. Then, we pointed out the definitions of the different page components (as described in Section 5.1) in relation to the W3C homepage and explained the weights' role in the menu generation process. At this point we asked the users to perform three adaptations of the W3C homepage with different weights:

- *All the weights at minimum (1);*
- *All the weights at maximum (100);*
- *User setting of free weights.*

As the next task we asked the users to perform two adaptations of the MSN homepage while fixing the *menu items limit* to 100 and 7. The last part of the test focused on the evaluation of the Desktop-to-Vocal customization parameters. We proposed the following definitions: *Voice Speed*, allows changing the speed of the synthesized voice in the final vocal interface; *Voice Gender Menu/Content*, permits selecting the voice gender for the menu and for the contents sections; *Type of Multiple Choice*, enables the user to choose between different kinds of multiple selection (Yes/No Question and Form Grammars); and *End Dialog Sound*, allows adding a sound at the end of each vocal dialog.

At the end of the test, users were asked to complete a questionnaire consisting of both multiple-choice and open questions. The questionnaire principally focused on the evaluation of the three main aspects of the tool: 1) the *pre-conversion*, 2) the *menu generation*, 3) *the desktop-to-vocal mapping*.

We asked for feedback about the clarity of the interface labels and about the usefulness of the associated customization. Moreover, we asked for feedback on the simplicity of the model of the cost-based structure adaptation and its visualization. Furthermore, we investigated the global perception of the proposed solution, asking the users to provide any kind of criticism/suggestion.

7.2 Evaluation Results

On a scale of 1 to 5, the user satisfaction for each of the feature discussed in Section 7.1 was: Pre-Conversion, Average: 4.13, Standard Deviation: 0.84; Menu Generation, Average: 4.26, Standard Deviation: 0.44; Desktop-to-Vocal Mapping, Average: 4.33, Standard Deviation: 0.34. The best result was obtained for *Desktop-to-Vocal Mapping* while *Pre-Conversion* and *Menu Generation*, despite good results, seem to need improvements. This is probably due to the fact that the first two sections of the customization interface are more technical and less intuitive for non-expert users. The overall evaluation of the Customization Tool was ~4.42 (scale [1-5]) with a standard deviation of ~0.54.

We gathered a number of positive user impressions and criticisms that gave us further motivation to investigate the field of adaptation for vocal interfaces. About 80% of the users underlined the lack of a zooming feature for a complete visualization of the tree; 50% asked for some helping mechanism, such as popups or tooltips; 70% found it quite hard to define appropriate weights in

a few attempts; and asked for a more immediate feedback from the environment.

For the considered Web sites, the time to perform the adaptation was on average ~40.55 seconds. We expect that this time can be influenced both by the original page dimension as well as by its intrinsic structure. In any case further work will be oriented to reducing processing time both by optimizing the transformations algorithm and adding caching features to the customization tool. Currently, every time that new weights are selected, the tool performs the entire process from the reverse engineering to the VoiceXML generation. The idea is to permit choosing the adaptation part to perform dynamically.

While all the users considered the weights range for the cost calculation suitable, more than half the users asked for a smaller range in the choice of the menu items. Instead of [1..20] they proposed [1..10]. 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 re-prompting and help hints when the user seems to stop during navigation.

8. CONCLUSIONS AND FUTURE WORK

Our automatic adapting process has been applied to twenty Web sites. The generated VoiceXML vocal interfaces have been tested with the VOXEO Voice Browser 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 specific 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 are interested to exploiting machine learning techniques in order to investigate the possibility to find the correct weights configuration and in the pattern analysis for the headings identification.

Acknowledgments

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

REFERENCES

[1] Bergenti F., Lazzari L., et al: Enabling Vocal Interaction in a Web Portal Environment. Lecture Notes in Business Information Processing, Volume 1, Part II, 204-213. Springer, (2007).

[2] British Broadcasting Corporation (BBC), http://www.bbc.co.uk/

[3] Buyukkokten O., Kaljuvee O., Molina H., Paepcke A. and Winograd T.: Efficient Web browsing on handheld devices using page and form summarization. ACM Trans. Inform. Syst. 20, 1, pp. 82-115, (2002).

[4] C. E. Cirilo, A. F. do Prado, W. L. de Souza, L. A. M. Zaina, Model Driven RichUbi - A Model Driven Process for Building Rich Interfaces of Context-Sensitive Ubiquitous Applications, Proceedings SIGDOC 2010, pp.207-214

[5] M. Ferati, S. Mannheimer, D. Bolchini, Acoustic Interaction Design through "Audemes": Experiences with the Blind, Proceedings SIGDOC 2009, pp.32-28

[6] Fonseca J.M.C. (ed.): W3C Model-Based UI XG Final Report, May 2010, available at http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui-20100504/.

[7] Franz. A., Milch B.: Searching the web by Voice. In proceeding of the 19th international conference on Computational Linguistic - Volume 2, pp. 1-5, Stroudsburg, PA, USA, (2002).

[8] Honkala M., Pohja M.: Multimodal interaction with XForms. Proceedings ICWE, pp. 201-208, (2006).

[9] Yesilada Y., Stevens R., Harper S. and Goble C.: Evaluating DANTE: Semantic transcoding for visually disabled users. ACM Transactions on Human-Computer Interaction 14, 3, Article 14, (2007).

[10] Leporini B., Paternò F.: Increasing usability when interacting through screen readers. Universal Access in the Information Society 3(1): pp. 57-70, (2004).

[11] 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).

[12] Paternò F., Sisti C.: Deriving Vocal Interfaces from Logical Descriptions in Multi-device Authoring Environments. Proceedings ICWE 2010, LNCS 6189, Springer Verlag, pp.204-217, Vienna, July 2010

[13] Spiliotopoulos D., Stavropoulou P., Kouroupetroglou G.: Acoustic Rendering of Data Tables Using Earcons and Prosody for Document Accessibility. HCI (7) pp. 587-596, (2009).

[14] Stanciulescu, A.: A Methodology for Developing Multimodal User Interfaces of Information Systems. Ph.D Thesis, University of Louvain, (2008).

[15] A. Talarico Neto and R. Pontin de Mattos Fortes, Improving multimodal interaction design with the MMWA authoring environment, Proceedings SIGDOC 2010, pp. 151-157

[16] Trewin S., Richards J., Bellamy R., John B. E., Thomas J., Swart C. and Brezin: Towards modeling auditory information seeking strategies on the Web. In Extended Abstracts of SIGCHI Conference on Human Factors in Computing Systems (CHI '10), ACM Press, NY, Extended Abstracts, pp. 3973-3978, (2010).

[17] Uwe D. R., Hartmut. R. P., Text preprocessing for speech synthesis. In Proceedings of the TC-STAR Workshop on Speech-to-Speech Translation, Barcelona, Spain, pp. 207–212, June (2006).

[18] Voxeo Voice Browser, http://www.voxeo.com/.