Speech-enabled Web Forms

Mikko Honkala, Mikko Pohja

Helsinki University of Technology
Telecommunications Software and Multimedia Laboratory
[mikko.honkala|mikko.pohja]@tml.hut.fi

Abstract

Using Web applications with multiple modalities, and especially speech is a difficult problem. For instance, the navigation differs from GUI based navigation, since in speech, the user interface has to be serialized. Current web technologies make applications usable only with one modality at the time (HTML vs. Voice XML).

This paper presents an idea of using XForms as the UI description language, and automatically creating UIs for different modalities based on a single XForms description. Three approaches of realizing speech UIs are described, and dialog-based approach is chosen to suit best the automatical creation of a speech UI.

There are several features in XForms, which make the task easier compared to HTML forms. XForms provides datatypes to the filled data, and it supports dynamic changes in the UI based on user input.

The paper also describes an implementation of this idea. The implementation is now part of the open-source X-Smiles XML browser. The implementation allows multimodal interaction on any XForms. It uses Sphinx-4 and FreeTTS as the ASR and TTS implementations, respectively.

A use case was developed to demonstrate the functionality of the implementation. The use case is a airline ticket reservation system. The use case demonstrates input and output of several datatypes, as well as navigation within a form.

1. Introduction

The diversity of devices accessing the Web as well as the applications running on the Web is increasing. Currently there is a lot of interest in delivering Web applications to different devices and usage scenarios regardless of the modality. For instance, Web applications could be used with voice while driving a car.

1.1. Research problem and goals

The main research problem is how to implement multimodal user interfaces in the Web context using a high-level user interface description language, such as XForms. The research was conducted in two phases. First, a literature study is done, in order to have a good understanding of the requirements. Second, an XForms implementation will be extended to include voice recognition features, in order to demonstrate the approach.

The main problems are related to navigation of a random form, which is not designed for speech interaction. Also, some types of input can be difficult to implement with voice. An example is a free-form text input.

The main result is an implementation that reads in any XForms compliant form, and allow filling and submitting it with voice input and output.

2. Background

2.1. Speech UI

As a modality, speech is quite different from graphical UI. The main difference is the navigation within the UI. In graphical model, the whole UI can be viewed at the same time, or at least the navigation between different parts is always possible. On the other hand, spoken command input allows intuitive shortcuts. It has been shown an efficiency increase of 20-40 per cent using speech systems compared with other interface technologies, such as keyboard input [1, 2]. Although speech interaction seems to be promising, the potential technical problems with speech interfaces may irritate the user and reduce task performance [3]. That is why speech is often used by combining it with other modalities to offset the weaknesses of them [4]. Especially, GUIs are often combined with speech modality [5].

Basically, there are three different approach to realize speech interface: command, dialog, and natural language based. Human-computer interaction through speech is discussed in [6]. The basis of the paper is that perfect performance of a speech recognizer is not possible, since the system is used e.g. in noisy environments with different speakers. Therefore, dialog and feedback skills are important to a recognizer because both human and computer may misunderstand each other. Conclusion is that a speech interface should provide adaptive feedback and imitate human conversation.

2.2. VoiceXML

Voice Extensible Markup Language (VoiceXML) Version 2.0 [7] is currently the best candidate for high-level speech UI definition language.VoiceXML is designed for creating audio dialogs that feature synthesized speech, digitized audio, recognition of spoken and DTMF key input, recording of spoken input, telephony, and mixed initiative conversations. Its major goal is to bring the advantages of Web-based development and content delivery to interactive voice response applications.

An VoiceXML **application** is a set of documents, which share the same root document. The root document is automatically loaded when one of the application documents are loaded, and it contains information, which is available to all documents in the applications (e.g. main navigation, etc.). The application contains a set of **dialog states**, from which only one is active at a time. Each dialog specifies how to transition to the next state. **Menu** is a simple dialog, which allows the user to select from different transitions to other dialog states. **Form** is a more complicated dialog, which allows the user to fill in different values and to submit the filled values to a server-side process.

The biggest problem with VoiceXML's usage in the Web is that all user interfaces have to be programmed twice; once for the Graphical browser using HTML and once for the voice browser, using VoiceXML.

2.3. XForms

XForms 1.0 Recommendation [8] is the next-generation Web forms language, designed by the W3C. It solves some of the problems found in the HTML forms by separating the purpose from the presentation and using declarative markup to describe the most common operations in form-based applications. It can use any XML grammar to describe the content of the form (the instance data). Thus, it is also possible to create generic editors for different XML grammars with XForms. It is possible to create complex forms with XForms using declarative markup, thus not resorting to scripting.

XForms is an abstract user interface description language. One of it's design goals was not to mandate a certain modality. Therefore it can be suited to describe user interfaces which are realized in different modalities, such as the GUI and Speech.

3. Requirements

The objective of the work was to integrate speech recognizer and synthesizer into X-Smiles. That enables dialog based interaction between user and the browser. The interaction includes navigation in a document and filling of a form. User must be able to focus to a form element by speech. The forms are usually hierarchical so the navigation has to be possible both to siblings and to parent and children. Since current recognizers are grammar based, the implemented form controls must have limited set of selections. That include selection lists, datatype specific controls like dates and numbers, and triggers (e.g., submit).

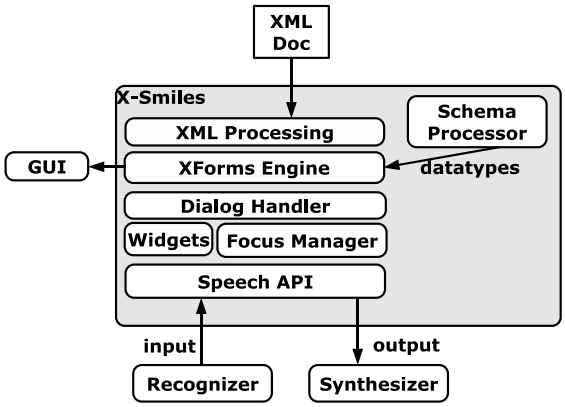

Figure 1: The components of the implementation.

4. Implementation

The research problem was addressed by taking a free XForms implementation and designing and implementing speech interface into it. The open source XML Browser X-Smiles was used as the XForms implementation. Sphinx-4 was used as the speech recognition engine and FreeTTS as a speech synthesis engine. All the projects are written in pure Java, so they were rather straight-forward to integrate. All the components are shown in Figure 1.

An XML document is parsed by the XML parser, which creates the DOM presentation of a document. The DOM document is given to the XForms engine. Schema processor provides datatypes for XForms engine. Speech widgets are created for each element according to the datatypes. Widgets store the inputted data to a form in correct format. To integrate speech components into X-Smiles, we defined a Speech API. The recognizer and synthesizer are used through the API. The dialog handler is build on top of XForms Engine. The handler uses speech widgets and focus manager to provide a speech UI.

4.1. Design

The speech implementation works on top of the XForms DOM. The implementation consists of a focus manager and speech widgets. The duties of the focus manager are to search the focus points of a document and hold the current focus point. The focus point is an XForms element. Elements, which does not have a label like HTML blocks, cannot be focus points. The focus points of a sample document are depicted in Figure 2.

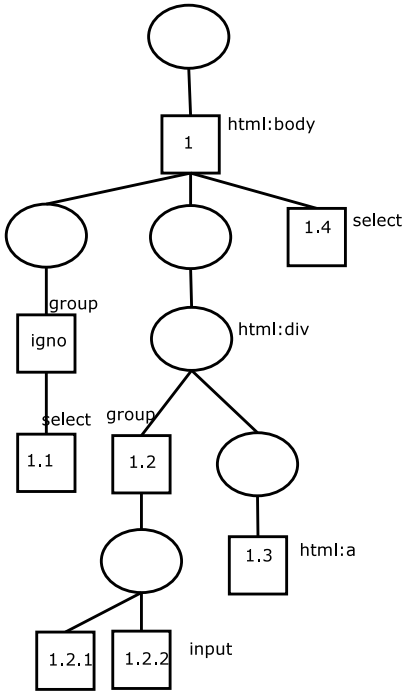

Figure 2: Focus points of a document.

The focus manager provides all the possible focus points from current focus point. The possible points are successors of the current point. In the example (cf. Figure 2), the possible focus points from point *1* are *1.1*, *1.2*, *1.3*, and *1.4*. Note that the items with only one choice are ignored (e.g., ancestor of *1.1*).

The speech widgets are the interface between XForms DOM and speech recognizer and synthesizer tools. Their responsibilities include generation of a dialog when an element is focused. Widgets also generate grammars for speech recognizer according to the possible selections. JSGF grammar format was used, since that is what Sphinx-4, and Java-based tools in general, support. User's answer is returned to a widget as a Java String, and the widget parses it and interprets to a corresponding element.

Every time focus changes, the dialog handler must create a new grammar for the recognizer. Creation of a grammar depends on a form control. Certain types of control have a constant grammar (e.g., date and numbers). In that case, the grammar is fetched from a file. If possible selections are defined in a form, the grammar must be formed according to those. In every cases, navigational commands are added to the grammar in the end (e.g., back).

4.2. Operation

The speech implementation operates as follows. When user focuses an element in a document (root element in the beginning), the dialog handler creates a question or command list and delivers it to the speech synthesizer. The question consists of label of an element and possibly the selections that can be made. The selections can be for instance navigable focus points or items of a selection input. Dialog hander creates grammar through a widget for recognizer according to possible replies.

The dialog handler receives the reply from the recognizer and delivers it to a corresponding widget. Widget parses the respond and stores it to the form in correct format. Finally, widget moves the focus to back to its ancestor.

5. Use case

We made a simple airline ticket reservation system to demonstrate the implementation. The graphical user interface of the system is shown in Figure 3. There are different input types in the document. Departure and destination can be selected from a prefined list, date is a calendar type input, and number of passengers must be inputted as integers. The adults and children integer input elements are within group element. Finally, there are two triggers for submit and reset the document. A possible ticket reservation dialog between system and user is represented below.

System: SmilesAir ticket reservation. ”Departure”, ”Destination”, ”Date of travel”, ”Number of passengers”, ”Rate”, ”Submit”, ”Reset”
User: departure
System: Departure . Select from: ”Helsinki”, ”Oslo”, ”Stockholm”, ”London”, ”Paris”, ”Copenhagen”
User: oslo
System: You selected: Oslo
SmilesAir ticket reservation. ”Departure”, ”Destination”, ”Date of travel”, ”Number of passengers”, ”Rate”, ”Submit”, ”Reset”
User: date of travel
System: Date of travel November 29, 2004. Select a date.
User: seven december two thousand and four
System: Date of travel : seven december two thousand and four
SmilesAir ticket reservation. ”Departure”, ”Destination”, ”Date of travel”, ”Number of passengers”, ”Rate”, ”Submit”, ”Reset”
User: number of passengers
System: Number of passengers. ”Adults”, ”Children”
User: adults
System: Adults 2. Select a number.
User: four
System: Adults: four
Number of passengers. ”Adults ”, ”Children”
User: back
System: SmilesAir ticket reservation. ”Departure”, ”Destination”, ”Date of travel”, ”Number of passengers”, ”Rate”, ”Submit”, ”Reset”

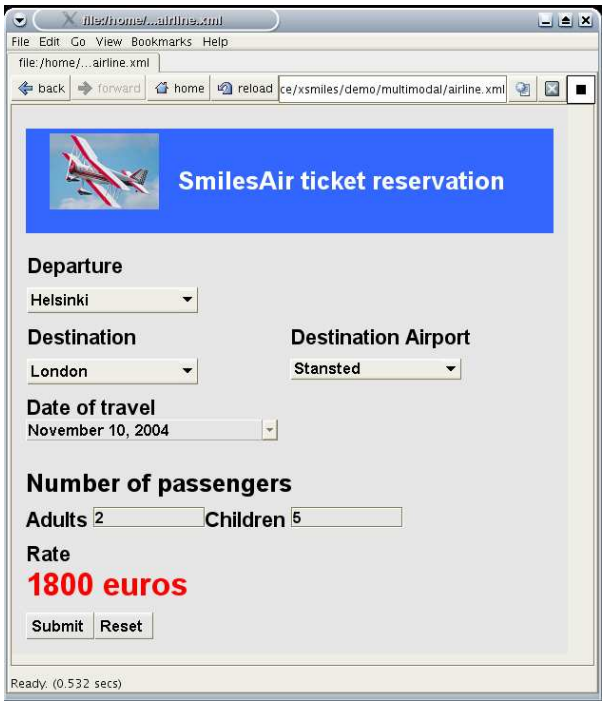

Figure 3: An airline ticket reservation system.

User: submit
System: Are you sure you want to :Submit ?
User: yes
System: Doing: Submit

6. Conclusions

Using Web applications with multiple modalities, and especially speech is a difficult problem. Current web technologies make applications usable only with one modality at the time (HTML vs. Voice XML). This paper presents an idea of using XForms as the UI description language, and automatically creating UIs for different modalities based on a single XForms description.

There are several features in XForms, which make the task easier compared to HTML forms. First, XForms provides datatypes to the filled data, which makes it possible to create efficient speech grammars for input and output. Second, XForms supports dynamic changes in the UI based on user input. This means that the same instance of the form can stay at the client longer, thus retaining the speech focus point. In HTML forms, the form has to be recreated by the server, thus losing any speech focus points.

We created an implementation of this idea. The implementation is now part of the open-source X-Smiles XML browser. The implementation allows multimodal interaction on any XForms. It uses Sphinx-4 and FreeTTS as the ASR and TTS implementations, respectively.

A use case was developed, to demonstrate the functionality of the implementation. The use case is a airline ticket reservation system. The use case demonstrates input and output of several datatypes, as well as navigation within a form.

6.1. About the tools

The prototype shows that pure Java tools are feasible to use in speech UI, when using modern desktop computers (e.g. over 1GHz Pentiums with 256MB or more memory). We suspect that these tools are not optimized enough to be run on smaller mobile systems, such as smartphones.

Sphinx-4 is under constant development, and the developers are keen to respond to outside requests. For instance, they implemented switchable grammar partly because of our request.

Java platform is also going through development. It was not possible to get Sphinx-4 and FreeTTS to coexist in JDK 1.4.2, because of bugs in javax.sound, but upgrading to the latest 1.5.0 helped.

We have also found X-Smiles to be a good platform to experiment with different XML-based markup languages and UIs of different modalities.

7. Future work

Navigating large forms using the speech focus, presented above, can be tedious. More work is required to study the possibility of creating shortcuts in the navigation sequence. Also, some author control of speech input and output is needed. Using sXBL [13] to bind in VoiceXML constructs and using CSS Voice [10] and EMMA [11], taking into account the Multimodal Interaction Framework [12] might help.

The implementation, as it stands, could be improved and extended. More widget types, such as range, and textarea should be added. Better datatype support (i.e., time, duration, float, etc.) could be added as well. Also, currently the implementation handles only XForms, and support for XHTML could be added.

We have successfully used open-source tools to do ASR and TTS, but some work is required there as well. For instance, Sphinx-4 does not support unknown word recognition. For instance, the dictionary we used, did not have pronunciation rules for ”Stansted”, so it could not be recognized at all from user input. There are methods to support unknown word recognition [14]. That paper uses the pronunciation dictionary to find out probabilities between letters and phonemes, and then constructs the most probable pronunciation for that word with accuracy of 80% (English), and 95% (Germany). For limited grammar (e.g. ”Stansted—Heathrow”), that accuracy should be enough. This method is supported by FreeTTS, and the same method should be implemented in Sphinx-4.

8. References

[1] Martin, G. L., ”The utility of speech input in user-computer interfaces,” International Journal of Man-Machine Studies, Vol. 30, pp. 355-375, Academic Press Ltd, 1989.

[2] Visick, D. et al, ”The use of simple speech recognizers in industrial applications,” in the Proceedings of INTERACT’84, London, UK.

[3] Shneiderman, B., Designing the user interface: strategies for effective human-computer interaction, 2nd edition. Addison-Wesley, 1992.

[4] Cohen. P. R., ”The role of natural language in a multimodal interface,” in the Proceedings of the 5th annual ACM symposium on User interface software and technology, Monteray, California, United States, pp. 143-149, ACM Press, 1992.

[5] Mane, A. et al, ”Designing the user interface for speech recognition applications,” ACM SIGCHI Bulletin, Vol. 28, pp. 29-34, ACM Press, 1996.

[6] Brennan, S. E. and Hulteen, E. A., Interaction and feedback in a spoken language system: a theoretical framework, Knowledge-Based Systems, vol. 8, no. 2, pp. 143-151, 1995.

[7] McGlashan, S., et.al. Voice Extensible Markup Language (VoiceXML) Version 2.0, W3C Recommendation 16 March 2004.

[8] Dubinko, M., XForms 1.0, W3C Recommendation 14 October 2003.

[9] Daniel C. Burnett, et.al. (eds.), Speech Synthesis Markup Language (SSML) Version 1.0, W3C Recommendation 7 September 2004.

[10] Raggett, D., D. Glazman (eds.), CSS3 Speech Module, W3C Working Draft 27 July 2004.

[11] Chou, W., et.al., EMMA: Extensible MultiModal Annotation markup language W3C Working Draft, 1 September 2004.

[12] Larson, J. A., et.al. (eds), W3C Multimodal Interaction Framework, W3C NOTE 06 May 2003.

[13] Ferraiolo, J., et.al. (eds.), SVG’s XML Binding Language (sXBL), W3C Working Draft 22 November 2004.

[14] Black, A., Lenzo, K., and Pagel, V.,Issues in building general letter to sound rules. In Proceedings of the 3rd ESCA/COCSADA Workshop on Speech Synthesis, 1998. pages 77–81, Jenolan Caves, Australia.