MULTIMODAL BROWSING

Giuseppe Caccia, Angelo Cicchitto, Rosa Lancini

CEFRIEL-Politecnico di Milano Via Fucini,2 20133 Milan – ITALY
Phone: +39.02.23954.209 - mailto: rosa@cefriel.it

ABSTRACT

With the increasing development of devices such as personal computers, WAP enabled wireless telephones and personal digital assistants connected to the World Wide Web, end users feel the need to browse the Internet through multiple modalities. We intend to investigate on how to create a user interface and a service distribution platform granting the user access to the Internet through standard I/O modalities and voice simultaneously.

1. ACCESS TO THE WORLD WIDE INTRODUCTION

Web is mainly achieved through personal computers, such as desktops and notebooks. A PC enables the user to navigate the Internet through visual browsers, which present output information on a monitor and receive input commands from keyboard or mouse. It is now possible to search for information on Internet through wireless telephones devices that support **W**ireless **A**pplication **P**rotocol (WAP) browsers, equipped with a small display. On such devices output is presented to the user on the small display, and input commands are given by pressing the dial tone keypad.
There are situations in which it would be easier to access an Internet browser through voice commands and receive an audio description of the browsed page content [6, 7] keeping enabled the standard I/O modalities (display and keyboard/keypad). This leads us to the need of a device dependent browser supporting multiple I/O modalities, now on referred to as **Multimodal Voice Browser**. This approach to Internet browsing seems useful in the following situations.

- Users with physical disabilities might find it much easier to access a browser with voice commands and be guided by an audio assistant.
- Using a small keypad on a cellular phone might be very frustrating. It seems easier to give voice commands to a WAP browser rather than pushing long sequences of buttons on a keypad. Also users not able to freely use their hands, perhaps drivers, would find it much more comfortable to keep their hands busy with driving and navigate the browser through voice.
- Even on a personal computer equipped with headphones and microphone a multimodal browser would have some great features. Think of an e-commerce site giving an audio assistant to the customer, or the possibility to skip frequent operations like checking for new mail, while it would be easier to give an appropriate voice command.

2. MULTIMODAL VOICE BROWSING

Multimodal access has been formally defined by W3C's Voice Browsing Working Group [1] as a combination of one or more speech modes (speech recognition, speech synthesis, prerecorded speech) with one or more standard I/O modes (dtmf, keyboard, small screen pointing device, other). It is clear that the type of deployable application is strongly device dependent.
Apart from the system architecture every speech application must be equipped with a **T**ext **T**o **S**peech (TTS) synthesizer and an **A**utomatic **S**peech **R**ecognizer (ASR) in order to implement voice interaction with the user (Figure 1).
A TTS synthesizer is a software package that accepts text strings as input and outputs the correspondent vocal message. An ASR is responsible for the inverse procedure accomplished by the TTS. It recognizes messages spoken by the user matching the extracted sound with a series of phonems from a defined grammar.

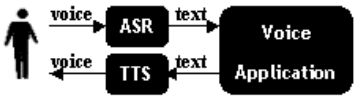

Figure 1. ASR and TTS interface.

3. SYSTEM ARCHITECTURES

A multimodal platform is obviously a distributed architecture, delivering the information stored on a server to a proper client. We must then take in consideration the following topics:

- **M**arkup **L**anguage for the data and presentation logic;
- Client-Server architecture;
- TTS and ASR implementation.

3.1. Markup Language

There are several possible approaches for data and presentation logic.
A first possible approach might be to create a specific markup language [8,5], perhaps an XML based language, to handle both

the data logic and the presentation logic, supporting multimodal I/O. This is the direction encouraged by W3C. This solution seems very robust but is hard to implement. In fact, towards this architecture, every HTML document should be rewritten in the new **M**arkup **L**anguage, or at least be dynamically generated by a proper gateway. Further, such a solution needs a specific browser in order to properly parse the multimodal document.
Another possible solution is to associate a VoiceXML [3] document to the HTML document to be rendered as multimodal. The corresponding VoiceXML document may be specifically created by an author or dynamically created by a proper gateway, which first analyzes the HTML content [4].

3.2. Client-Server

The client architecture is strongly device dependent. In the case of multimodal browsing through PC the client will certainly hold the multimodal browser. According to the chosen markup language the browser may vary:

- if we choose to develop a brand new language we must also develop a corresponding browser supporting both visual and aural I/O modalities (Figure 2);
- if we choose to associate a VoiceXML document (.vxml) to each HTML document, we may use the current Internet browsers (Explorer©, Navigator©, Opera©, ...) to access the .html documents and implement a specific browser to access the .vxml documents (perhaps via Java™ applet) (Figure 3).

Figure 2. Client-server architecture with a Multimodal Markup Language.

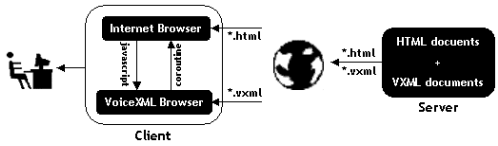

Figure 3. Client-server architecture with HTML and VXML.

In the case of multimodal browsing through a WAP phone, it seems almost impossible to implement a new browser on the device. It is much more reasonable to use the WAP browser for standard I/O modality, and open a vocal channel with the server for voice interaction.
The server is responsible of holding the documents being accessed by the client. Depending on the system architecture it is also responsible of the business logic, properly behaving to particular events. Perhaps it may store collected data from forms or contact other remote servers. As for WAP phones interaction, the server may also push information (WML documents or voice) towards the end user device.

3.3. TTS & ASR implementation

In a client-server architecture there are two possible implementation choices for TTS and ASR:

- a first choice is to leave TTS and ASR on the server in order to deploy a strongly device independent service;
- a second choice is to embed TTS and ASR directly inside the browser on the client side.

3.4. Proposed architectures

Both the architectures explained in section 3.3 are implemented and compared. As we discussed in section 3.2, if we chose to develop a new markup language for multimodal browsing, we need also an appropriate multimodal browser that is not compatible with WAP technologies. In order to compare both the WAP and the PC solutions, the multimodal markup language is not feasible. In this section we give a brief description of our implementation strategies.
Concerning the implementation with TTS and ASR on the client (see Figure 4) we used a Java Applet embedded directly inside the html page. Using Javascript we allow communication between particular events like mouse or keyboard interaction with the VoiceBrowser, TTS and ASR. The voice events is caught from the applet and, in keeping with VoiceBrowser policy, an event (like following an hyperlink or filling a form) is sent to html page.

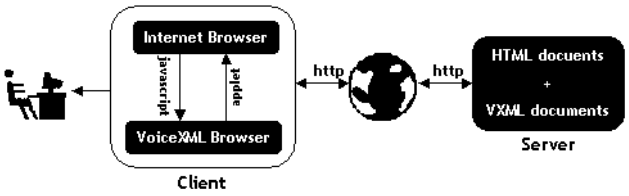

Figure 4. Client side implementation.

In the second architecture (server side implementation) the Voice browser is on the client and voice logic (TTS and ASR) are on the Server (Figure 5).

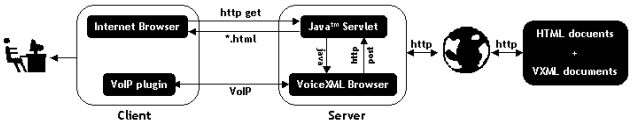

Figure 5. Server side Implementation.

The interactions between the requested html page and the associated voice events are implemented though a Java servlet (see Figure 6).

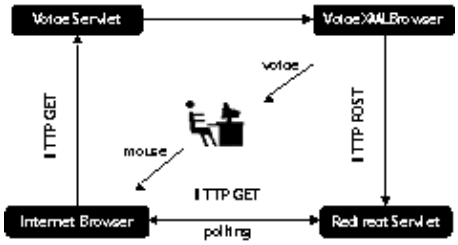

Figure 6. Servlet architecture.

4. RESULTS

The research activity focused on the comparison between Multimodal Voice Browsing on PC and on WAP. The proposed architectures associate VoiceXML with standard languages in order to take advantage of the wide diffusion of both HTML and WML.

4.1. Architecture Validation (requirements and services)

Our research was focused on evaluating of the performance of the following architectures:

- Client-side implementation of TTS and ASR;
- Server-side implementation of TTS and ASR.

The proposed architecture are implemented using Java technology [9, 10] ensuring system portability. The VoiceXML browser will be an enhanced version of the standard VoiceXML browser developed at CEFRIEL research laboratories. The adopted TTS is Actor© developed by ELOQUENDO laboratories and the adopted ASR was Spinet© developed by IRST-ITC laboratories.

4.1.1. Client side implementation

The client side architecture needs both TTS and ASR installation on the client. With the present technologies, it is not possible install a robust and complete version of a TTS and ASR software on a WAP devices. Moreover the Java applet that allow voice and html interaction is quite big and needs a long time for downloading though the Internet. A relevant issue regards the communication between client and server. In this implementation only a standard TCP/IP channel has to be opened to allow the downloading of the requested html pages. Voice interaction is performed between the user and the client (see Figure 4). The advantage of such a solution is performance. Multiple clients accessing the server do not overload it with heavy computations (necessary especially for the recognizer), and only a single communication channel between client and server is necessary.

4.1.2. Server side implementation

The server implementation does not need heavy software installation on the client. ASR and TTS are on the server (an application server is needed instead of a standard web server) so a new channel that allows voice communication between client and server was needed. In our testing implementation both a standard voice channel and a TCP/IP connection are opened simultaneously. This choice is due to packets losing during the transferring of voice information that could make the recognition process critical. WAP devices do not allow the opening of two channels, so our tests are performed over GPRS testing platform. As for PC clients, we may consider using a **V**oice **o**ver **IP** architecture (see Figure 5) or a synchronized POTS telephone channel.

4.1.3. Architecture comparison

According to the application environment (PC or WAP browsing though the Internet) the more feasible solution seems to be the server side logic architecture. Respect the fact that we have a growing of computational load of the server for an high number of connections, we can have a device independent architecture that allows a quick access at the requested information without a long waiting for applet downloading.

5. CONCLUSIONS

This article intended to be an overview of the possible architectures and technologies capable of delivering multimodal services on different devices. Two architectures have been proposed in order to cover the possible approaches and to outline every characteristic. Both solutions were being implemented, tested and validated. The results were be compared in terms of flexibility, performance and quality of service.

6. REFERENCES

[1] W3C "Voice Browser" Activity, http://www.w3.org/Voice/.

[2] Multimodal Requirements, http://www.w3.org/TR/multimodal-reqs.

[3] VoiceXML Forum, http://www.voicexml.org.

[4] S Goose, M Newman, C Schmidt, L Hue, Enhancing Web Accessibility Via the Vox Portal and a Web Hosted Dynamic HTML-VoxML Converter, http://www.www9.org/w9cdrom/354/354.html.

[5] S Rollins and N Sundaresan, AVoN Calling: AXL for Voice-enabled web Navigation, http://www.www9.org/w9cdrom/55/55.html.

[6] S McGlashan, Position paper - Standards for voice browsing, http://www.w3.org/Voice/1998/Workshop/ScottMcGlashan.html.

[7] R Agarwal, Y Muthusamy, V Viswanathan, Voice Browsing the Web for Information Access, http://www.w3.org/Voice/1998/Workshop/RajeevAgarwal.html.

[8] M Ishizuka, T Tsutsui, S Saeyor, A multimodal presentation markup language with character agent control

functions, http://www.miv.t.u-tokyo.ac.jp/MPML/en/.
[9] Java Speech API Programmer's Guide, http://java.sun.com/products/java-media/speech/index.html.
[10] Cascading Style Sheets, level 2 CSS2 Specification, http://www.w3.org/TR/REC-CSS2/.