

AN ARCHITECTURE FOR A MULTI-MODAL WEB BROWSER

Cristiana Armaroli², Ivano Azzini¹, Lorenza Ferrario², Toni Giorgino¹,
Luca Nardelli², Marco Orlandi², Carla Rognoni¹

¹ Dipartimento di Informatica e Sistemistica, Università di Pavia, Pavia, Italy.

² ITC-irst, Pantè di Povo, Trento, Italy.

{ivano, toni, carla}@aim.unipv.it, {armaroli, ferrario, lunarde, orlandi}@itc.it

1. INTRODUCTION

The very rapid evolution of telecommunication technology is leading to the convergence of fixed and mobile networks and devices. At present, it is very widespread for people to access the Web with Internet connections using HTML and/or WML (*Wireless Markup Language*) browsers, and present portable devices (e.g. PC/PDA, GPRS/WAP phones) offer a range of features (e.g. large memories, graphical displays, friendly user interfaces, communication interfaces, including the possibility to install Internet browsers) that make them suitable for hosting quite all of the applications that can be normally performed by standard PCs. Their main limitation is related to the reduced input/output capabilities, since they frequently lack of an alphanumeric keyboard and have very small displays. In this case, the development of multi-modal browser with voice input/output capabilities and/or making use of other devices (e.g. graphic pointing, touch screens, small numeric keyboards, etc.) should satisfy a large variety of user requirements.

The idea we propose consists in the definition (and consequent realization) of an architecture capable of handling multi-modal browsing through the synchronization of HTML and VoiceXML documents. In doing this, we have to consider issues related to the variability of user/terminal profiles, as well as issues related to the layout adaptation to different presentation modalities (e.g. spatial/temporal axes and *hyperlinking*). VoiceXML enables users to browse documents by speaking and hearing on a phone, but does not support a graphic interface, as HTML or WML do. We propose to synchronize different documents through a specific platform instead of adding new features to existing HTML, WML or VoiceXML documents. This approach has the advantage of allowing, in a quite general way, multi-modal browsing of existing HTML documents by developing corresponding VoiceXML documents. In any case, we do not exclude the possibility of defining an XML schema that will include, on a general basis, both HTML and VoiceXML syntax, thus allowing a multi-modal definition of an application in a single document. What we want to point out is that the system we are proposing can be used in a quite general way, provided that the Markup documents describing the web service to realize are correctly interpreted by specific components.

The work presented in this paper has been partially developed inside the E.U. project *Homey* [2, 3]. The purpose of this project is to monitor the clinical state of chronic patients (in

particular patients affected by hypertension pathologies), by means of the telephone, as will be described in section 3. Another application under investigation is to use the multi-modal browser for accessing the WebFabIS information system (a system for the data management described in section 3). The benefits resulting from the adoption of mobile devices are currently being investigated.

2. SYSTEM DESCRIPTION

In ITC-irst, recently, a client-server architecture (called *SPINET - Speech INTO Enriched Text*) for developing voice applications has been realized. The SPINET Server can handle multiple simultaneous connections with applications that need speech recognition resources. Furthermore, a VoiceXML interpreter has been realized in the Java programming language, and application programmers can use it by means of a Java API (Application Programming Interface).

The architecture of the multi-modal browser we are proposing is shown in Figure 1. A client uses a traditional Web browser (e.g. Netscape or Internet Explorer) to interpret HTML documents. The client should also be able to acquire and transmit the voice signal to the server (voice channel). Note that the speech signal can be acquired/transmitted either through PSTN (Public Switch Telephone Network) or through TCP/IP connections; this implies the presence of telephone or decoding capabilities on the server side.

The server side hosts:

- a conventional Web server, handling requests for HTML documents and the corresponding VoiceXML counterparts. The Web server manages all the resources needed for ASR and TTS functions (grammars, speech files, etc);
- the Speech Server, which manages both ASR and TTS resources;
- the multi-modal browser process, which integrates the VoiceXML Interpreter.

The multi-modal browser consists of 3 components:

- a Voice Gate that manages the voice channel;
- a Visual Gate that manages a TCP/IP connection with the Web browser;
- an Interpreter Manager that corresponds to the VoiceXML Interpreter Context and it integrates the VoiceXML Interpreter.

The two Gates (voice and visual) interact with the VoiceXML Interpreter through the Interpreter Manager.

Our solution for synchronizing HTML and VoiceXML documents (see fig. 1) consists in opening and maintaining a TCP/IP connection between the Web browser, on the client side, and the multi-modal browser on the server side. Information is exchanged along this connection in order to:

- modify incoming HTML pages according to the input provided by the user's utterances;
- send commands to the multi-modal browser to update the VoiceXML context according to "non voice" input provided by the user via keyboard and/or mouse.

For example, when the user utters one or more values for filling fields in a displayed form, the system modifies the loaded Document Object Model (DOM, it allows to access the structure of HTML documents) pages in order to show the uttered data. On the other hand, if the user types in the value of a field, the Web browser communicates the event to the multi-modal browser on the server side through the Visual Gate, which in turn updates the corresponding VoiceXML field item variable. This process requires that each HTML page received by the client includes an appropriate JavaScript file and a Java applet.

The JavaScript functions allow both to update DOM objects on demand (e.g. the *value* of an input field), and to dispatch the events occurring when users interact with a Web page (e.g. through a button press or a text field modification). The Java applet manages the TCP/IP connection between the client and the server, by transmitting and receiving commands according to a specific communications protocol. Thanks to the Live Connect technology [4], the applet and the JavaScript functions can interact with each other.

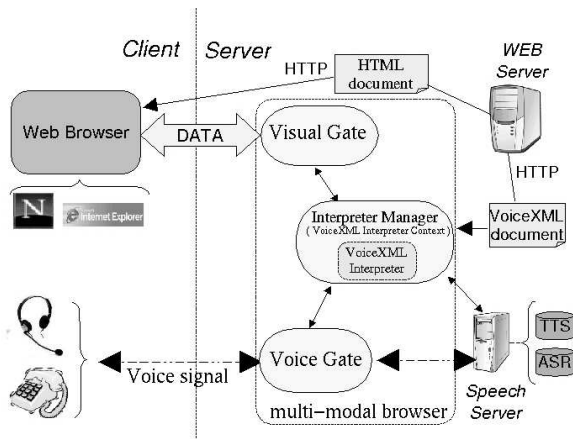


Figure 1: Architecture of the proposed multi-modal browser.

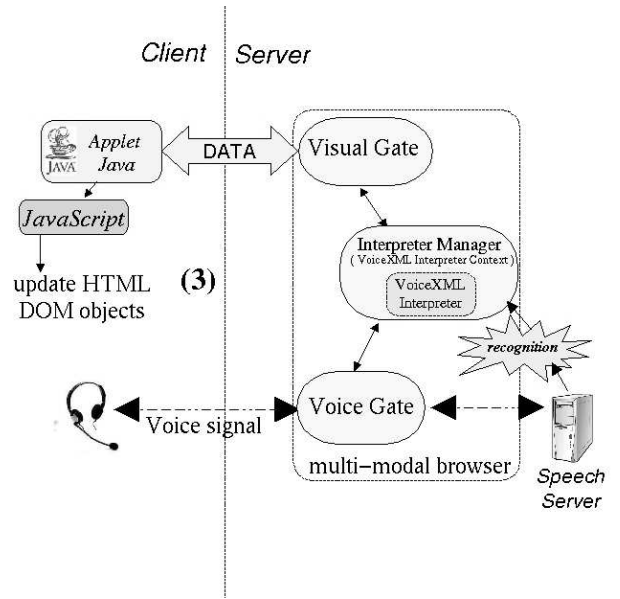


Figure 2: Voice interaction.

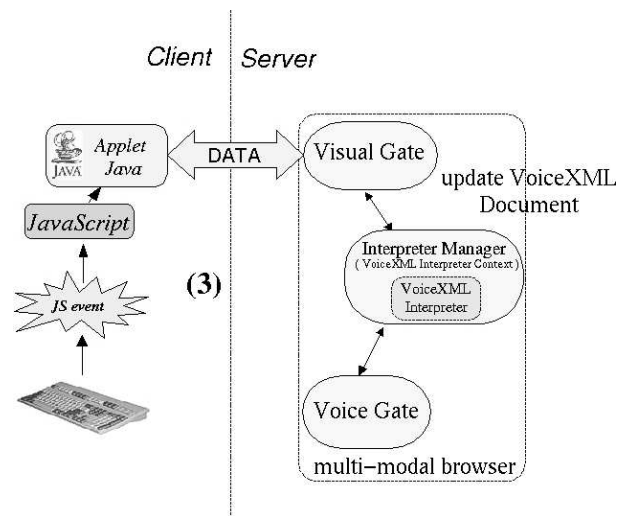


Figure 3: Non-voice interaction.

The user actions with a Web page are notified to the multi-modal browser through JavaScript function calling. This function uses the applet methods to require to the server to update a VoiceXML document (see fig.3). The Visual Gate analyses the request and indicates to the Interpreter Manager to execute the necessary operations. A correct voice interaction generates a recognition event, which the VoiceXML Interpreter uses in order to modify the context of the current VoiceXML document. The multi-modal browser, through the Interpreter Manager and the Visual Gate, uses this event to update the corresponding HTML document through the TCP/IP connection opened with the applet. Finally, the applet calls the JavaScript

functions that modify the Web page (see fig.2).

Note that, there is a problem on Markup Language syntax to point out: it is not always possible to automatically generate a visual component from the verbal component (e.g. voice prompts, grammars) and vice-versa. We are currently investigating the possibility to include VoiceXML and HTML pages in a single document (a possible solution is to use the XSL markup language proposed in [7]).

Furthermore, in this project we intend to investigate the possibility of distributing the resources in different ways, particularly on the client side (e.g. the ASR/TTS), thus avoiding to transmit the speech signal to the Web server. Our final goal is to use our multi-modal browser on portable device (e.g. PDA and UMTS terminal)

3. APPLICATION SCENARIOS

3.1 The Homey Project

The Homey project is currently investigating the use of voice recognition technology in the context of hypertension care.

Recent guidelines [1] suggest that patients affected by high blood pressure should have their blood pressure values monitored relatively frequently (e.g. once a week). The Homey collaboration has developed a database, which keeps a record of a wide range of clinical data associated to patients under treatment for hypertension. The database stores patient anagraphic records, detailed cardiovascular and familiar anamnesis, and habits. The physician also records the prescribed clinical exams and the respective outcomes, therapies assigned, side effects possibly observed, besides, of course, blood pressure and heart rate values. Most of this data is used to update three standard risk indicators (e.g. Framingham); the history of these indicators can be monitored in time in order to check therapy effectiveness. The database consists of approximately 30 tables, not counting those used for decoding.

While most of the records of the database will be in control of the physician, the database also has an interface towards a telephone gateway (see figure 4). Patients will be able to dial a toll free number from home and enter some relevant data by themselves, such as blood pressure values and side effects of his/her therapy possibly observed. The data-entry happens via a mixed initiative dialogue, which takes place through TTS and ASR technology; the latter is provided by a SPINET server.

We are currently evaluating the possibility of providing to physicians the database voice access as well as to patients. The idea is that the most frequently used subset of the database could have a multi-modal interface, so that health care personnel could browse and fill it via both hands-on as well as hands-free operation, without the need of explicitly switching the input modality. To achieve this goal, according to what has been discussed above, we will create a version of the database GUI and a description of the desired vocal interaction, respectively in the HTML and VoiceXML languages. These two documents will be generated dynamically according to the appropriate alignment rules so that the input fields and the field variables can be kept synchronized by the multi-modal browser. The dynamic generation of these two documents shall happen on-demand as they will be requested to the Web server during

the user interactions. The generation of both files may be handled by established server-side technologies like CGI scripts, JSP and Java Beans.

This multi-modal interface will be first developed and tested on a desktop environment. The ultimate goal of multi-modality in our domain is to make the human-machine interaction more comfortable; we shall therefore study the impact of issues like the avoidance of repetitive or most error-prone inputs, when they could be avoided. We are also evaluating the opportunity of giving PDA devices to health care personnel. Such personnel could benefit a much reduced version of the database input forms, so that they could be displayed properly on wireless networked appliances. Multi-modal browsing could apply to these devices, as well, as long as their Web browser has appropriate support to Java and JavaScript, and the network architecture provides an appropriate sustained bandwidth towards the speech recogniser, or sufficient local sound encoding resources (e.g. real-time GSM compression).

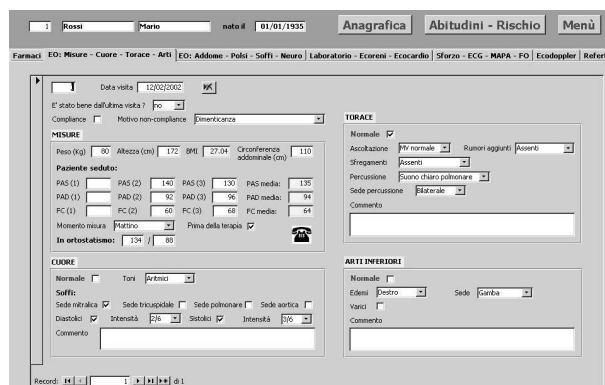


Figure 4: A part of the graphical user interface of the Homey database, and its reduced version suited for PDA displays. The small telephone icon indicates that the data entry was performed by the patient himself, dialling into the automatic dialogue system. The most frequently used input fields will be voice-enabled by the multi-modal technology explained in the article.

3.2 The WebFabIS Project

WebFabIS is an information system for clean-room microelectronics laboratories. It is aimed to handle technical and organizational information: device recipes, process control, yield indicator, safety control, and resources optimisation. It is used by the clean-room staff in order to have device fabrication instructions and from the research team in order to store and maintain new device design and recipes. It has been developed for R&D labs, but can be easily adapted to production organization, with no structural modifications.

In the microelectronics laboratory where WebFabIS is employed could happen that the operators cannot interact with the system through the keyboard. In this environment the operators wear uncomfortable cloths and gloves and have very often their hands busy with fragile silicon wafers. For this reason the possibility to interact with the system with the voice is much more than a "funny gadget". The architecture adopted for WebFabIS allows an easy integration with the multi-modal browser. In fact both systems are modular, Web based and realized with Java (WebFabIS makes extensively use of HTML and of the Servlet technique). Note that for each HTML traditional page, the corresponding VoiceXML has to be generated.

WebFabIS has been designed with the most advanced software engineering techniques, e.g. UML and the Object-Oriented paradigm, and can of course be considered a Web Application, since it can be accessed through Web browsers with no need of local applications (see Fig. 3). Strong attention has been paid to the modularity and the flexibility of the system. It results highly dynamic thanks to the use of the Java Servlets. Thanks to them, indeed, the same servlet can be used to interface the user to all the types of equipment and process steps.

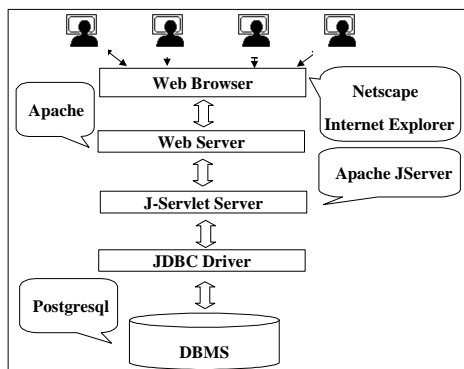


Figure 3: General overview of the WebFabIS architecture

The tools used to build the first release of the system are:

- a relational database: where all the information of the system are stored, e.g. technological process data, users password (PostgreSQL 6.5.3);
- a database driver: it allows the connection from the web server to the tables (JDBC);
- java and java servlets: used for the implementation of the browser interface (in "html" pages) and the processing of the information (JAVA2, JSDK2.0);
- web server (Apache Server 1.3.11);
- java servlet engine: it manages through the Hyper Text Transfer Protocol "http" the communication between the client (user interaction with browser) and the server (database and java code) (Apache JServlet Server 1.1);
- web browser: it allows for user-system interaction (Netscape and Internet Explorer).

Among the quality targets of the system obtained thanks to the design techniques adopted:

- portability;
- access through the net;
- object-orientation;
- WEB based;
- easiness to use also to not trained people;
- it allows the contemporary access, also in write mode, to the same table from different users, without deadlock problems.

4. REFERENCES

- [1] *The Sixth Report of the Joint National Committee on Prevention, Detection, Evaluation, and Treatment of High Blood Pressure*. NIH Publication No. 96-4080, November 1997;
- [2] Ivano Azzini, Daniele Falavigna, Roberto Gretter, Giordano Lanzola, Marco Orlandi. *First steps toward an adaptive spoken dialogue system in medical domain*, EUROSPEECH 2001, Aalborg, Denmark, September 2001 - IRST Tech. Rep. No. 0104-21;
- [3] E.U. Project "HOMEY" *Home monitoring through intelligent dialog system*. EC Fifth Framework project IST-2001-32434.
- [4] <http://www.netscape.com/eng/mozilla/3.0/handbook/plugins/>
- [5] G. Booch, J. Rumbaugh, I. Jacobson, *UML: The unified modeling language - User Guide* (Addison-Wesley 1999)
- [6] S. Khoshafian, *Object-oriented databases* (Wiley, New-York, 1993)
- [7] <http://www.w3.org/Style/XSL/>
- [8] L. Ferrario, C. Armaroli, M. Zen "WebFabIS: A Web system for Microelectronics Laboratories Activity" RM2001- iasted conference - Cancun Mexico May 2001