ON THE USAGE OF AUTOMATIC VOICE RECOGNITION IN AN INTERACTIVE WEB BASED MEDICAL APPLICATION

C. Eccher, L. Eccher, D. Falavigna, L. Nardelli, M. Orlandi, A. Sboner

ITC-irst, Istituto per la Ricerca Scientifica e Tecnologica
Via Sommarive, 18, 38050 - Povo (TN), Italy

ABSTRACT

We will describe the multi-modal browsing system, developed by us, that allows to add automatic speech recognition and text to speech functions to standard Internet browsers. The system is based on the temporal synchronization of HTML and VoiceXML documents. It was developed starting from a real Web application designed for a medical domain (i.e. an electronic patient record adopted in the oncology unit of an Italian hospital). We have recently introduced the possibility to define the multi-modal interaction by means of a single XML document. System evaluation is going to be carried out on data collected during the usage of the system in the hospital.

1. INTRODUCTION

The paper describes the evolution of the system reported in [1] that allows to add automatic voice recognition functions to standard Internet browsers (e.g. "Internet Explorer" or "Netscape"). The basic idea, underlying the system design, consists in the definition and consequent realization of a software architecture capable of handling multi-modal interactions through the synchronization of HTML and VoiceXML documents. HTML documents define a usual interaction based on standard devices, such as: graphic monitor, keyboard, mouse and touch screen, while VoiceXML documents define a corresponding interaction based on voice. The multi-modal browsing system detects events coming from various types of input devices, including a microphone, and provides output according to predefined spatial and temporal layouts. In general, the spatial layout is specified in HTML documents, while the temporal layout is specified in VoiceXML documents. Note that, in this way, the user can freely interact by using the preferred device (e.g. the mouse, for selecting an item from a short list of options, or voice for filling the fields of a form) and the system is able to provide output in a coherent way.

For both testing the system and tuning its parameters we have chosen a medical application scenario, where the goal is that of entering data of laboratory test results into a patient database. To cope with this application, namely a distributed Electronic Patient Record (EPR), we developed a system [2] in the past that utilizes a HTML based web interface for accessing the patient database. Hence, to add voice browsing capabilities to the system, we only need to define a VoiceXML interface corresponding to the given HTML one (i.e. we must write a corresponding set of VoiceXML documents[1]).

In general, the usage of EPRs implies a large amount of work by people inside hospitals, for storing clinical data collected in different formats: text reports, numerical values, images, etc. Moreover, EPRs allow the user to easily go across a huge amount of information to find the needed one. Hence, EPRs are usually organized in sections, in which information is structured in categories that are homogeneous from the medical knowledge perspective. Whatever the EPR structure is, data entry and retrieval is primarily accomplished by using keyboard and mouse. In several cases this approach can be very time consuming and error prone. Adding voice recognition capability might largely improve the efficiency of EPRs and, for this reason, EPR applications are particularly suitable for testing our multi-modal browsing architecture. Finally, as far as we know, automatic speech recognition in the medical field has been mainly related to continuous speech dictation applications (e.g. dictation of radiological reports); up to now, no voice technology has been extensively used to develop interactive services for accessing data organized in medical records.

The paper is organized as follows: section 2 describes the application scenario related to the developed oncological EPR, section 3 describes the system architecture and gives the details of the visual/voice synchronization mechanism, section 4 concludes the work.

2. APPLICATION SCENARIO

The EPR system was specifically designed to support patient management in the oncology unit of the S. Chiara Hos-

e-mail:{cleccher, eccherlo, falavi, lunarde, orlandi, sboner}@itc.it

[1]For the specification of the VoiceXML Markup language see http://www.voicexml.org

pital of Trento (Italy). This unit is structured in either an inpatient ward and in a day hospital ward, during which chemo-therapy care is delivered to outpatients. The workflow of the day hospital patient management consists of four tasks: patient admittance, patient visit, chemo-therapeutic drug preparation and drug administering.

1. The patient admittance task is committed to a reception nurse. If a chemo-therapy session is scheduled for a certain date, the patient presents herself/himself at the day hospital reception, handing a list of prescribed laboratory test results to the nurse. The nurse has to retrieve the patient clinical record (either electronic medical record or paper medical record) and insert the test values. The patient's eligibility for a chemo-therapy treatment is determined controlling the blood test results. The admittance task is successfully completed when the nurse commits to an oncologist the patient visit task.

2. The oncologist visits the patient to asses her/his conditions for either prescribing a new chemo-therapy or deciding if an ongoing therapy has to be continued, modified or interrupted. During the visit the oncologist needs to consult the patient's data contained in her/his clinical record. The visit ends when the oncologist delivers the list of drugs, for the prescribed therapy, to the preparation room.

3. The preparation of the patient specific chemo-therapy drugs is performed by a nurse. This task is completed when the administering room is provided with the prepared drugs.

4. The last task of the workflow is the availability of the prescribed drugs. The administering task is committed to a nurse, who prepares the patient and inoculates the drugs. This task can lasts several hours.

The usage of voice technology could be effective during the patient admittance task. This consideration is supported by two main arguments.

1. The admittance task is mainly a heavy data entry task, in which there is the necessity to search patients in the database and to enter many numerical values related to the patient's laboratory test results. Automatic speech recognition could speed the data entry phase reducing the error rate.

2. Since reception nurses have to enter data by reading from patient's documents, it would be very useful for them to keep their hands free.

3. SYSTEM DESCRIPTION

In the following, the EPR system and the multi-modal browsing system will be described.

3.1. EPR system description

The EPR system is a 3-tier web-based system. This ensures system flexibility and scalability. The general architecture is illustrated in figure 1. The client side is mainly a web-browser able to manage JavaScript code, dynamic HTML, Document Object Model (DOM) and eXtensible Stylesheets Language (XSL) files. XSL stylesheets describe how to transform XML documents for generating, for instance, HTML web pages[2]. XSL stylesheets are also used for defining, in a single XML document, both visual and voice interactions, as will be explained below.

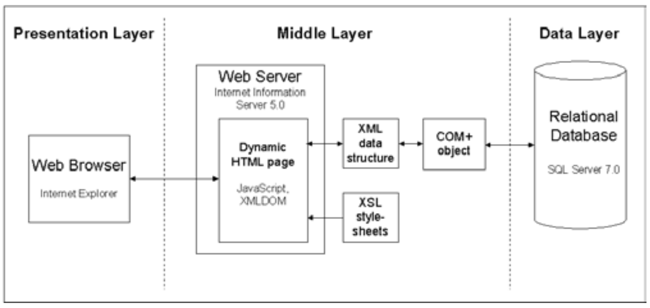

Fig. 1. General architecture of the EPR system.

In our investigation we used Microsoft Internet Explorer version 6.0. On the server side, the middle-layer consists of a web-server (Microsoft Internet Information Server) and a Microsoft COM+ component, whereas the data layer is made up by a relational database (Microsoft SQL Server). The middle-layer performs two tasks:

1. the conversion of tabular data format to XML data format and vice versa;

2. the creation of dynamic HTML pages, that include both XML and XSL stylesheets, needed by the client to display the information with the correct layout.

Whenever the user requires a certain information about a patient through the web-browser, the web-server first activates the COM+ component. This component retrieves the data from the relational database, creates the XML hierarchical structure and inserts the data into the XML structure. The filled XML structure is then embedded into the dynamic HTML page together with the corresponding XSL stylesheet.

[2]For more details about XSL stylesheets see http://www.w3.org/Style/XSL/

On the client side, the XML data are managed by an ActiveX object (MSXMLDOM), implementing W3C DOM interfaces. By means of Javascript, we can call its member functions performing the transformation from XML data structure to HTML data structure defined in the XSL stylesheet.

By default, the data are graphically presented. When the user would like to insert or modify the data, e.g. by simply clicking a button, a new stylesheet is downloaded from the web-server. The new stylesheet provides the new data layout, showing the dynamic HTML tags that are used to input data, such as text and radio buttons. The information inserted by the user, by means of HTML tags, are directly bound to the corresponding node in the XML structure. Hence, to save the data, it is sufficient sending the filled XML data structure to the server, where the COM+ component converts them into the more usual tabular data form and stores them into the relational database. The great advantage of this solution, with respect to traditional web based client-server applications, is that data remains on the client side and only the XSL stylesheets are downloaded from the server to provide different views of the data themselves.

3.2. Multi-modal browsing system description

As seen above, the multi-modal browsing system [1], developed in ITC-irst, can interpret VoiceXML documents, as a standard Internet browser does with HTML or XML documents. A VoiceXML interpreter enables user browsing by speaking and hearing, but does not support a graphic interface, as HTML does. We propose to synchronize different documents through a specific platform instead of adding new features to existing HTML, XML 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.

The architecture of the multi-modal browsing system we are proposing is shown in figure 2. 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 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);

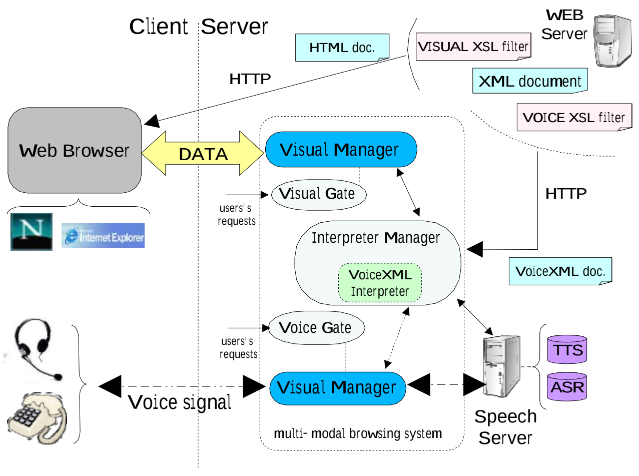

Fig. 2. Architecture of the multi-modal browsing system.

- the Speech Server , which manages both ASR and TTS resources;
- the multi-modal browser process, which integrates the VoiceXML Interpreter.

We point out that both the ASR and the VoiceXML interpreter have been developed in ITC-irst. This last one is compliant with release 1.0 of the VoiceXML specification and was entirely developed in the Java language [3, 4].

The multi-modal browsing system consists of 3 main components:

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

The two Managers (voice and visual) interact with the VoiceXML Interpreter through the Interpreter Manager. Two Gates listen, on two different sockets ports, for user requests and create new instances of the appropriate Managers.

Our solution for synchronizing HTML and VoiceXML documents consists in opening and maintaining a TCP/IP connection between the Web browser, on the client side, and the multi-modal browsing system on the server side. This TCP/IP connection is indicated in figure 2 as the “DATA” connection. 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 browsing system 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 browsing system on the server side through the Visual Manager, which in turn updates the corresponding VoiceXML field item variable. This process requires the presence of both an appropriate JavaScript file and a Java applet loaded in a hidden frame. 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[3], the applet and the JavaScript functions can interact with each other.

The user actions with a Web page are notified to the multi-modal browsing system through JavaScript function calling. This function uses the applet methods to require to the server to update a VoiceXML document. The Visual Manager 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 browsing system, through the Interpreter Manager and the Visual Manager, uses this event to update the corresponding HTML document through the TCP/IP connection (the “DATA” connection in figure 2) opened by the applet. Finally, the applet calls the JavaScript functions that modify the Web page. 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.

It could be useful to have the possibility to define a single XML document that comprises all of the information needed to generate different multi-modal interactions, by means of several XSL stylesheets. Then, as seen above, XSL stylesheets can be used for specifying, according to a given formalism, both visual and voice layouts. In this way, HTML and VoiceXML documents can be directly derived from the given single XML document, as shown in figure 2. This solution has the advantage to separate the layouts (temporal layout defined by VoiceXML documents and bi-dimensional layout defined by HTML documents) from the data structure (XML document) and to easily allow the integration of new interacting modes (e.g. user’s gesture recognition, talking head, etc...) by defining the appropriate XSL filter.

[3]For more details about Live Connect see http://wp.netscape.com/eng/mozilla/3.0/handbook/plugins/index.html

4. CONCLUSIONS AND FUTURE WORK

We have presented a system architecture that allows to add ASR and TTS capabilities to a standard Internet browser. The system has been developed for a data entry application in a medical domain, however its usage is general, provided that an appropriate description of the given application is developed in both HTML and VoiceXML languages.

The described system is currently used, without ASR functions, for handling the admittance phase of an oncological day hospital ward. During the system usage the overall time requested for entering laboratory test results by keyboard is stored. Furthermore, for carrying out the data entry task, the multi-modal browsing system will soon be used (i.e. data entry will be done by means of either voice or keyboard) and performance will be evaluated according to: speech recognition accuracy, time needed for data entering and some measures for expressing the overall efficiency of the system.

Finally, we are adapting the whole system architecture to support interactions with Personal Digital Assistants (PDAs). In this case the addressed application scenario consists of a computer network (e.g. a LAN) where PDAs are connected through a wireless transmission protocol. Note that, also in this case, the ASR and TTS resources are located on the server side.

5. REFERENCES

[1] C. Armaroli, I. Azzini, L. Ferrario, T. Giorgino, L. Nardelli, M. Orlandi, and C. Rognoni, “An architecture for a multi-modal web browser,” in *Proc. ICSLP*, Denver, CO, USA, 2002, pp. 2553–2556.

[2] F. Demichelis, F. Berloffa, C. Eccher, M. Galvagni B. Larcher, A. Sboner, A. Graiff, and S. Forti, “Implementation of a regional tele-oncology project,” *Journal of Telemed Telecare*, vol. 6, no. 1, pp. 71–73, 2000.

[3] R. De Mori, *Spoken Dialogues with Computers*, Academic Press, London, 1998.

[4] D. Falavigna, R. Gretter, and M. Orlandi, “Mixed language model for a dialogue system over the telephone,” in *Proc. ICSLP*, Beijing, China, 2000, pp. 585–588.