VoiceXML in a Real Automatic Meteorological Information System

Luis Villarejo, Javier Hernando, and Núria Castell

TALP Research Center, Universitat Politècnica de Catalunya
Campus Nord - A0, Jordi Girona 1-3, 08034 Barcelona, Spain
{luisv, javier, castell}@talp.upc.es

Abstract. This paper describes the work done in developing a real automatic meteorological information system by means of building a VoiceXML framework, or browser, over an open source VoiceXML interpreter. The system provides real-time telephone access to meteorological data and a warning service which keeps users informed on the weather conditions. The interpreter has been integrated with the telephony platform, the text-to-speech engine and the automatic speech recognition engine. So we have obtained a fully functional framework for VoiceXML applications quick development and deployment. In this paper, after the description of the system functionalities, we present the framework by describing its architecture and the effort done in integrating the voice technology with the interpreter as long as in overcoming the difficulties found from the industrial point of view.

1 Introduction

The spread of voice applications has reached many fields, not only in research but also in education and industry. Meteorology has not been an exception, for a long time it has been provided by means of weather reports on television, radio and lately on the internet. Nowadays speech access permits making this information available to a large number of customers anytime and anyplace from their mobile phone. The VoiceXML standard[1] is taking an important role in giving support to this kind of interactive speech systems developments which cover a great range of domains. To quickly review some VoiceXML-based systems we should mention the INSPIRE[2] (Infotainment management with speech interaction via remote-microphones and telephone intefaces) project, which focuses on speech dialogue-based assistant for wireless command and control of home appliances. Or the FAZ.NET Fonservice[3] (Frankfurter Allgemeine Zeitung) project which focuses on giving speech access to this German newspaper information. Lately some systems have focused on meteorological information like the JUPITER[Zu00] project, not developed in VoiceXML but being a reference work in this field.

In this paper we describe aTTemps[1] [PH02], a VoiceXML system which provides

[1] This work has been supported by the Catalan Secretary for the Information Society and by the Spanish Government (TIC2000-1735-C02-01).

not only a set of personalized real-time meteorological data but also a fully configurable warning service activated either by time or by meteorological conditions. The system makes use of a wide range of the VoiceXML 2.0 dialog capabilities. We start by presenting, in section 2, the functional goals of the system, its information sources, its geographic coverage and a typical dialogue example. In section 3 the system architecture is described. After that, we discuss the integration issues involved with implementing the framework in section 4. Section 5 gives a general overview on the dialogue manager. And at the end of the document, section 6, we present the conclusions and further work.

2 aTTemps scenario

aTTemps covers two main functional requirements: offering real-time data and setting up a warning message. In more detail, the first one, as can be seen in Figure 1, is to provide personalized real-time data on a set of meteorological conditions/variables on each place of the Catalan geography in the Catalan language via phone. So users can ask the system for meteorological conditions/variables such as temperature, wind speed, rain forescast... taking place at calling time. A dialogue is initiated by the system in order to understand the user request and provide the desired information. The second functional requirement, as we stated before, is to establish a notice and alarm system on the meteorological conditions. So users can ask the system to be notified (with an SMS or a voice message) either when some user defined meteorological conditions take place or at certain date and time about some meteorological conditions. In order to give support for both scenarios the system exploits different information sources from the *Servei Meteorològic Català* (SMC[4], Catalan Meteorological Service).

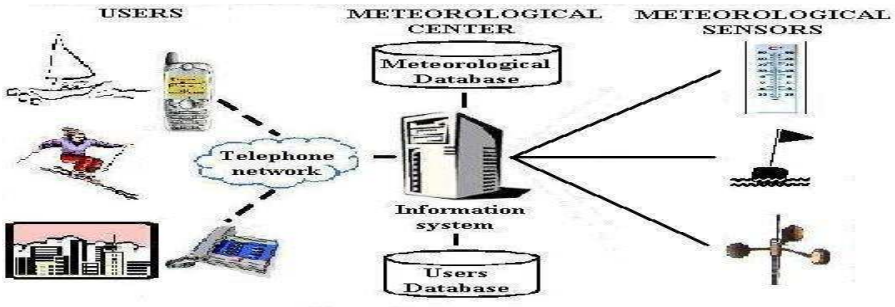

Fig. 1. Real-time meteorological data on demmand

2.1 Information Sources and Geographic Coverage

Data queried by the users are obtained from four different information sources offered by the SMC:

1) Two meteorological ground nets, made up of ninety one automatic ground stations, scattered over the Catalan geography. These stations are connected via satellite or modem to the meteorological centre where an average of the data acquired constantly by the stations is received every half an hour.

2) A meteorological sea net, made up of four automatic buoys scattered over the Catalan coast. Constantly acquiring information, an average of the data is sent to the meteorological centre via satellite every hour.

3) Rain forecast, updated twice a day, providing information about rainfalls for the next 12, 24 and 36 hours.

4) Weather alarms, updated every half an hour, on special situations like risk of fires, hailstorms, danger of floods, avalanches...

All this information is offered about nearly one thousand Catalan municipalities and a set of toponyms of special interest like airports or tourist places. Of course, there is not one meteorological station for every municipality, so an inference system has been developed in order to set the station which best suits for every municipality.

2.2 Example of Use

A typical dialogue, translated to English, is described below in order to give an idea of the basic interaction between the user and the system. In this case we present a user who wants to be warned if significant rainfalls take place in Badalona city.

System: Welcome to the attemps meteorological warning and information system. You can always say help, correction, repetition or exit. From which Catalan place do you want information, or do you want to establish a warning?
User: I would like to activate a warning.
System: Tell me, digit by digit, the telephone number where you want to receive the warning. You can also dial it on your telephone.
User: 6 6 9 5 1 6 1 5 8
System: To the 669 516 158, about which Catalan place do you want to receive the warning?.
User: Ummmm.... about Badalona, please.
System: About Barcelona, in which information are you interested?
User: No, not from Barcelona, I want to correct.
System: About which Catalan place do you want to receive the warning?
User: I want a warning about Badalona.
System: About Badalona, in which information are you interested?
User: I need information on rainfalls.
System: Do you want to receive the message if it rains, if it doesn't rain or depending on

rainfalls quantity?.
User: Notify me if rainfalls are over 5 liters per square meter.
System: Which kind of warning do you want to receive? at a specific date and time, every time rainfalls at Barcelona reach over 5 liters per square meter, or just the first time?
User: I want you to notify me every time, please.
System: Do you want to be notified with an SMS or a phone call?
User: An SMS will be ok.
System: Ok, you will be notified with an SMS to the 669 516 158 every time rainfalls at Badalona reach over 5 liters per square meter. Is this correct?
User: Yes.
System: The warning has been correctly registered, do you want any other information?
User: No, I don't. Thanks.
System: Thanks for calling, good bye.

3 aTTemps Architecture

System architecture follows the typical VoiceXML architecture, that means that it is organized around the *VoiceXML framework* (detailed in next section) which requests the *web server* for the VoiceXML documents that make up the *dialogue manager*. Those documents are generated in the server side with dynamic content from the *system database* which is updated every half an hour by the *data acquisition module* that queries the SMC server. In order to understand later sections a short overview of all modules (excluding the framework and the dialogue manager which are detailed in next sections) is done in this section. For a deeper knowledge of the system see [Vi02].

System data is stored in two different databases. The first one, a relational database, stores the meteorological information and is updated by the data acquisition module. While the second one, an LDAP (Lightweight Directory Access Protocol) database to optimize searches, stores the user profile kept for every warning and is updated every time that either a warning takes place or a user activates or deactivates a warning.

The data acquisition module updates the meteorological database every half an hour with the information acquired from the SMC server. This module stands as the guarantor for real-time data.

4 VoiceXML Framework

The main goal in building this framework has been setting up a structure that facilitates and accelerates as much as possible the development and deployment of oral dialogue managers by making them independent from the voice technology. Firstly, we identified VoiceXML as the language for our framework because it standardizes voice applications development taking advantage of web technologies and making

the dialogue manager independent not only from voice technology but also from application logic. Once the language was chosen, OpenVXI 2.0 [Eb02] from Speech-Works[5] was chosen as the toolkit to interpret VoiceXML because of its portable open source design. So, as can be seen in Figure 2, the framework is based on the interpreter which acts as the skeleton where to integrate the technology dependent components. These are the telephony platform, the text to speech engine (TTS) and the automatic speech recognition engine (ASR). The effort done in integrating these technology dependent components is made by means of different application programming interfaces (API) and is briefly described in the next three sections.

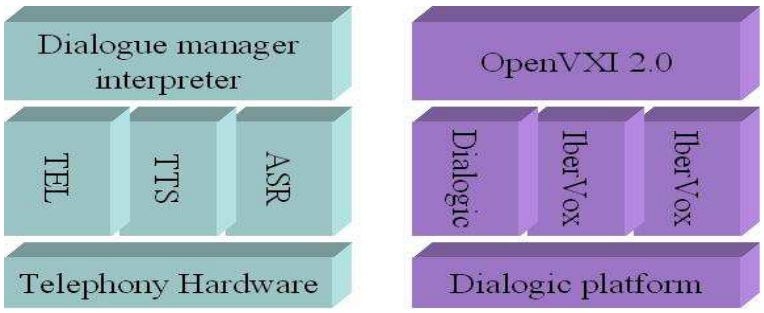

Fig. 2. VoiceXML framework architecture and technology dependent components used

4.1 Telephony API

The building of this API involved the integration of Dialogic[6] telephone card, as the framework telephony platform. The work done in this API mainly consisted in firstly, initialize the telephony platform and wait for a phone call to come. Secondly, once a phone call has been received, a logical channel is assigned to it and notified to the interpreter. And last but not least, the handling of telephony events during the life of the call.

4.2 Text To Speech API

The development of this API implied the integration of IberVox[7] as the framework TTS engine. This software provides asynchronous message reproduction which was incorporated. The main goal here is to provide text-to-speech services, that is, taking care of prompting management and providing the audio to the telephony services for playback.

The management strategy consisted firstly in preprocessing every prompt in order to translate the Speech Synthesis Markup Language (SSML) marks into the specific TTS engine language. And secondly in maintaining a buffer where prompts

are stored.

The strategy used when a play event is detected is made up of four steps. First of all, the content of the buffer holding the prompt basic block is recovered and the buffer released. Secondly, a silence is added at the end of the prompt if the barge-in property is on (this is justified in the ASR API section). Thirdly, the system waits for the channel to be free, this is waiting either for any messages to finish its reproduction or for any speech recognition action to finish. And finally, an asynchronous reproduction event is sent to the TTS engine.

4.3 Automatic Speech Recognition API

The building of this API entailed the integration of IberVox as the framework ASR engine which is in charge of processing caller input obtained via the telephony services. The two tasks to tackle here are the speech recognition services handling (explained in the next section due to its high barge-in dependency), and the coordination of the interpreter and the ASR engine to manage grammars.

One of the interpreter deficiencies was grammar management. We found that the syntactic parser offered by the interpreter only admitted word lists, that is, with no syntactic structure. That was totally insufficient if real applications were to be implemented over this framework, so an ABNF parser was implemented and incorporated in order to accept a wider and richer range of grammatical constructions from the user. This parser accepts both in-line and external grammars giving support to dynamic grammar construction. The grammar processing involves the following steps for each loaded VoiceXML document: 1) getting grammar content 2) doing syntactic parsing and 3) doing a transcription into an internal format facilitating its handling. Last step represented a bottle neck due to the time taken by this process when big grammars, like the one holding one thousand places, are involved. That is commonly avoided by preprocessing big grammars and making its internal representation persistent so as compilation only has to be done once. Once all grammars in the document have been processed, the interpreter determines in every moment which of them are active. Processing just active grammars is an interpreter improvement which speeds up the framework.

4.4 Supporting barge-in

ASR and TTS API's must be coordinated, specially to handle the barge-in property. Depending on its value two different reproduction policies are applied:

1) With barge-in on, speech reproduction and recognition must be thrown simultaneously. This means that grammars must be compiled before playing any message. In order to simplify the prompt management the system stores prompts in a buffer by means of basic blocks, where a basic block is a sequence of prompts not cut by any play. So the prompts are not reproduced as soon as the system processes them

but when a play event is thrown (which is usually followed by a speech recognition action). Recognition and play functions are stopped when a termination event from either is detected, letting the user to interrupt the system messages whenever s/he wanted.

2) With barge-in off, in order to avoid uncomfortable and unnecessary silences caused by big grammar processing or long calculus performance, and taking advantage of the TTS asynchronous capabilities prompts are reproduced as soon as they arrive to the buffer. So the system can go on while reproducing a message.

5 The dialogue manager

The dialogue manager is made up of VoiceXML files which are generated with dynamic content (from databases) in the server side using CGI scripts. Designing the dialogue several factors must be taken into account like the dialogue flow, the confirmation policy, the language generation, the helping features and the application structure into VoiceXML documents. Among them, here we would like to point out the last two. For a deeper description see [VCH03].

The helping system has been based in two main features. Firstly, setting up an automatically adaptable help messaging service taking advantage of VoiceXML features. This service aim is to provide helping messages exactly when needed avoiding annoying the user with unnecessary messages. So when welcome is done, user behaviour is briefly explained and after that, help is only provided when user keeps quiet, when user input is not understood by the system or when user explicitly asks for help.And Secondly, a set of Catalan key/words expressions, shown in Table 1, has been set up to support most frequent user actions anytime.

Table 1. Key words/expressions

Key forms (Translated to English)	User needs	System reaction
"Again please?", "Repeat please"...	Repetition	Repeats last message
"I want to rectify", "Correction"...	Correction	Asks for the last value given
"Help me", "I want help"...	Help	Provides help
"Good bye", "Exit"...	Exit	Finishes the interaction

Application structure has been determined by the main interpreter bottle neck, that is the grammar compilation. As we stated before, the interpreter compiles all grammars in the current document without taking into account whether each grammar is active or not. Considering this limitation and the fact that loading a document takes a negligible time we decided to separate grammars as much as needed in different documents in order to speed up the document processing. Specially the one containing the places grammar which is the biggest.

6 Conclusions

A working meteorological information VoiceXML-based system has been implemented covering the functional requirements stated. In addition to giving meteorological data, the system provides a warning service which can be activated in a quite friendly way. We have presented not only the meteorological aplication, operated via a natural language interface, but also the integration process needed to build the voice framework which supports it. As a result of that, this framework serves as an implementation platform for applications dealing with the VoiceXML standard.

While integrating speech technology in the interpreter, some deficiencies have been detected. A major issue was the interpreter and the ASR engine coordination to solve grammar management. The policies implemented to overcome these deficiencies have been presented. These policies go from the need of developing an ABNF grammar parser to the necessity of precompiling very huge grammars, due to both interpreter and speech technology software lacks. In order to improve the naturalness of the system, the grammars should be enriched to recognize a greater range of utterances from the user. Moreover, the addition of the already developed text generation module will increase the system response variability.

References

[Eb02] Eberman, B., Carter, J., Meyer, D., Goddeau, D.: "Building VoiceXML Browsers with OpenVXI", 2002. http://www2002.org/CDROM/refereed/260/index.html

[PH02] Padrell, J., Hernando, J.: "ATTEMPS: Access to Meteorological Information by Telephone", ICSLP 2002, vol. 4, pp. 2713-2716.

[Vi02] Villarejo, L.: "Gestor de diálogo de un sistema de información meteorológica", in Spanish, Master Thesis in Informatics Engineering, FIB, 2002.

[VCH03] Villarejo, L., Castell, N., Hernando, J.:"Dialogue Management in an Automatic Meteorological Information System", IEA/AIE 2003, LNCS vol. 2718, pp. 495-504.

[Zu00] Zue, V., Seneff, S., Glass, J.R., Polifroni, J., Pao, C., Hazen, T.J., Hetherington, L.: "Jupiter: A Telephone-Based Conversational Interface for Weather Information", IEEE Transactions on Speech and Audio Processing, vol. 8, n. 1, pp. 85-96, 2000.

1. W3C, Voice eXtensible Markup Language (VoiceXML) version 2.0, http://www.w3.org/TR/voicexml20, February 2003.
2. INSPIRE project: http://www.inspire-project.org
3. FAZ.NET project: http://www.hltcentral.org/page-1058.shtml
4. Servei Meteorològic Català http://smc.gencat.es.
5. SpeechWorks, speech recognition solutions provider. http://www.speechworks.com
6. Dialogic, supplier of computer telephony products. http://www.dialogic.com
7. Ibervox from Atlas-CTI. http://www.atlas-cti.com/es/ibervoxasr.htm