Appl Intell (2006) 24: 253–261
DOI 10.1007/s10489-006-8516-5

Architecture and dialogue design for a voice operated information system

Luis Villarejo · Javier Hernando · Núria Castell · Jaume Padrell · Alberto Abad

Abstract In this paper we present a real automatic meteorological information system that, not only provides friendly voice access to real-time data coming from automatic sensors, but also establishes an automatic warning service on the weather. It aims to extend the availability, personalization and friendliness of the meteorological information by means of a reusable easy-to-use friendly oral natural language interface. This interface takes advantage of the improvements in speech processing, dialogue handling and the great growth of mobile telephony. After the description of the functionalities of the system and its architecture, we present in detail the features of the dialogue manager. The main goals we have considered are: to provide the right information and to design a friendly interface.

Keywords Intelligent interfaces · Natural language processing · Speech processing and VoiceXML

1. Introduction

Meteorological information has long been provided by television and radio as weather reports scheduled at a fixed timetable giving impersonal information. Nowadays there are telephonic systems that offer some personalized information but in an unnatural way. Usually these systems offer some menus operated via the telephone keyboard and have a very limited vocabulary. These features condemn the system to failure in the real world due to a lack of naturalness and fluency in the dialogue. This kind of systems usually provide general information and are updated less frequently than desired.

The interest and research on interactive speech systems has increased in recent years due to the extended use of telephone information systems. We should mention the TRINDI [1] (Task-Oriented Instructional Dialogue) project, which focuses in generic technology for the creation of a dialogue movement engine. Nowadays there are systems that offer a good level of interaction during the information exchange. Some examples are the following: ARISE [2] (Automatic Railway Information Systems for Europe), TRAINS [3], and BASURDE [4] (Spontaneous Speech Dialogue System in a Semantically Restricted Domain), all of them are about railway information, and ATIS [5] (Air Travel Information System), is about flights. Some systems are designed to offer meteorological information in a user-friendly way. A reference work in this field is the JUPITER [6] project.

The system described here provides personalized real-time data, in the Catalan language, on a set of meteorological conditions on each place of the Catalan geography through an easy-to-use natural language interface. It also provides an alarm and warning system, based on the same interface, that keeps the user informed on the variables of his/her interest whenever they occur. The whole system has been constructed on the basis of VoiceXML (the voice standard promoted by the World Wide Web Consortium), specifically the voice framework and the dialogue manager.

The VoiceXML standard [7] 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 [8] (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. Also the

L. Villarejo (✉) · J. Hernando · N. Castell · J. Padrell · A. Abad
TALP Research Center, Universitat Politècnica de Catalunya
Campus Nord - A0, Jordi Girona 1-3, 08034 Barcelona, Spain
e-mail: {luisv, javier, castell, jaume, alberto}@talp.upc.es

FAZ.NET Fonservice [9] (Frankfurter Allgemeine Zeitung) project, which focuses on giving speech access to this German newspaper information. By building a voice framework based on VoiceXML, the dialogue manager becomes independent not only of the voice technology but also of the application logic facilitating the system reusability. The development of this system has been promoted by the Catalan government, under the name of aTTemps [10] project.

In this paper we start by presenting, in Section 2, the services offered by the system and its meteorological data source, the Catalan Meteorological Service. In Section 3, the system architecture is presented and a brief description on how the system works is done. After that, the issues related to the dialogue manager are discussed in Section 4. Section 5 introduces the language processing. In Section 6 the evaluation of the system is discussed. At the end of the document, Section 7, we present the conclusions and further work.

2. Meteorological information and services

The information offered by the system is collected by the *Servei Meteorològic Català* (SMC) [11] from four different sources:

(1) The Meteorological Net and the Atmosferic Pollution's Surveillance and Forecasting Net nets consists of ninety one automatic meteorological ground stations, scattered over the Catalan geography. These stations are constantly acquiring information on different variables, such as temperature or direction and force of the wind, in order to supply measures of each of them every half an hour (via satellite or modem) to the meteorological centre.
(2) The Meteorological and Oceanographic Instruments Net net consists of four automatic meteorological buoys, scattered over the Catalan coast. These buoys are constantly acquiring information on maritime variables, such as the sea disturbances, the ground swell and the height of the waves, to supply measures of each of them every sixty minutes (via radio) to the meteorological centre.
(3) The rain forecast which provides information, updated twice per day, on the rain intensity and areas for the next thirty six hours.
(4) The weather alarms which provide information, updated every half an hour, on risk situations such as hailstorms, danger of floods, avalanches or risk of fires.

2.1. aTTemps services

As we stated in the Introduction section, the system developed under the aTTemps project can be used in two ways, depending on the user needs:

(1) Asking for immediate information: The user needs to know the values of some meteorological variables or the rain forecast. The system initiates a dialogue to get the desired variable and the location of interest in order to provide the information.
(2) Asking for activation/deactivation of a warning. The user needs to be notified (with an SMS or a voice message) on the weather conditions either when some meteorological conditions occur or at a certain time of the day. The system initiates the dialogue and saves the user warning profile in its database.
(3) In all cases the user can ask for information not only in all the Catalan municipalities (949), but also in larger areas (regions, mountains, natural parks, etc.), other spots which have a particular tourist interest (airport, etc.), and popular abbreviations of names for a total number of 1376 places availables. In addition, if the system does not have information from the requested place, information from the closest place (climatologically speaking) will be provided.

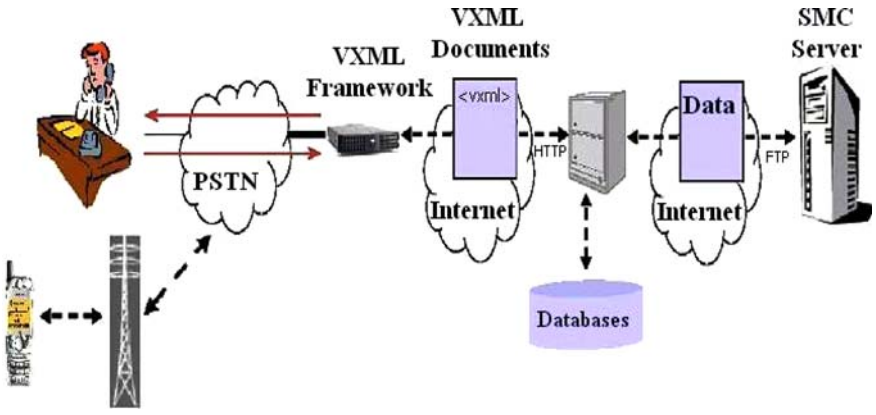
2.2. aTTemps dialogue examples

In this section we present a typical dialogue example, translated into English, for each aTTemps service in order to give an idea of the basic interaction between the user and the system. In the first one the user is interested in knowing the temperature in Barcelona; in the second one, the user is interested in being warned when rainfalls in the city of Blanes go over a certain threshold.

Dialogue 1: Asking about the temperature.

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 am interested in Barcelona, please.
System: From Badalona, which information do you need?
User: I want to correct, not from Badalona.
System: From which Catalan place do you want information?
User: I want to know the weather right now in Barcelona.
System: From Barcelona, which information do you need?
User: I need the temperature, please.
System: In the last measure at 14:30 h, in the meteorological station of Barcelona, the temperature was 20 degrees. Do you need any other information?
User: No, that's all. Thanks.
System: Thanks for calling. Good bye.

Fig. 1 System architecture



Dialogue 2: Activating the warning system.

System: Welcome to the aTTemps...
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: About Blanes, please.
System: About Blanes, 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 when 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 Blanes 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 Blanes 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. System architecture

Two separate developments were done for the dialogue managing which affected the whole system architecture. The first one was based on a traditional architecture while the second one was based on a VoiceXML-oriented architecture. In this paper we are going to talk about the VoiceXML-oriented architecture.

The system is made up of four main blocks , as can be seen in Fig. 1, which correspond with: (1) the VoiceXML framework, which integrates the speech processing and the telephony hardware, (2) the VoiceXML dialogue manager (made up of VoiceXML documents and the application logic both in the web server), which integrates the dialogue control and the language generation, (3) the system databases and (4) the data acquisition module, which provides the real-time meteorological contents by accessing the SMC server.

A short overview of all modules (excluding the one of the dialogue manager that is going to be explained in detail in Section 4) is done in this section. A more detailed overview of this system can be found in [12].

The VoiceXML framework is made up, as can be seen in Fig. 2, on the basis of the OpenVXI 2.0 [13] interpreter from SpeechWorks [14]. This interpreter allows easy integration of telephony and speech processing components by means of APIs where specific developments are done in order to integrate the Dialogic [15] components for telephony purposes and the Ibervox [16] (cf. Section 4.5) components for speech processing purposes.

The databases module is composed of two main databases which store the two kinds of information that the system must maintain: the meteorological data and the user's profiles for warnings. The first one (referred from now on as meteorological database) is stored in a relational database which is updated constantly by the data acquisition module. While the second one (referred from now as user database) is stored in

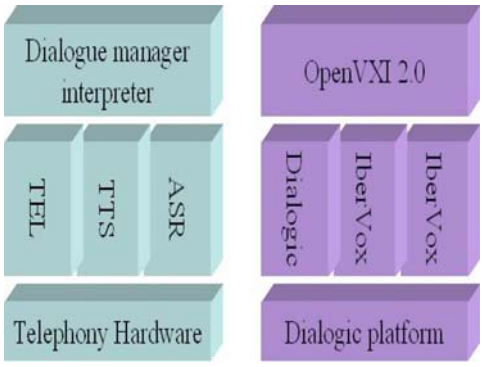


Fig. 2 VoiceXML framework structure

a LDAP (Lightweight Directory Access Protocol) database in order to optimize searches. This database is updated every time that a user activates or deactivates a warning message.

The data acquisition module consists of a process that acquires, every half an hour, the meteorological data in real time from the SMC databases and updates the local meteorological database. This process guarantees that we get the new data as soon as there is in the remote database.

The system works as follows:

(1) The framework waits for a telephone call.
(2) When a call is received, the root VoiceXML document of the dialogue manager is executed by the framework.
(3) This execution causes the reproduction of the welcome message (by means of the voice technology integrated in the framework) that gives a little notion about how to interact with the system.
(4) Then the system asks to the user about which information he/she wishes.
(5) The grammars that should pick up the user's answer are loaded.
(6) When the user answers, his/her reply is given back to the document. Once the reply is received, the execution of the document can go on.
(7) Depending on the user needs, the dialogue manager will ask another question to the user or will do a request on the server to get data from the database. In that last case, the query is sent to the database interface and the result is picked up and used, in the server, to build a document with dynamic content that will be sent to the framework for execution.
(8) Once all user requested data has been collected, the system provides the meteorological information (contained in the dynamically generated document) to the user in natural language or takes record of the warning profile for that user.

4. The dialogue manager module

The main goals of the dialogue manager are: to provide the right information in the minimum time, to be a friendly oral natural language interface that facilitates as much as possible the interaction with the user, and to guide the user in order to avoid situations where he/she would be lost. Setting up the dialogue manager strategy we have to take into account several factors including the dialogue flow, the confirmation policy, the amount of data per turn, the helping features and the language generation.

4.1. Dialogue flow

The dialogue is made up of turns in which typically a question, as clear as possible, is made to the user guiding him/her to the kind of answer expected by the system in each turn. The scope of the turn is defined in such a way that there is no possible ambiguity in the user utterance. Once the user has answered, the system processes the input and, if succeeded, initiates the next turn that can contain another question or a message to the user. If the system is unable to process the input, then the helpling policy is activated as will be explained later. As it is seen, the automatic system takes the initiative guiding the dialogue, but it lets the user to answer with a great range of syntactic possibilities. This range covers all the usual expressions in Catalan language for each turn. In the case of using an unknown expression for a given turn, the system is able to ignore the expression and parse the input as long as a key word (e.g. temperature, a time, etc.) has been recognized. So, the semantic interpretation is made by trying to match the parser's output to semantic templates, and, if this fails, by working directly on the parser's output looking for key words. An important feature of the system is the fact that it lets the user to interrupt the system's messages at any time in order to ask for help or repetition or even to answer a question before it is finished. This last property, called *barge-in*, strengthens the fluency, naturalness and speed of the dialogue by shortening turns when a mistake is done or when the system faces an expert user who knows what the system is going to say, just listening to the beginning of the message. Taking into account all this factors, the flowchart of the dialogue was designed in order to improve naturalness, as can be seen in Fig. 3.

4.2. Confirmation policy

Every utterance got from the user must be confirmed in order to be sure that the system successfully understood what the user was saying. This is done by means of two different confirmation policies: explicit and implicit. Explicit confirmations are used to confirm either critical information or a big set of data by means of a direct question to the user,

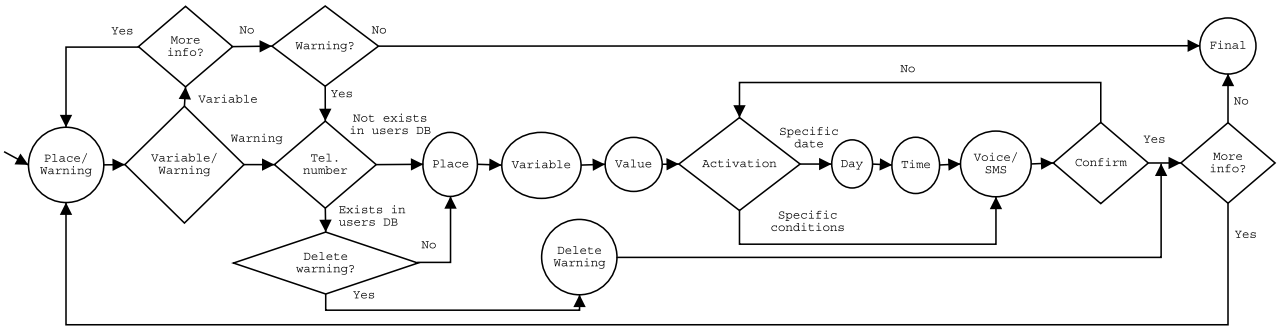


Fig. 3 Dialogue flow

asking him/her whether the last information acquired was correct or not. Implicit confirmations can be used to confirm every data got from the user, the data appear by means of a short sentence at the beginning of the next question and thus without penalizing the fluency of the dialogue with an extra question.

In this work we mixed both policies, using implicit confirmations as a general rule, while explicit confirmations were used to confirm critical information or the final step of a whole process. Giving as a result not only an effective interface, but also a natural dialogue flow between the human and the system. Examples of both policies can be seen in Table 1.

4.3. Amount of data per turn

The amount of data captured by the system in one turn is closely related, on the one hand, to the user's predisposition and naturalness in giving more than one data in the same turn, and on the other hand, to the complexity of the task associated with each turn. We kept the balance between both aspects by recognizing more than one data only in turns that did not involve a high complexity task, like the one of getting the temperatures between the user wants to be notified, but isolating hard tasks, such as the one that gets the municipality, in order to improve its recognition success.

4.4. Helping features

The helping features have been focused on two points. Firstly, a set of Catalan key words/expressions, as it can be seen in Table 2, has been defined in order to help the user to interact with the system. This key words can be accessed by saying the exact word or just an expression containing a key word (e.g. "Help", "Help me, please", "I need some help"). Secondly, an automatically adaptable helping message policy has been set up.

During the preliminary evaluation of the system, we detected that the users missed more messages telling them what they could do in every moment. However, repeating in each turn helping information involves penalizing the fluency of the dialogue because we could substantially increase the time spent in most dialogues. So we decided to take an intermediate solution. In the beginning, the system provides an initial message where welcomes the user, introduces itself

Table 1 Confirmation policies used

Implicit confirmations	Explicit confirmations
User: I am interested in Barcelona. System: About Barcelona, which information do you want?.	System: Do you want to delete this warning? User: Yes.
User: 669 516 158. System: To the 669... about which Catalan place do you want...?	System: ok, you will be notified with an SMS to the 669... is this correct?. User: Yes.

Table 2 Catalan key words/expressions

Key word or expression (Translated to English)	What the user needs	What the system does when detects the situation
"I need help", "Help"...	Help	Throws a helping message
"Could you repeat?", "Repeat"...	Repetition	Throws the last message
"I want to correct", "Correction"...	Correction	Asks for the last value given

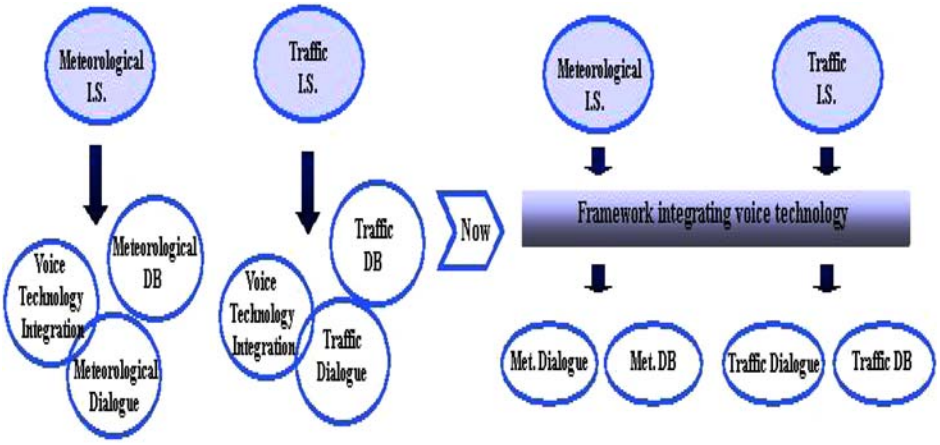


Fig. 4 Developing IS without and with voice framework

and notices the user about the key words/expressions and its use: *"Welcome to the Catalan government meteorological warning and information system. You can always say help, correction, repetition or exit*". During the dialogue, an automatically adaptable help service only provides help when the system detects the user did not understand something or has problems to continue with the dialogue. This help service is activated when one of the following three different situations is detected: the user keeps quiet, the user says something that is not understood by the system, or the user explicitly asks for help.

4.5. Language generation

The language generation has been done using templates that are dynamically filled with the adequate data to originate the final messages reproduced to the user. An example of template is "In the meteorological station of [place] the [variable] at [hour] is [value] [measure]" which, once filled, can result for instance in *"In the meteorological station of Blanes the temperature at 8:30 is 20 degrees"*.

However, in order to introduce variability and more naturalness in the system's answer, the history of the dialogue and a set of different templates with the same meaning is kept. So in each turn of the dialogue the system gives its answers using different templates for the same kind of information. In addition, the system is ready to include a complete text generation module [17] designed for aTTemps, and based on linguistic components. This text generation module is already integrated in the non VoiceXML-oriented system development.

4.6. Implementation

Structurally the dialogue has been built by means of a set of VoiceXML dialogue documents, stored in a web server, that contained each one a differentiated part of the dialogue. For example, the initial document gives the welcome message, gets the first utterances from the user and invokes the document containing the part of the dialogue that fits the user needs. This VoiceXML document has been included in the Appendix.

As we stated before, we decided to make the development of the dialogue manager as independent as possible from the managing of the speech technology in order to clarify the dialogue manager, and also to build a dialogue independent platform that can be easily reused by dialogue managers for other domains than the meteorological one. So we built a framework for dialogue managers and the dialogue manager itself based on the VoiceXML language, enabling telephony applications to be developed in an open-standards based environment. In Fig. 4 it is shown how the portable framework among different domains application changes the development of this kind of information systems (IS). Without the voice technology integration framework, both the voice technology and the dialogue manager development had to be mixed. While now, with the framework, we can develop the dialogue manager much more independently. For a deeper description of the framework see [18].

5. The language processing

As we stated in previous sections, Ibervox has been used as the text-to-speech (TTS) and automatic speech recognition (ASR) engine in the Catalan language. The ASR engine is based on Hidden Markov Models, uses non-stochastic grammars and filling words, and for every utterance provides a confidence recognition measure. When this measure reaches over an empirically tuned value, the utterance is validated (for more details on how to get this value see [19]). A major issue here was the grammar management, which involved coordination not only between the VoiceXML interpreter and the ASR engine, but also between the ASR and TTS engine to support the *barge-in* property.

The ASR grammars have been developed following the Augmented Bacus-Naur Form (ABNF) [20] that is used to specify languages, protocols and text formats. All grammars have been designed in order to give the user as much flexibility as possible. A grammar example, inside a VoiceXML document, can be seen in the Appendix.

Regarding to the interpreter and ASR engine coordination, we found that the syntactic parser offered by the interpreter only admits word lists, that is, with no syntactic structure. That is totally insufficient if real applications are going to be implemented over this framework. So an ABNF parser was implemented and coordinated with the ASR engine in order to accept a wider and richer range of grammatical constructions from the user.

Concerning the ASR and TTS coordination, in order to give full support to *barge-in* property, two different reproduction policies were applied depending on the property activation:

1. With *barge-in* activated, speech reproduction and recognition must be thrown simultaneously. This means that grammars must be compiled by the ASR engine 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 which are not cut by any play action. Therefore 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* defused, taking advantage of the TTS asynchronous capabilities, prompts are reproduced as soon as they arrive to the buffer in order to avoid uncomfortable and unnecessary silences caused by big grammar processing or long calculus performance. Therefore the system can go on while reproducing a message.

   As a rule, we always maintain *barge-in* activated except when an explicit confirmation is required from the user.

6. Evaluation

Two evaluations campaigns were done while developing the system in order to accomplish three general objectives: to assess the effectiveness of the user-machine communication, to evaluate the global performance of the system, and to identify the possible lacks of the system that may have kept unnoticed to the people involved in the project. These evaluations became a part of the development process that helped to improve the system having into account the user opinion.

Each evaluation consisted in giving user satisfaction surveys to the users, who answered filling a form on a web site. We decided to carry out an opinion poll letting two different groups of users interact with the system and polling them afterwards in order to determine which were the lacks of the system. The first group was made up of 17 colleagues not involved directly in this work; and the second one, with general public (19 persons). In the web site, the users were asked to specify the degree in which they agree (from 1 for "completely disagree" to 6 for "completely agree") with 14 statements about the system. The form was divided into two parts: a mandatory one, with 6 statements; and an optional one, with 8 more statements. These statements were oriented to determine whether the speech recognition and the text-to-speech component were working properly, the flow of the dialogue and the behaviour of the system were natural and predictable, the user knew in every moment the actions that he/she could take, the help provided was useful, the repair strategy was useful and easy, and which were the parts of the dialogue where the user had a higher difficulty to success. In this preliminary evaluation we obtained an average of 3.68 and 3.43 (over 6) for each evaluation campaign. These results are good enough taking into account that have been obtained while developing the system.

7. Conclusions and future work

We have described a working automatic information system that provides two different services: on one hand, to obtain information about real-time and personalized meteorological data, and on the other hand, to manage a meteorological warning service. Both services are accessed through a natural language interface, in the Catalan language, and help is provided when needed.

The main advantages of the architecture of our system are: it separates the dialogue control, not only from the managing of the voice technology but also from the application logic, it provides a portable dialogue manager among different voice technologies, it provides an easily changeable interface, and it provides a portable voice framework among different domains application. All this occurs as a result of the VoiceXML standard based design and its architecture that clearly encapsulates every functional area in an independent component empowering the system reusability.

There are some points that can be improved. Firstly, the grammars can be enriched to recognize a greater range of utterances from the user. Secondly, regarding extensibility, system messages and grammars can be translated to other languages in order to incorporate speech tools in other languages satisfying tourists needs. And finally, the text generation module [17] should be included to introduce more naturalness in the system answer.

Acknowledgments This project has been developed in collaboration with the Catalan Meteorological Service, Mensatec and Atlas companies, and the Phonetic Group of the Department of Filología Española at the Autonomous University of Barcelona. In the complete development of aTTemps system have actively participated Febrer A., Rodríguez H., Bonafonte A., Mariño J.B., Nadeu C. and Fonollosa J.A.R.

Appendix

This is the VoiceXML application main document which welcomes the user, asks for some data and redirects the execution to the appropriate auxiliary documents or cgi's. Default treatment for events are defined at the beginning of the document, and can be overlapped with the ones defined in each field. Every field has one or more grammar associated which can be inline or external.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml version="1.0">
  <property name="bargein" value="false"/>

  <link event="correct">
    <grammar type="application/grammar"
      src="http://localhost/gramatics/correct.grm"/>
  </link>
  <link event="help">
    <grammar type="application/grammar"
      src="http://localhost/gramatics/help.grm"/>
  </link>
  <link event="repetition">
    <grammar type="application/grammar"
      src="http://localhost/gramatics/repetition.grm"/>
  </link>

  <catch event="correction">
    <prompt>Correction not available at this part of
     the dialogue.</prompt>
    <reprompt/>
  </catch>
  <catch event="help">
    <prompt> Say help if you need more information
          about a turn, say repetition to hear again
          the last turn or say correction to modify last turn.
           </prompt>
    <reprompt/>
  </catch>
  <catch event="repetition">
    <reprompt/>
  </catch>
  <catch event="nomatch">
    <prompt>Can you repeat, please?</prompt>
    <reprompt/>
  </catch>
  <catch event="nomatch" count="2">
    <prompt>Sorry, I can not understand you. Can you
      repeat, please?</prompt>
    <reprompt/>
  </catch>
    <catch event="noinput">
      <prompt>Sorry, I didn't hear you. You should
           answer the question.</prompt>
      <reprompt/>
    </catch>

   <form id="aTTemps">

     <prompt> Welcome to the aTTemps meteorological
           warning and information system. </prompt>
    <field name="place">

     <catch event="help">
      <prompt> You should say the whole name of the
          Catalan place from which you want information.
       For example: "Sant Feliu de Guixols". Or say
       "warning" to establish a warning. </prompt>
       <reprompt/>
     </catch>

     <prompt> From which Catalan place do you
          want information <break msecs="100"/> or
do     you want to establish a warning? </prompt>

     <grammartype="application/grammar"
      src="http://localhost/gramatics/place.grm"/>
     <filled>
      <if cond="place == warning'">
      <assign name="variable" expr="false"/>
      <else/>  <goto nextitem="variable"/>
      </if>
     </filled>
    </field>

    <field name="variable">

     <catch event="correct">
      <clear namelist="place"/>
      <reprompt/>
     </catch>
     <catch event="help">
      <prompt> You should say the meteorological
       variable <break msecs="100"/> in which you are
       interested. For example: "temperature".</prompt>
```

```
      <reprompt/>
    </catch>

    <prompt>About <value expr="place"/>, which
      information do you need?.</prompt>]
    <grammar>
      #ABNF 1.0 ISO8859-1;
      $SIL = "[SIL] " {} | "[H]" {};
      root $variables = $SIL* ([$filler_1] |
      la_direcció_del_vent {wind's direction}
          | la_velocitat_del_vent {wind's speed} |
      l_alcada_de_les_onades {waves' height}
          | la_temperatura {the temperature} |
      les_precipitacions {the rainfalls}
          | l_estat_de_la_mar {sea state} |
      la_mar_de_fons {groundswell} ) | $SIL)

$SIL*;
      $filler_1 = (vull {want} | voldria {would like})
        ( (saber {to know} | coneixer{to know})
     ([quina_es {which is}] | [quines_son{which are}]) |
        informació {information}
        [(sobre{about} | de{from})]) | de {from};
    </grammar>
    <filled>
      <submit
       next="http://localhost/cgi-bin/cgi_Acimet"
       method="get" namelist="place variable"/>
     </filled>

    </field>

    <field name="warning">

    <catch event="correction">
     <clear namelist="variable"/>
     <clear namelist="place"/>
     <reprompt/>
    </catch>
    <catch event="help">
     <prompt> You should say the complete telephone
          number <break msecs="100"/> to which
       you want to be notified. You can also dial the
          number on the keyboard.</prompt>
     <reprompt/>
    </catch>

    <prompt>Say or dial digit by digit the telephone
      number to which you want to be notified.</prompt>

      <grammar type="application/grammar"
     src="http://localhost/gramatics/telf_dtmf.grm"/>
    <grammar type="application/grammar"
      src="http://localhost/gramatics/telf_voice.grm"/>
    <filled>
     <submit next="http://localhost/cgi-bin/
          cgi_Acimet_warnings"
       method="get" namelist="warning"/>
    </filled>

    </field>
   </form>
  </vxml>]
```

References

1. TRINDI project: http://www.linglink.lu/le/projects/trindi
2. Lamel L, Rosset S, Gauvain JL, Bennacef S (1999) The limsi arise system for train travel information. In: Proc. of ICASSP, pp. 501–504
3. Allen JF, Miller BW, Ringger EK, Sikorski T (1996) A robust system for natural spoken dialogue. In Proc. of ACL, pp. 62–70
4. Álvarez J, Arranz V, Castell N, Civit M (2001) Linguistic and logical tools for an advanced interactive speech system in spanish. In: Proc. Int. Conf. IEA/AIE, LNAI 2070, pp. 519–528.
5. Cohen M, Rivlin Z, Bratt H (1995) Speech recognition in the ATIS domain using multiple knowledge sources. In Proc. of the ARPA Spoken Language Systems Technology Workshop, Austin, Texas, pp. 257–260
6. Zue V, Seneff S, Glass JR, Polifroni J, Pao C, Hazen TJ, Hetherington L (2000) Jupiter: A telephone-based conversational interface for weather information. IEEE Trans. on Speech and Audio Processing, 8(1) 85–96
7. W3C, Voice eXtensible Markup Language (VoiceXML) version 2.0, Feb. 2003. http://www.w3.org/TR/voicexml20
8. INSPIRE project: http://www.inspire-project.org
9. FAZ.NET project: http://www.hltcentral.org/page-1058.shtml
10. Padrell J, Hernando J, aTTemps access to meteorological information by telephone. In: Proc. of ICSLP, Denver, EEUU, vol. 4 pp. 2713–2716
11. Servei Meteorològic Català. http://smc.gencat.es
12. Villarejo Muñoz L, (2002)Gestor de diálogo de un sistema de información meteorológica, in Spanish, Master Thesis in Informatics Engineering, FIB
13. Eberman B, Carter, Meyer D, Goddeau D (2002) Building VoiceXML J. Browsers with OpenVXI www2002.org/CDROM/refereed/260/index.html
14. SpeechWorks, provider of over-the-telephone automated speech recognition solutions. http://www.speechworks.com
15. Dialogic, supplier of computer telephony products. http://www.dialogic.com
16. Ibervox from Atlas-CTI http://www.atlas-cti.com/es/ibervoxasr.htm
17. García Zorrilla P (2002) Sistema d'accés en llenguatge natural a informació meteorològica, in Spanish, Master Thesis in Informatics Engineering, FIB
18. Villarejo L, Hernando J, Castell N (2003) VoiceXML in a real automatic meteorological information system. Berlin XML Days. Berlin, Germany
19. Hernando J, Padrell J, Bonafonte A, Castell N, Mariño JB, Nadeu C, Fonollosa JAR, Rodríguez H, Abad A, Villarejo L, aTTemps: A meteorological information service through the telephone networ, U.P.C. Politechnical University of Cataluña, Spain, Internal Report
20. http://www.w3.org/TR/speech-grammar/