# THE DEVELOPMENT OF A MULTI-PURPOSE SPOKEN DIALOGUE SYSTEM

**João P. Neto** **Nuno J. Mamede** **Renato Cassaca** **Luís C. Oliveira**

Instituto Superior Técnico / INESC ID Lisboa
L²F - Spoken Language Systems Laboratory
INESC ID Lisboa, R. Alves Redol, 9, 1000-029 Lisboa, Portugal

{Joao.Neto, Nuno.Mamede, Renato.Cassaca, Luis.Oliveira}@l2f.inesc-id.pt
**http://l2f.inesc-id.pt**

## Abstract

In this paper we describe a multi-purpose Spoken Dialogue System platform associated with two distinct applications as an home intelligent environment and remote access to information databases. These applications differ substantially on contents and possible uses but gives us the chance to develop a platform where we were able to represent diverse services to be accessible by a spoken interface. The implemented voice input/output possibilities and the service independence level opens a wide range of possibilities for the development of new applications using the current components of our Spoken Dialogue System.

## 1. Introduction

Only recently the Spoken Dialogue Systems (SDS) started emerging as a practical alternative for a conversational computer interface, mainly due to the progress in the technology of speech recognition and understanding [1]. They are more effective than an IVR system since they allow a more free and natural interaction and can be combined with other input modalities and visual output.

In the last few years in our lab we have been building different SDS tailored to very specific applications. We started by developing an application for voiced-based interactive control of a set of home appliances as lights, air conditioning and hi-fi, creating an intelligent home environment. Later on, this application was extended to also include spoken access to an e-mail server, which enabled a user to navigate through his/her own messages using voice and also listen to them. In parallel, other applications were developed in quite different domains such as the remote access to databases with weather information [2], cinema schedules and bus trip information [3].

Despite the differences between all these applications, namely in terms of the type of inputs, our systems were built using an architecture with a set of common features. This experience led us to develop a multi-purpose SDS platform that is actually being used as a show case for different applications and demonstrations by our group.

This platform makes use of a generic Audio Manager where different pairs of input/output devices can be plugged-in: table, head mounted or wireless microphones, standard audio speakers or headphones, fixed or GSM telephones or PDA devices.

On the other end of the platform, a generic device representation was created using an XML description to be used by the Service Manager. This description can be used to represent devices of very different types like a light control module or an e-mail messaging service.

Hopefully, the generalization level at the input of the system will make these services available to anyone, anywhere at anytime.

In the future, we plan to incorporate additional modalities and to port the system or, at least, part of it to hand held devices, by creating small footprint versions of the different modules.

This paper is divided into five sections. The next section describes the structure of the overall platform. Its central block, the spoken dialogue system itself, is described in detail in Section 3. The following section is a brief discussion on the advantages of device independency achieved by this architecture, and in Section 5 some concluding remarks.

## 2. System Structure

This section describes the overall structure of our platform which is divided into 5 main blocks: Input/Output, Audio Manager, Spoken Dialogue System, Service Manager and Services. The next subsections describe each block.

### 2.1. Input / Output

One of our main objectives was to create a platform available to anyone, anywhere at anytime. To accomplish this goal we need different input/output facilities, adaptable to the different possible uses of the system. In our system we have available a set of different audio interfaces:

- head or table mounted microphone with speakers;
- lapel mounted wireless microphone with speakers;
- fixed telephone;
- GSM telephone;
- PDA using the built in mic and speakers;
- Bluetooth mic to a direct access to the system or through GSM telephone or PDA.

In the near future, we plan to incorporate also VoIP facilities in order to make the system available from the Web. These different accessibilities could imply complete different uses of the system according to the specific application.

### 2.2. Audio Manager

Figure 1 shows the positioning of the Audio Manager as an interface between the input/output devices and the Spoken Dialogue System itself.

The Audio Manager connects the different input/output devices with the ASR and TTS in the Spoken Dialogue System. With this layer we create an independency between the specific devices and the processing that will be done on the speech signal.

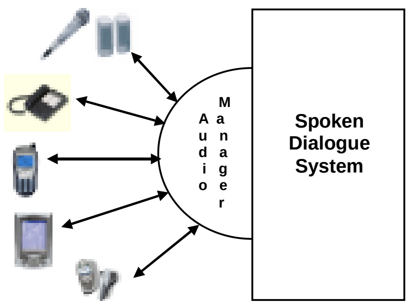

*Figure 1*: Functional diagram of input/output connections to the SDS through the Audio Manager.

### 2.3. Spoken Dialogue System (SDS)

Figure 2 presents a general block diagram of our implementation of the SDS.

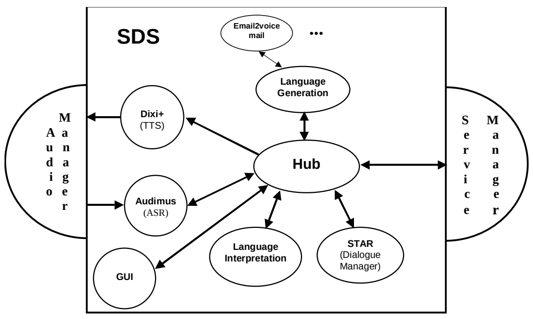

*Figure 2*: Block diagram of SDS.

This system connects the Audio Manager, from where it receives the speech input and generates speech output, to the Service Manager, where the requests from the user are executed.

This system is based on a hub framework similar to the Galaxy II architecture. The speech input coming from the Audio Manager is passed to the *Audimus* [4] recognizer, whose output is transferred to the Language Interpretation module, responsible for transforming the recognized text into speech acts [5]. The speech acts are interpreted in the STAR [3] Dialogue Manager. When the dialogue manager completes filling up all the necessary information about the requested service, it sends a demand to the Service Manager, which is responsible for the service execution. If the service outputs a message to the user, that information is dully formatted in the Language Generation block and synthesized through the DIXI+ [6] text-to-speech system. The generated audio is then played to the user, using the Audio Manager and the active output device. The individual components of the SDS are described in detail in section 3.

### 2.4. Service Manager (SM)

This block establish the connection between the SDS and the services and/or devices representing the specific applications, as depicted in Figure 3.

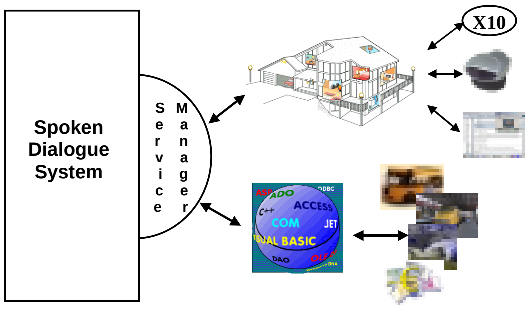

*Figure 3*: View of the Service Manager as the interface between the SDS and the real services.

Since we want to use the same SDS across different applications we need to create a representation level that guarantees the independency of the SDS relative to the services. This question is discussed in more detail in section 4.

### 2.5. Services

The flexibility of the multi-purpose platform can be illustrated by its application to two significantly different types of service: in a home environment, and on database retrieval. In the first, based on X10 and IRDA protocols, we control a wide range of home devices, such as lights, air conditioning, hi-fi, TV, etc.. We can extend the application to include any infra-red controllable device or whose control functions may be programmed by the X10 protocol.

The second type of application, information retrieval via voice from remote databases, has been tested with weather information, cinema schedules and bus trip information. This type of application can easily be extended to other different domains.

## 3. Spoken Dialogue System (SDS)

In Figure 2, we sketch the block diagram of our SDS implementation. In this section we will describe in detail each block.

### 3.1. *Audimus* recognizer

*Audimus* [4] is a hybrid speech recognizer that combines the temporal modeling capabilities of Hidden Markov Models (HMMs) with the pattern discriminative classification capabilities of multilayer perceptrons (MLPs). This same recognizer is being used for different complexity tasks based on a common structure but with different components.

The acoustic models are dependent on the input facilities since we are using separated models for telephone speech or for microphone speech. The same occurs with the lexical and the language models which are dependent on the specific application domain. Since we can have several subsequent uses of the system, we built a pool of components for each appropriate model that *Audimus* makes active according to the needs. These models can be associated to single domain or multiple domains.

The domains where we are applying our system requires different capabilities from the ASR. There are simple actions, as turn on the lights, but there are domains, as the e-mail, where is necessary a large vocabulary and language model and adapted acoustic models.

Having different components implies the availability of material (spoken and/or written) collected for that domain. *Audimus* is based on the same type of context-independent acoustic models of phone-like units, independently of the task complexity. That is, even simple tasks that could be addressed using whole word models are handled via generic sub-word models.

### 3.2. Language Interpretation

This module is responsible for extracting the intentions of the user's utterances. The text version of the user's utterance is sent through a process pipeline. The module first lemmatises the text using an external dictionary. The resulting text is then passed to a post-morphological processor that detects and forms special groups according to recomposition and correspondence rules. The text is sent to a syntactic analyser that, using a surface grammar, groups the phrase constituents. The last process, the Speech Act Finder (SAF) extracts the speech acts candidate list, which will be sent as the output of the Language Interpretation module. SAF uses the services of the Service Manager to assign a meaning to each object referred to in speech acts.

We are only recognizing "forward-looking acts" (assertions, re-assertions, offer, commit, opening, close), and "backward-looking acts" (accept, accept part, reject-part, reject, signal-non-understanding, signal-understanding, acknowledge, repeat-rephrase, completion, correct-misspeaking). As in [7] we use decision trees to identify the speech acts.

The structure of the speech acts handled by our system is described in Table 1.

*Table 1*: Speech Act Structure.

| | |
|---|---|
| Type | The speech act type |
| Subject | The author of the utterance |
| Object | The objects referred within the speech act, and their role on the domain |
| Start | The objects that may identify a departure place (optional) |
| Destination | The objects that may identify an arrival place (optional) |
| Start Time | The objects that may identify the starting time of an action (optional) |
| End Time | The objects that may identify the ending time of an action (optional) |
| Action | The action implicitly associated with the speech act (optional) |
| Quantity | The object quantifiers or the number of times an action must be executed (optional) |
| Optional | The objects that may identify the start time of an action (optional) |

### 3.3 STAR Dialogue Manager

Our dialogue manager is able to handle multiple domains simultaneously, and consists of three main modules: Interpretation Manager (IM), Discourse Context (DC) and Behavioural Agent (BA).

We use frames to represent both the domain and the information collected during the interaction with the users [3]. Each domain handled by the dialogue system is internally represented by a frame, which is composed by slots and rules. Slots define domain data relationships, and rules define the system's behaviour. Rules are composed by operators (logical, conditional, and relational) and by functions.

To keep the filling of the frame slots consistent, it is necessary to indicate the set of values with which a slot can be instantiated. To avoid invalid combination of values, we have defined a meta-language to express the constraints that must be satisfied. So each frame definition includes a set of RECOGNITION_RULES, used to specify the set of values that each slot may hold; a set of VALIDATION_RULES, to express a set of domain restrictions, i.e., invalid combination of slot values; and CLASSIFICATION_RULES, used to specify the actions that must be performed when some conditions are satisfied, i.e., the valid combinations of values. This approach has the advantage of making the dialogue control independent of the domain.

The IM receives a set of speech acts and generates the correspondent interpretations and discourse obligations. Interpretations are frame instantiations that represent possible combinations of speech acts and the meaning associated to each object it contains. To select the most promising interpretation two scores are computed The recognition score to evaluate the rule requirements already accomplished, and the answer score, a measure of the consistency of the data already provided by the user. A more detailed description of this process can be found in [8].

The DC manages all knowledge about the discourse, including the discourse stack, turn-taking information, and discourse obligations.

The BA enables the system to be mixed-initiative: regardless of what the user says, the BA has its own priorities and intentions. When a new speech act includes objects belonging to a domain that is not being considered, the BA assumes the user wants to introduce a new dialog topic: the old topic is put on hold, and priority is given to new topic. Whenever the system recognizes that the user is changing domains, it first verifies if some previous conversation has already taken place.

### 3.4. Language Generation

This module coordinates the generation activities, having to find the best way to express what the Behavioral Agent has decided. For the moment, we are using pre-defined templates with blanks to produce a natural answer from prewritten questions and response templates.

In this module we also plug in an email-to-voicemail module where we were able to enrich an email message through a set of tags.

### 3.5. DIXI+ text-to-speech

Our TTS module (DIXI+) [6] is a concatenative-based synthesizer, based on the Festival framework. This framework supports several voices and two different types of unit - fixed length units (such as diphones), and variable length units. This latter data-driven approach can be fine tuned to a limited domain of application, by adequate design of the corpus.

This is the case of weather information, since we have a limited number of possibilities in terms of information to supply to the user, but completely different from an e-mail, where the possibilities of message subjects are very large.

A significant effort has been recently devoted to the development of a small footprint version of DIXI+, based on Flite.

### 3.6. Graphic User Interface (GUI)

The GUI serves as an administration tool to the system. It can be used to monitorize all the messages in the hub and extract a log with all the necessary information to evaluate the behavior of the system and of its sub-blocks. It can also be used to install new services and control existing ones.

## 4. Device independency

We designed the Service Manager (SM) as the interface between the spoken dialogue platform and a set of heterogeneous devices. As we are working with different types of devices there is a need for a representation that gives us a uniform access to all of them.

The SM is the only block with a direct interaction with the devices. The other blocks on the SDS only have access to a description of the set of available services. The SM is responsible for the mapping between the services and the corresponding device.

The SM was built from three main blocks:

- Connection to the hub to receive the requests and send the answers to the SDS.
- A processing stage for the management of the description structure associated to the devices (processing of queries to the device functionalities and the generation of the most probable hypothesis to satisfy the service).
- Management of the communication with each of the devices routing the requests and answers.

The representation of the services available from each device is based on an XML description, according to a generic DTD, mapping the device functions to be executed for each service. DOM trees, resulting from the XML processing, are used for the internal representation of each device. A device has more than one state associated, and at a given instant it is only possible to have access to the hierarchy of services under the actual state of the device. The device is responsible to update its state on the SM. When the device executes a service that results in a new state it should inform the SM of that new state. Based on that update a new set of services corresponding to that new state may become available.

The addition of a new service is very simple, being only necessary the XML description of services and the piece of code to be executed, a kind of device driver. The installation of the service on the system occurs through administration facilities available from the GUI.

## 5. Concluding remarks

The work reported in this paper results from an integration of several components being developed in our lab. This system is the result of a collaborative effort between people working in the different technologies, and it became a common platform where the different components can be associated to produce multiple applications.

This system is currently being used in our “intelligent” demonstration room where a set of appliances and services can be controlled by voice. It is also being used on a demonstration system, accessible by telephone, where people can ask for weather conditions, stock hold information, bus trip information and cinema schedules.

We expect to conduct some formal evaluation of our system in order to quantify the user satisfaction.

## Acknowledgements

This work was partially funded by FCT project POSI/33846/PLP/2000. INESC ID Lisboa had support from the POSI Program of the "Quadro Comunitário de Apoio III".

## References

[1] M. McTear, "Spoken Dialogue Technology: Enabling the Conversational User Interface", in ACM Computing Surveys, pp. 1-85, 2002.

[2] P. Cardoso, L. Flores, T. Langlois and J. Neto, "Meteo: a telephone-based Portuguese conversation system in Weather domain", in Proc. PORTAL2002, Portugal, 2002.

[3] P. Madeira, M. Mourão, N. Mamede, "STAR Frames - A step ahead in the design of conversational systems capable of handling multiple domains", in Proc. ICEIS, Angers, France, 2003.

[4] J. Neto, C. Martins, H. Meinedo and L. Almeida, "AUDIMUS - Sistema de reconhecimento de fala contínua para o Português Europeu", in Proc. PROPOR IV, Évora, Portugal, 1999.

[5] J. Allen, G. Ferguson, A. Stent, "An architecture for more realistic conversational systems", in Proc. of Intelligent User Interfaces (IUI-01), Santa Fe, USA, 2001.

[6] S. Paulo and L. Oliveira, "Multilevel Annotation Of Speech Signals Using Weighted Finite State Transducers", in Proc. 2002 IEEE Workshop on Speech Synthesis, Santa Monica, USA, 2002.

[7] M.Poesio and D.Traum, "Towards an axiomatization of dialogue acts", in Proc. of TWENDIAL, Twente, 1998.

[8] M. Mourão, P. Madeira, N. Mamede, "Interpretations and Discourse Obligations in a Dialog System", in Proc. Propor’2003, Faro, Portugal, 2003.