Integration of Dialogue Moves and Speech Recognition in a Telephone Scenario

José F. Quesada, J. Gabriel Amores, Rafael Ballesteros
{jquesada,jgabriel}@cica.es; bono@fing.cica.es
Universidad de Sevilla

Abstract Spoken Dialogue Management systems constitute one of the main research trends in Computational Linguistics. The integration of speech recognition, natural language understanding, dialogue management, natural language generation and speech synthesis poses a crucial challenge to these systems. This paper describes the implementation of a spoken dialogue system based on the Information State Update point of view of dialogues for an automatic telephone operator scenario.[1]

1 *Introduction*

Spoken dialogue systems enable users to interact with computer systems via natural and intelligent dialogues, as they would with human agents. Development of such systems requires a wide range of speech and language technologies, including automatic speech recognition, to convert audio signals of human speech into text strings, natural language and dialogue processing, to determine the meanings and intentions of the recognized utterances and to generate a cooperative response to them, and text-to-speech synthesis, to convert the system utterance into actual speech output.

The central objective of the SIRIDUS project is to expand the understanding of and develop computational tools to support the development of more robust, functional and user-friendly dialogue systems. Within this framework we are designing and implementing an interface in Spanish to a telephone using a Natural Command Language (NCL).

We take Natural Comand Languages as the set of input and output natural language expresions which are acceptable in a given application domain. Among their most remarkable properties are the use of natural language vocabulary and syntactic structure. NCLs reflect natural syntactic order and more precisely the patterns of spoken language. They constitute a sub-language, with reasonably clear boundaries but also one in which new functionality can be added without altering the structural properties of the language.

The Scenario for this natural command dialogue system is a relatively large organizations such as private or public companies, universities, government departments, etc., with a Private Automatic Branch Exchange (PABX) that provides local extensions to the individuals who are part of that institution. These local extensions allow the users to perform certain advanced functions, like transferring incoming calls, conference calls and so on.

In the following sections we will describe the dialogue moves and types within the Automatic Telephone Operator Scenario and the architecture of the system. We will also talk about the meaning extraction strategies that have been implemented and the results obtained using three different speakers. Finally we will draw our conclusions and outline some future lines of work.

2 *Dialogue Moves and Types: Natural Command Languages for an Automatic Telephone Operator Scenario*

2.1 Command-oriented moves

- askCommand: the system requests the user to specify a command or function to be performed.
- specifyCommand: this move takes place when a specific function has been selected. In the telephone scenario, seven

[1]This work has been supported by the European Commision under Project Siridus IST-1999-10516.

different types have been defined as part of the user specifications:

  - Dialling: TelephoneCall. The possible parameters for this type are a Name, a four digit Extension number, or a nine-digit external number.
  - Redialling: Redial. This function has no parameters. The dialogue manager will specify that it must look for a previous Name, Extension or Number.
  - Tranferring Incoming Calls: TransferCall. The possible parameters for this type are a Name, or a four-digit Extension number.
  - Cancellation of Incoming Calls Transfer: CancelTransferCall. This function has no parameters. The dialogue manager will cancel any incoming calls transference associated with the current terminal.
  - Conference Call: ConferenceCall. The possible parameters for this type are two names, two four digit extension numbers, or a name and a four-digit Extension number (in any order).
  - Office Number Look Up by Name: LookUpOffice. The only possible parameter for this type is a name.
  - E-mail Address Look Up by Name: LookUpEMailAddress. The only possible parameter for this type is a name.

- informExecution: actual execution of the command.

2.2 Parameter-oriented Dialogue Moves

- askParameter: The system asks for the value of a specific parameter.
- specifyParameter: This move corresponds to the assignment of some value to one parameter. Possible parameter types are: Name, and Extension.

2.3 Interaction-oriented Dialogue Moves

- askConfirmation
- answerYN
- askContinuation
- askHelp
- answerHelp

3 Architecture of the System

This section describes the hardware involved in the project. It comprises three hardware elements. This system will allow personnel in a large or medium sized company to perform simple and some advanced telephone functions, as well as some information query functions, using their voice over the phone and natural language (in particular commands issued in a natural way). These three components are:

1. A PABX providing:
   - Internal telephone extensions for the people in an institution.
   - External telephone lines for external telephone calls.
2. A PABX Control Unit (PCU) that will control the PABX in order to perform the telephone functions that the system will provide.
3. A Voice Response Unit (VRU) in charge of the user interaction. As this interaction will be held over the phone and using only speech and natural language, this unit will be responsible for:
   - Speech recognition.
   - Speech synthesis.
   - Natural language processing.
   - Dialogue management.

   The Voice Response Unit (VRU) will be in charge of the system–user interaction. This interaction will be held entirely via voice over telephone lines. Therefore, it will need software to perform voice recognition and voice synthesis over the phone.

4 Multi-Agent Distributed Architecture

The software has been divided into agents that play a particular role within the complete system, and that communicate with the others in order to obtain the global functionality.

Each agent is a separate program that communicates with other agents through KQML ASCII text messages. KQML stands for Knowledge Query and Manipulation Language, and is a standardised language for inter-agent communication. Different specification versions of this language can be found in [Finnin *et al* 1993].

This is a centralised agent architecture, in the sense that there is a central agent and all the messaging will go through it. This central agent routes the messages to their destinations, and provides additional services to the rest of the agents integrating the system.

This choice seems to be consistent with the trend of some of the most advanced dialogue systems. For instance TRAINS [Trains Web Page, Ferguson *et al* 1996], TRIPS [Trips Web Page, Allen *et al* 2001], and the GALAXY architecture [Seneff *et al* 1999] make use of multi–agent distributed architectures. Moreover, the architectures they use also make use of a central agent that controls the communications.

The TRAINS system and the TRIPS system also use KQML as the communication language between the agents.

As a result of the adoption of a multi-agent distributed architecture for our this system, the software developed can be described as a set of independent programs or agents that run concurrently in the same or different machines and interchange KQML messages in order to achieve the desired functionality.

The following section describes the set of agents in charge of the natural language processing and the interpretation of the user's input.

5 Dialogue Management Agent Design

From a functional point of view, the overall system can be seen as an interface to a telephone service, where the dialogue manager acts as an automatic telephone operator.

This design is based on the analysis of the domain, user requirements and the illustrative corpus outlined in *User Requirements on a Natural Command Language Dialogue System* [Torre, Amores & Quesada 2000].

From an operational point of view, the design of the dialogue manager is based on and makes an extensive use of the notion of *dialogue move*, and specifically of the set of dialogue moves described in *Dialogue Moves in Natural Command Languages* [Amores & Quesada 2000].

5.1 The Dialogue Management Agent in the Multi Agent-based Architecture

At a high level of abstraction, the dialogue manager may be considered as an agent communicating dynamically with several other agents in the overall architecture of the system (Figure 1), such as:

- The Database Agent, which allows consulting operations in the general (organisation) directory.
- The Input/Output Agent, allowing for a functional abstraction about the specific characteristics of the components in charge of the real input and output. The interface between the dialogue manager and the input/output agents may be seen as an asynchronous channel.
- The PABX Agent, which will actually execute the commands detected and confirmed by the Dialogue Management Agent.
- The Process Manager (or Agents Manager) Agent in charge of the control of the different agents involved.

Figure 2 below shows the relations between these four agents, including the control and data flow of information.

5.2 Main Components of the Dialogue Management Agent

Coming down in the level of abstraction, this section will analyse the main components of the Dialogue Management Agent (DMA). At this level, we will differentiate five modules:

- Speech Prospector,
- Input Analyser (Natural Language Understanding),
- Dialogue Move Selector,
- Dialogue Manager, and
- Output Generator,

two main stores:

- Dialogue Moves Input Pool, and

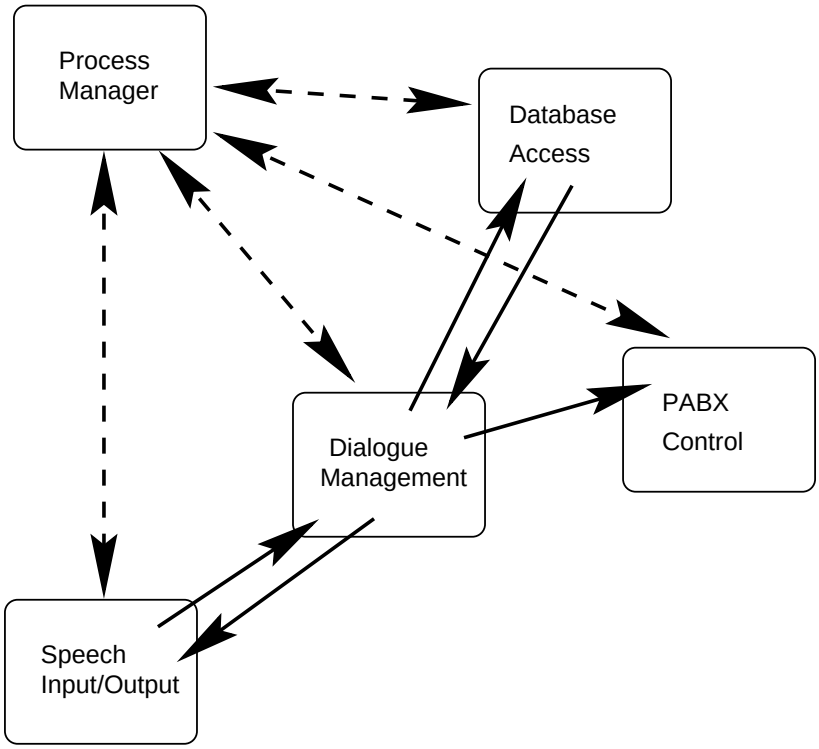

Figure 1: The Dialogue Management Agent in the Multi Agent–based Architecture

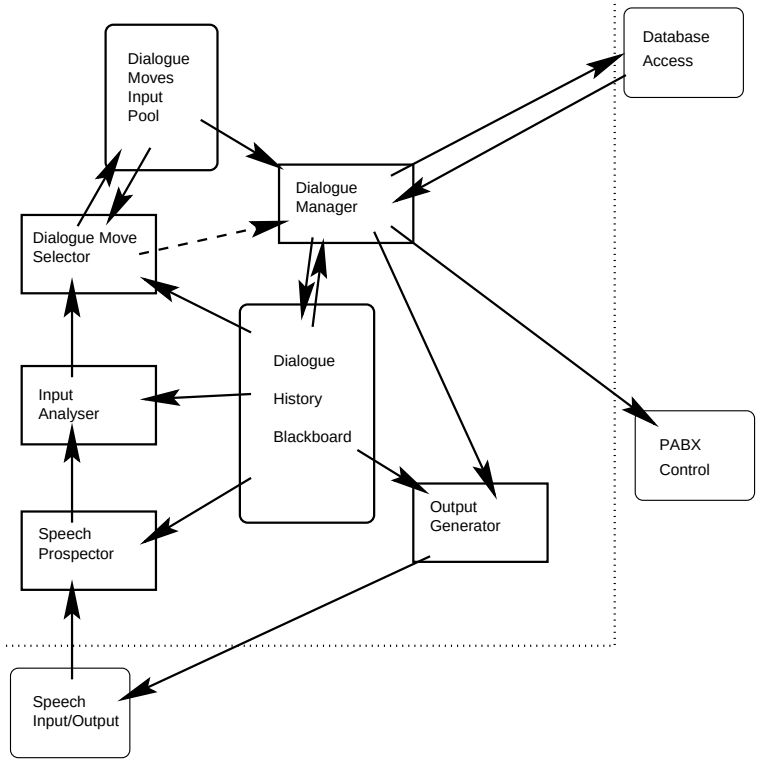

Figure 2: Main Components of the Dialogue Management Agent

- Dialogue History Blackboard,

and an external resource (controlled by the Database Agent):

- Directory and User Database.

6 *From Speech-Recognition to Dialogue Moves*

In order to have a general picture about the functions and relationships of the different components, let's assume that at some point in the man–machine interaction, the user says something (asks a command, replies to a question, etc.).

The user's input itself will be processed by the Input/Output Agent. In the real scenario of the target application, the input module will be a speech recogniser. Therefore, at first glance, the result of the first input analysis should be a string of words containing the textual transcription of the most likely sentence recognised by the speech recogniser.

Nevertheless, one of the main goals is to take advantage, at any stage of the whole dialogue management system, of the integration of the different components. At this moment, the speech recognition task may take advantage of the current informational status of the whole system (last dialogue moves, current active expectations, dialogue history, etc.) and the domain knowledge (set of possible commands and the parameter model for each command, current user's device configuration, special or high priority commands, etc.)

In order to accomplish this functionality, the Dialogue Management Agent includes a Speech Prospector. In the ideal situation, this module will have access to the internal representation of the speech recogniser (word lattice, timing marks, acoustic scores, and so on). As a result of the integration of this information and the dialogue history we expect that the Speech Prospector will improve the semantic adequacy of the string of words obtained from the analysis of the user's input.

A string of words will be the input to the Input Analyser (Natural Language Understanding) module. The goal of this module is to obtain the dialogue move, or the set of dialogue moves in case of a multi–functional or ambiguous input. The architecture proposed is based on a unification–based grammar. That is, the Input Analyser contains a lexico–morphological submodule, a parser and a unifier. They use the DTAC protocol for the representation of the dialogue moves [Amores & Quesada 2000].

The Input Analyser module sends the dialogue moves obtained to the Dialogue Move Selector. This module controls the Dialogue Moves Input Pool. This store contains the ordered set of dialogue moves not yet processed by the Dialogue Manager module.

The Speech Prospector, the Input Analyser and the Dialogue Move Selector modules may be connected using a sequential and synchronous strategy, defining what should be called the User Input Analysis block of the Dialogue Management Agent.

Nevertheless, this Agent has two main barriers of synchronism. The first one appears in the interface between the Dialogue Manager and the Dialogue Move Selector (in fact, between the Dialogue Manager and the User Input Analysis block as a whole). The idea is that the Dialogue Manager may be processing a dialogue move while (just at the very same time) the user makes a new utterance, activating (in parallel to the Dialogue Manager), the Input Agent, and, in a cascade, the Speech Prospector, the Input Analyser, and the Dialogue Move Selector.

So, one of the main goals of the Dialogue Moves Input Pool is to allow this asynchronous and parallel work of the User Input Analysis block of modules and the Dialogue Manager. To allow this, the first function of the Dialogue Move Selector is to store each dialogue move received from the Input Analyser in the Dialogue Moves Input Pool.

The Dialogue Move Selector must also reorder the dialogue moves in the Input Pool according to the priority level of each dialogue move, maintaining a totally ordered set of dialogue moves. The score used for this operation may be statically assigned by the Input Analyser (for instance, if the user requests the cancellation of the current command, this dialogue move should have a very high priority level, in order to be able to interrupt the Dialogue Manager). Also, the Dialogue Move Selector may assign the scores dynamically taking into account the relationships between the different dialogue moves in the pool, the status of the Dialogue Manager and the dialogue history stored in the Dialogue History Blackboard store.

The Dialogue Manager module will get the

top dialogue move in the Dialogue Moves Input Pool (with the highest priority level) and will analyse it using the domain knowledge previously stored as a set of dialogue rules and the dialogue history. During this operation, the Dialogue Manager may use additional information stored in the external Data Base resource (for instance, to consult a directory entry, or to update the last call of the user in order to retry it later).

The processing of the Dialogue Manager may be interrupted by a higher priority dialogue move by a control call from the Dialogue Move Selector.

The successful completion of the Dialogue Manager will generate a dialogue move as a result. That is, the dialogue manager is a function that uses as parameters the dialogue move under processing, the set of dialogue rules, and the dialogue history, and returns the dialogue move that contains the information state corresponding to the integration of the new dialogue move in the dialogue history according to the dialogue rules.

The resulting dialogue move will be stored in the Dialogue History Blackboard. This store marks the second asynchronism in the whole Dialogue Management Agent. The dialogue history contains the time–based ordered set of dialogue moves that defines the evolution of the information state of the Dialogue Management agent. This information is not merely anecdotic but very informative, as it contains the set of open dialogue moves (corresponding to actions triggered by the user but not yet completed) and the previous commands and parameter values, which may be used to solve anaphoric references.

The Dialogue Manager is also connected to the Output Generator, which will have to translate the dialogue move obtained by the Dialogue Manager into a sentence that will be communicated to the user by means of the Output Agent (a Display Agent or a Speech Synthesiser).

6.1 Results' Overview

The following tables illustrate the results obtained over a set of utterances. These utterances were recorded by three different speakers to simulate the telephone user moves which have been implemented in the system. They were passed through the speech recognizer and after through the NLU module.

Tables 1 to 2 below illustrates the initial experiments realised over the first version of the system.

The initial goal of this experiment is to study the robustness of the overall architecture. With this goal in mind, each table contains a set of sample sentences for the corresponding function. Each sentence has been tested with three different speaker. The column Recognizer specified the WER (Word Error Rate) expressed as the number of correct recognised words and the total number of word in the sentence. Finally, the column NLU-DTAC indicates whether the semantic–driven grammatical strategy has been or not successfull, both at the level of command (+/- C) and the level of argument (+/- P).

At this stage, we have focused on a qualitative evaluation of the model. This initial evaluation will help on the tuning of the speech recognizer. Once we have finished this stage, we plan to perform a quantitative and more detailed evaluation.

7 Conclusion

A spoken dialogue system for the automatic telephone operator scenario has been presented. The paper has focused on the main linguistic agents of the overal agent–based architecture specified for the dialogue system. Specifically, we have presented the flow of information through the components: Speech Recogniser, Speech Prospector, Natural Language Understanding module, Dialogue Move Input Pool, Dialogue Manager, Dialogue History Blackboard, Output Generator and Speech Sinthesyser. Some results about accuracy, robustness and semantic coverage of the system has also been presented.

References

[Allen *et al* 2001] Allen, J., Ferguson, G. & Stent, A. 2001. An Architecture for More Realistic Conversational Systems. *Proceedings of Intelligent User Interfaces 2001.* Available on http://www.cs.rochester.edu /research /trips /papers.

[Alvarez *et al* 1997] Alvarez, J., Tapias, D., Crespo, C., Cortázar, I., & Martínez. 1997. Development and Evaluation of the ATOS Spontaneous Speech Conversational System. *International Conference on Acoustics, Speech and Signal Processing*, 1139–1142.

[Amores & Quesada 2000] Amores, J. G., Quesada, J. F. 2000. *Dialogue Moves in Natural Command Languages.* Deliverable 1.1. Siridus project.

[Cooper *et al* 1999] Cooper, R., S. Larsson, C. Matheson, M. Poesio & D. Traum. 1999. *Coding Instructional Dialogue for Information States.* TRINDI (LE4-8314) Deliverable D1.1, February 1999.

[Ferguson *et al* 1996] Ferguson, G., Allen, J.F., Miller, B.W., & Ringger, E.K. 1996 *The Design and Implementation of the TRAINS-96 System: A Prototype Mixed-Initiative Planning Assistant*, TRAINS Technical Note 96-5. Available at http://www.cs.rochester.edu:80 /u /ringger /research /tn96-5.html. University of Rochester, Computer Science.

[Finnin *et al* 1993] Finnin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzon, R., McKay, D., McGuire, J., Pelavin, R., Shapiro, S., & Beck, C. 1993. *Specification of the KQML Agent-Communication Language.* Draft 15 June 1993. Available at http://www.cs.umbc.edu/kqml. University of Maryland Baltimore County.

[Heeman 1997] Heeman, P. A. 1997. *Speech Repairs, Intonational Boundaries and Discourse Markers: Modeling Speakers' Utterances in Spoken Dialog* Ph.D. Dissertation. Department of Computer Science, University of Rochester, New York.

[Lewin *et al* 2000] Lewin, I., Rupp, C. J., Hieronymus, J., Milward, D., Larsson, S., Bermann, A. 2000. *Siridus System Architecture and Interface Report (Baseline)* Deliverable 6.1. Siridus project.

[Noord *et al* 1998] Noord, G. van, G. Bourna, R. Koeling & M. J. Nederhof. 1998. Robust Grammatical Analysis for Spoken Dialogue Systems. *Natural Language Engineering*, 1(1), 1-48.

[Poesio *et al* 1999] Poesio, M., R. Cooper, S. Larsson, D. Traum & C. Matheson. 1999. Annotating Conversations for Information State Updates. In *Proceedings of the Amstelogue´99 Workshop on the Semantics and Pragmatics of Dialogue.* Amsterdam University, May 1999.

[Quesada, Torre & Amores 2000] Quesada, J. F., Torre, D. & J. G. Amores. 2000. *Design of a Natural Command Language Dialogue System* Deliverable 3.2. Siridus project. December 2000.

[Seneff *et al* 1999] Seneff, S., Lau, R., & Polifroni J. 1999. Organization, Communication and Control in the GALAXY-II Conversational System. *Proc. Eurospeech 99*, Budapest, Hungary, September 1999. Available at http://www.sls.lcs.mit.edu /sls /publications /index.html.

[Torre, Amores & Quesada 2000] Torre, D., J. G. Amores & J. F. Quesada. 2000. *User Requirements on a Natural Command Language Dialogue System.* Deliverable 3.1. Siridus project.

[Trains Web Page] TRAINS project web page, http://www.cs.rochester.edu /research /trains. University of Rochester, Computer Science.

[Traum *et al* 1999] Traum, D., Bos, J., Cooper, R., Larsson, S., Lewin, I., Matheson, C., Poesio, M. 1999. *A model of dialogue moves and information state revision.* TRINDI (LE4-8314) Deliverable D2.1, November 1999.

[Trips Web Page] TRIPS project home page, http://www.cs.rochester.edu /research /trips. University of Rochester, Computer Science.

Function	Sentence	Speaker	Recognizer	NLU-DTAC	
TelephoneCall	1	1	2/4	+C -P	Incomplete parameter
		2	1/4	+C -P	Wrong parameter
		3	3/4	+C -P	Incomplete parameter
	2	1	6/6	+C +P	OK
		2	0/6	+C -P	Wrong parameter
		3	6/6	+C +P	OK
	3	1	6/6	+C +P	OK
		2	0/6	+C +P	OK
		3	6/6	+C +P	OK
	4	1	3/3	+C +P	OK
		2	2/3	+C +P	OK
		3	3/3	+C +P	OK
	5	1	8/8	+C +P	OK
		2	5/8	-C +P	No specify command
		3	8/8	+C +P	OK

1. llama a Agustín Díaz
2. llama al seis ocho seis dos
3. desearía hablar con el señor Díaz
4. al señor Ballesteros
5. ponme con el noventaicinco cinco sesentaiuno cuarentaidós trentaiocho

Table 1: Telephone Call Function

Function	Sentence	Speaker	Recognizer	NLU-DTAC	
Redial	1	1	1/3	+C +P	OK
		2	1/3	+C +P	OK
		3	1/3	-C -P	Wrong specify command
	2	1	3/3	+C +P	OK
		2	1/3	-C -P	Wrong specify command
		3	3/3	+C +P	OK
	3	1	2/3	+C +P	OK
		2	2/3	+C +P	OK
		3	2/3	+C +P	OK
	4	1	2/3	+C +P	OK
		2	2/3	+C +P	OK
		3	2/3	+C +P	OK

1. vuelve a llamar
2. llama otra vez
3. repite la llamada
4. inténtalo de nuevo

Table 2: Redial Function