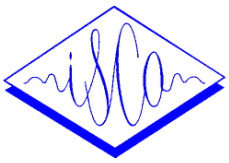

3rd European Conference on
Speech Communication and Technology
EUROSPEECH'93
Berlin, Germany, September 19-23, 1993

HABITABLE INTERACTION IN GOAL-ORIENTED MULTIMODAL DIALOGUE SYSTEMS

Philippe Morin[1,2] and Jean-claude Junqua[1]

[1]*SPEECH TECHNOLOGY LABORATORY, Panasonic Technologies Inc., Santa Barbara, California, 93105, USA*
[2] *CRIN-CNRS / INRIA Lorraine, 54506 Vandoeuvre les Nancy Cédex, France.*

ABSTRACT

An understanding of what makes a user interface habitable and natural is needed. Man-machine interfaces must be helpful and cooperative to be usable, especially when difficult situations occur in the interaction. User acceptance of these interfaces depends on their error robustness and their ability to provide assistance. This is especially important for error-prone input such as speech. In this paper we describe several generic mechanisms that have been developed to provide user guidance and improve naturalness in the PARTNER man-machine dialogue system. It is shown how these mechanisms can be used to ease the interaction with naive users and facilitate error recovery. Finally we present some results obtained in the case of an object manipulation tool developed with PARTNER.

Keywords : *Help, Error recovery, Multimodal dialogue*

10.21437/Eurospeech.1993-376

1 INTRODUCTION

The introduction of text and speech input and output channels in applications responds to a growing need for user-friendly interfaces. A more natural interaction between man and machine is necessary to make the machines more accessible. The traditional conception of man-machine interaction shows drastic limitations. With ever more advanced functions being incorporated into various home or office electronic equipment, it is apparent that, for the average user, operating these devices fully has become a too complex, difficult or burdensome task. As a result many functions are unused. A user interface should be simple, flexible and cooperative. Consequently there is a tremendous potential for a dialogue technology that will facilitate the development of such interfaces. However these interfaces represent a fundamental change and require new architectures and software concepts. Speech input and output can greatly contribute to improve naturalness. Experiments made on voice-based systems have shown that speech itself does not guarantee the success of an interface and that users expect a *complete dialogue environment*. In addition, due to the limitations of speech recognition, strategies have to be developed to overcome recognition errors. Realistic user and dialogue models have to be designed to ensure that the interaction is viable. It requires the design of a flexible input model in charge of the discourse analysis, but also the integration of assistance and error correction mechanisms.

These issues have globally been addressed in PARTNER since we believe that the success of an interface depends on the presence of each of these mechanisms. In this paper we focus on the development that has been made in our system to provide guidance and naturalness in the interaction. Generic mechanisms have been implemented. In the following sections, after an overview of the system's architecture, we present several assistance mechanisms including 1) dynamic proposal, 2) automatic assistance, 3) contextual help, 4) explanation and 5) language completion. Finally, we briefly highlight our concern for a lively interaction and present some techniques that have been implemented to reduce monotony in the dialogue.

2 OVERVIEW OF PARTNER

The framework of our contribution is a real-time multimodal dialogue system which is independent of the application : the PARTNER system [1]. It has been designed to facilitate the conception and the implementation of goal-oriented applications using speech as the primary communication channel. PARTNER is a generic system that provides a configurable dialogue environment. We believe that it is important to provide a general technology applicable in different contexts. PARTNER is in charge of the dialogue activity (i.e. input/output processing and discourse analysis) and in control of the applications. It supervises the user-application interaction so that applications can focus on their specific task. Its architecture is based on a complete separation of dialogue and application domains. An application specification model has been developed and allows applications to be described logically. The system has been implemented as a dialogue server to which client applications can connect. A dialogue toolkit provides access functions to the server.

2.1 Application development

In PARTNER, an application model is used to describe applications. The model is based on the definition of 1) a set of scripts, 2) a set of application functions and 3) a dialogue database. Scripts represent the control part (i.e. application scenario) and are evaluated by the server. They are composed of elementary *dialogue instructions*. Application functions represent the processing part and allow the dialogue system to interact with the application universe. In particular actions can be triggered in the application or information can be obtained from the application model.

Applications are responsible for the design of their user interface since the dialogue server has no direct interaction with the user. The toolkit provides functions to configure the dialogue server and control the interaction. In particular functions to transmit application-dependent input events (e.g. mouse, touch screen and text inputs) to the server are provided.

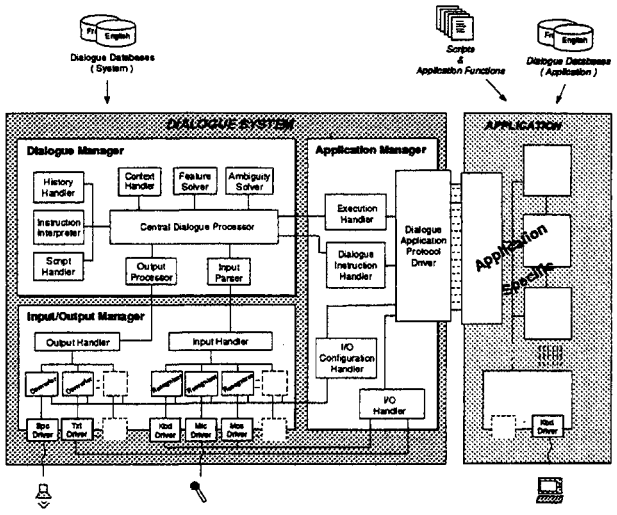

Figure 1: Generic architecture of PARTNER

2.2 Architecture of PARTNER

Figure 1 shows PARTNER's generic architecture. The server is structured around three major components : 1) an input/output manager, 2) an application manager and 3) a dialogue manager.

The ***I/O manager*** provides drivers to handle the information flow between the dialogue manager and the user. The current version controls four input drivers (continuous speech, text, mouse and touch screen inputs) and two output modes (synthesized speech and text outputs). All input channels are available at any time and can combined to produce actions. To describe speech and text input/output events, a restricted natural language is being used.

The ***application manager*** controls the communication between the server and the applications. In particular it contains functions to load the dialogue scripts from the application and the application-dependent language database.

The ***dialogue manager*** implements a dialogue model. It interprets the input events coming from the I/O manager in context and pilots the execution of the application. Its main objective is to assure that a coherent dialogue is being run, especially in case of incomplete user inputs, errors and ambiguities. On top of the application language a *meta-language* is defined to take care of the dialogue phenomena (e.g. requests for correction, explanation, help or repetition).

3 TOWARDS A COOPERATIVE INTERACTION

3.1 Introduction

Cooperative interaction is a key factor that greatly conditions the usability of most applications. In PARTNER, several complementary assistance mechanisms have been introduced to keep the dialogue active and coherent. They are generic and part of the dialogue model. Most of the mechanisms are activated when needed at the user's initiative. It is important for users to be able to specify the degree of assistance they need (i.e. from naive mode to expert mode). However, when problematic situations are detected by the dialogue server, some of these mechanisms might be engaged automatically at the system's initiative.

In this framework assistance techniques offer significant benefits not only in terms of ease-of-use but also in terms of error recovery and learning. When dealing with voice-based applications, it is important to provide ways that will help users acquire the application language.

To improve the user's ability to use an application, the following generic assistance mechanisms have been developed. Note that the dialogue server controls their execution, but that the knowledge sources are defined in the application model. Depending on the mechanism used, the server can get the information needed from the scripts, through specific application functions or in the dialogue database.

1 ***Dynamic proposal***
This mechanism can be to used to propose an answer to the user when he is asked a specific question. In case of a good guess, users can validate the proposal by entering a limited answer such as *Yes*.

2 ***Automatic assistance***
Automatic assistance is a mode that can be turned on or off. When activated a list of possible answers will be provided for each question registered with that mode in the subsequent dialogue at the user interface level.

3 ***Contextual help***
Users can enter the help mode to obtain a list of possible choices at the dialogue level. In this case simple answers can be used (e.g. *Yes/No* type) to validate the desired entry.

4 ***Explanation***
Users can use the *explain* meta-command to get additional information on the return status of the last command.

5 ***Language completion***
This mechanism allows the user to build valid inputs word by word (available for text and speech channels).

Sys: What do you want ?
Usr : ***Paint the cube alpha in yellow***
Sys : Ok
Usr: ***Store it***
Usr : ***Store alpha***
Sys : Sorry, impossible request
Usr : ***Why***
Sys: The cube alpha is already stored
Usr : _

Figure 3: Example of explanation request

These five standard mechanisms are available for any application developed with PARTNER and must, whenever possible, be

Sys : What else ?
Usr : ***Paint alpha***
Sys : In which color ?
Usr : ***In red***
Usr : ***Paint the cube gamma***
Sys : In which color ? Is it red ?
Usr : _

Note : In this example, the color red used to paint the previous object is proposed

Figure 2: Example of dynamic proposal

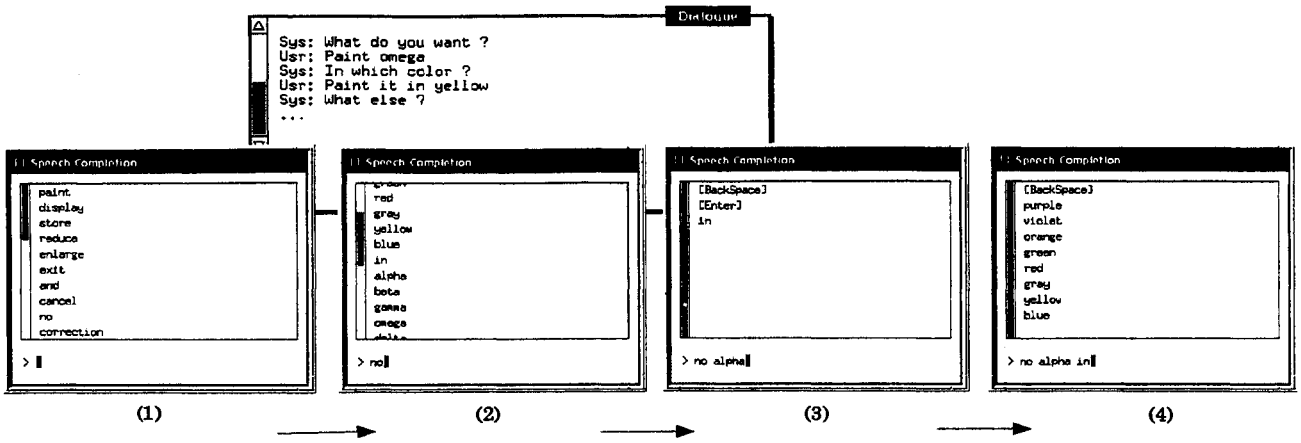

Figure 5: Example of speech completion

integrated. On the other hand we believe that application-dependent assistance mechanisms should not be neglected. The design of an interface and the user feedback it provides are essential as well.

In the following sections we detail three mechanisms : automatic assistance, language completion and contextual help. Figure 2 illustrates the generation of a dynamic proposal and figure 3 shows an example of an explanation request.

3.2 Automatic assistance

The automatic assistance mode is a mode inside the dialogue server that users can activate to get a continuous assistance from the interface. When activated, the interface will be asked to display the possible choices corresponding to the current immediate dialogue context, as a pop-up window for instance. From that moment users can use any input mode to enter the information, in particular they can click on the entry.

When a question is being asked to the user, the server checks in the application scripts if an application function of the type *assistance* has been registered with the missing attributes; in which case the function will be automatically called by the server to get the contextual list from the application representing the possible choices. After some processing it will be sent back to the application for display. It is the application programmer's responsibility to declare assistance application functions in the scripts and assistance may be or may not be present at selected points in the dialogue. Figure 4 shows a typical assistance session.

3.3 Language completion

A language completion mechanism is provided to build valid inputs interactively. It corresponds to a mode inside the dialogue server that can be turned on or off at the user's initiative. It has been designed to allow naive users to operate an interface and acquire the application language progressively. In this mode, speech and text inputs can be edited and entered word by word. At each step the dialogue server provides the application with the next possible word candidates completing the initial input. The completion is contextual and only valid inputs that make sense in the dialogue context are generated. Valid inputs are inputs that are syntactically and semantically correct. They are predicted by the dialogue manager from the current application context and the dialogue history. The semantic constraints given by the dialogue manager are used to generate the syntactic structures from the application language.

Note that the list of possible word candidates is sorted in plausibility order such that words related to the immediate dialogue context are on top of the list.

The text and speech completion modes can be turned on or off at any time. Inside a completion session, other modalities (e.g. touch screen or mouse mode) can be used if found more appropriate. Figure 5 shows a typical completion sequence.

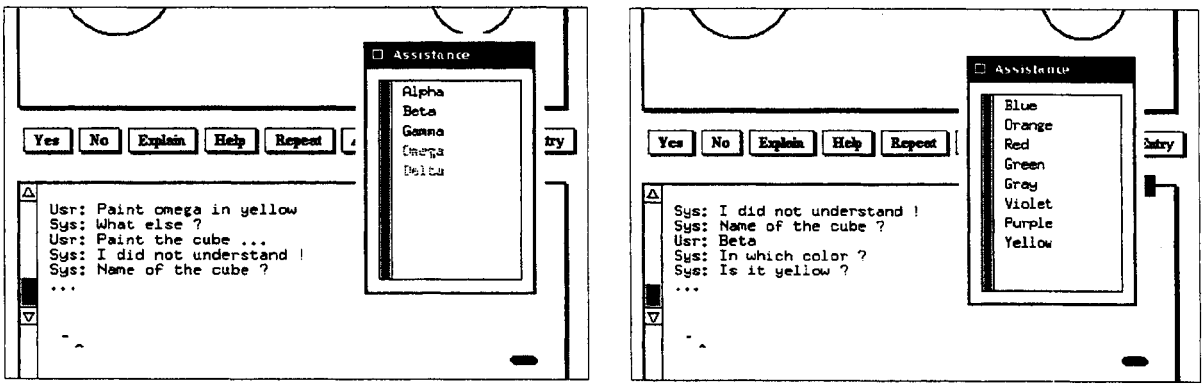

Figure 4: Example of automatic assistance

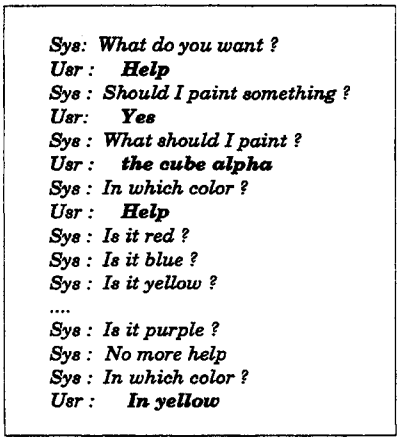

Figure 6: Example of contextual help

3.4 Contextual help

The help mechanism allows the user to be more precisely guided in the dialogue. When activated, the dialogue manager will ask specific questions corresponding to the possible choices available in the immediate dialogue context so that limited answers (e.g. Yes/No type) can be used. Its implementation is very similar to the assistance mechanism described earlier and application programmers can register application functions of the type *help* in the scripts. In PARTNER the help mechanism can be activated at the user's or system's initiative.

- ***User's initiative***
  When the user does not know what to answer or what are the possible choices to choose from, the help meta-command will activate the mechanism.
- ***System's initiative***
  After two successive non-interpretable inputs, the server will automatically engage the help mechanism to provide assistance and keep the dialogue active.

Figure 6 shows an example of help activation at the user's initiative. Note that the scope of the help mechanism is limited to the current input.

4 NATURALNESS IMPROVEMENT

In man-machine communication robustness and naturalness are two crucial issues that primarily require the development of 1) flexible discourse models accepting natural input, 2) error management techniques and 3) assistance mechanisms to achieve usability. However another aspect of naturalness is variety. For a better acceptance a lively interaction is desirable and some techniques have been introduced in our system to help reducing monotony in the dialogue and make it more fluent. It includes :

- Development of meaning-to-text algorithms which produce varied syntactic structures,
- Activation of proposal,
- Use of default values to reduce the number of exchanges assuming that explicit correction is always possible,
- Randomness in the execution of the application scripts to obtain different outputs.

We believe that important efforts need to be pursued in this direction. Figure 7 shows an example of variable output.

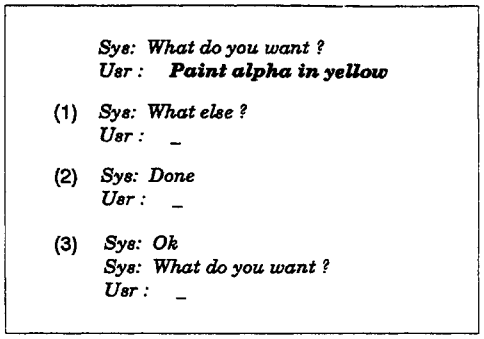

Figure 7: Example of variable output

5 EXPERIMENTS

Two applications have been developed using the PARTNER architecture. Currently the assistance mechanisms are being assessed in the case of the second application which an *Intelligent Home Appliance Controller* (IHAC) designed for voice control of various home equipment such as compact-disk player, VCR, telephone or answering machine.

6 CONCLUSION

In the domain of man-machine communication it has been shown that a cooperative interaction is needed to ensure the success and the usability of voice-based applications. In this paper we presented five generic assistance mechanisms that have been developed in the application-independent multimodal dialogue system PARTNER. The proposed mechanisms are complementary and correspond to different user needs and different user level (e.g. naive, occasional or expert users). Our latest results seem promising. However field tests have to be conducted to evaluate the system in terms of performance and user satisfaction. We are currently extending the language completion mechanism described in this paper to integrate contextual dialogue prediction mechanisms into speech recognizers.

REFERENCES

[1] P. Morin, J. Junqua, and J. Pierrel. A Flexible Multimodal Dialogue Architecture Independent of the application. In *ICSLP-92*, pages 939-942, 1992.

[2] S. Young. The MINDS System: Using Context and Dialog To Enhance Speech Recognition. In *Speech and Natural Language, DARPA Workshop*, pages 131-136, February 1989.

[3] K. Fellbaum, R. Heinstein, and H. Loebner. Speech Dialogue Systems - State of The Art and Selected Applications. In *EUROSPEECH-89*, pages 433-436, 1989.