In Proceedings of
Esprit Conference, 1990, Elsevier.

Project No. 2474

ARCHITECTURE OF A MULTIMODAL DIALOGUE INTERFACE FOR KNOWLEDGE-BASED SYSTEMS.

J-L. Binot, BIM, Everberg Belgium,
P. Falzon, INRIA, Voluceau - Rocquencourt, France
R. Perez, ISS, Barcelona, Spain
B. Peroche, Ecole des Mines, Saint-Etienne, France
N. Sheehy, University of Leeds, UK
J. Rouault, CRISS, Grenoble, France
M. Wilson, Rutheford-Appleton Laboratory, Chilton, UK

ABSTRACT. This paper describes the architecture and the first implementation results of a multimodal dialogue interface for knowledge based systems developed in the context of Esprit II project MMI2. The paper reviews the basic principles of the architecture of the system and the approach taken by the project with respect to user modeling issues, then describes individually each communication mode.

1. Overview of aims and basic approach

This paper outlines the first results of ESPRIT II project P2474: "MMI2: A Multi Mode Interface for Man Machine Interaction with knowledge based systems." These results were obtained through the cooperative efforts of all researchers involved in the project: Jean-Louis Binot, Fabienne Balfroid, Lieve Debille, David Sedlock and Bart Vandecapelle (BIM, Belgium), Gerard Henneron, Genevievre Lallich-Boidin, Rosalma Palermiti, Jacques Rouault, Jean-Louis Zinger (CRISS, France), Helmi Ben Hamara, Christian Bertin, Christine Jouve, Dominique Michelucci, Bernard Peroche (Ecole des Mines de Saint-Etienne, France), Bernadette Cahour, Francoise Darse, Pierre Falzon (INRIA, France), Alica. Manzanera, A. Moneta, Ricardo Perez, David Trotzig, Juan-Carlos Ruiz (ISS, Spain), Farah Arshad, N. Ghali, Mark Howes, K. Marida, Noel Sheehy (Univ. Leeds, U.K.), Helen Chappel, Graham Doe, Gordon Ringland, and Michael Wilson (Rutheford Appleton Laboratory, U.K.).

A multimodal system. The MMI2 project aims to build a man/machine interface for different kinds of users, integrating several modes of communication supported by modern workstations: natural language, command language, graphic and gesture. The interface will provide simultaneously modes suitable to support the efficiency of experienced, professional users (command languages, menus) and natural communication modes well suited to naive users, such as graphics and natural language. Natural language modules are being developed for English, French and Spanish.

Difference between modes and media. It seems first necessary to clarify the distinction that we make between the meanings of the two words medium and mode. A number of projects have already

studied multi-media phenomena and in particular multi-media interfaces. The multi-media concept is present as soon as a computer system can deal with more than one type of input/output support. Multi-media communication, however, does not imply multi modal communication.

While a medium is only an information support, a mode is a means of expression and thus a means to convey information: a mode is built on a lexicon, syntax, semantics etc. Communication with a computer through graphic mode requires not only a graphic medium but the definition of one or several graphic modes using that graphic medium.

Dialogue management. Advances are aimed at in each mode separately. However the main source of improvements to interface technology will come from the integration of the different modes. To reach such an integration, one of the main aims of the project is to develop a dialogue management and mode selection system which uses knowledge of the specificities of individual modes, knowledge of the context of previous interactions, and knowledge of the application domain to interpret the input, determine the content of system output and select the most appropriate mode in which to present particular information. A user modeling module will interact with dialogue management, so that the system will react appropriately to different classes of users and individual users.

Knowledge-based backend application. On the machine side, the interface is primarily aimed to be connected to applications such as Prolog based expert systems (although we expect many of the results of this project should be usable, at least indirectly, for many other kinds of workstation application software). In order to focus on real practical problems, the interface prototype is being connected to a specific application, also developed within this project. This application, called NEST, is an expert system in computer network design. Such a system, besides having a very high intrinsic interest in its own right, given the current trends in information technology, has a great variety of potential users and offers many opportunities for multimodal interaction, including natural language and graphics.

Finally, the interface is designed to be portable across a range of potential applications of Prolog based KBS. Special emphasis is put on designing a flexible and portable architecture having well defined interconnection points with the application and on developing a set of tools for the rapid adaptation of the interface to a new application.

2. Architecture of the MMI2 system

A significant part of the work done in the first year of the project has been concerned with the definition of a clear, modular and conceptually sound architecture for the whole system. The architecture of the system is based on the notion of ***"expert module"***. The name "expert" should be clearly understood. We are not proposing an architecture of "cooperating experts", or "multiple agents", which, we believe, fall outside the scope of this project. What we call an expert is simply a module performing specific tasks and with its own private data structures, and which represents a sufficiently coherent set of processes to be gathered in a single module. While such a notion is clearly not new, the identification of the nature of the basic modules constituting the multimodal interface, and of the interactions between them, has been a crucial step in the project. The resulting architecture is illustrated in figure 1 below.

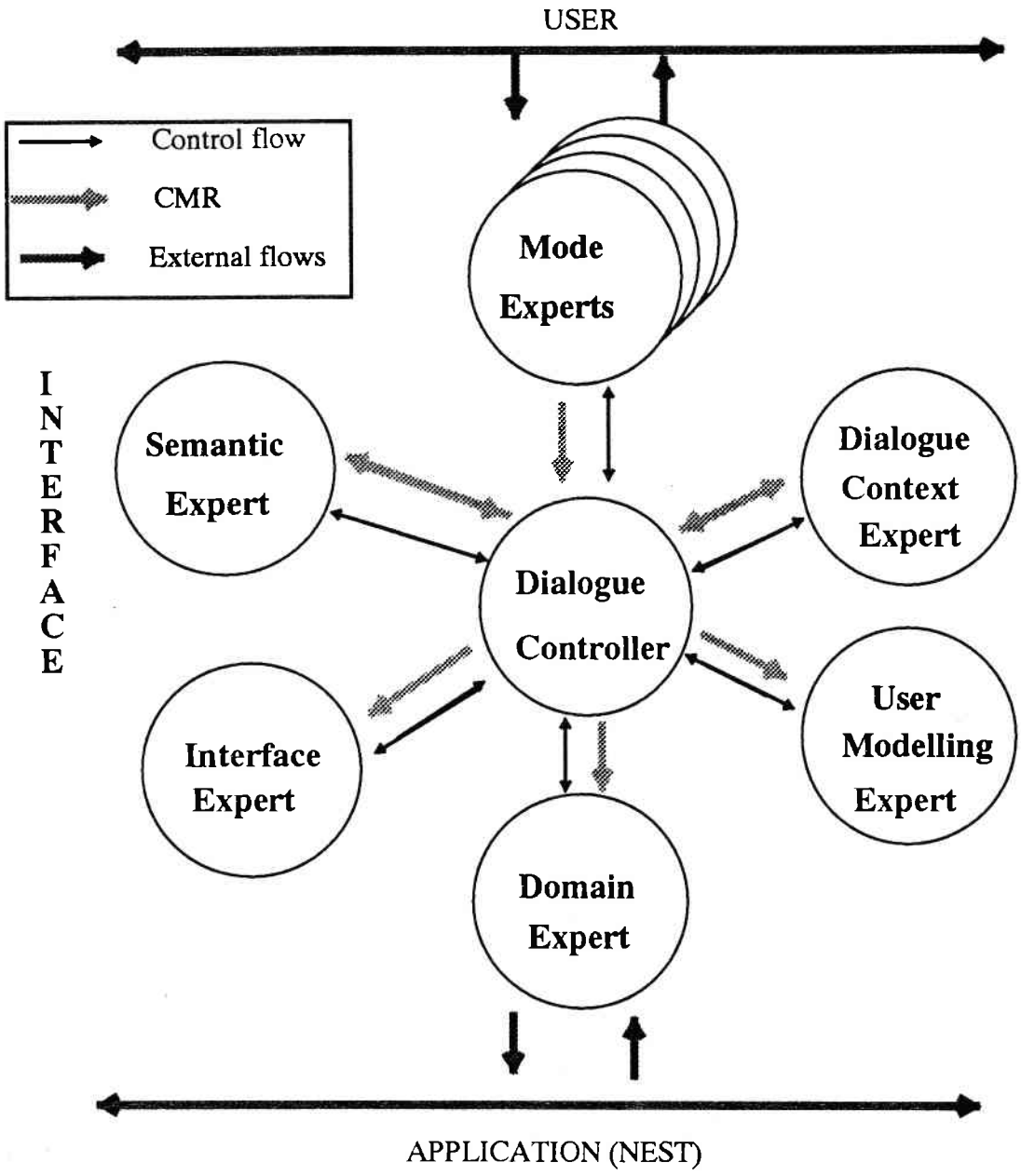

Figure 1: General architecture of the MMI2 system

The basic idea of this architecture is that every input should be cast into some suitable meaning representation, then forwarded to the dialogue controller, which will decide what to do with it This should hold not only for input in natural language mode, but also for inputs in command and/or graphical mode (thus graphical or typed in commands, for example, should not be executed directly

but go through the dialogue controller). The dialogue controller will call the dialogue context manager to update the dialogue model according to the new interaction, then reason (on communication acts, input content, state of dialogue and user models) to decide what to do with the input, how to gather results, what answers to present and in which output mode to present them.

To make such an architecture possible, there has to be a **meaning representation formalism** common to all modes, which is used as a vehicle for internal communication of the semantic content of interactions inside the interface itself and also used as a support for semantic and pragmatic reasoning. This meaning representation formalism is called the CMR and will discussed in more details in a later section.

The main functionality of each of these expert modules can be described as follows:

- The *dialogue controller d*eals with:
  - choosing and managing the structure of the dialogue
  - performing response determination and output mode selection
  - activating whatever experts are necessary to support dialogue interaction

- The ***dialogue context manage***r manages everything which has to do with:
  - identifying the dialogue structure and its various components
  - recording that structure
  - extracting relevant information from it

- The ***user modeling expert*** maintains and exploits the user model.

- The ***domain expert*** has all the expertise about the domain, including:
  - what is covered by the domain
  - how to translate the internal meaning representation into domain terms
  - how to manage "task plans" describing problems to be transmitted to the application.

- The ***interface expert*** has the knowledge about the interface itself, including
  - features/capabilities/configuration of the interface
  - current physical interface layout on the screen

- The ***semantic expert*** has all the knowledge about the general (domain-independent) properties of meaning concepts, and of semantic inferences that can be performed on them.

- The various ***mode experts*** perform input and output for each mode.

The modules being part of what is generally called the "dialogue management" process are the dialogue controller, dialogue context manager, semantic expert and domain expert. The following sections discuss further some basic aspects of the main modules enumerated above. But, in order to provide a practical context, we first start with some words about the application chosen as a testbed for the project.

3. Testbed application: an expert system in computer network design

The application chosen for MMI2 and developed within the project is an expert system for Network Design. It has been chosen for its own interest and for its interest in the context of a multi-mode interface development.

Indeed, information technology is evolving fast toward distributed systems. Configuring computer networks is, in this context, a crucial and difficult task both from economical and technical points of view. The development of expert system tools assisting such a task can certainly contribute significantly to the progress of European information technology. On the other hand, an expert system such as the one considered here has a great variety of users: technicians and commercials, beginners and experts. It deals with graphic and text information and must allow request and updating of data. So it supposes at least natural language, command language, graphics and gesture use. Thereby, it constitutes a credible practical test for a multi-modal interface.

Different components are involved in a design system. We decided to start the development of the application by the analysis component. This component is currently implemented. It analyzes *local area networks* using *Ethernet* technology and checks if those networks are correct or satisfying according to different evaluation criteria such as *technological validity, extensibility, client-server relation, departmentalization* or *cost*. Networks which have to be analyzed are described by using an object oriented model defined in BIM_Probe, an object oriented tool built on top of BIM_Prolog.

The next step of the application will be devoted to the development of the configurated component able to compute a local area network configuration respecting some given constraints such as those specified by the customer (budget, building limitations,...) or the technological ones already considered in the analysis tool.

4. Mode integration and dialogue context management

The MMI2 architecture is based on the fundamental assumption that

> **mode integration should mainly be achieved by an integrated management of a single, generalized, discourse context.**

The basic idea is that any interaction, or any "discourse" between the user and the interface, in any of the modes, takes place in a common "discourse world" (which may but is not necessarily connected to the real world, or to an application). Any entity mentioned in the course of an interaction acquires, by the sole virtue of having been mentioned, an existence in this discourse world, where it shall be called a "discourse referent".

Various operations may apply to discourse referents. They may be "introduced" (or "created", or "brought into the discourse"), referred to by using a "description" of a referent, and even possibly "forgotten" ("deleted"). They may also be brought in or out of focus, and different sorts of foci can possibly be distinguished.

These notions are familiar to people involved in Natural Language Processing. What we argue is that they are common to other modes as well, and are the most natural way to perform mode

integration. Let us give a few examples about each of the mode.

In natural language, new discourse referents are typically introduced by indefinite noun phrases and referred to by definite noun phrases (many special cases can be found in the literature, but will not be discussed here). Thus, in

Add a server to the network. Connect this server to....

the indefinite expression "a server" will cause the introduction of a new discourse referent in the discourse world, and this referent will receive a unique identifier. The expression "this server" can then refer to the newly introduced discourse referent, and can possibly bring it into focus.

For NL interaction, the referencing operation is done mainly through the use of descriptions which take the form of (definite) noun phrases. The basic problem is to relate such descriptions to unique referent identifiers: this process is usually known as "noun phrase resolution".

In ***the graphical or gesture mode***, any basic graphical operation will have an effect on discourse referents. Creating a graphical object will obviously introduce a new referent. Selecting an object with the mouse will perform a reference to a existing referent. Moving a graphical object will bring the corresponding discourse referent into focus. Changing the graphical display (zooming, displaying another network, selecting another window where something else is displayed) would bring a different set of discourse referents into focus.

For the graphical mode the referencing operation is done by selection: the referent has a graphical description (as it could have one or several NL descriptions) and selecting the graphical description leads to the identification of the referent. The problem of identifying the referent selected is trivial, as each graphical object should have as associated property the unique identifier of the referent it represents. This association is easy to maintain: when a graphical object is created, it is either to display an existing referent (in which case the referent identifier is known) or to create a new one, in which case a new referent identifier name should be created automatically by the system.

In the ***command language mode***, finally, basic operations on discourse referents are easy to identify, and bear close similitude to graphical ones. Thus a command to add an object will create a discourse referent for that object; commands performed on objects will tend to bring these objects into focus. Reference to existing discourse referents can be done by using the unique identifier of a referent as parameter in a command. The referencing operation for command language is also simple, as the referent identifier itself can serve as description of the referent.

Reasoning in terms of discourse referents and focus provides already for very interesting mode integration, which goes further than the more or less traditional graphical deixis (simply selecting something with the mouse and asking "what is this"). Thus in a sequence like:

System:	*<Graphical act to display new network>*
User (NL)	*What is the server?*
System:	*<Graphical act to display an icon>*

the first (graphical) utterance will bring a new set of discourse referent into focus. The NL query will then attempt to resolve the noun phrase "the server" against the discourse context and the

current focus. The response determination module of the dialogue controller would then decide of a mode (graphics) and an appropriate graphical "communication act" (highlighting) to provide the answer.

We are thus led to study the conditions governing creation, focusing, reference and possibly destruction of discourse referents across all communication modes. A first attempt to organize the factors controlling these events is indicated in the figure below.

	Creation	Focus	Reference	Destruction)
NL input	(ind.) noun phrases	sentence content & structure	(def.) noun phrases	forgetting?
Graphics	copy, paste, draw	display, zoom	click	
Command	COPY, CREATE	command content & structure	discourse referent id	

Figure 2: Operations on discourse referents across modes

5. Representing interactions - the Common Meaning Representation

A second source of integration arises from the fact that many interactions can be expressed equivalently (from the point of view of meaning, if not of ease of expression) in several modes, as the following examples illustrate:

NL: *Suppress the SUN3 connected to server Ella*
Graphics: *<Click on icon and select delete option>*

NL: *Augment performance of Ella by 100%*
Graphics: *<Select appropriate bar in a bar chart about computer performance and modify it>*

NL: *Suppress the blue servers (on color screen)*
Graphics: *<Click on icons and select delete option>*

Although expressed quite differently, all these input must have the same effect on the application (knowledge-based system) and on the dialogue context. Thus one of the results of MMI2 is to establish a taxonomy of actions across modes (e.g. the verb "suppress", a DELETE option in a menu, a DELETE command or a gesture of crossing out something with the mouse all refer to a "deleting" operation). But integrating the representation of interactions across modes can go further than that. A second basic architectural principle of MMI2 is that

there is a meaning representation formalism, common to all modes, which is used as a vehicle for internal communication of the semantic content of interactions inside the interface itself and also used as a support for semantic and pragmatic reasoning.

This meaning representation formalism is called the CMR. The purpose of the CMR is to represent the meaning of interactions between system and user, or user and system. Such interactions are called "communication actions". In MMI2 a communication action is a graphical action, a command language/gesture action, or a natural language action, which can be carried out by either the user or the system. When the communication action is expressed in a natural language, the action is an utterance.

The proposition expressed by a communication action consists of content and logical form. Following many other practical natural language processing systems, we have chosen to express the propositional content of a communication action in a language based on a first order typed predicate logic where relations are, as a general rule, reified. Specialized languages, such as frame or semantic network languages, fall short of the expressive power we expect to find used in our application. (See Chapter 2 in [Genesereth 1987]. Of course, there are extensions of these specialized languages, but these extensions end up looking like predicate logic.) On the other hand, we do not expect to need the extra expressive power that a second order language affords.

Our approach to representing meaning can thus be regarded as **logical**. However, we recognize that there are aspects to communication that are difficult, perhaps impossible, to capture on a purely logical approach. Therefore, we have decided to include extra information in CMR concerning illocutionary force, enunciation conditions and other things that we call "annotations". So a CMR expression contains four sorts of information: illocutionary force, propositional content, logical form, and various other annotations. A CMR expression is also part of a larger data structure that possibly includes the following additional information: processing status, mode, time of action, user presuppositions, user mistakes, and some syntactic information. The following figure illustrates one example of CMR representation. Although we shall not describe it in more details here, the full specification of the CMR language has been completed and is one of the major results of the first year of the project.

User input: What do the machines on the network cost?

```
CMR(
  [CMR_exp(
          [request,referent([var(x1)])],
          [anno(x3,[definite_plural])],
          (desc(the,x2,MACHINE,
                  (desc(the,x3,NETWORK,true),
                  (desc(some,x4,IS_ON,true),
                  and(
                        [pred(SUBJECT,[var(x4),var(x2)]),
                         pred(LOC,[var(x4),var(x3)])])))),
          (desc(null,x1,QUANTITY,true),
          (desc(some,x5,HAS_COST,true),
          and(
                  [pred(PRESENT,[var(x5)]),
                   pred(SUBJECT,[var(x5),var(x2)]),
                   pred(OBJECT,[var(x5),var(x1)])])))))],
  ok,
  English,
```

time(1,1,1,1,1,1990),
none,
none,
none)

Figure 3: Example of a CMR representation

A series of communication actions forms a "dialog". CMR is **not** meant to be a representation of a dialog, although it is supposed to lend itself to the construction of dialog representations. Of course, how you define actions or utterances is not so clear in general. In practice, however, each action is individuated by an illocutionary force: one force, one action, one thing to be represented in a CMR expression.

6. Dialogue flexibility and dialogue control

The application, being a knowledge-based system, has its own data and knowledge structures. A typical knowledge-based system has usually a "problem space", where the initial data of the problem to be solved are placed, a "solution space", where the expert system would build its solution, and a knowledge base containing general knowledge and expertise about the domain. What, then, should happen when the user starts to specify a problem?

Typically, for an expert system, the specification of the problem may require a set of information of different types. If the dialogue is application-driven, these information will presumably be asked to the user in some systematic order. But, in a real problem acquisition dialogue, the user may shift topics, answer questions by other questions, provide ambiguous or incomplete answers that will require subdialogues, request help, or even change his mind. It is obviously the job of the interface, and not of the application, to deal with such problems. If the user-provided data were sent directly to the application, they might well be in the wrong order or provide wrong values that would have to be corrected later. We have thus decided that, for reasons of flexibility, there should be a level of representation of the problem in the interface itself.

This kind of problem has started to interest researchers in dialogue management, and has been discussed notably by [Julien and all 89], who illustrate it with an example taken from financial advising expert systems:

SYSTEM: *How much do you want to invest in an emergency plan*
USER: *Let us talk about my car loan instead!*

In the face of such an answer, the interface must either enforce a strict dialogue schema, or accept a shifting of topic, which supposes that it should be able to detect it, and to remember to come back later to the "emergency plan" topic if this one is essential for the formulation of the problem.

To implement this kind of behavior, we decided to provide the interface with three basic kinds of data structures, a "task plan", and a "communication plan", and a "discourse referent space", the two first being inspired from [Julien and all 89]. The position of these structures in the architecture is

illustrated in figure 4 below:.

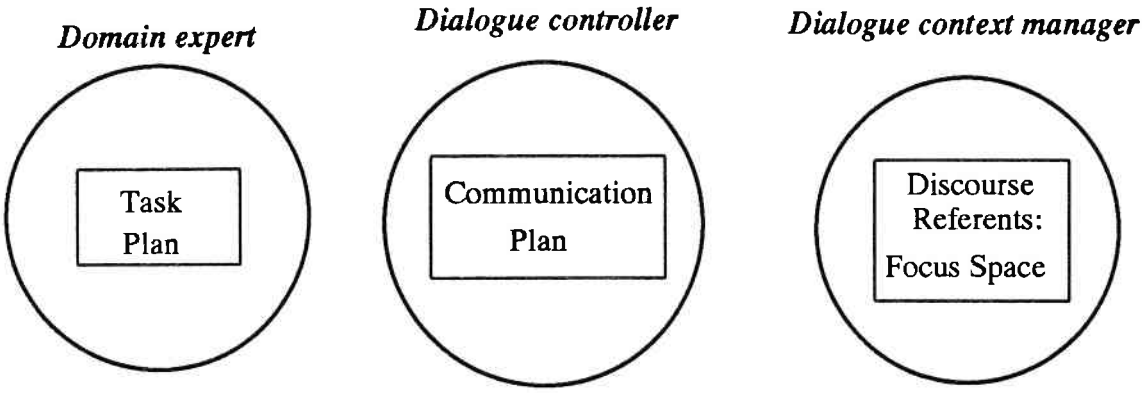

Figure 4: Key structures to support interaction with user

The basic role of the space of discourse referents has been described before.

The ***task plan*** is a model of the kind of information that need to be provided to the application in order to specify a well-formed problem. It can be seen as a kind of "skeleton" of a typical application problem, which will be progressively instantiated in the course of the dialogue. Task plans are of course domain dependent, and are thus managed by the domain expert. Several task plans may be available for a single domain, corresponding to the different kinds of problems one can submit to the application. The task plans will have to be provided as part of the application dependent information and will be stored in some part of the domain model. Normally, the task plan to be applied should be identified through interaction with the user.

The ***communication plan*** is a plan structure for guiding a dialogue with the user. Once a task plan is identified, there will be a communication plan activated to acquire the information necessary for the task plan. There will be at least a communication plan for every task plan available in the current domain; but the selection of such communication plans will also depend on the user model. There can also be communication plans for other parts or subparts of the dialogue; for example there could be one for greeting the user and ascertaining the kind of problem he wants to solve. As communication plans have to do with controlling the structure of the dialogue, they will be stored in the dialogue controller, in a "communication plan library". At any time there will be one active communication plan, which will be executed, and if needed updated by the dialogue controller.

7. User modeling

7.1. THE USER MODELING EXPERT

In MMI2, the role of the user modelling module is to increase the cooperativity of the system. The user modelling expert builds up and maintains the user model. The user model is the knowledge

source in the system that contains explicit assumptions on all aspects of the user that may be relevant t the behavior of the system. User models will be stored between sessions as User Profiles which will be downloaded for experienced user at logon.

The major uses of the information in the user model are for:

- The Dialogue Context Expert to identify speech acts, ellipsis and other pragmatic features on the basis of the user's goal and beliefs.
- The Mode Experts to present information using the symbols, lexical items and style the user prefers.
- The Dialogue Controller, to use the user's current goal to identify the user's current plan and thus provide co-operative responses. Responses will also be tailored on the basis of the user's beliefs according to Grice's maxims [Grice 1975] to avoid redundancy and to emphasize and explain the user's misconceptions.
- The Domain Expert will use items of user specific information in completing domain plans which will form queries to the knowledge based system.

The basic framework of the user modelling component comes from General User Modelling System - GUMS - of Finin and Drager [Finin and Drager 1986]. This provides four essential features of the user model itself:

- An interface between the user model and the rest of the system;
- A mechanism for overridable default inheritance from stereotypes with optional negation as failure;
- A truth maintenance mechanism to manage the updating of inheritance from different superordinate classes during a session as the user's membership of a class changes;
- A mechanism for storing user and class models between sessions.

Two advances on the GUMS model are required fro the underlying framework of the user modelling component in the MMI2 architecture. Firstly, the GUMS model only supports inheritance from a single stereotype which is limiting since users could be both a member or a class with knowledge about the domain, and another with knowledge about the interface and system itself. Inheritance from multiple stereotypes has been added to the GUMS system. Multiple inheritance brings with it the possibility of conflicts between information in different stereotypes. These are handled by searching for certain information rather than just default values; if these conflict then the certain information overrides the default. Similarly, where negation as failure is used to show the lack of knowledge or belief, and knowledge or belief is stated, then the presence of knowledge or belief is returned. If certain but contradictory knowledge is stated in different places, a temporary unsatisfactory heuristic is applied that the first version encountered in the bottom up breadth first search is returned. The second change follows from the use of multiple stereotypes. Sparc Jones [1987] succinctly the problems with assuming that the user of the system who is entering the commands is actually the user of the system for both the interface and the reasoning components of the system. Her potent example is one of a social welfare worker entering information on behalf of a client where the view of the user act as a filter on the information obtained about the client; either approving or otherwise. The overcome this complexity, the leaf node of the user model in the present system represents the combination of the user and the client. Therefore this user model inherits from both the user and the client nodes in the lattice, each in turn from other stereotypically defined nodes.

A major requirement of the MMI2 system is to produce co-operative responses for the user. The class of co-operative responses expected follows [Kaplan 1983, Allen 1983, Carberry 1985, Pollack 1986]. Their generation require representations of the user's goals; the plans for achieving them; the user's beliefs; an evaluation of whether they are correct knowledge of misconceptions with respect to the systems view of the world which is assumed to be rue; some decision relevant information to be used in task plans for the KBS application; and various attitudes and preferences of the user for explanation style, lexical item choice, and interface style and options.

The user's beliefs are stored in the prototypes and the user's own model. These can be acquired explicitly or implicitly as suggested above. The rules for implicitly acquiring user beliefs follow those used by Kass [1988, 1989]. In order to use user beliefs to interpret input to the interface or to tailor output for the user it is useful to identify the beliefs as true to the world, and therefore user knowledge, or false in the world, and therefore misconceptions. The test of truth in the world is made by comparing the user's beliefs with the representations in the other experts and change their status if a judgment can be made. If no comparison is possible, the beliefs remain beliefs and are not classified as knowledge or misconception.

7.2. USER MODELING EXPERIMENTS

The set of classes and rules for determining if users belong to them, along with the sets of possible relevant goals, preferences, decision relevant information, knowledge and beliefs which must be included in this architecture are established for the demonstration application by the experiment described below.

The experiment included two steps: simulation of interactions between users and a system, and postverbalization supported by a record of the activity.

One expert simulated the system and different types of subjects (or 'users') requested his advice to solve two problems of physical network design that had been previously defined in collaboration with the expert: the first problem consists of designing a network for a research department and the second one consists in designing a network connecting different buildings in a university. According to the expert, these problems are typical and of average difficulty. The 10 subjects differed in computer education and in type and level of knowledge about network design.

The expert used MUSK as a tool for interaction between the subjects and the expert. MUSK is a program (designed at Rutherford Appleton Laboratory) that allows both graphic and written interaction between several users simultaneously. Each of the ten subjects solved the two problems successively in the same order, the interaction for each problem taking about two hours. Every ten minutes the expert was given evaluation sheets on which he quickly noted the user's level and type of competence (on-line evaluation).

The second phase of the experiment consisted of a post-verbalization supported by the transcript of the dialogues: the expert was brought to look at the transcriptions of the interactions (text and graphics) and comment on them line by line, the on-line evaluations constituting further support for his comments. The aim of this method is to re-create the consultation situation the expert participated in. The expert's comments were focused on interlocutor modelling: he was asked to point out he clues he used for elaborating the model of the interlocutor and to stress the effects of this model on the interaction. A second expert was asked to comment on the dialogues in order to

check the reliability of the evaluations.

Gathering dialogues and having the expert commenting on them have proven to be very beneficial. This method yields rich information concerning the modelling process to be gathered [Cahour 1989]. For MMI2, the analysis of these protocols allows:

- the specification of the content of the user model, i.e. the predicates and arguments that must be included, the stereotypes that categorize the different users and the inference rules that trigger them.
- the description of the elaboration of the UM, i.e. the clues in the discourse or in the drawings of the "user" on which the expert bases his elaboration of the UM.
- the identification of various effects of the modelling on the interaction, i.e. the use of the user model for identifying misconceptions, adapting the explanations (level and quantity), identifying users' plans, managing initiative distribution and disambiguations.

8. Input - Output modes

Once the architecture was defined, design and implementation work has started in parallel on all input - output modes. The following sections review briefly the aims and the first results achieved for each mode.

8.1. GRAPHICS MODE

When designers consult with clients with problems such as our test application of computer network design, they not only talk to them but draw copiously on paper. In the expertise acquisition interviews we made, these notes contain plan and elevation views of buildings, with initially none and then reducing granularities of description of the designed network. The notes also contain lists of numbers and tables for showing the component costs, calculating composite lengths and load on the network. During a consultation both the client and the consulting expert refer to these diagrams and tables verbally, by pointing at them and by drawing signs or 'gestures' on them in 'designer shorthand'.

An analysis of this class of data from design consultations clearly shows that MMI2 must represent the plan and elevation building geography, and the designed network at different granularities. The information displayed in tabular form is sometimes used to calculate exact figures, but is often used to show relative costs, lengths or loads, so not only must tables be used, but also pie charts, histograms and graphs which more effectively convey ratios and relative values. This range of graphical tools should not only present the information to the user, but also allow it to be changed and manipulated. For example, the histogram tool should not present a set of data, but allow the user to drag a bar up or down to change the value represented; the tool representing the network must allow the system to present a designed network, but must also allow the user to state the requirements by placing devices which ought to be connected, and to modify the system's design suggestion. Therefore five graphics tools have been developed to display tables, histograms, line and scatter graphs, and building and network geography.

In order to generalise the intelligent interface to other domains, new CAD tools should be incorporable, and other removable. The graphics tools have therefore been developed to meet 3

design constraints:

1. Individual tools must be accessible for different functions
2. A Graphics Manager which provides a clear interface between all tools and the rest of the MMI2 interface must be provided. New tools can be added to this manager.
3. A window manager is used to facilitate portability. The current window manager used is Sun-View although their are plans to port the system to the more general X-windows manager

In operation, the graphics manager receives CMR packets from the rest of the interface, which are decomposed so a tool can be selected to display the information. The CMR is then translated into the internal representation required for that tool which opens a window and presents the information graphically. The selection of the appropriate tool is made by a rule set which assesses the structure of the information and accesses the User Modeling Expert to determine the preferences of the user. The rules determine the structure of graphs and charts draw on the methods proposed by Tukey (1977), Cleveland (1985), Beach (1985) and Mackinlay (1986). When a user makes a selection or modification this is translated into CMR by the graphics manager which passes the packet down to the Dialogue Controller. Figure 5 shows an interaction with the system illustrating the range of graphics tools.

The graphics tool to allow the presentation and manipulation of building and network geography is the most complex. This allows users to enter building geography as a free hand sketch which is then digitally sampled and adjusted to turn wavy free hand lines into straight lines and the square up angles between these lines. Linked planar maps are used to represent the building and network geography so that either may be viewed and modified separately, as well as together. This approach of allowing free hand input with a smoothing process was chosen rather than the conventional use of a menu selected 'straight line' tool with handles to move it, since it fits the details of the style of the observed experts better.

8.2. INTERFACE EXPERT

The Interface Expert serves three main roles in the MMI2 system:

Declarative Knowledge of the MMI2 Interface is stored in a knowledge base in the Interface Expert. This is called upon by the dialogue controller when the user asks questions about the limitations, abilities, structure or components of the interface itself rather than the domain. For example, if the users asks "What natural languages can I user here?" the knowledge that English, French and Spanish are available would be provided by this knowledge base.

Screen layout management is performed by the interface expert in as far as it provides windowing tools with a position in which to appear on the screen. This overrides the default algorithm in the window manager to position windows in task relevant positions rather than progressively across the screen. The need for this role arises since the windows within the application are overlapping rather than tiled to allow more flexibility in dialogue.

Text interaction in natural language and command language is performed in a pair of windows which are part of the interface expert. Once text is input and a carriage return typed, the string is sent to the appropriate natural or command language mode which returns a CMR packet that is sent to the Dialogue Controller. This provides a uniform image to the user which may not occur if separate

Insert Figure 5 here

Figure 5: Example of an interaction with MMI2. The windows in the upper left and center show a toc in which the user can enter details of buildings and computer networks for those buildings, and i which MMI2 can display design solutions. The windows on the right and bottom show tools for th system to display answers as charts or tables. The window in the middle left supports comman language and natural language interaction. Users can perform design gestures in any of these windows

interaction windows were developed for each text mode. These windows are part of the interface expert rather just another graphics tool since they include the main event handling procedures for the interface, and support the gesture mode in different windows. Consequently the graphics manager actually passes its CMR packets through the Interface Expert to the Dialogue Controller. This provides a clear interface between all modes and the dialogue controller as shown in figure 6.

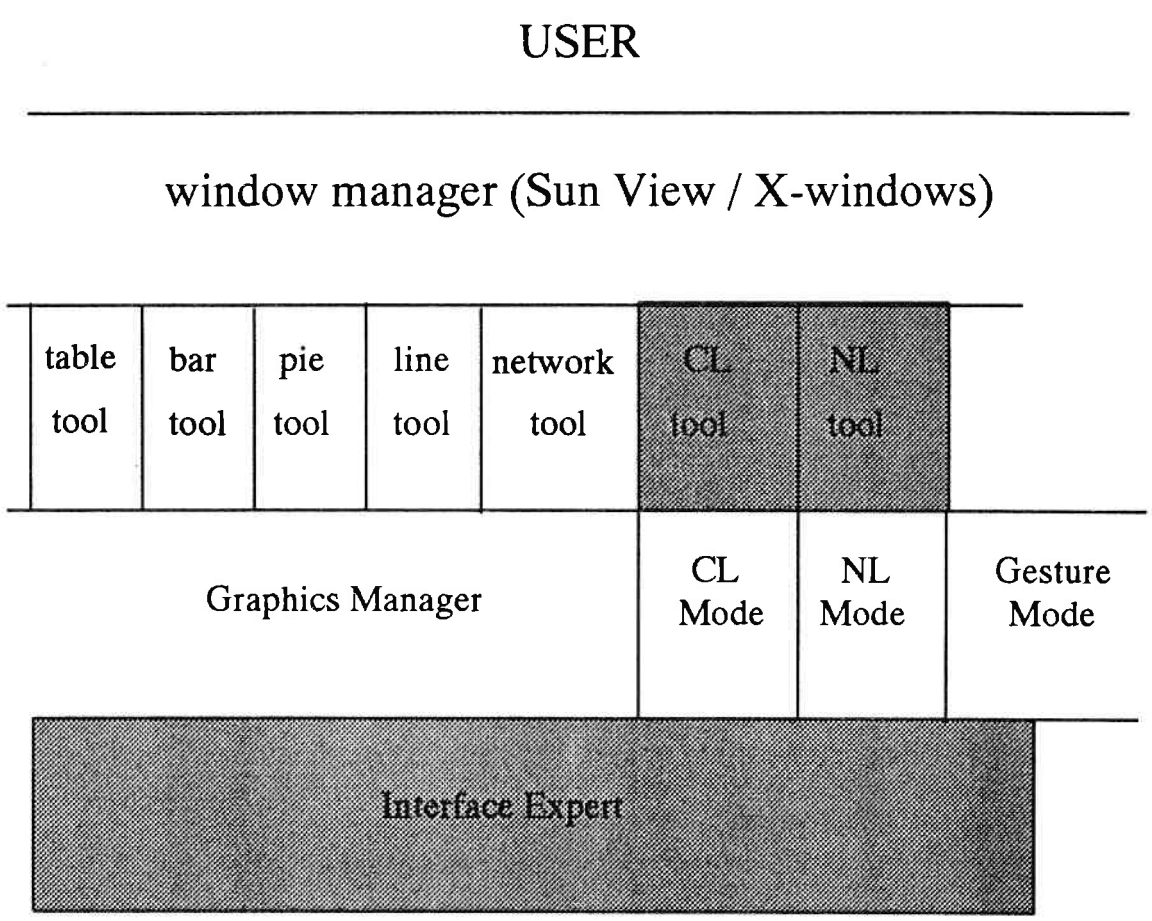

Figure 6: Layered interaction of the graphics and text tools with the window manager. Shaded areas are part of the Interface expert.

8.3. COMMAND LANGUAGE

The purpose of the command language (CL) is to provide users with a language based mode but without the computational overhead associated with a full natural language. The improvement in speed is offset by a cost to the user in terms of syntactic structure: users are required to comply with the syntactic constraints of the CL. The CL comprises a syntactic part and a semantic part. The syntactic part comprises two data sets: a set of operators/actions or commands such as 'add', 'connect' etc. and a set of application objects which are common or proper nouns. A command comprises an action-object pairing. In order to determine whether the command conforms with the

semantics of the application the command is coded in CMR and passed for evaluation. The Semantic Expert determines whether the pairing is legitimate and returns and evaluation.

8.4. GESTURE MODE

As the complexity of graphical representations of a network increases the efficiency the efficiency of natural language modes of dialogue tends to decrease while the use of direct manipulation tends to increase. Designers routinely use a wide range of non-standard but familiar graphical annotations to their work plans in order to edit and modify them. Moreover, they usually retain a cumulative record of this part of the design process using it as an aid to quality control in a design audit trace. Thus, the use of 'design shorthand' is an integral part of the design process and the MMI2 interface permits designers to continue to use this mode of dialogue.

An algorithm has been implemented which allows designer to draw up to 19 different shapes. The shapes are recognized as commands (e.g. 'delete', 'transpose') within the application. Users can draw onto a graphical representation of a plan and make technical modifications. For example, designers can remove graphical objects by 'scribbling' over them or transpose objects by drawing the appropriate shape around the objects to be transposed. The algorithm recognizes the shape drawn by the user (the syntactic part of the process) and pairs it with the objects around which it has been drawn (the semantic part) for subsequent evaluation by the Semantic Expert. The algorithm is scale, location and style invariant. In other words the shapes can be drawn to different scales, in any location using a wide range of individual drawing styles. Improvements in the algorithm will satisfy rotational invariance, allowing users to draw the shapes in a variety of orientations. The algorithm can be used in a variety of applications where designers routinely employ graphical, gestural shorthand within the design process.

8.5. SPANISH LANGUAGE MODE

The Spanish System is envisaged around two components: an analysis module and a generator. The latter is to be implemented from 1992 on.

The analysis module includes a morpholexical component that does an extensive morphological analysis of Spanish - including resolution of collocations and idioms like *cable delgado* (thin cable), *de todas formas* (anyway), *tener en cuenta* (to take into account), etc.

The result of the morphological process is a graph that reveals the set of categorical ambiguities in a given input text. It is worth saying that some of these ambiguities will be solved with the aid of statistical methods. Lexical lookup is performed upon that pruned graph. This component is entirely written in C. The approach adopted is extremely lexicalist. The assumption is that lexical entries are rather huge structures containing a lot of information (selectional restrictions for arguments, semantic typing for CMR formulas, etc.)

The output information from the morphological component is the input to a syntactic parser (developed in BIM_Prolog) that yields a CMR representation by means of a compositional feature-based formalism grounded on functional dependency grammar (cf. Kaplan and Bresnan [1982], Kay [1985]). A formalism has been devised that extensively supports several types of feature manipulation: unification, constraint evaluation, and value overlapping. This parser is currently under development.

Some parts of the parser will interact with other modules of the MMI2 system, in order to solve problems like word sense and attachment ambiguities ellipsis or anaphora, that need contextual and world knowledge.

8.6. FRENCH LANGUAGE MODE

The French processing system will include both the analysis of NL utterances in order to reach a CMR expression and the generation of the expert system answers. The objective of NL analysis is not to build any possible parse, but only the right ones. Concurrently, the generation system has to build a NL answer according to the user expectations.

The approach to French NL analysis has the following characteristics:

- the analyzer is modular; it is made of little modules associated with limited tasks.
- the analysis is driven by the content of the NL sequences: at each step, all the accurate informations are stored, and are used for local prediction, in order to avoid to build spurious solutions.

The steps of the analysis are:

- preprocessing, which normalizes the input NL sequence,
- morphological analysis, associated with a 50 000 words dictionary which tags each word with its lexical entries, grammatical categories, etc.
- disambiguation of the sequence of categories based on a Markovian filter,
- segmentation of a sentence in clauses,
- parsing of each clause through a Earley's automaton which is supervised by a linguistic expert system,
- transformation of the constituent structures in a categorical formalism. Inside this formalism, some operations like anaphora resolution, modification of the word order..., are done in order to reach a CMR expression.

The first state of the analysis, producing the constituent structures, has been implemented in C; the rest of the parser is currently under development.

8.7. ENGLISH LANGUAGE MODE

The English NL input mode of MMI2 will use the English parser of the LOQUI system, developed first within the context of the LOKI ESPRIT project (P 107) [Binot et al. 88], then within an internal development project called BIM_LOQUI.

LOQUI consists of a number of heavily interrelated modules, among which the following main components can be distinguished: morphology, parser and interpretation module on the analysis side and response determination module and generator on the generation side. These processing modules make use of several sources of knowledge: the lexicon, a body of morphological rules, a body of "database mapping rules", a "world model" including general semantic knowledge about word meanings as well as pragmatic knowledge of the application domain and a dialogue memory holding the discourse structure representation.

The parser for English is mainly based on the theoretical principles of Generalized Phrase Structure

Grammar (GPSG) [Gazdar et al. 85] but allows mechanisms from other theories as they seem useful, the aim being to develop an implementation of English grammar which is not only theoretically sound but also computationally efficient. Prolog itself is the formal expression language of the English grammar, mainly for reasons of efficiency.

The work required to adapt the LOQUI parser to MMI2 is rather limited, due to the portable nature of the system. This work, which is currently under way, mainly includes:

- defining an appropriate lexicon for the domain of the application (computer network design);
- defining an appropriate conceptual model for the domain of the application; this is related to the design of the "Semantic Expert" introduced in the general architecture;
- modifying the output of the parser so that it generates CMR expressions;
- improving some aspects of the parsing itself, mainly from the point of view of extending its coverage and/or efficiency.

9. Conclusions

We have presented what should probably be considered as an ambitious architecture for a multimode interface system, including concerns such as user modeling, communication planning, multimodal meaning representation, dialogue context management, etc. We do not necessary expect to develop all above mentioned features to the same degree of completeness within the scope of this project. But we see the architecture itself, which we tried to describe in a clean and modular way, as a kind of reference framework for further research and implementation work on these topics.

The MMI2 project itself will eventually produce an interface prototype illustrating (maybe at various stages of completion) all aspects of the proposed architecture. In this respect, the current phase of the project is focused on the implementation of prototypes for the various components of the system, including the various input - output modes and on the integration of some of these prototypes in a first version of the multimodal system.

10. References

Allen J. (1983) Recognizing Intentions from Natural Language Utterances. In M. Brady & R.C Berwick (Eds) "Computational Models of Discourse". Cambridge MA: MIT Press.

Beach, R.J. (1985), Setting tables and illustrations with style. PhD.thesis, Department of Computer Science, University of Waterloo, Canada

Binot J-L., Demoen B., Hanne K.H., Solomon L., Vassiliou Y., von Hahn W., Wachtel T., (1988), LOKI: A logic Oriented Approach to data and knowledge bases supporting Natural Language Interaction, Proceedings of the ESPRIT88 Conference, North-Holland.

Cahour B.(1989) "Competence modelling in consultation dialogues", Proceedings

of the "Work With Display Units" Second Conference, Montréal, 11-14 Sept. 1989. Amsterdam: Elsevier

Canberry, M.S. (1985) "Pragmatic Modelling in Information System Interfaces." Ph.D. thesis, University of Delaware.

Cleveland (1985), The elements of Graphing Data, Wadsworth Advanced Books and Software: Monterey, California.

Finin T. and Drager D. (1986)."GUMS: A General User Modelling System." Technical Report MS-CIS-86-35, Department of Computer and Information Science, U. of Pennsylvania

Gazdar G., Klein E., Pullum G. and Sag I., (1985), Generalized Phrase Structure Grammar, Blackwell.

Genesereth, M.R., Nilsson, N.J. (1987), *Logical Foundations of Artificial Intelligence* (Morgan Kaufmann Pub., Los Altos, CA).

Grice, H.P. (1975) Logic and Conversation. In P. Cole and J.L. Morgan (eds.) "Syntax and Semantics", volume 3, pages 64-75, Academic Press, New York

Julien C. and Marti J-C. (1989),: Plan revision in person-machine dialogue, Proceedings of the Fourth ACL European Chapter Conference, Manchester.

Kaplan R. and Bresnan J. (1982), Lexical-functional grammar: A formal system for grammatical representation, in J. Bresnan (ed.) The Mental representation of Grammatical Relations. The MIT Press.

Kaplan S.J. (1983) Cooperative Responses from a Portable Natural Language Database Query System. In Brady, M. and Berwick, R.C. (eds) "Computational Models of Discourse". Cambridge MA: MIT Press.

Kass, R.J. (1988) "Implicit Acquisition of User Models in Cooperative Advisory Systems" Technical Reports MS-CIS-878-05, Department of Computer and Information Science, U. of Pennsylvania

Kass, R.J. (1988) "Acquiring a Model of the User's Beliefs from a Co-operative advisory Dialogue."Technical Reports MS-CIS-88-104, Department of Computer and Information Science, U. of Pennsylvania

Kay M. (1985), Parsing in functional unification grammar, in D. Dowty, L. Karttunen and A.Zwicky (eds.) Natural Language Parsing. Cambridge

U.Press, pp.251-278.

Mackinley, J. (1986), Automating the Design of Graphical Presentations of Relational Information, ACM Transaction on Graphics, 5(2), 110-141

Pollack, M.E. (1986) "Inferring Domain Plans in Question-Answering". Technical Report MS-SIC-86-40, Department of Computer and Information Science, Univ. of Pennsylvania.

Sparck Jones K. (1987) "Realism about User Modelling". Technical Report no.111, University of Cambridge Computer Laboratory.

Tukey (1977), Exploratory Data Analysis, Addison-Wesley: Reading, Mass.