Technical Report
RAL-TR-96-011

Intelligent Multimedia Presentation Systems: A Proposal of Reference Model

S Ruggieri M Bordegoni G Faconti T Rist P Trahanias and M Wilson

February 1996

COUNCIL FOR THE CENTRAL LABORATORY OF THE RESEARCH COUNCILS

Enquiries about copyright, reproduction and requests for additional copies of this report should be addressed to:

The Central Laboratory of the Research Councils
Library and Information Services
Rutherford Appleton Laboratory
Chilton
Didcot
Oxfordshire
OX11 0QX
Tel: 01235 445384 Fax: 01235 446403
E-mail library@rl.ac.uk

ISSN 1358-6254

Computer Graphics Network, Task2,
Knowledge Engineering and Graphics Recognition

Intelligent Multimedia Presentation Systems

A proposal of reference model

Salvatore Ruggieri [1]
Monica Bordegoni [2]
Giorgio Faconti [3]
Thomas Rist [4]
Panos Trahanias [5]
Michael Wilson [6]

February 6, 1996

Correspondence address: Giorgio P. Faconti, CNR - Istituto CNUCE, Via S.Maria 36, 56126 Pisa ITALY. e-mail G.Faconti@cnuce.cnr.it

The work reported on this paper was undertaken as part of Task 2 of the ERCIM Computer Graphics Network on Graphics and Knowledge Engineering partially funded by Grant number CHRXCT93-0085 from DG XII of the EC under the Human Capital and Mobility Program.

[1] Dipartimento di Informatica, Università di Pisa, Italy and Rutherford Appleton Laboratory, United Kingdom.
[2] Consiglio Nazionale delle Ricerche ITIA, Milano, Italy
[3] Consiglio Nazionale delle Ricerche CNUCE, Pisa, Italy
[4] DFKI, Saarbrücken, Germany
[5] FORTH-ICS, Greece
[6] Rutherford Appleton Laboratory, United Kingdom.

Contents

1 Introduction

This document presents a proposal of a *reference model* for *intelligent multimedia presentation systems (IMMPSs)*. Many papers in the literature consider these systems, and some large-size prototypes have been developed (MMI^2, WIP, MIPS, and COMET). However, no generic model has been proposed so far. Each project began from scratch, relying only on the past experience of the members. This has meant a lack of uniform logical approach to the problem. Consequently, there is no agreement on the terminology to be used, on the functional definition of an IMMPS and on its logical architecture.

This paper attempts to fill this gap, by proposing a reference model for these systems. There are several advantages for an agreement on a reference model. The development and the analysis of systems will benefit from a uniform approach to the problem. The modular development of large-size programs will be feasible, as each module has a well defined role, with well defined interfaces and communication language with the other modules. From a theoretical point of view, the analysis and comparison of systems will be made on the basic of a unique, general architecture and by means of a common terminology. These considerations are universally valid for any reference model. In the case of IMMPSs, however, they assume a deeper significance, as the literature is still confusing on the basic terminology.

What we model is that part of a computational system whose tasks are to present information to the user in an *effective* way. The presented information and the way it is presented have to be tailored to the user, i.e to his needs, knowledge, attention, etc., and to the context.

The survey by Roth and Hefley [24] provides a first attempt to overview these systems in a uniform and general way. However, the aims of this document are different. First of all, we are going to give *precise* definitions concerning the terminology and the architecture, and the background necessary to understand them. We start from a definition of presentation systems, and then incrementally add the notions of multimedia and intelligence. The latter is characterized by the use of explicitly encoded knowledge.

The reference architecture for IMMPSs is composed of four layers - control, content, layout and presentation - and four "expert modules" - application, user, context and design. The generic notion of "expert module" is introduced, with the intention to model the provision of shared knowledge sources in a client-server (or ask-tell) fashion.

The guidelines in our proposal are aimed towards:

- giving a modular organization of the system, in order to make feasible the development and the comparison of practical large-size systems,
- introducing a minimum set of explicitly encoded knowledge, called *knowledge sources*, in order to characterize the term "intelligence" in *intelligent multimedia presentation systems*,
- locating an owner component for each knowledge source - when possible,
- separating each shared knowledge source in a distinct component, called "expert module", which provides the knowledge in a client-server - or ask/tell - fashion. These components can be shared with other systems. As an example, the component storing the user knowledge is likely to be shared among all the applications running on a system (personal computer,

distributed system, or even network.) We model these components taking into account the mentioned considerations.

- identifying relations with existing or potential standards. On the one hand, some black box in it may be filled by existing standards, such as the ones for output devices (ex., *Computer Graphics Reference Model* ([6]), MIDI, etc.), and standards for knowledge exchange (like the *Knowledge Query and Manipulation Language* ([1])). On the other hand, the model can itself be a component of other models. For instance, it fills most of the *Run-time Layer* of the *Dexter Hypertext Model*, or any of its extensions, like the the *Amsterdam Hypermedia Model.*

Many concepts in this paper are not totally new. Wherever possible, we consistently adopt terminology and concepts from other studies on information ([15]), multimedia ([2, 23, 5]), computer graphics ([6]), user models ([10, 16]), and knowledge based systems ([11, 32, 30]).

In conclusion, the model is intended to fit *real* systems, like the cited ones. This is the main aim of this work, and we constantly have been guided towards such a goal. However, this *does not necessarily mean that the proposed architecture is to be followed in an implementation.* It is only a logical view of these systems. A reference model cannot be forcing the implementation of a system.

The paper is organized as follows. First, we give some basic definitions for *media terms* and *presentation systems.* Then we add knowledge moving towards *intelligent multimedia presentation systems.* In section 3, a reference architecture is proposed: it is composed of a *knowledge server* - specified in section 4 - and four layers - specified in section 5. In the next section, some aspects of real systems are described in terms of the reference model. Finally, in appendices A and B the MMI^2 and WIP systems are presented with our terminology and architecture, respectively.

2 Presentation Systems

A way of defining IMMPSs is to start from a definition of *presentation systems*, and then incrementally add the notions of *multimedia* and *intelligence*. In this section, we give a *functional* definition of IMMPSs by means of this strategy. The understanding of the reader should be helped by the clarity and relative simplicity of each step, as only one new concept is added at a time. The following section will define a reference architecture according to the fundamental notions we are going to introduce.

2.1 Media terms

First of all, we need to agree on some basic definitions on media terms. Even for the very fundamental notion of *media*, the definitions presented in the literature do not agree. Very often the term is confusing: Most of the times the term is distincted into two components, media and modality, which are not clearly perceivable as two distinct concepts. The reason of this misunderstanding lies on the fact that the term "media" is used with different meanings in different contexts, such as in computer science, psychology and telecommunications. We refer the reader to [24] for a deeper discussion of this topic.

We avoid the distinction between *media* and *modality*, since we do not see any way of characterizing properly the boundary among these terms.

---

medium a single mechanism by which to express information (e.g. spoken and written language, tables, maps, photographs, pie charts, graphs, movies, etc.)

multimedia adj., referring to the use of multiple media.

simple exhibit what is produced by a one invocation of a medium (e.g., a paragraph of a text, a diagram, a computer beep, a picture etc.)

exhibit a collection or composition of several simple exhibits.

information carrier that part of a single exhibit which communicates the relevant piece of information.

carried item that piece of information represented by the carrier.

context the background to a communication.

substrate the background to a simple exhibit which establishes physical or temporal location, and often the semantic context, within which information is presented.

channel an independent dimension of variation of a particular information carrier in a particular substrate.

parallelism the number of channels gives the total number of independent pieces of information the carrier can convey. This is a measure of the degree of parallelism of a medium.

fission the process of splitting an exhibit into simple exhibits.

fusion the process of interpreting a collection of simple exhibits as a whole.

abstraction a process that transforms information into information whose semantic content and scope are richer/higher than the content and scope of the initial information.

Discussion

Other terms

We do not claim that this list of terms is complete. Several other notions - e.g., *cross-media references, coherency between media, anaphora, diexis, redundancy, etc.* - are currently used. Nevertheless, an extensive characterization of media terms is out of the scope of this paper. We restrict ourselves to consider a set of very basic terms, which should be sufficient for our purposes.

Information carrier, carried item and substrate

The information carrier of a particular simple exhibit may be different for the producer and the consumer of that exhibit. The same can happen for the notions of carried item and substrate. When this mismatch arises the communication between producer and consumer is likely to fail.

2.2 Presentation systems

In the following we introduce the fundamental notions of *goals* and *presentation systems*. These are designed for achieving goals by means of a presentation.

---

goal a high level description of a collection of data together with the purpose or intention for communicating information.

achieving a goal presenting to the user the data described by a goal in a way that achieves the purpose or intention for communicating information.

presentation the result, perceivable by the user, of achieving a goal.

presentational command a command to the system whose execution affects the presentation process (ex., any-time interruptions, design constraints, etc.)

application the external system producing application data as input to the presentation system.

goal formulation the external system producing goals and presentational commands as input to the presentation system.

presentation system a system for achieving goals and executing presentational commands.

rendering function of transforming an internal logical representation of a presentation in the primitives of a set of devices.

devices interfaces between the presentation system and the user. Generally, they are the physical artefacts necessary to the system to deliver the presentation (e.g., display, speaker, printer etc.)

user a person interpreting a presentation.

multimedia presentation systems a presentation system that generates multimedia presentations.

Discussion

Goal formulation

Actually, the purpose to communicate information can be partly embedded in the presentation system (e.g., let the user be aware of some information, etc.)

User as a person

The definition of *user* could be stated requiring a user to be *any* external object interpreting a presentation. In principle, indeed, it could be another computational system. However, as we are dealing with high-level communications it seems correct to assume the user to be a person.

Presentational commands

Through these commands, we model requests like interruptions of the presentation process, or communications of constraints, etc. These interactions are hardly formalizable in terms of the only notion of goal.

2.3 Adding knowledge

The last point towards a full definition of IMMPSs is to introduce the notion of *knowledge*. This is fundamental in order to characterize IMMPSs as "intelligent" systems and to differentiate them within the larger class of presentation systems.

---

knowledge justified true beliefs of an abstraction of data.

knowledge sources the following explicitly encoded knowledge:

user model knowledge about the user (goals and plans, capabilities, attitudes, knowledge or belief.)

context model knowledge about the context (discourse model, context references.)

application knowledge knowledge provided by the application and knowledge for reasoning about application data (term interpretation, data characterization.)

design knowledge general knowledge about the design of multimedia presentation (design constraints, cognitive theory, media model, device model.)

media specific design/realization knowledge knowledge about the design/realization for a specific media in a multimedia presentation.

system model knowledge about the system's characteristics relevant to the presentation process.

media selection knowledge knowledge for identifying an effective and coordinated collection of media for the representation and presentation of application data.

intelligent multimedia presentation system (IMMPS) a multimedia presentation systems that designs a presentation exploiting the knowledge sources to achieve goals.

Discussion

Other knowledge sources

The knowledge present in an IMMPS may include other explicitly encoded knowledge than those described as *knowledge sources*. For instance, we will discuss in section 4.2 that a *planning* and a *content selection knowledge* might be present in a system. However, the classified knowledge sources seem to be the most relevant knowledge, and the most used in real application. Therefore, we defined IMMPSs as systems exploiting *at least* the knowledge sources. Additional knowledge may be present - and often is - but we do not force it to be in every IMMPS.

Intelligence of an IMMPS

The degree of knowledge exploited in an IMMPS gives an idea of how "intelligent" it is. Unfortunately, it is impossible to give a formal definition of a "measure of intelligence", as it depends on how well the goals are fulfilled. This "how well" is related to the user perception and understanding of the presentation, and then it is a relative concept.

IMMPSs and presentation systems

It is worth noting that there is no difference between IMMPSs and presentation systems with respect to their interface with the goal formulation. This component does not have to know whether or not the presentation systems is "intelligent", in order to generate a goal. The alternative choice of differentiating the interface for IMMPSs would not be modular, since it would not allow to replace a presentation system with a (further revised) intelligent presentation system.

3 Architecture model

Until now we considered "what" an IMMPS is. From now on, we deal with "how" an IMMPS is conceptually designed. Our objective is to give a reference architecture, i.e. a generic model of this class of systems. Designing a reference architecture implies relevant practical and theoretical consequences.

The design and implementation of new systems will gain enormous advantages from the existence of a reference model. The design process is more clear, the role and the functionalities of each part of the systems are well-understood and unambiguous, the life-time cycle of the project needs less feedback between design and implementation. Finally, huge-size projects are made feasible by the existence of an unambiguous and modular organization of the system.

Theoretically, a reference model allows for comparing systems, by means of a single terminology and a single conceptual architecture. This is a step forward in understanding and characterizing a class of systems.

The guidelines in our proposal are aimed towards:

- giving a modular organization of the system, in order to make feasible the development and comparison of practical large-size systems,
- locating an owner component for each source of knowledge - when possible,
- separating each shared knowledge source in a distinct component, called "expert module", which provides the knowledge in a client-server - or ask/tell - fashion. These components can be shared with other systems. As an example, the component storing the user knowledge is likely to be shared among all the applications running on a system (personal computer, distributed system, or even network.) We model these components taking into account the mentioned considerations.

In the following, we give some basic definitions about architecture terms, knowledge and computation and client/server architecture. Next, a general view of the IMMPS reference architecture is presented.

3.1 Basic glossary

---

reference architecture, layer, module, submodule, component a reference architecture is a collection of guidelines and constraint rules for a computational system. A layer is a conceptual, gross categorization of building blocks for the system. These blocks are either modules or interconnection structures. A module is composed of sub-modules and interconnection structures, and so on. A component is a layer, or a module or a submodule.

notification message stating the result of an operation, possibly together with additional informations - e.g., what caused the failure, possible repair strategies, statistics, etc.

3.2 Knowledge and Computation

First of all, we recall some definitions regarding knowledge based systems (KBSs). The essential characteristic of a KBS is that knowledge representation and the means for knowledge manipulation are separated.

---

knowledge base resource comprising a collection of machine-processable explicitly encoded knowledge, often in the form of rules and facts that describe relations and phenomena in a domain, and possibly also methods and heuristics for problem solving.

inference engine module managing, storing and retrieving explicit and implicit knowledge in a collection of knowledge bases.

integrity checking module for observing a (collection of) knowledge base(s) and creating a triggering action when its state changes in an inconsistent way.

knowledge based system collection of (several) knowledge bases, an inference engine and possibly an integrity checking.

We anticipate that we will be interested in servers which provide knowledge. They will be a basic component of our model. Therefore, we need to introduce some concepts related to client-server interactions.

---

client a component that asks for services.

server a component that provides services.

service function provided by a server for a client, directly or indirectly. Different sets of services may be provided to different clients or to the same client in different times.

knowledge exchange format a language and protocol for exchanging information and knowledge. In particular, in a communication between a knowledge base system acting as a server and a client component, we distinguish several classes of messages: *ask, tell, assert, unassert, deny, justify, bundle, answer* ....

ask the client wants to know all the values for which a sentence with possibly free variables is true in the knowledge base.

tell the server reports the answer to an *ask* request.

assert the client wants to assert an axiom in the knowledge base.

unassert the client wants to remove an axiom from the knowledge base.

deny the client wants to make a sentence neither an axiom of the knowledge base, nor derivable by inference.

justify the client wants a justification for a sentence which is true in the knowledge base.

tell-justify the server reports the answer to a *justify* request.

ask-do-not-care the client wants to know one value for which a sentence with possibly free variables is true in the knowledge base.

tell-do-not-care the server reports the answer to an *ask-do-not-care* request.

Knowledge exchange format

The exchange of information among the components of the system we will introduce is by means of some *knowledge exchange format* (e.g., the Knowledge Query and Manipulation Language [1].) This format is not explicitly dealt with. We only assume it underlies each arrow between communicating components.

Services

The classification of the messages between a client and a server gives a gross characterization of the input-output functionalities of those components. A further phase in the development of a formal reference model should investigate, by means of some specification language, more detailed the specification of each black box in our architecture.

3.3 IMMPS = Layers and Experts

Figure 1 introduces the IMMPS reference architecture. It is composed of four layers - control, content, layout and presentation - which, in addition to their private knowledge, exploit explicitly encoded knowledge provided by the knowledge server - composed of four "expert" modules: application, context, user and design. The system receives goals and presentational commands from the *goal formulation*, designs a presentation asking the *application* for application knowledge, and finally communicates the presentation to the user.

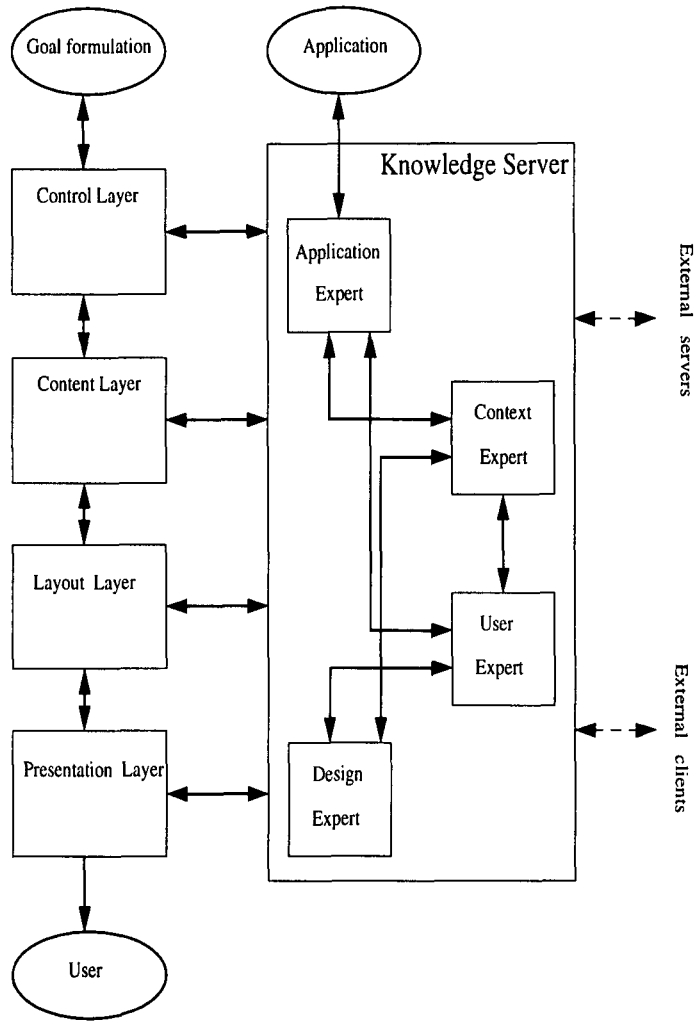

Figure 1: *IMMPS: general architecture*

As a first convention, we stipulate that the arrows that are not pointing anywhere in the charts are intended to point to each component of the picture. Moreover, the exchange of information among the components of the system is by means of some *knowledge exchange format* (e.g., the Knowledge Query and Manipulation Language [1].) This format is not explicitly dealt with. We only assume

it underlies each arrow between communicating components.

IMMPS reference architecture the reference architecture defined in this document. Figure 1 shows a gross view of its components. It is composed of four layers - control, content, layout and presentation - and the knowledge server - composed of four expert modules: application, user, context and design.

Discussion

Goal formulation.

Although the *goal formulation* and the *application* play different roles, in most real systems they are the same object, usually called "application".

Our distinction underlines this difference: application is strictly that part of the system concerning the application knowledge - often consisting of a knowledge based system, or only of a database - whereas the goal formulation is that part concerning the use of that knowledge.

As an example, in the MMI^2 system - see appendix A and [20] - the application fits the *application KBS* and the goal formulation fits the input-interpretation part of the system.

External servers/clients.

It is necessary to formalize some interactions of the system with *external servers/clients*. This is the case, for example, when the user expert acquires knowledge directly from the user, or from an input subsystem, or when it is shared with other systems. All the other experts have such kind of interactions as well. In the pictures, the interactions with unspecified systems - called *external servers/clients* - are denoted by dashed arrows. Our solution is analogous to the one adopted in the Computer Graphics Reference Model [6], where the unspecified interactions are modeled by means of *data capture metafiles.*

Knowledge in an expert module.

We anticipate that the main characteristics required for the knowledge stored in an expert module are: explicitly encoded, user and context -dependent, shared. The generic architecture of an expert module will be shown in section 5.1. Here, our concern is to make clear that the knowledge in the knowledge server is not the *only* knowledge in an IMMPS. Each component of a layer has its own private knowledge; for instance, we will show that the content layer stores the media selection knowledge.

4 Layers

The *expert modules* act as servers of knowledge for the four layers of the reference architecture - control, content, layout and presentation. In addition to the knowledge provided by the *knowledge expert*, each layer exploits some other private knowledge to the end of carrying out a presentation of a goal, and executing presentational commands.

Each layer acts a transformation of the input, as the following figure sums up.

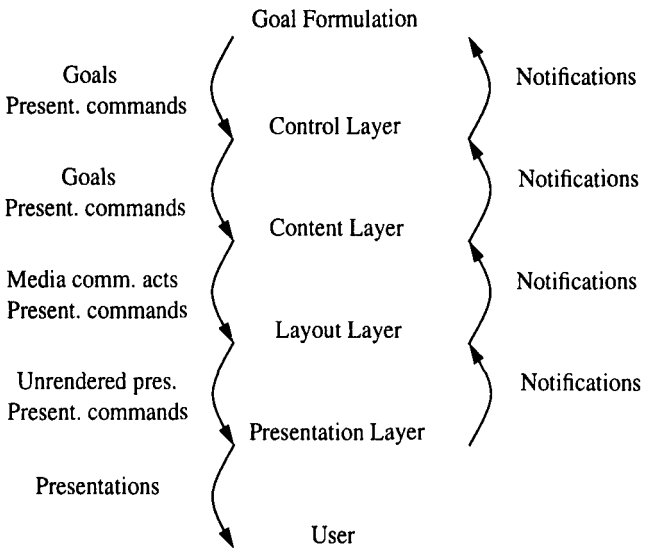

Figure 2: *Information flow through the layers*

Starting from a goal given by the goal formulation, the control layer transforms it into simpler goals. For each of these, the content layer plans a collection of media communication acts, i.e. communication acts enriched with semantic/logic content, media selection and relations between them. The layout layer designs the layout of the presentation, transforming the media communication acts into an internal representation of the presentation - called unrendered presentation. Finally, the presentation layer renders the presentation to the user.

On the other hand, each layer notifies to the layer above the result of its processing - success or failure - possibly together with some additional informations - causes of failure, recovery strategies, explanations, etc.

The definitions of the information exchanged through layers are to be understood in a broad sense. We will give only an abstract characterization of such "data". It is not our intention to force any implementation. For instance, we do not force, in a real system, the *goals* passed from the *goal formulation* to the *control layer* to be of the same type as the *goals* passed from the *control layer* to the *content layer*. We require only that they satisfy the definition of goal given in section 2.2. The same reasoning applies to the notifications passed upward by the layers: We do not require all of them to be expressed in the same format.

4.1 CONTROL LAYER (CL)

The control layer consistently coordinates the presentation process in time: Its task is to choose the next goal to be achieved or the next presentational command to be executed.

Goals and presentational commands from the goal formulation are analysed - possibly transformed or decomposed into simpler ones - and stored in the context model. Then the next goal to be achieved is chosen and sent to the content layer. From this, the control manager receives notifications, which are analysed and collected to produce notifications to the goal formulation.

The presentational commands are sent by the control layer to each component of the system involved in their execution. In principle, we assume that these interactions happen through the connections of the reference architecture. The control layer is directly connected to each expert of the knowledge server; the layout and presentation layer are reached by simply passing through the content and the layout layers, which will pass forward the presentational commands and backward the related notifications.

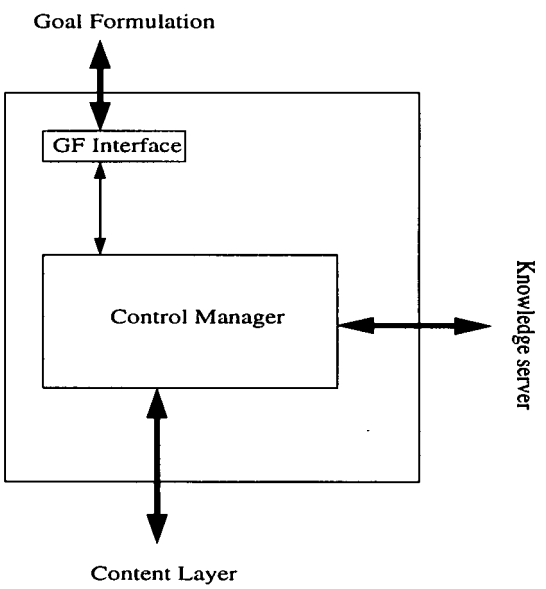

Figure 3: *The Control Layer*

Control Layer the task of this layer is to choose the next goal to be achieved or presentational command to be executed. It receives goals and presentational commands from the the *goal formulation*, and notifications from the *content layer*. It notifies to the *goal formulation* the result of the presentation process.

GF Interface the interface between the *goal formulation* and the *control manager*, performing simple scheduling operations on the interactions and/or transformations of the content of the messages.

Control Manager this modules receives goals and presentational commands from the GF Interface. They are possibly transformed - split, decomposed in simpler ones - and stored in the *context expert*. The control manager decides which is the next goal to be achieved or presentational command to be executed. Then it sends the goal to the content layer, or the presentational command to the components involved in its execution. From them and from the content layer, the control manager receives notifications, which are analysed - possibly generating a recovery strategy in case of failure - and collected to produce notifications to the GF Interface.

Discussion

Next goal to be achieved

The goals received by the goal formulation may be split in the control manager in a collection of "simpler goals". Therefore, the goals passed to the content layer may not be the same with the ones received by the goal formulation.

GF Interface

The intended role of this component is to convert the messages to be exchanged between goal formulation and control manager in the appropriate format. This task is not carried out by the underlying knowledge exchange format, since the transformation depends on the specific representation language(s) used in the IMMPS.

Knowledge in the control manager

Deciding the next goal to be achieved can be a very simple task: It could involve only popping a goal from the discourse model in the context expert. On the other hand, some private knowledge may be present in the control manager, if the decision involves more complex reasoning.

4.2 CONTENT LAYER (CnL)

The objective of the content layer is to build up a sequence of actions to the end of achieving a goal by means of generating media communication acts. These are communication acts enriched by semantic/logic content, media selection and relations between acts.

Communication acts are an extension of speech acts [25] to multimedia communications. For a gross classification of communication acts we refer the reader to [19].

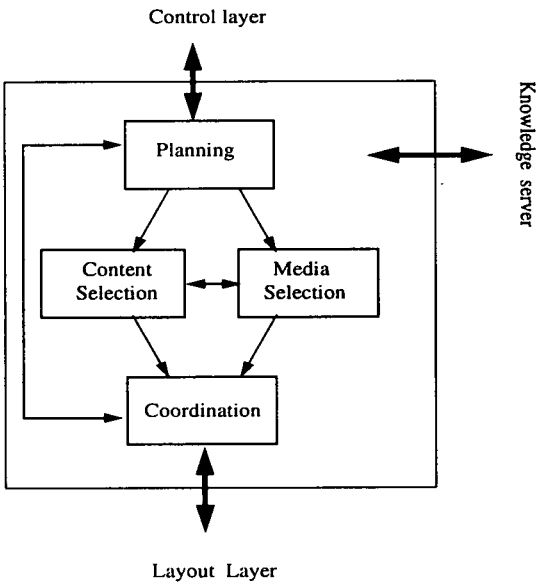

Figure 4: *The Content Layer*

Content Layer starting from a goal passed by the control layer, the content layer generates media communication acts as an input to the layout layer. The task is accomplished by means of coordinated planning, content and media selection. It receives notifications from the layout layer and notifies to the control layer.

Media communication acts communication acts enriched with semantic/logic content, media selection and relations between acts.

Planning starting from a goal passed by the control layer, a sequence of actions is planned to the end of achieving the goal. *Communication acts* and *relations between them* are produced as an input to content and media selection and coordination. From the coordination, the planning receives notifications. If the result of the following stages in the presentation process is positive, then the notification is passed to the control layer. Otherwise, an alternative strategy can be followed in order to achieve the goal.

Content selection the function of this module is to effectively select a content for (parts of) communication acts, according to the relations between them, and in coordination with the media selection. As a result, communication acts enriched with semantic/logic content are passed to the coordination.

Media selection the function of this module is to effectively select a coordinated collection of media - i.e., a collection of media related by some spatial, temporal, cognitive relationships - for (parts of) communication acts, according to the relations between them, and in coordination with the content selection. As a result, communication acts enriched with the selected media are passed to the coordination. This component stores the media selection knowledge.

Coordination this submodule merges the communication acts passed by the content and the media selection, according to the relations between them passed by the planning. The results - called media communication acts - are passed to the layout layer. From this, it receives notifications, which are passed to the planning.

Discussion

Planning and coordination

The strategy followed by the planning module can be influenced by the possibility to perform some actions. If an action (selecting the appropriate content/media, structuring the layout in a way that any two objects do not intersect, etc.) cannot be performed (there is lack of information to retrieve the correct content/media, the layout has an unwanted structure, etc.) then the coordination communicates to the planning the failure - possibly with some additional information. A trade-off is initiated among planning, content/media selection, coordination and layout layer.

Planning and content selection knowledge

We did not introduce a "planning knowledge" or a "content selection knowledge". Basically, the function of these two components consists of asking the application expert for the interpretation of

the goal or the suitable content. However, it may happen that the application expert has not enough knowledge to perform the task, or the communication acts and the relations between them are too complex to be simply passed forward, or - as in the MMI^2 system - the planning performs some kind of analysis (mainly, plan recognition, heuristics for error detection or uncomplete specification etc.) on the goal by exploiting some private knowledge.

4.3 LAYOUT LAYER (LL)

In this layer, a layout for a presentation is designed, according to the media communication acts passed by the content layer. The layout layer is composed of the layout manager, the media design and realization and the coordination. The *Layout manager* designs the general layout of the presentation. It also passes medium specific communicative acts to the responsible generation components, eg. for text, graphics, audio, etc. Each generation component is further subdivided into a design and a realization part. The result is coordinated to the end of generating an unrendered presentation to be passed to the presentation layer. From this, the layout layer receives notifications. Finally, it sends notification to the content layer, to report the result of elaborating media communicative acts.

A trade-off is possible - and in a real system, often necessary - among layout design, media design, media realization and coordination.

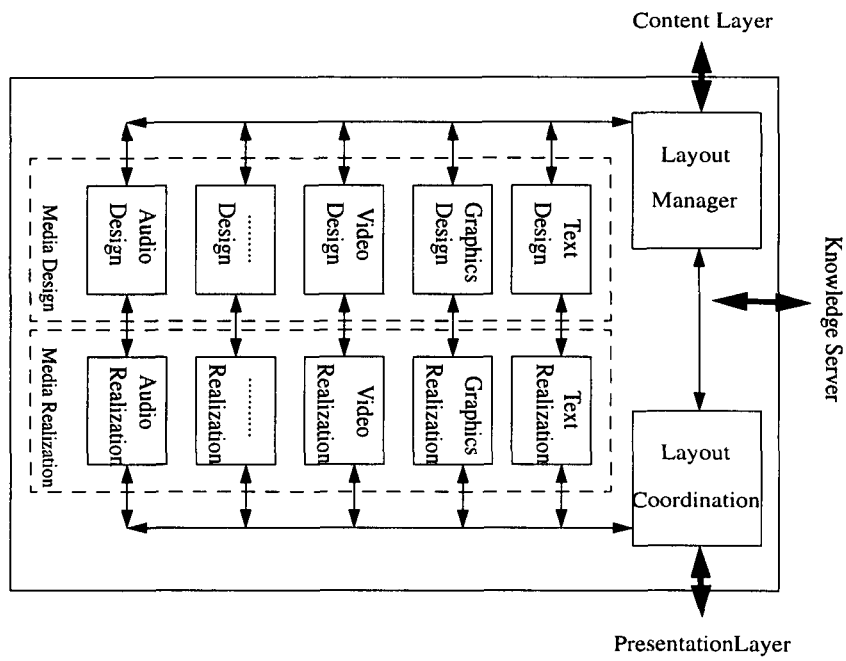

Figure 5: *The Layout Layer*

Layout Layer the task of this layer is to satisfy the media communication acts passed by the content layer by means of coordinated layout design, media design and media realization. As a result, (some) unrendered presentations are passed to the presentation layer. It is notified by the presentation layer and notifies to the layout layer the result of elaborating the media communication acts.

Unrendered presentation an internal representation of a piece of presentation, with some presentation instructions.

Layout Manager starting from the media communication acts passed by the content layer, the task of this module is the design of the layout structure of the presentation, together with dispatching of the media communication acts to the media design and realization involved, and notifying to the content layer.

Media Design collection of coordinated design components, one for each media. Each of these components stores its own media specific design knowledge.

Media Realization collection of coordinated realization components, one for each media. Each of these components stores its own media specific realization knowledge.

Layout Coordination coordination of the resulting media realization and layout structure, to the end of producing an unrendered presentation as an input to the presentation layer. It receives notifications from the presentation layer and notifies to the layout manager the result of the presentation process.

Discussion

Reply to a communication act

A media communication act may require an answer to the content layer - communicated as a notification message. This way we model those situations in which the planning or the media/content selection depend on information stored in the layout layer. As an example, if the content layer needs to know if two pictures intersect then it has to query - with a media communication acts - the layout manager, which will retrieve this information from its own knowledge or from the media design/generation components.

Unrendered presentation.

Achieving a goal means producing a presentation. This could involve splitting the goal in the *control layer* into several simpler ones and separately achieving each of them. In this sense, the unrendered presentation cannot be said to be an internal representation of *a* presentation, but only of a piece of it, since at the *layout layer* we only have a restricted view of the presentation process - i.e., achieving simpler goals.

4.4 PRESENTATION LAYER (PL)

The presentation layer renders a presentation, i.e. transforms an internal representation of it in terms of the device primitives, according to some presentation instructions. A coordination module dispatches each piece of presentation to the suitable device interface(s), according to the presentation instructions and in an effective way - exploiting the knowledge provided by the knowledge server. For each device, a device interface is present, e.g. *display, speaker, printer, etc.*

The outputs of the device interfaces are directed to the user. In Fig. 6 the arrows are merged to make clear that the result perceivable to the user is the *coordinated* fusion of those outputs.

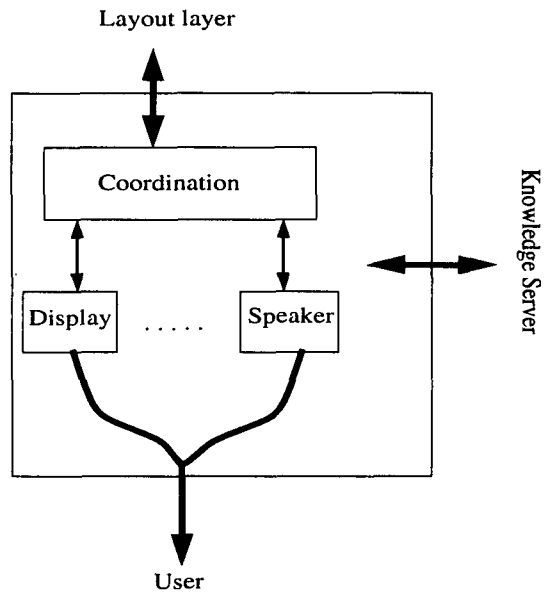

Figure 6: *The Presentation Layer*

Presentation Layer rendering of an *unrendered presentation* passed by the layout layer, according to its presentation instructions. At the end of the process it notifies the result to the layout layer.

Coordination coordination of the rendering process, and notification of the result to the *layout layer*.

Display, Speaker, etc. device interfaces.

Discussion

Coordination.

Coordination has to be understood in a broad sense. As an example, it dispatches pieces of the unrendered presentation to the device interfaces, according to the presentation instructions and using the knowledge provided by the knowledge server, such as user's preferences. Also, another functionality might be temporal coordination.

Notification.

The result of the presentation process is not directly communicated from the presentation to the control layer for three reasons.

The first is that the two layers are not directly connected.

Even though they were directly connected, we point out that achieving a goal could involve more than just the rendering of only one *unrendered presentation*, as discussed in the last section.

Finally, if the rendering process fails, then the content or the layout layer might decide to follow an alternative strategy. In this case, they have to be notified.

5 Knowledge server

The *Knowledge server* provides the layers of the reference architecture with several types of explicitly encoded context- and user-dependent knowledge which is shared among several (or even all) components.

The *Knowledge server* consists of four expert modules, designed along the lines of *knowledge bases*. Each of which represents knowledge on a particular aspect of the presentation process: application, user, context and design. They all share the same general structure. In addition to the core knowledge base system, there are some interfaces with clients - for providing services - and servers - for requiring services.

The experts provide the clients with their knowledge in a client-server (or ask-tell) fashion, tailoring the answers to the user model and to the context. They may provide and require services from external knowledge sources, such as the application or the user himself or other system. In this sense, they are independent modules, easily integrable with other systems which share the same knowledge.

In the following, we will present the general structure of an expert module, and will instantiate it for each module of the knowledge server.

5.1 Expert modules

The expert modules all share the same general structure in their logical architecture. The overall structure is independent from the particular systems we are describing, namely IMMPSs and, in principle, the modules differ only in the knowledge which they store and in the interface operations. In the following we present this structure together with a functional definition of its components, its instantiation in each layer, and a characterization of its interface operations.

A generic architecture

Figure 7 shows the overall logical structure of a generic expert module.

The core of the expert module is obviously, the knowledge it represents. This knowledge can consist of a number of logically distinct knowledge bases, each for a logically different aspect to be dealt with, or only a single knowledge base, when such distinction is not relevant. The *inference engine* provides a uniform and general view of the stored knowledge and of that inferable from it. Maintenance involves incorporating new knowledge, which could be inconsistent with the current state. If this is the case, then inconsistency must be resolved by some triggered action performed by an *integrity checking* sub-module.

The three components - knowledge base, inference engine and integrity checking - are referred to as the Knowledge Base System (KBS) of the expert. This is the usual terminology in the literature on the KBSs.

The other components of an expert module are interfaces: *server, user expert, context exper* and *acquisition interface*. The first one is the interface with the clients, whereas the other three ones are interfaces with servers.

The experts provide services to the other components of the system - the clients - through the *server manager*. This module receives requests from clients. The clients are the layers of

the generation pipeline or other experts or external entities, namely the external sources or the application. The *server manager* transforms the interface operations into (a collection of) messages to the *inference engine*, collecting the answers, and responding to the client.

On the other hand, an expert module requests services from other servers.

As its knowledge has to be tailored to the user and context, two interfaces - respectively with the user expert and the context expert - allows the inference engine to acquire such knowledge.

Finally, the *acquisition interface* supports access to other servers. This is necessary when the knowledge of the expert depends on other factors in addition to user and context. As an example, the application expert has to interact with the application - as a client - to request or to assert application data.

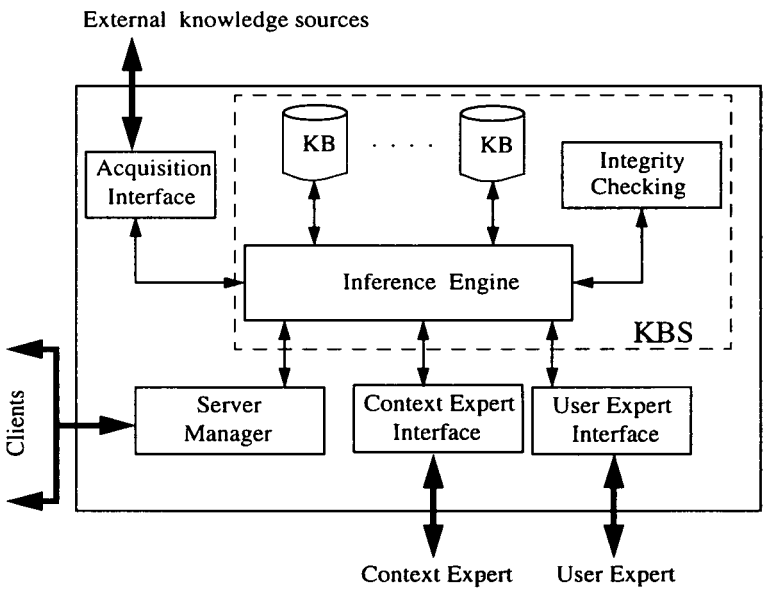

Figure 7: *General structure of an expert module*

In the following, a functional definition of each component of the architecture is given.

Expert module a module for the acquisition, maintaining, and providing of explicitly encoded user and context -dependent knowledge shared among the components of a system.

External Knowledge Sources the servers an expert module is connected to through the *acquisition interface.*

Clients the clients an expert module is connected with through the *server manager.*

Knowledge Bases a number of knowledge bases storing the knowledge of the expert module. They can be logically distincted or not, depending on the degree of analysis required.

Integrity checking the integrity checking component for the knowledge base system of the expert module.

Inference Engine the inference engine for the knowledge bases. It acts as a server of the server manager, and as a client for the user, context and acquisition interfaces.

Knowledge Base System of the expert module it is composed of the Knowledge Base, the Integrity checking and of the Inference engine. In Fig. 7 it is surrounded by dashed lines.

Acquisition Interface interface with servers other than the user and context experts.

User expert interface interface with the user expert acting as a server (of course, it is absent in the user expert itself.) The exchange of information is necessary to tailor the knowledge of the expert to the user preferences, and to communicate inferred user preferences.

Context expert interface interface with the context expert acting as a server (of course, it is absent in the user expert itself.) The exchange of information is necessary to resolve context-references, and to communicate new ones.

Server manager this module provides the clients with the services of the expert module, performing some scheduling of the interactions. The interface operations with the clients are transformed into (a collection of) messages to the KBS. Then the answer(s) from the KBS is collected, re-transformed, and sent to the client.

Discussion

Several design choices need to be commented and justified.

Knowledge in an expert module.

We want to stress the characteristics required for the knowledge stored in an expert module: Explicitly encoded, user and context -dependent, shared.

Explicitly encoded refers to the fact we are considering knowledge based systems.

User and context -dependent refers to the two main factors involved in obtaining an *effective* presentation.

Shared refers to the fact the knowledge is used by more than one components and/or system. Therefore, it does not have to be too specific for a particular use.

The reference resolution is performed in the expert.

Alternative approaches are to require that the references are resolved by the client, or by the interface between the client and the expert. The latter makes the interface too specific to our systems, whereas with the current choice we can use any knowledge interchange format. The former is a valid solution to the problem but, to our opinion it does not respect the principle of modularity of the system. In fact, it implies that a client in a layer should know *what to resolve* and *what not to resolve*. In other words, it should know the internal knowledge representation formalism of the experts.

Inside the expert, the reference resolution could have been designed as part of the server manager, simply connecting the server interface to the context interface. Also in this case, we have the same problem; therefore the only modular solution is to let the inference engine resolve the context-dependent references.

Several logically distinct knowledge bases.

It is sometimes useful, and intuitive, to have a logically structured view of a knowledge base. This is the reason of the presence of *several* knowledge bases. Actually, this is not an implementative requirement: We do not require that the logical distinction reflects the real implementation (for instance, the implementation could confuse all knowledge in a single knowledge base, or structure it under other points of view - i.e., a knowledge base for each user, etc.). In the description of the four experts, we will see that it is natural to have such a classification of the knowledge in order to have a clear idea and a simpler definition of it.

The architecture of an expert module is generic.

The architecture of an expert module is not directly constrained by the particular systems we are describing, namely IMMPSs. This means that the notion of *expert module* is generic and suitable for other applications. We actually constrain its architecture when we instantiate clients and external knowledge sources. However, we are still in a general setting, since we can model interactions with unspecified servers/clients. This is actually the role that the *external servers/clients* play in our architecture: They are supposed to model interactions with unspecified systems. This allows us to merge our user expert with a functionally equivalent module of another application - for instance, the input system.

5.2 USER EXPERT (Ue)

To present the user expert as an instantiation of the generic architecture of an expert module, we only have to instantiate *the clients, the external knowledge sources* and *the knowledge bases.*

The user expert is the maintainer of the user model knowledge (see Fig. 8). This is classified in four categories, according to [16, 10].

Goals and plans (Ugp). The user's goal is a state of affairs he wishes to achieve, while a plan is a sequence of actions or events that he expects to result in the realization of a goal.

User capabilities (Uca). Physical or mental ability of the user to perform some actions.

User attitudes (Uat). Preferences of the user.

Knowledge or belief (Ukb). Modeling of the user knowledge.

The clients of the user expert are the layers of the presentation systems, and all the other experts. In addition, interactions with unspecified components are modeled by means of the *external servers/clients.*

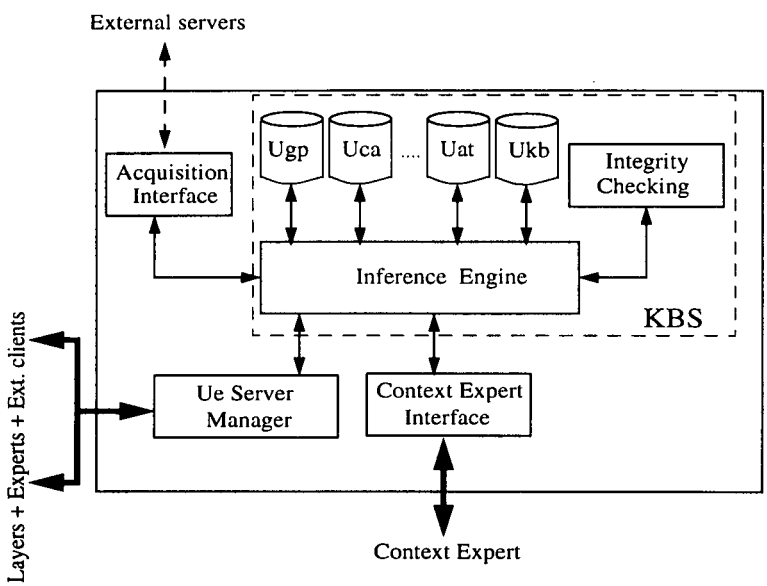

Figure 8: *The user expert module*

User Expert expert module for the acquisition, maintaining, and providing of the user model knowledge.

Clients the layers of the presentation system, the other expert modules, and the external clients (other computational systems, the developer, the site manager, the user himself.)

Knowledge Bases the knowledge bases of the *user expert* are *goals and plans, capabilities, attitudes* and *knowledge or belief*.

External knowledge sources they are the external servers (*the application developer, the site manager,* and the *user* himself may be asked for services by the KBS.)

Discussion

Knowledge bases

There is no fundamental reason to justify the chosen classification of the user-model knowledge. It seems, however, as a good compromise: A grosser categorization would be too generic and helpless in design systems ; a finer one would be too specific and constraining. We point out again that such a classification is only from a logical point of view, and it is aimed to have a clear view of what a user-model is. It absolutely does not constraint any implementation or other view of the user-model knowledge.

KBS

In the case of the user expert module, the KBS is a well-studied component in the literature. It is connected with a client - the Ue Server Interface - and with two servers - the Context Expert Interface and the Acquisition Interface. Many proposed reference models in the literature [10] will fit this component.

We are not aware of the existence of any similar proposal for the application, context and design experts. It would be interesting to compare this proposal with reference models for those components.

Interface operations: examples

Here are some examples of the services that may be offered by the Ue server manager
ask: user_knows?, user_stereotype?, user_preferences? ;
assert: user_knows!, user_prefers!;
deny: user_knows!, user_prefers!.
For instance, a request like *ask(user_knows?(mouse))* is supposed to ask the KBS about a user knowledge. A reply could be something like *tell(user_knows?(mouse),FALSE).*

Role of the Ue in an IMMPS

The user expert affects the presentation process in many ways. It may interact with *the application expert*, by affecting what is expressed;

the context expert, by affecting what is expressed. As an example, the presentation of a particular information could be omitted if the user can easily infer it from the context;

the context expert, the control and content layers, by affecting the way the dialog proceed;

the design expert, the content, layout and presentation layers, by affecting how the data are presented.

5.3 APPLICATION EXPERT (Ae)

Following the same approach of last section, we present the application expert as an instantiation of the generic architecture of an expert module. We only have to instantiate *the clients, the external knowledge sources* and *the knowledge bases.*

The application expert provides the components in the layers with the application knowledge, i.e. knowledge provided by the application and knowledge for reasoning about application data.

The knowledge bases are classified accordingly.

Term interpretation (Ati). Knowledge for interpreting a high level description of a collection of application data.

Data characterization (Adc). Characteristics of application data relevant to the presentation design.

The clients of the application expert are the layers of the presentation systems. In addition, interactions with unspecified components are modeled by means of the *external servers/clients.* The fundamental external knowledge source is obviously, the *application.*

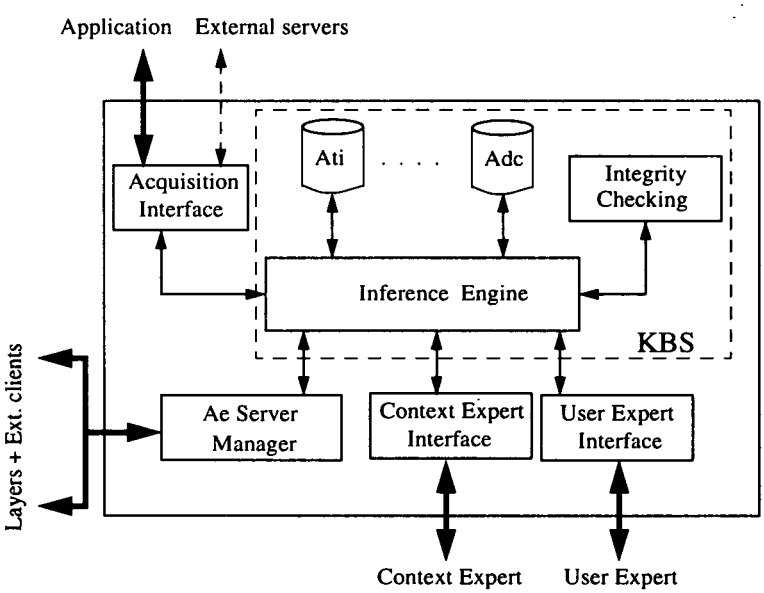

Figure 9: *The application expert module*

Application Expert expert module for the acquisition, maintaining and providing of the application knowledge.

Clients the layers of the presentation system, and the external clients.

Knowledge Bases the knowledge bases of the *application expert* are *term interpretation* and *data characterization*.

External knowledge sources the fundamental external knowledge source is the *application*. In addition, some *external servers* can play this role.

Discussion

Knowledge bases

Again, the classification of the application knowledge is arbitrary and not intended to be constraining from an implementation point of view.

Interface operations: examples

Here are some examples of the services offered by the Ae server manager

ask: items(X, cost(X) < 100), type_of_data? ;

assert: (NOT ALLOWED FOR IMMPSs);

deny: (NOT ALLOWED FOR IMMPSs).

For instance, a request like *ask(items(X, cost(X) < 100))* is supposed to ask the KBS for the items which cost less than 100 units. A reply could be something like *tell(items(X, cost(X) < 100), { it_1, ..., it_k })*.

It is worth noting no *assert* or *retract* service is offered by the application experts, since IMMPSs are concerned with *presentation of data* and not with *acquisition of data*.

Role of the Ae in IMPPSs

The application expert affects the presentation process in many ways. It may interact with:

the content and layout layer, by affecting what is expressed, since most of content selection is guided by application data and knowledge;

the content layer, by affecting the way the dialog proceed - since different presentation strategies can be followed for different characteristics of the application data ;

the content and layout layer, by affecting how the data are presented - since (media and data) selection and layout design depend on the characteristics of the application data.

5.4 CONTEXT EXPERT (Ce)

To present the context expert as an instantiation of the generic architecture of an expert module, we have to instantiate *the clients, the external knowledge sources* and *the knowledge bases.*

The context expert is the maintainer of the context knowledge. This is classified in two categories.

Discourse model (Cdm). Modeling of the past interactions of the system with the goal formulation.

Context references (Ccr). Modeling for the resolution of context-dependent references - like anaphora, diexis, ontology etc.

The clients of the context expert are all the layers of the presentation system, and all the other experts. In addition, interactions with unspecified components are modeled by means of the *external servers/clients.*

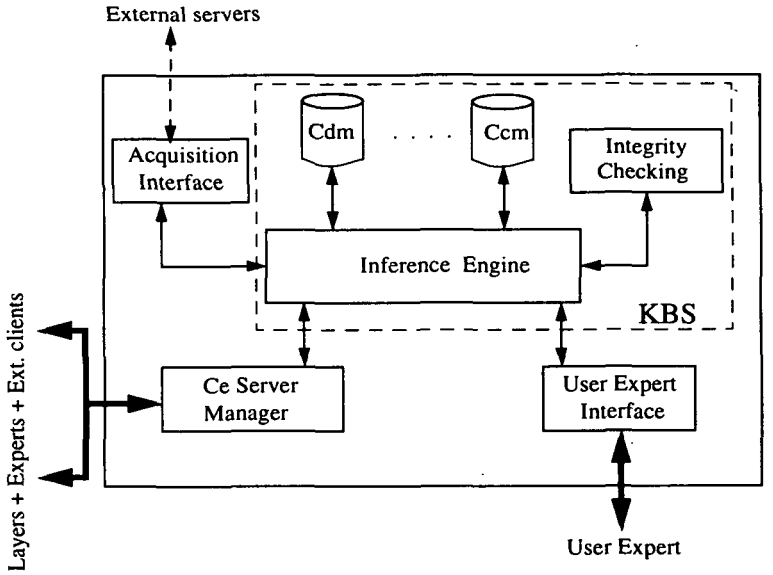

Figure 10: *The context expert module*

Context Expert expert module for the acquisition, maintaining, and providing of the context knowledge.

Clients the layers of the presentation system, the other expert modules, and the external clients.

Knowledge Bases the knowledge bases of the *context expert* are *the discourse model* and *the context references*.

External knowledge sources external servers.

Discussion

Knowledge bases

Again, the classification of the context knowledge is arbitrary and not intended to be constraining from an implementation point of view.

Interface operations: examples

Here are some examples of the services offered by the Ce server manager

ask: reference(Variable), focus? ;

assert: reference(Variable, computer), dialogue("interaction_description") ;

deny: reference(Variable, computer).

For instance, a request like *ask(reference(Variable))* is supposed to ask the KBS for the current *focus of attention* of the dialogue.

Role of the Ce in an IMMPS

The context expert is a fundamental component. Every layer of the presentation system and every other expert refer to it for asserting, retracting and resolving context references.

5.5 DESIGN EXPERT (De)

The fourth expert is the design expert. Like the other experts, it is an instantiation of the generic architecture of an expert module.

The design expert is the maintainer of the design knowledge. This is classified in four categories.

Design constraints (Dds). Modeling of the design constraints during the presentation process.

Cognitive theory (Dct). Knowledge on the cognitive impact of multimedia techniques (ex., clarity and complexity of the presentation, cognitive issues, how the user attention is attracted, etc).

Media model (Dmm). Modeling of the media characteristics.

Device model (Ddm). Modeling of the devices characteristics and availability.

The clients of the design expert are all the layers of the presentation system. In addition, interactions with unspecified components are modeled by means of the *external servers/clients.* For instance, we need an external server - namely the operating system - to be the server of the knowledge about the devices available at a certain moment.

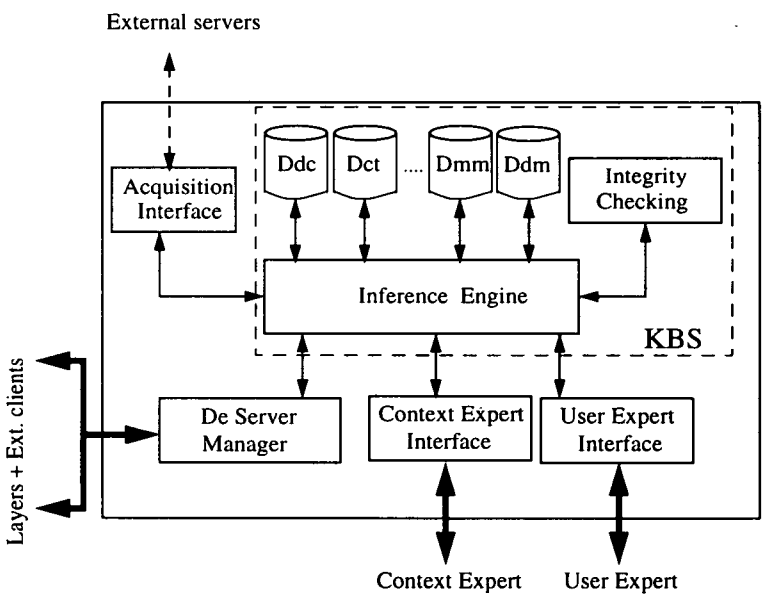

Figure 11: *The design expert module*

Design Expert expert module for the acquisition, maintaining, and providing of the design knowledge.

Clients the layers of the presentation system, and the external clients.

Knowledge Bases the knowledge bases of the *design expert* are *design constraints, cognitive theory, media model* and *device model.*

External knowledge sources external servers.

Discussion

Knowledge bases

Again, the classification of the design knowledge is arbitrary and not intended to be constraining from an implementation point of view.

Interface operations: examples

Here are some examples of the services offered by the De server manager
ask: device_available(mouse), is_visible(clock) ;
assert: visible(clock) ;
deny: visible(clock).
For instance, a request like *ask(is_visible(clock))* is supposed to ask the KBS whether or not a clock is visible on the screen. A reply could be something like *tell(is_visible(clock), TRUE).*

Role of the Ae in an IMMPS

The design expert plays a central role in the presentation process. The layers of the presentation system interact implicitly through this expert, simply by stating design constraints to be followed.

5.6 The Knowledge Server

In the previous sections we described the four expert modules - application, user, context and design - as instantiations of the presented generic structure.

Here, we briefly sum up this instantiation process by giving a broad view of the overall *knowledge server*. Next table shows the fundamental differences among experts, namely their knowledge and the clients/servers they are connected to in the architecture of an IMMPS.

MODULES OF THE KNOWLEDGE SERVER

Expert	*Ext. Know. Sources*	*Clients*	*Knowledge*
Application	Application External servers	Layers External clients	Ati Adc
Context	External servers	Layers Other-Experts External clients	Cdm Ccr
User	External servers	Layers Other-Experts External clients	Ugp Uca Uat Ukb
Design	External servers	Layers External clients	Ddc Dct Dmm Ddm

LEGEND

Ati	Term interpretation	Adc	Data characterization
Cdm	Dialogue model	Ccr	Context references
Ugp	User goals and plans	Uca	User capabilities
Uat	User attitudes	Ukb	User knowledge and belief
Ddc	Design constraints	Dct	Cognitive theory
Dmm	Media model	Ddm	Device model

Knowledge Server collection of four experts - user, application, context and design - connected as shown in Figure 12, each of which acts as a server to the layers of the presentation system. Unspecified interactions are modeled through the *external servers/clients*.

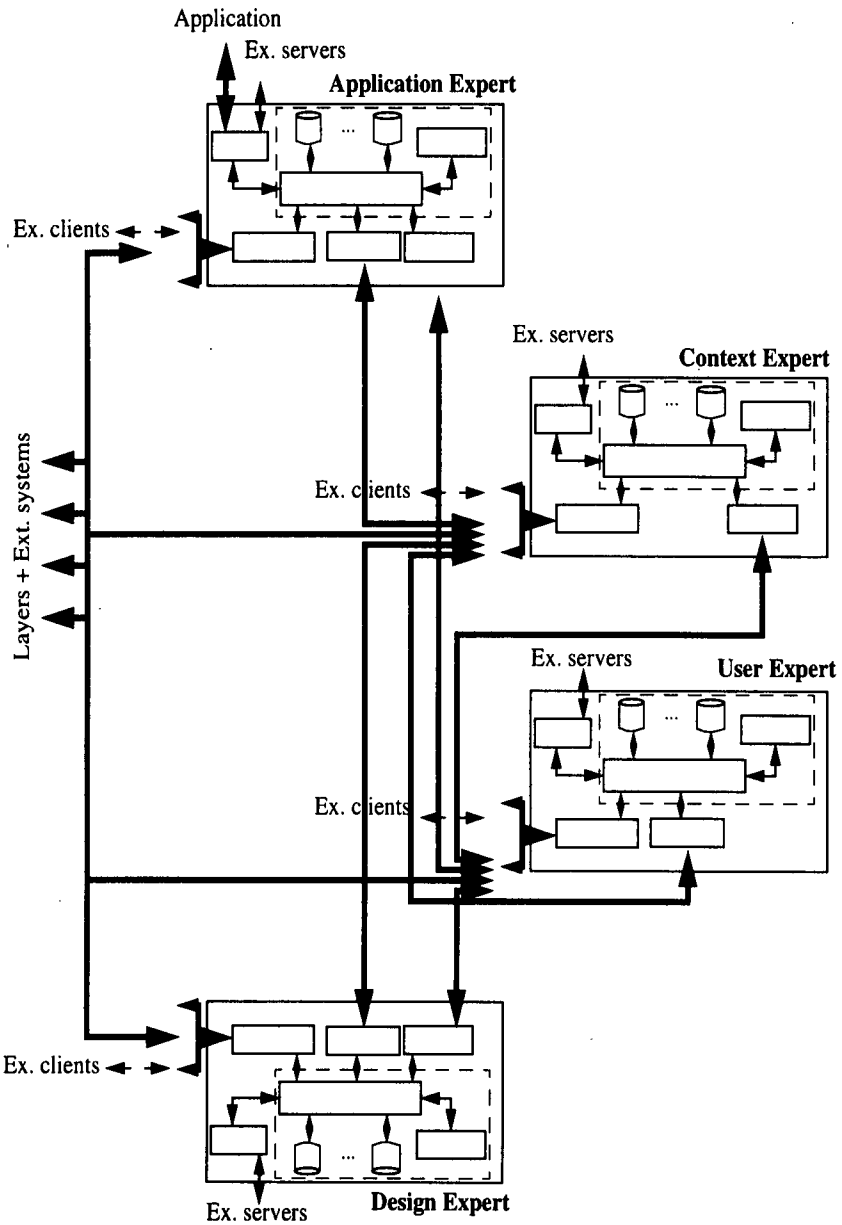

Figure 12: The Knowledge Server

5.7 Knowledge sources location

The knowledge sources introduced in section 2.3 are stored partly in the knowledge server - if they are shared among several components - and partly in components of the layers. The following table sums up where they are located.

---

user model user expert,

context model context expert,

application knowledge application expert,

design knowledge design expert,

media specific design/realization knowledge media design/realization in the layout layer,

system model split in several components: user expert, design expert, layers.

media selection knowledge media selection in the content layer.

We tried to set an owner component for each knowledge source. However, it is difficult to force the system model in a unique component. For its own nature, the system model is split among the user expert, the design expert and many components in the layers. In fact, user model and system model are not disjoint knowledge sources. Neither are design knowledge and system model.

6 Relating the model to real systems

The objective of this section is to relate the model described in this paper with aspects of real systems. For example, we will show how to express in terms of our reference architecture phenomena like integration with other reference models, interactive systems, acquisition of user preferences, and other discussions.

Computer Graphics Reference Model

The Computer Graphics Reference Model [6] defines the notion of *computer graphics* as *the creation of, manipulation of, analysis of an interaction with pictorial representation of objects and data using computers*, and defines a logical architecture of computer graphics interfaces.

The architecture for computer graphics is logically distincted in five layers: *construction, virtual, viewing, logical* and *realization*. Each of them applies a transformation on its input, starting from *application data* provided to the construction layer and obtaining a *complete definition of the image* as result from the realization layer.

We can integrate such a model in the black box *graphics realization* in the layout layer, *and* in the black box *display* in the presentation layer. In the former, the graphics generation produces as input to the computer graphics reference model the data necessary for developing an unrendered presentation. Since at this stage the generated text is not yet available, the *graphics realization* deals only with graphics objects, and produces only an abstract *complete definition of the image*. Further, in the *display* module of the presentation layer, the text and other generated object are integrated with the graphics by means of another instance of the computer graphics model, whose output is now directly perceivable to the user.

Acquisition of user's preferences

In all practical system, there is a window in which the user can modify his preferences, together with some other parameters. This interface is not directly related to the presentation system. How can we model its interactions with the IMMPS in our model ? There are two ways we can do that.

The first one is to model the interface as an external client, which acquires the user's preferences and then communicates them to the expert modules involved.

The second way is to assume the interface to be part of the goal formulation and then interpret the user's preferences as presentational commands, which are sent through the control layer to the components involved. This second possibility is slightly more general. Whereas the first one allows only to modify knowledge in the expert modules, the second way allows to reach every component of the system - through the control manager.

Coping with Interactive Presentation Systems

The IMMPS reference architecture does not allow to pass information directly from the user to the presentation layer. The way we can model interaction with the user - like anytime interruptions, active participation in the presentation, browsing, etc. - is to assume the goal formulation to interpret the appropriate user inputs as presentational commands or goals to send to the IMMPS.

Functionalities of the components

A further phase in the development of a complete formal reference model is to express in a semi-formal notation the functionalities of each black box component of the system. For instance, an analogous work is under construction for the Computer Graphics Reference Model, and it is called PREMO [23] - Presentation Environment for Multimedia Objects. The specification language employed is Object Z [27].

However, we point out that a complete formal specification can take place only after a strong agreement on the reference architecture.

As an example, let us consider the *control manager* module in the control layer. Below we report a semi-formal specification of parts of its interactions (in the style of CSP [14]).

[

□ GF-Interface?$Goal(goal)$ →
  $goal' = Decompose(goal)$;
  ContextExpert!$Push(goal')$

□ GF-Interface?$Command(com)$ →
  $com' = ElaborateCom(com)$;
  ContextExpert!$Push(com')$

□ ContentLayer?$Notification(not)$ →
  if $OK(not)$ *then*
    $not' = ElaborateNot(not)$;
    GF-Interface!not';
  else
    $rep = Repair(not)$;
    ContextExpert!$Push(rep)$

□ ContentLayer?$Ready()$,
  ContextExpert?$NextGoal(goal)$ →
  ContentLayer!$Goal(goal)$;

⋮

]

Intuitively, the *control manager* nondeterministically receives goals, and presentational commands from the GF-Interface, Notifications and Ready messages from the content layer. When receiving goals, these are decomposed and stored in the context expert. Also, the commands are elaborated and stored. The notifications are analysed: if they report success then they are elaborated and eventually sent to the GF-Interface. If any problem occurred then a repair strategy is extracted from the notification and pushed into the context expert. Finally, if the content layer is ready to start with a new goal and the context expert has a goal to be achieved, then the goal is sent to the content layer.

A MMI² in terms of the model

In this and in the next appendix, we will show how two real systems fit our model. On the one hand, the broad applicability of the model is checked. On the other hand, the two systems are described by a common reference architecture and terminology.

MMI²: A Multi-Modal Interface for Man Machine Interaction with Knowledge Based Systems

The MMI² system was developed with the purpose of demonstrating the architecture and development method required to produce large scale co-operative interfaces to Knowledge Based Systems [33]. The system supports multiple channels, parallelism, abstraction, fission and fusion, and the use of rich contexts. Within the project, two demonstrators about local and wide area network design were produced [20]. The first one - released in October 1991 - is a tool to design local area networks for institutions such as hospitals, universities or offices. The architecture of the second demonstrator was a subset of that of the first since it did not include all the interaction modes, nor a detailed user model.

The main concerns with the MMI² project were the architecture notion of "expert" and the use of a common meaning representation (CMR). By "expert" it is meant a module performing specific tasks and with its own private data structures, and which allows a sufficiently coherent set of processes to be gathered in a single module. This corresponds to our notion of "module".

CMR is the common communication language among the components of the system. Communication between the application and the application expert is in the language of the application. With our terminology, the CMR is a Knowledge Exchange Format. It is used to support fission and fusion of information between media and to supply a common discourse context through which to resolve references made within and between media. Each CMR packet contains one or more CMR_acts, along with the status, mode and time for those acts. Each CMR_act contains an utterance type, one or more CMR-expressions, and a slot for mistakes. Each CMR-expression represents a possible interpretation of an ambiguous utterance. Each CMR-expression then contains annotations, a logic formula, and syntactic information. The status field is used for internal error checking and the time field to co-ordinate output. A media field identifies the media through which the packet was received as user input, or the one for which it is destined as system output. The utterance type is used to define the further processing of the CMR_act.

The MMI² systems was devised for co-operative dialogues. This implies that its components are designed for input-output - or two-way - interactions with the user and the application. As we are dealing only with the output generation, we will isolate and describe this part of the system, even though actually it is strongly merged with the input subsystem.

Goal formulation

The *goal formulation* manages input acquisition and application data updating. In MMI², a goal is not directly specified by the user. The *goal formulation* processes the (multimedia) inputs from the user, building up a goal (or a presentational command) as the result of the fusion of several coordinated inputs concerned with output generation. Using the terminology of [26], only the "attitudes" *User wants to know* and *User wants* are passed to the presentation system. They respectively correspond to goals and presentational commands. They are represented as CMR packets.

Control Layer

The *control manager* (called *dialogue controller*) classifies the CMR passed by the *goal formulation* into goal or presentational command. A pair consisting of a CMR and its classification is called *user-desire*. The classification is done on the basis of the form of the CMR, and the form of previous system presentations.

The *discourse model* - stored in the context expert - is structured simply as a stack. The control manager simply pops the top of the stack - if not empty - to decide the next action - achieving a goal, executing a presentational command, or notifying to the goal formulation.

A notification from the content layer can contain some recovery strategies if the presentation process fails. These strategies are organized as a collection of sub-goals to be achieved and notifications to the goal formulation, which are pushed in the *discourse model*. As an example, let us suppose that the goal formulation asks for presenting to the user the cost of a computer, and the system does not know the cost of the monitor. Then the recovery strategy can be to notify to the goal formulation that the system is expecting that the application database being correctly updated.

Finally, the control layer replaces anaphoric expressions in a CMR packet through relating it to an appropriate antecedent. Anaphoric expressions are resolved in the context expert.

Content Layer

The role of the content layer is to convey the system's intentions to the user in such a way as to follow the rules of cooperativity in dialogue.

The planning module performs first an analysis (called *informal semantics*) of the goal in input - exploiting some private knowledge - in order to provide pragmatic, dialogue oriented functionalities beyond the narrow range of selecting the data described by the goal (called *formal semantics*). The main functionalities are repair (identification of error condition which prevent straightforward handling of goals) by means of *task plans*, explanations (the determination and provision of presentation that are more informative than formal semantics alone would give), and clarification (assessing goals, checking their validity, etc.). A notification to the control layer is reported if problems occur, together with a recovery strategy in the form of a collection of goals and notifications.

After this analysis, a strategy tailored to the user and the context is planned (by a sub-module called *communication planner*) producing communication acts - grouped in a CMR packet -, to the content selection and media selection modules.

The content selection interprets the communication acts by decomposing them in terms of simple predicates directly interpretable by the application expert, and then collecting and composing the answers from it. It can ask the layout manager - through the coordination and the planning - for the meaning of media objects, whose knowledge is stored in the media design and realization of the layout layer.

Media selection exploits its media selection knowledge based on cognitive studies (see [7] for an overview), operationalized by means of several rules for media selection. Finally, coordination collects the media communication acts from media and content selection and send them - as CMR packets - to the layout layer. The notifications from it are sent to the planning.

To some extent, it is correct to say that planning, content and media selection, and coordination are three *sequential* tasks. Instead, content and media selection work concurrently, with some synchronization.

Layout Layer

The layout manager (called *interface expert*) maintains a record of window positions and provides locations for new windows. It passes media communication acts received from the content layer to the media design, media realization and coordination. From these, it receives notifications about the presentation process, possibly together with additional information, which are collected and passed to the content layer.

The media in the MMI² are: english language, french language, spanish language, graphics (actually, in addition to these output media, the *gesture* and *command language* input media are present). Let us briefly overview the english language and the graphics. The former was not developed within the MMI² project. It was lifted out from the Loqui system, BIM's English natural language interface to relational databases, and ported to the MMI² application domain. This involved building a new lexicon and defining the interfaces for communication with other MMI² modules. The graphics media design and realization is supervised by a *graphics manager*, which has at its disposal a number of graphics tools. The graphics manager

- translates the CMR packets representing media communication acts passed by the layout manager into a graphical action by requesting a change on one of the existing window graphics tools (e.g., displaying a new network on the network tool, moving a machine from one room to another.) or by displaying a new graphics tool (e.g., the system tries to generate a new chart whose carried item is more effectively perceivable by the user.);
- manages a library of graphical data structures used in some of the tools;
- provides a library of functions that all the graphical analysis tools use;
- provides an interface to answer to media communication acts asking about the graphics media (e.g., objects currently visible, relative position of machines, etc.)

The graphic tools include tables, bar chart, pie chart, scatter plots, and network tools.

Presentation layer

The coordination module mainly consists of a window manager (SunView/X-windows), together with a component dispatching the unrendered presentations to the suitable device interface. The only (output) device is the display.

User expert

The user expert dynamically acquires and stores knowledge about the users. This knowledge enables the MMI² system to respond more cooperatively to the current user. The knowledge of the current user that is acquired and stored is the user's general knowledge of the domain of application (e.g., which domain objects the user knows about), the user's knowledge of the MMI² system (e.g., which MMI² commands the user knows about), and user's preferences (e.g., which currency the user would prefer).

Information about the current user is derived from the discourse - stored in the context expert - , or even by the user himself (e.g., changing his stereotype) through a graphical interface to show the inheritance network of user stereotypes, the knowledge within a particular user model, the predicates permitted in user models, and any inconsistencies between beliefs in a user model, or derived from its parents. The graphical interface is modeled as an external client of the user expert.

From the information in the user model, it may be possible to categorise the user as a certain type of user. Further knowledge is available because of the assumptions that can be made based on stored knowledge about different types of users. The hierarchy of user types allows multiple inheritance from different stereotypes.

From studies on how to obtain knowledge from human experts and users in the application domain, the following knowledge was obtained and used within the user expert:

- the stereotype hierarchy of users in the domain,
- what domain knowledge would be known by a member of a particular user stereotype,
- the rules that allow a human expert to infer the state of knowledge of an interlocutor from the dialogue,
- the way that human experts decide that an interlocutor belongs to a particular stereotype.

Application and Application expert

The role of the application expert is to provide the data described by a (part of a) goal and the denotations of the symbols used in the CMR packets, by exploiting the application knowledge, according to the user preferences (e.g., currency, unit of measure preferred etc.) and the context. Moreover, it communicates to the context expert the relations in the reified world of the application. The application is an expert system (called NEST: a Network design Expert SysTem) providing knowledge on network design (location of machines, graphics objects, etc.) In particular, its knowledge base contains all the definitions of the needed objects, i.e. both the various network components and the topological information relative to the buildings.

Context expert

This module provides contextual functionalities - essentially anaphora and ellipses resolution - which are involved in the contextual processing of each move, and joins an MMI^2 standard reified world with a reified world of the application. The MMI^2 standard world contains conceptual labels representing the reified objects of the layout, the graphics objects, and the word sense representatives that are relevant to the communication for english, french and spanish languages. The world of the application is represented by means of the relevant terms of the application and the word senses representatives that are relevant to the communication for english, french and spanish languages.

Utterance containing anaphorical or elliptical phenomena are represented by the so-called incomplete CMRs. The context expert contributes to their resolution by proposing possible candidates for completing them. These candidates are extracted from the stored discourse model. Contextual functionalities are accessed through a focusing mechanism, which emphasizes the role of phrases and relations that link together (anaphora and ellipsis) in discourse.

Design expert

The design expert provides knowledge about design constraints, cognitive impact of the used media, device model, and media characteristics. The knowledge is tailored to the user preferences and to the context.

B WIP in terms of the model

The current prototype of WIP [3] (WIP stands for *Knowledge-Based Presentation of Information* - in German) generates multimedia explanations and instructions on assembling, using, maintaining or repairing physical devices. WIP is currently able to generate simple German or English explanations on using an espresso machine, assembling a lawnmower, or installing a modem, demonstrating the claim of language and application independence.

The major design goals of the WIP [3] system are the generation of coordinated presentations from a common representation, the adaptation of these presentations to the intended target audience and situation, and the incrementality of all processes constituting the design and realization of the multimedia output.

It is an important objective not to simply merge the verbalization results of a natural language generator and the visualization results of a knowledge-based graphics design component, but to carefully coordinate natural language and graphics in such a way that they generate a multiplicative improvement in communication capabilities. In addition, page layout is addressed as a rethorical force, influencing the intentional and attentional state of the reader.

WIP is a highly adaptive interface since all of its output is generated on the fly and customized for the intended target audience and situation. Another important design goal was that the incremental generation of a multimedia presentation should be supported. Incremental generation is the immediate realization of parts of a stepwise provided input. The authors claim that for systems like WIP incrementality is essential. On the one hand, WIP must be able to begin outputting words and graphical elements before the input is complete, when the information to be expressed arrives in a stream from the backend system. On the other hand, the system should be prepared for cases when the goal and the media communication acts are changed in the course of the media generation process. Such a change might be due to new high priority goals, or the user's reaction to the presentation generated so far.

In WIP, the design of multimedia presentation is viewed as a subarea of general communication design. The fact that communication is always situated (i.e., depends on some context) is taken into account by considering user's preferences and design constraints (called *generation parameters*.) The current system includes a choice between user stereotypes (e.g., novice, expert), target language (German vs English), layout formats (e.g., hardcopy of instruction manual, screen display), and output modes (incremental vs complete).

One of the important insights in the development of WIP is that it is actually possible to extend and adapt many of the fundamental concepts developed to date in artificial intelligence and computational linguistics for the generation of natural language - like coherence, speech acts, anaphora, rethorical relations - in such a way that they become useful for the generation of graphics and text-picture combinations as well. A basic assumption behind the WIP model is that not only the generation of text and dialog contributions, but also the design of graphics and multimedia presentations are planning tasks.

In WIP, no recovery strategy is provided to the *goal formulation* when the presentation process fails to achieve a goal.

In the following, we discuss more in detail the WIP system in terms of the components of our

reference model.

Goal formulation

In WIP the *goal formulation* is a menu-guided interface with the user. This interface allows the user to modify the generation parameters and to choose a goal to be achieved (in MMI^2, instead, the goal is a built up by the goal formulation as a function of the user's inputs.) A goal is expressed in terms of a modal logic for modeling believes and intentions. For instance, the goal

(Goal P (BMB P U (LOCATION SWITCH-2 ?LOCATION)))

stands for: the presenter P has the goal to inform the user U of the location of the switch denoted by the constant SWITCH-2. *(BMB P U x)* is the predicate of mutual belief, i.e. P believes x and P believes that U believes x, etc.

Control Layer

Control tasks such as choosing the next goal to be achieved from an input queue or the decomposition of presentation goals are performed by WIP's component for presentation planning (see also next paragraph Content Layer). The presentational commands are sent to the components involved. For instance, the generation parameters are sent to the design expert. Differently from MMI^2, no recovery strategy is provided by the content layer when the it fails to achieve a goal.

Content Layer

At the heart of the presentation system is a parallel top-down *planning* module. It tries to find a presentation strategy for the given goal by incrementally generating a refinement-style plan in the form of a directed acyclic graph (DAG) - stored in the context expert - by means of some presentation strategies. They reflect general presentation knowledge - stored in the *planning* module - or they embody more specific knowledge of how to present a certain subject - knowledge provided by the application expert.

An example of a presentation strategy is shown below:

Header: (Introduce System User ?object Graphics)
Effect: (BMB System User (Isa ?object ?concept))
Applicability Conditions:
(Bel System (Isa ?object ?concept))
Main Acts:
(S-Depict System User ?object ?pic-obj ?picture)
Subsidiary Acts:
(Label System User ?object ?medium)
(Provide-Background System User ?object ?pic-obj ?picture Graphics)

The header is a complex communicative act, its effect refers to an intentional goal. The applicability conditions specify when a strategy may be used and constrain the variables to be instantiated. The main and subsidiary acts form the kernel of the strategies. The slot reporting the media allows to define medium-independent presentation strategies simply by filling it with a variable, which will be instantiated only when stronger constraints are required (for instance a strategy - like that shown - which instantiates the media slot.)

The leaves of the planned DAG are specifications for elementary multimedia communication acts, which are elaborated by the media design and realization in the layout layer. From these, through the coordination, the planning module can acquire knowledge about relative position of objects, etc.

Since the planning module has no direct access to the knowledge of the media design and realization components in the layout layer, it cannot consider this information when building up a candidate strategy. Consequently, it may happen that the results provided by the generators deviate to a certain extent from the planned strategy (e.g., they are unable to perform the required task, etc.) Such deviations are handled by dedicated mechanisms which are: *output sharing* - part of a presentation is reused for different purposes - , *structure sharing* - part of a DAG is shared - and *structure adding* - split of part of the DAG. Restructuring methods are applied by the *coordination* module.

Whereas in the MMI^2 system the planning, selection and coordination tasks are performed *sequentially*, in the WIP system they are performed concurrently. This is due mainly to the concerns for incremental processing.

Layout Layer

The WIP layout manager stores a set of document types, together with some layout constraints for each, which constrain the media design and realization. In addition, communicative acts asking for reply - e.g., is_visible(object) - are answered by the layout manager, after dispatching the request to the appropriate media design and realization.

The media design and realization include the following media: graphics, German natural language and English natural language. In illustrated instructions for technical equipment, graphics are used in order to accomplish presentation tasks, such as depicting a domain object in a certain state, showing an object's location, or visualizing the course of an action. The authors operationalized certain 2D and 3D illustration techniques frequently used by human illustrators: the formalization is based on a compositional semantics of pictures. Using graphical design strategies, graphics design is in principle a goal-driven planning process. However, it does not seem feasible to strictly separate a graphics design and realization phase, as some realization operators have side effects which are computationally expensive to anticipate. A solution to this problem is to interleave graphics design and realization and to allow for feedback.

As for graphics, the design and realization of English and German natural languages are strongly influenced by the quest for incremental reasoning. Thus the form and size of basic processing units, data flow, and the interaction between the components were determined by this incremental processing scheme.

Finally, coordination between text ad graphics generation - namely, *multimedia referring expressions* and *crossmedia referring expressions* - is performed in order to express cross-media deictic assertions like

The on/off switch is located in the upper-left part of the picture

or the multimedia referring to world objects like

The switch on the frontside.

The authors developed a model of referring which takes into account that user's and system's knowledge about the identity of objects doesn't necessarily coincide. In order to describe the links a user has to infer in understanding a referring expression, the predicates *(Coref rep1 rep2), (Encodes means information context-space)* and *(EncodesSame p1 p2 context-space)* are stored in the context expert.

Note that in the publications on WIP the term "layout" is used for describing the spatial arrangement of text picture blocks on a page (either paper or screen) only. Thus, the component called "Layout Manager" in WIP performs a subset of the tasks which are to be accomplished by the modules "Layout Manager" and "Layout Control" in Fig. 5. Tasks such as the monitoring of media generators, or the design of crossmedia references are done by WIP's presentation planner.

Presentation layer

The output of WIP's Layout component (which is a specification of text/picture blocks together with constraints for their spatial arrangement) can be sent either to a window manager, or to a (postscript) printer. Furthermore, the incrementally generated text output can be sent to a speech synthesizer.

User expert

The WIP stereotype user model distinguishes *novice* and *expert* users. User's goals, preferences and knowledge are stored in the knowledge bases.

The user can modify his stereotype and preferences. This can be modeled either as a presentational command given through the goal formulation, or by modeling the interface for acquiring the user preferences as an external client of the user expert.

The user model is updated after a goal has been achieved. From that time on, the user is supposed to know the information conveyed by the presentation of the goal. For instance, the user model is updated after the goal "instruct how to fill the watercontainer" has been achieved, since the user is assumed to know that concept.

Finally, the user model is not supposed to be integrated with other systems. Therefore, no external client is present. There is, however, a context expert interface to resolve context-dependent references.

Application and Application expert

WIP's application knowledge comprises knowledge about physical domain objects (eg. a modem, a lawnmower, an espressomachine), and knowledge about operation procedures for that objects. The representation of this application knowledge is hybrid. While the ontology of the domain objects and the operation procedures are represented in RAT [13] - an extension of a Terminological Logics - wireframes are used to represent geometrical information of objects and object constellations.

The knowledge represented in RAT is used both for the generation of text and graphics as the main source of knowledge about the domain. Besides the domain plans the entire information concerning the domain terminology is represented in RAT. In order to support the user modeling, a partitioning mechanism is provided to reason about the potentially conflicting views of the world the user an the system may have.

Many of WIP's presentation strategies are general in nature, however, there are also strategies which are application dependent. In the current reference model one may therefore distinguish between the general strategies which form a private knowledge source of WIP's presentation planner, and a set of application dependent strategies stored in the application expert."

To a certain extend, the same goes for the knowledge sources of WIP's medium specific generators. For example, there is also application dependent knowledge how to choose a suitable view of an object if it's depiction has to meet standards in the area of Technical Drawing.

Context expert

The context knowledge consists of a *document design plan*, and some predicates for managing referring expressions.

The former is a directed acyclic graph incrementally built by the *planning* module in the content layer, and restructured by the *coordination* module in the same layer. The leaves of the *document design plan* are specification for elementary acts of presentation, namely graphics and text design and realization.

The predicates for managing referring expressions are

- coreference relations - like *coref(rep1, rep2)* - to express that two representations - *rep1* and *rep2* - refer to the same world object,
- semantic relationships - like *(Encodes means information context-space)* - between a textual or graphical means and the information the means is to convey in a certain context space,
- cohesive relationships - like *(EncodesSame p1 p2 context-space)* - between two presentation parts *p1* and *p2*, satisfied if and only if *p1* and *p2* encode the same object.

Design expert

The WIP design expert stores knowledge on *generation parameters* - constraints on the document layout, like short/long presentation, etc. - given by the user, layout constraints dynamically inferred from the context and the user preferences, and finally knowledge on device availability - acquired from the operating system modeled as external server/client.

References

[1] KQML Advisory Group. An Overview of KQML: A Knowledge Query and Manipulation Language. http://retriever.cs.umbc.edu:80/kqml/.

[2] The System Modelling Glossary. AMODEUS Project, Esprit Bra 7040. Document System Modelling/WP26, 28 June 1995.

[3] E. Andrè, W. Finkler, W. Graf, T. Rist, A. Schauder, and W. Wahlster. WIP: The Automatic Synthesis of Multimodal Presentations. In M. T. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 3. AAAI/The Mit Press, 1993.

[4] E. Andrè and T. Rist. The Design of Illustrated Documents as a Planning Task. In M. T. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 4. AAAI/The Mit Press, 1993.

[5] Y. Arens, E. H. Hovy, and M. Vossers. On the Knowledge Underlying Multimedia Presentations. In M. T. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 12. AAAI/The Mit Press, 1993.

[6] Computer Graphics Reference Model. International Standard Organization, ISO/IEC IS 11072, 1992.

[7] H.R. Chappel, M. D. Wilson, and B. Cahour. Engineering User Models to Enhance Multimodal Dialogue. In J.A. Larson and C. Unger, editors, *Engineering for Human-Computer Interaction*, pages 297–313. Elsevier Science Publishers, Amsterdam, 1992.

[8] G.C. Van der Veer, T.R.G. Green, J.M. Hoc, and D.M. Murray, editors. *Working with Computers: Theory versus Outcome.* Academic Press, 1988.

[9] S. K. Feiner and K.R. McKeown. Automating the Generation of Coordinate Multimedia Explanations. In M. T. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 5. AAAI/The Mit Press, 1993.

[10] T. W. Finn. GUMS - A General User Modelling Shell. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems.* Springer-Verlag, Berlin, 1989.

[11] G. Gazdar and C. Mellish. *Natural Language Processing in PROLOG.* Addison-Wesley Publishing Company, 1989.

[12] W. D. Gray, W. E. Hefley, and D. Murray, editors. *Proceedings of the 1993 International Workshop on Intelligent User Interfaces.* ACM Press, 1993.

[13] J. Heinsohn, D. Kudenko, B. Nebel, and H. J. Profitlich. RAT - representation of actions using terminological logics. Technical report, DFKI, Saarbrücken, Germany, 1992.

[14] C.A.R. Hoare. *Communicating Sequencial Processes.* Prentice Hall International, 1985.

[15] I3 (Intelling Integration of Information) Glossary. 8 Nov. 1994.

[16] R. Kass. *Acquiring a Model of the User's Beliefs from a Cooperative Advisory.* PhD thesis, Dept. of Computer and Information Science, University of Pennsylvania, PA, 1988.

[17] A. Kobsa and W. Wahlster, editors. *User Models in Dialog Systems.* Springer-Verlag, Berlin, 1989.

[18] J. Lee, editor. *Preproceedings of the First international workshop on Intelligence and Multimodality in Multimedia Interfaces: Research and Applications*, 1995. http://www.cogsci.ed.ac.uk/ john/IMMI/.

[19] M. T. Maybury. Planning Multimedia Explanations Using Communcative Acts. In M. T. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 2. AAAI/The Mit Press, 1993.

[20] The MMI^2 Demonstrator Systems: A Multi-Modal Interface for Man Machine Interaction with Knowledge Based Systems. Technical Report RAL-94-016, Rutherford Appleton Laboratory, UK, 1994.

[21] J.D. Moore. *Participation in Explanatory Dialogues.* ACL-MIT Press Series in Natural Language Processing, 1995.

[22] J.D. Moore and C.L. Paris. Planning Text for Advisory Dialogues. 1989.

[23] Presentation Environment for Multimedia Objects (PREMO). International Standard Organization, ISO/IEC 14478, 25 Aug 1995.

[24] F. Roth and E. Hefley. Intelligent Multimedia Presentation Systems: Research and Principles. In M. T. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 1. AAAI/The Mit Press, 1993.

[25] J.R. Searle. What is a speech act ? In M. Black, editor, *Phylosophy in America*, pages 221–239. 1965.

[26] D. Sedlock, G. Doe, M. Wilson, and D. Trotzig. Formal and informal interpretation for cooperative dialogue. Submitted for publication.

[27] M. Spivey. *The Z Notation.* Prentice Hall International, 1987.

[28] J.W. Sullivan and S.W. Tyler, editors. *Intelligent User Interfaces.* ACM Press, 1991.

[29] A. Sutcliffe and P. Faraday. Designing Presentation in Multimedia Interfaces. In *CHI94*, 1994.

[30] D.S.W. Tansley and C.C. Hayball. *Knowledge Based Systems Analysis and Design.* Prentice Hall International, 1993.

[31] M.M. Taylor, F. Nèel, and D. G. Bouwhuis, editors. *The Structure of Multimodal Dialogue.* North Holland, 1989.

[32] A. Walker, editor. *Knowledge Systems and Prolog.* Addison-Wesley Publishing Company, 1987.

[33] M.D. Wilson. Enhancing multimedia interfaces with intelligence. *Multimedia systems and applications*, 1995.