The Role of User Models in Cooperative Interactive Systems*

Robert Kass
Center for Machine Intelligence, Ann Arbor, MI
Tim Finin
Paoli Research Center, Unisys Corporation, Paoli, PA

For interactive systems to communicate in a cooperative manner, they must have knowledge about their users. This article explores the role of *user models* in such systems, with the goal of identifying when and how user models may be useful in a cooperative interactive system. User models are classified by the types of knowledge they contain, several user modelling characteristics that serve as *dimensions* for an additional classification of user models are presented, and user model representations are discussed. These topics help to characterize the space of user modelling in cooperative interactive systems—addressing *how* they can be used—but do not fully address *when* it is appropriate to include a user model in an interactive system. Thus, a set of design considerations for user models is presented, while a final example illustrates how these topics influence the user model for a hypothetical investment consulting system.

I. INTRODUCTION

Computer systems have never been particularly cooperative with their users. The tendency to "do what I say," rather than "do what I mean" often leads to great frustration (and aggravation!) on the part of computer users. One reason for this communication difficulty is that computer programs do not have a good model of who they are talking to. They fail to recognize the knowledge, beliefs, goals, and plans of their human users, hence they frequently are not as cooperative as they could be. For some time, researchers have recognized the need to incorporate a model of the people who use computer programs into the programs themselves. Such a *user model* would contain specific information to aid the program in cooperating with its users.

This article analyzes the role of user models in *cooperative interactive* systems: systems that interact with a user and that seek to help the user achieve

* This work was partially supported by grants ARMY/DAAG-29-84-K-0061 from the Army Research Office, and DARPA/ONR-N00014-85-K-0807 from DARPA.

his goals. Cooperative interaction may be useful in a wide range of applications, such as database retrieval, help and advisory systems, intelligent tutoring, or expert systems. The need for user models in cooperative systems is particularly strong, since such systems must reason about the goals, plans, and beliefs of their users. This article explores *how* a user model may be used to support a cooperative interactive system, and *when* it is appropriate to use such models.

A. An Overview of this Article

The range of uses for (and requirements of) user models is very broad. The remainder of this section presents a general characterization of user models encompassing the range of things to which the term "user model" is usually applied. Section II addresses the question "What is to be modelled?" looking at the types of information contained in a user model. Section III discusses the types of user models themselves, focusing on several dimensions along which user models can be classified, Section IV discusses methods for representing user modelling knowledge, and Section V presents several considerations that affect the design of a user modelling system. Section VI illustrates the issues discussed in this article with an example of how a user modelling system might be used for a hypothetical investment advisory application. The concluding section summarizes some basic lessons that can be learned from the issues discussed here, and considers directions for future work in user modelling.

B. What is a User Model?

Defining "user model" is not an easy task. This section presents a general definition for "user model," then expands on that definition to give a stronger characterization of user models.

The term "user model" has been used in many different contexts to describe a range of program components. Some user models have simply assigned numeric values to rate the expertise of a user with the program, as in the Scribe Advisor,[1] or an explanation facility for NEOMYCIN.[2] Others seek to maintain substantial amounts of information about each individual who uses the system, as in Grundy,[3] the Real-Estate Agent,[4] or GUMS$_1$.[5,6] User models also differ in whether they differentiate between each user (as Grundy, the Real-Estate Advisor, and GUMS$_1$ do), or treat all users initially the same (as most intelligent tutoring systems do).[7–10]

Any definition for "user model" that includes this range of possibilities will necessarily be vague. However, an imprecise definition seems better than one that might preclude the examination of interesting aspects of the domain labelled "user modelling." Wahlster and Kobsa[11] propose a general definition for "user model" in the context of natural language dialogue systems. They suggest that:

A *user model* is a system knowledge source that contains explicit assumptions on all aspects of the user that may be relevant for the dialog behavior of the system.*

Although this general definition will be used throughout the article, user models that share the following features with *knowledge bases* will be of particular interest.

- *Separate Knowledge Base*–Information about the user is represented explicitly in a separate module rather then distributed throughout the system.
- *Multiple Use*—Since the user model is explicitly represented as a separate module, it can be used in several different ways (e.g. to support a dialogue or to classify a new user). This requires that the knowledge be represented in a more general way that does not favor one use at the expense of another. The goal is to express the knowledge in a general way that allows it to be reasoned about as well as reasoned with.
- *Explicit Representation*—The knowledge in the user model should be encoded in a representation language sufficiently expressive to support the ways it will be used. Typically, this means the representation will include a set of inferential services to draw further conclusions about the user beyond those explicitly represented.
- *Support for Abstraction*—The modelling system should provide ways to describe abstract as well as concrete entities. For example, the system should be able to discuss classes of users and their general properties, as well as individuals.

C. How Can User Models be Used?

The knowledge about a user that a model provides can be used in a number of ways in an interactive system, as illustrated by the taxonomy in Figure 1. At the top level, user models can be used to support (1) the task of recognizing and interpreting the information seeking behavior of a user, (2) providing the user with help and advice, (3) eliciting information from the user, and (4) providing information to him. Situations where user models are used for many of these purposes can be seen in examples throughout this article.

This broad characterization of user models admits a wide range of user modelling factors for consideration. These factors provide dimensions for plotting the various types of user models. In Section III these dimensions will be explored to get a better feel for the range of user modelling possibilities. Given

* Wahlster and Kobsa's original definition was presented only in the context of natural language systems. We have modified it slightly to encompass any interactive system.

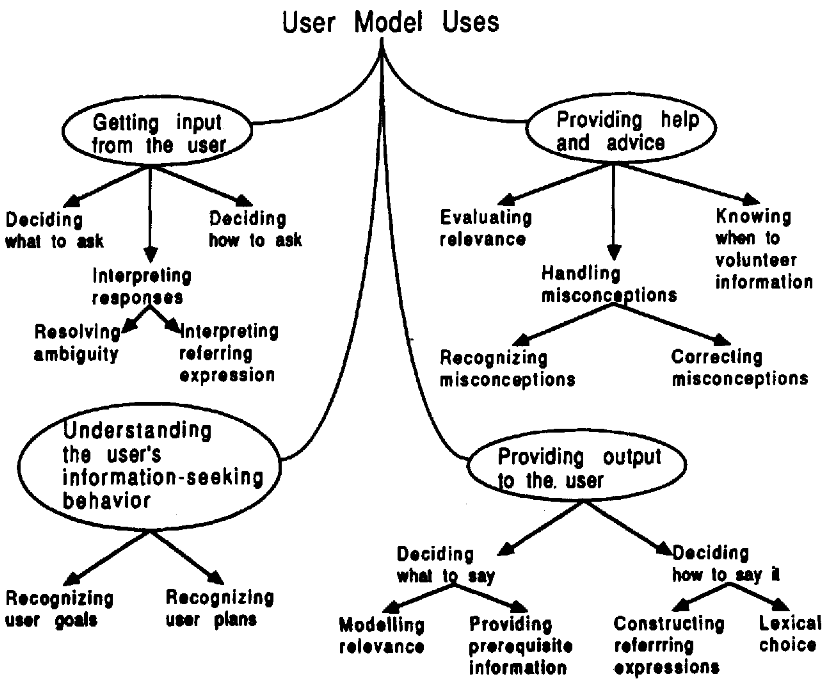
User Model Uses
Getting input from the user
Deciding what to ask
Deciding how to ask
Interpreting responses
Resolving ambiguity
Interpreting referring expression
Understanding the user's information-seeking behavior
Recognizing user goals
Recognizing user plans
Providing help and advice
Evaluating relevance
Handling misconceptions
Knowing when to volunteer information
Recognizing misconceptions
Correcting misconceptions
Providing output to the user
Deciding what to say
Deciding how to say it
Modelling relevance
Providing prerequisite information
Constructing referrring expressions
Lexical choice

Figure 1. Uses for knowledge of the user.

this range of possible types of user models, the factors influencing the feasibility and attractiveness of particular types of user models for given applications can be examined (Section V). First, however, the types of information a user model should be expected to keep are explored.

II. THE CONTENTS OF A USER MODEL

What information must be maintained in a user model? This section considers the primary types of information about a user that a system needs in order to interact in a cooperative manner. This information consists of knowledge about the user's goals, plans, and preferences, and about the user's knowledge and beliefs.

A. Goals

Recognizing the goal of a user in asking a question is essential for a computer system to provide an effective response. Frequently, a user will not explicitly state a goal when posing a question. For example, if a person walks up to an information booth in a train station and asks the attendant

"Can you tell me what time the next train to the airport departs?"

it is unlikely that he is trying to determine the level of knowledge about airport trains possessed by the attendant. Instead, the attendant infers that the inquirer wants to be told the time of the next airport train, probably so that he can board that train. The attendant infers these goals from his model of people who ask questions at train station information booths. This model may be very simple, but is still a user model under the loose definition we have given.

Sometimes goal inference requires a more complex model. Consider the following question posed to a course advisor[12]

"Is Professor Smith teaching Expert Systems next semester?"

A simple yes-no answer to this question is possible, but the questioner is probably more than just idly curious about Professor Smith and and the Expert Systems course. A cooperative response by the advisor could be:

1. "No, but Professor Smith is scheduled to teach it next year;"
2. "No, Professor Jones is teaching Expert Systems next semester;" or
3. "No, Professor Smith is teaching Natural Language Processing next semester"

depending on whether the advisor knows the questioner (1) wants to take the Expert Systems course with Professor Smith, (2) simply wants to take the Expert Systems course, or (3) wants to take a course with Professor Smith. In this case, the goal of the user cannot be determined from the interaction context alone. Rather, the advisor's response depends on what he knows already about the user from earlier information or other sources. For example, knowledge that the student has already taken the expert systems course eliminates the hypothesized goals (1) and (2).

User models are needed to aid in recognizing user goals in areas other than advisory systems. An intelligent help system must be able to determine what help the user needs before it can effectively respond.[13] This may be determined from an explicit statement by the user about what help is needed (as in the MACSYMA Advisor[14] or in an EMACS help system[15]), or by monitoring the user's activity and detecting when help is needed (as done by WIZARD, a help system for the VAX/VMS operating system[16]).

A data base query system must know about the goals of users to help determine what information to retrieve when a request is ambiguous. An example of this type of system is the Automated Yellow Pages Advisor (AYPA),[17] which must determine the correct categories of information to retrieve based on its understanding of the user's goal. For example, if the user tells the system

"My windshield is broken, help."

the system must infer that the user wants to replace the windshield, and hence needs to know about automotive repair shops that replace windshields, or glass shops that handle automobile glass.

Even expert systems need to know the goals of their users. Pollack, Hirshberg, and Webber[18] argue that "naturally occurring" expert system interactions include a negotiation process in which both the expert and the user come to an agreement on the goals of each party (as well as other things). Sparck-Jones[19] claims that as expert systems evolve to have natural, unstructured interactions with the user (such as accepting information, both solicited and unsolicited, from a natural language interface) knowledge of the user's goals will be necessary to help interpret the user's utterances and to guide the content and form of the explanations generated by the system.

Goals are not simple objects. Typically a user will have multiple goals when asking a question of the system. Allen[20] distinguishes between *task* goals and *communicative* goals in a discourse. The communicative goal is the immediate goal of the utterance. Thus in the question

"Can you tell me what time the next train to the airport departs?"

the communicative goal of the questioner is to discover when the next train leaves. The task goal of the user is to board the train. Carberry[12] makes a similar distinction between the *local* and *global* context of goals, and McKeown distinguishes between *current* and *relevant* goals.[21] Thus, question answering systems must be able to respond to goals at different levels, and hence user models need to be able to assist in determining a multiplicity of concurrent user goals.

The difficulty of obtaining goal information for a user model can vary greatly. For example, the simplest case is when the user explicitly states a goal when formulating the question, such as:

"I need to catch a plane at 5:15, when is the next train to the airport?"

The range of possible goals may also be strongly constrained by the domain of the system. Thus, boarding and meeting trains might be the only task goals of a train information system. This greatly simplifies the task of inferring the user's goals. On the other hand, the system itself may control the goals of an interaction, as is the case in some tutoring systems. In these cases it is possible for the user to have goals beyond the scope of the program, but the program cannot reasonably be expected to recognize and respond to such goals.

Inferring user goals is often much more difficult, however. Previous discourse may help identify a particular goal or at least constrain the range of possible goals. Another possibility is the use of stereotype information (discussed in Section III), in which a small number of facts may trigger a whole set of assumptions about the user.

B. Plans

Plans and goals are closely related. A plan is a proposed sequence of actions an individual has in mind to achieve a particular goal. A plan may have subparts that are also plans, with their own subgoals to achieve. Thus, situations where a user model is needed to infer user goals typically require the system to infer the user's plan as well.

Plan inference is necessary in some situations where goal inference is not, however. In intelligent tutoring systems (ITS's), the goal is usually implicit or defined by the system. For example, many systems operate in a problem solving environment where the ITS assumes the student's goal is to solve a problem posed by the system. However, many plans can be used to achieve the same goal; an ITS must be able to identify the plan followed by the student to determine whether or not it is valid. The detection of an invalid plan (a "buggy" plan) may then lead to an interruption by the system to initiate some level of tutoring.

Plan inference has been especially important in intelligent tutoring systems for novice programmers. Recent work in this area includes PROUST,[22,23] which detects bugs in novice programmer's Pascal programs; TALUS,[24] which detects bugs in Lisp programs; and GREATERP,[9,25] an interactive tutoring system for Lisp. Each of these systems has an "expert model,"* containing the knowledge of how to correctly solve the problems encountered by the student, and a "bug library," containing "buggy procedures" or "mal-rules"[26] for likely student mistakes. The correct model, together with the bug library, forms the user model in these systems. In the systems described, all model information is preencoded into the system.

The intelligent help system WIZARD[16] also must infer the plans of a user. In this case the steps of the plan may be interleaved with other operations or steps in other plans. WIZARD must be able to recognize when an action is part of a certain plan. The WIZARD system does this by keeping a library of inefficient plans it can recognize, building up partial plans as the user acts. When a complete, inefficient plan is recognized, WIZARD will interrupt to advise the user of better methods for achieving the same goal. The MACSYMA Advisor[14,27] also infers user's plans, using a history of the user's interaction with Macsyma. Once a plan is found, the Advisor uses a library of common errors and heuristics to try to determine the user misconceptions that led to the faulty plan.

Allen and Perrault[28] have focused on recognizing *obstacles* to user's plans. For example, suppose an information booth attendant hears the question

"I need to catch a plane at 5:15, when is the next train to the airport?

and recognizes that it is too late to get a train from the station in time to catch a 5:15 plane. Cooperative behavior requires that he inform the questioner of the problem with the plan, and perhaps suggest an alternative. This capability requires both detailed knowledge of plans and their components, and the ability

* Actually the expert model for GREATERP is an *ideal student* model, reflecting the abilities of a very good student, but not the sophistication of an expert in Lisp programming.

to infer such plans in the user. In systems with restricted domains (such as advice about trains) this is not too difficult. In more diverse domains, inferring plans is much harder.

One goal of Carberry's TRACK system[12,29] is to deal with plan inference in a more complex domain (in this case student course advising). She uses a tree structure of plan contexts to keep a history of the discourse as it progresses. The path from the root to the current context represents all active plans the user has at a given time, while other branches in the tree represent other possible or inactive plans. Thus, a plan history is remembered. If the user switches plans (as is often the case when a user has several goals with plans for each) the system can instantiate a new plan or find the old plan in the tree and reactivate it (if the user is returning to a previously mentioned plan).

Plan inference is thus similar to goal inference, but even more difficult. In domains where a large range of plans is possible, the common modelling solution is to encode all (or most) possible plans. This reduces plan inference to a matching problem, but requires an omniscience on the part of the system designers to anticipate all the plans a user might develop (both good and bad). Carberry's approach seems to be a good step towards handling plan inference in a more general way. Her work points to the importance of keeping a history of interaction and the context of that interaction, to aid in the inference and to be able to return to previous contexts if necessary.

C. Preferences and Attitudes

People are subjective. They hold beliefs on various issues that may be well founded or totally unfounded. They exhibit preferences and bias toward particular options or solutions. Interactive systems need to be able to "meet people at their own level" to communicate in an acceptable manner. Admittedly, recognizing and dealing with personal bias, preference, or perspective does not have the same level of importance as inferring user goals or plans. For this reason not as much attention has been paid to preferences in user modelling. Personal preferences do exist, however, and play a significant role in the behavior of the people that hold them.

Interactive systems need to consider personal attitudes when generating responses. The choice of words used, the order of presentation, and the presence or lack of specific items in an answer can drastically alter the impact a response has on the user. Jameson[30,31] addresses this to some degree in the IMP system. IMP plays the role of an informant responding to questions from a user concerned in evaluating a particular object (in this case, an apartment). IMP can be provided with a particular bias (for or against the apartment in question, or neutral) and uses this bias in responses it makes to the user. Thus, if IMP is favorably biased towards a particular apartment its responses will include additional related information that favorably represent the apartment, while attempting to temper negative features with qualifiers or additional nonnegative features. Thus, IMP strives to be a cooperative, biased system while appearing to be objective.

Swartout[32] and McKeown[21] address the effects of the user's *perspective* or *point of view* on the explanations generated by a system. In the XPLAIN system, built to generate explanations for the Digitalis Therapy Advisor, Swartout uses a very rudimentary technique to represent points of view. For each rule in the system, points of view that would find this rule meaningful or important are attached. When generating explanations for a user the system keeps *include* and *exclude* lists of the viewpoints held by the user; rules with a viewpoint on the include list are used in generating an explanation. McKeown uses intersecting multiple hierarchies in the domain knowledge base to represent the different perspectives a user might have. This partitioning of the knowledge base allows the system to distinguish between different types of information supporting a particular fact. When selecting what to say, the system will choose information supporting the point the system is trying to make and that agrees with the perspective of the user.

Utterances from the user must be considered in light of potential bias as well. Sparck-Jones[19] considers a situation where an expert system is used to compute benefits for retired people. The system is used directly by an *agent* who talks to the actual people under consideration by the system (the *patients*). In this case, the system must recognize potential bias on the parts of both agent and patient. The patient may withhold information or try to "fudge" information in order to improve their benefits (as well as a number of other reasons, such as distrust of government or computers), while the bias of the agent may color the way information is given to the expert system.

Rich,[3] and Morik and Rollinger[4] have built user modelling systems that model the preferences or dispositions of the users, but their systems are far from the situations described above. In both GRUNDY (Rich) and the Real-Estate Advisor (Morik and Rollinger), the domain of the application itself is the user's preferences, so the systems can directly interrogate the user about their preferences. In the systems envisioned by Sparck-Jones and Jameson, the bias of the user is "beneath the surface;" it is not possible for the application to directly interrogate the user about such biases without causing serious affront to the user. Thus, this type of information must be inferred directly from what the user says—a very hard task.

D. Knowledge and Belief

Any complete model of a user will include information about what the user knows, or believes to be true. In the context of modelling other individuals, an agent does not have access to objective truth, hence cannot really distinguish whether a proposition is *known* or simply *believed to be true*. Thus, the terms *knowledge* and *belief* will be used interchangeably.

Modelling the knowledge of a user involves a variety of things. First, there is the knowledge the user has of the domain of the application system itself. In addition, a user model may need to model information the user has about concepts beyond the actual domain of the application (*common sense* or *world* knowledge). Finally, any user, being an intelligent agent, has a model of other agents (including the system) and even of himself. These models are recursive, in the sense that the user's model of the system will include information about what he believes the system thinks about him, about what he believes the system believes he believes about the system, and so on. Each type of user knowledge is discussed in the following paragraphs.

Domain Knowledge

Knowing what the user believes to be true about the domain is useful for any system that must generate explanations of its behavior. The system needs to know what concepts and terms the user understands and is comfortable with, so that the explanation can incorporate such terms, while avoiding those terms that the system believes the user does not understand, or is confused about. This is especially true of intelligent help systems,[13] which must provide clear, understandable explanations to be truly helpful. Providing definitions of data base items (such as the TEXT system does[33]) has a similar requirement to express the definition at a level of detail and in terms that the user understands.

Modelling user knowledge of the domain is also important for detecting misconceptions the user might have about the meaning of terms or the relationship of concepts in the domain. McCoy's ROMPER system[34] does this in the domain of financial instruments. ROMPER must identify that the user holds a belief that is inconsistent with its own belief about the domain, then try to correct this misconception by providing an explanation refuting the incorrect information and supplying corrective information. A model of the user's domain knowledge will be very helpful in correcting the misconception since the system must avoid compounding the situation by an unclear explanation.

World Knowledge

Sometimes it is useful for a system to have knowledge about what the user knows about things beyond the narrow scope of the application domain. People seldom restrict themselves to the narrow confines of the domain in which they are working. Sparck-Jones[19,35] notes three types of knowledge that can be kept about a user.

- *Decision Properties* are those domain related properties the system believes the user knows.
- *Nondecision Properties* are properties of the user not used by the expert system in its decision making process, but may be useful for the interaction with the user. Examples of such properties include the name, age, or sex of the user.
- *Subjective Properties* are nondecision properties that tend to change over time.

Decision properties are the only properties an expert system can be expected to know anything about. Although nondecision properties and subjective properties are not directly needed by the expert system, they are important for appropriate interaction with the user. Without this information the expert system is much more limited in its ability to tailor responses to the user.

Detecting and using "foreign" concepts employed by the user is very hard. Until the day when computers have available vast stores of common sense knowledge about the world and about people, there will frequently be things a user says that do not have meaning to the system. The best a system can do in such a case is to try to learn from the user's utterance, to aid in this and further interactions, both with the same user and perhaps with others.

A special case of modelling information outside the domain of the application is when that information is closely related to the domain. Schuster has explored two situations where individuals try to use previous knowledge of a related domain when learning a new domain. In the first situation,[15] she discusses users familiar with other text editors attempting to use EMACS. An intelligent help system can detect errors the user is making, but frequently cannot provide the most helpful advice, because the expert does not realize the user has a model of how text editors work, based on experience with text editors other than EMACS. The user thus expects EMACS to work in a similar manner, which is sometimes not the case. If the advisor had knowledge of the user's previous experience (and models of other text editors), it would be able to provide much better help by directly addressing the misconception the user has due to previous experience. In the second situation,[36,37] she discusses a case where individuals learning a second language use their knowledge about the grammar of their native language as a model for the grammar of the new language. Again, if the system recognized that this misconception was occurring, much confusion could be avoided on the part of the user.

Knowledge of Other Individuals

Modelling what individuals believe about other agents is very important for question answering systems. Sidner and Israel[38] make the point that when individuals communicate, the speaker will have an *intended meaning* consisting of both a *propositional attitude* and the *propositional content* of the utterance. The speaker expects the hearer to recognize the intended meaning, even though it is not explicitly stated. Thus, a system must reason about what model the user has of the system when making an utterance, because this will affect what the system can conclude about what the user intends the system to understand by the user's statement.

A further complication in modelling a user's knowledge of other individuals are *infinite-reflexive* beliefs.[39] An example of such a belief is the following situation:

S believes that *U* believes *p*.
S believes that *U* believes that *S* believes that *U* believes *p*.
⋮

An important instance of such infinite-reflexive beliefs are mutual beliefs. A mutual belief occurs when two agents believe a fact, and further believe that the other believes the fact, and believes that they both believe the fact, and so on. Kobsa has pointed out that in the context of user modelling only *one-sided* mutual beliefs, i.e. what the system believes is mutually believed, are of interest.

For a system to determine the intended meaning of a user's utterance, five beliefs of both parties must be dealt with:[38]

(1) beliefs about the characteristics of the current situation;
(2) beliefs about the speaker's beliefs and goals;
(3) beliefs about the context of discussion (*discourse content*);
(4) beliefs about what conventions for action exist between speaker and hearer; and
(5) beliefs about what is mutually believed with respect to 1–4.

For a cooperative conversation to ensue, both parties in the conversation must reach mutual agreement on the content of these beliefs. Joshi[40] calls this process "squaring away." Squaring away may involve clarification subdialogs in which both parties explicitly explore their beliefs and seek to reach a mutual agreement (common beliefs) before continuing with the conversation.

To summarize, several types of knowledge may be required for an interactive system to effectively communicate with the user, including knowledge of the user's goals, plans, preferences, and beliefs. Not all of this information may be required for any given application, but each type is needed in some forms of interaction, while a truly versatile interactive system would use all forms.

III. THE DIMENSIONS OF A USER MODEL

User models are not a homogeneous lot. The range of applications for which they may be used, and the different types of knowledge they may contain indicate that a variety of user models exist. In this section, the types of user models themselves, classified according to several *dimensions*, are studied.

Several user modelling dimensions have been proposed in the past. Finin and Drager[5] have distinguished between models for individual users and models for classes of users (the *degree of specialization*), and between long or short term models (the *temporal extent* of the model). Sparck-Jones[19] adds a third, the modifiability of the model. *Static* models do not change once they are built, while *dynamic* models change over time.

Rich[1,3] likewise has proposed these three dimensions, but treats the modifiability category a little differently. Instead of static models, she describes *explicit* models, models defined explicitly by the user and that remains permanent for the extent of the session. Examples of explicit models are "login" files or customizable environments. She uses the term *implicit* model for models that are acquired during the course of a session, and that are hence dynamic. This characterization seems to mix two separate issues: the method of model

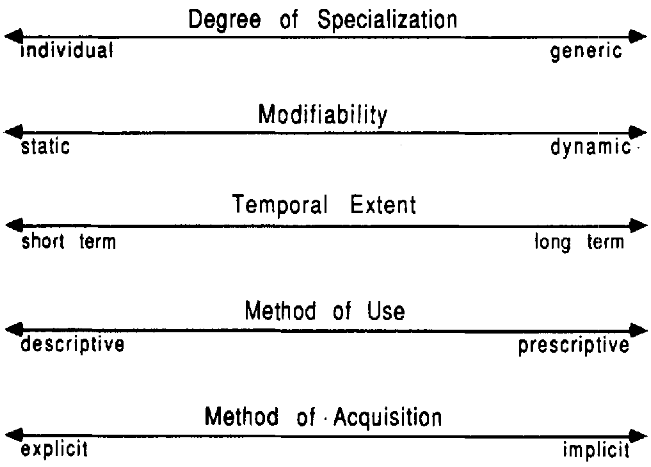

Figure 2. Dimensions of a user model.

acquisition, and the modifiability of the model. Thus the modifiability category will be limited to refer only to whether the model can change during a session, while the method of acquisition constitutes a separate category. An additional category that will be discussed is the method of use for the user model. Figure 2 summarizes these dimensions. The following sections will explore each of these user modelling dimensions in turn.

A. Degree of Specialization

User models may be *generic* or *individual*. A generic user model assumes the set of users is homogeneous—all individuals using the program are similar enough with respect to the application that they can be treated as the same type of user. Most programs have a generic user model, including many programs that keep an explicit model of the user. Examples of these are: many ITS programs,[7-9] the MACSYMA Advisor,[14] and question answering systems designed to infer goals and plans of the user.[12,17,20,28,41]

A generic user model is usually a collection of facts assumed to be true of the user, but it also can include rules used to derive new facts about the user from known facts.[5,6] A common extension to a generic user model is the use of a *set* of generic models, called *stereotypes*, each representing a separate class of users. This creates the additional problems of selecting the most appropriate stereotype from the set, and detecting and recovering from a bad choice.

A system may also keep an individual model of each user. In many systems, this is expressed as a "profile" of the user. Such a profile typically has initial default settings for the various attributes it contains (a sort of stereotype), and some mechanism for allowing the user or the system to change the attribute values.

Generic and individual models are often combined to take advantage of features of each. When initially building a user model, certain pieces of information from the individual model serve as *triggers*,[3] causing the system to attribute the features of a stereotype to the user. This enables a system to quickly build a robust user model by including the stereotype assumptions, yet enables it to override those assumptions if further information about the specific user is acquired. Systems that use stereotypes such as GRUNDY,[3] the Real-Estate Advisor,[4] and GUMS$_1$[5] further enhance the use of stereotypes by allowing them to be arranged in a hierarchy. As more information is discovered about the user, more specific stereotypes are activated (moving down the tree as in GUMS$_1$), or the user model invokes several stereotypes concurrently (as in GRUNDY).

A user modelling system could use a combination of these approaches as well. Consider a data base query system. A generic user model may be employed for areas where the user population is homogeneous, such as modelling the goals of users of the system. At the same time, individual models might be kept of the domain knowledge of the users, their perspective on the system and the level of detail they expect from the system.

B. Modifiability

Users models can be static or dynamic. Static user models do not change during the course of interaction with the user, while dynamic models can be updated as new information is learned. A static model might either be pre-encoded (as is implicitly done with most programs) or might be acquired during an initial session with the user before entering the actual topic of the discourse. Dynamic models will incorporate new information about the user as it becomes available during the course of an interaction. User models that track the goals and plans of the user are necessarily dynamic models. Rich's examples of login files or mail system parameters are essentially static, although it is possible for the user to explicitly change them during the course of the interaction.

As with the degree of specialization, different degrees of modifiability may apply to different types of model information. Sparck-Jones[19] refers to objective properties of the user (things like age and sex) that are not expected to change over the course of a session. Objective properties consisting of the Decision and Nondecision properties in her classification, require only static modelling, while subjective properties are changeable, hence dynamic.

C. Temporal Extent

At the extremes, user models can be short term or long term. A short term model might be built during the course of a conversation, or even during the course of discussing a particular topic, then discarded at the end. Generic, dynamic user models are thus usually short term since they have no facility for remembering information about an individual user.* On the other hand, individual models and static models will be long term. Static models by their nature are long term, while individual models are of little use if the information they retain from session to session is no longer available.

The temporal extent of information kept in a user model can be diverse. A user may have concurrent goals, each with time intervals during which they are active. Facts the system learns about the user are time sensitive as well. A good user modelling system will need the ability to determine the temporal extent of the information it holds and update itself appropriately when that information "expires."

D. Method of Use

User models can be used in two ways, *descriptively* or *prescriptively*. The descriptive use of a user model is the more "traditional" approach to user models. In this view, the user model is simply a data base of information about the user. An application queries the data base to discover the current view the system has of the user's state of beliefs, goals, plans, etc. Prescriptive use of a user model involves letting the model "simulate" the user for the benefit of the system.

An example of a prescriptive use of a user model is in *anticipation feedback loops*.[11] Anticipation feedback loops use the system's language analysis and interpretation components to simulate the user's interpretation of a potential response of the system. The HAM-ANS system[42] uses an anticipation feedback loop in its ellipsis generation component to ensure that the response contemplated by the system is not so brief as to be ambiguous or misleading. Jameson's IMP system[30] also makes use of an anticipation feedback loop to consider how its proposed response will affect the user's evaluation of the apartment under discussion.

E. Method of Acquisition

User models may be acquired *explicitly* or *implicitly*. Explicit models may be pre-defined by the system designer or user (such as login files or environment parameters), or acquired by directly querying the user. The use of bad plan libraries and stereotypes are examples of pre-defined user models, while Sleeman's UMFE system[43] (where the system explicitly asks questions about the user's knowledge before generating an expert system explanation) is an example of explicit acquisition from the user.

Implicit user model acquisition builds a model based on the observed behavior of the user, usually while he is interacting with the system. Many user

* Although it is conceivable that each interaction with an individual user might refine the generic model of all users in some way. Thus such a user model would converge on the "average user" after many sessions.

modelling systems use implicit acquisition to a small degree: to acquire information about the user in order to trigger a stereotype. GUIDON[44] used implicit acquisition to a greater degree, with specific rules to reason about the user's knowledge of medical diagnosis by comparing his behavior with what an expert (MYCIN) would have done. More recent work includes Lehman and Carbonell's work on building a model of user's grammar,[45] and our own work on implicitly acquiring knowledge about user beliefs in a cooperative advisory dialogue.[46,47]

As with most of the user modelling dimensions mentioned above, both methods of user model acquisition can profitably be used together in practice. Building a user model implicitly from scratch is impossible without significant constraints. If the range of information to be modelled is wide, much explicit information is necessary to "get going." This is certainly evident in human interaction, where people use many clues to quickly classify new individuals into previously built categorizations.

IV. REPRESENTATION OF A USER MODEL

Any explicit user model must represent the information it keeps about the user in some way. User models that keep information about user goals, plans, preferences, and beliefs require sophisticated knowledge representation capabilities. User modelling benefits, however, from the research on knowledge representation in general, which has considered how to represent such information. Thus, although representing such information in a user model can be difficult, the problems it poses are not unique to user modelling, meaning progress in knowledge representation will benefit user modelling as well.

User model representation is unique in some respects, however. In this section, four representational issues peculiar to user modelling are presented and discussed. These issues are: the user model's relation to the system's domain model, the problem of representing embedded belief structures, the inherent nonmonotonic nature of the knowledge in a user model, and default reasoning.

A. Relation to Domain Model

The first representational issue arises from the nature of the situations where user models are used, and from the difficulty in acquiring knowledge about the user. User models have been used almost exclusively in knowledge-based application systems. Usually, this means the system has knowledge about facts in the domain, or about plans and goals relevant to the application. In such systems, it is common to build the user model with respect to the system's domain model, since this can greatly reduce the amount of information that must be acquired specifically for the user model.

Two techniques may be used to build a user model with respect to the system domain model, resulting in overlay or perturbation models. *Overlay* modelling assumes that the information held by the user is a subset of the

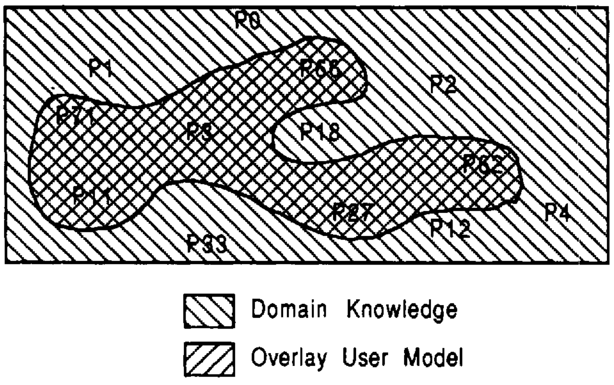

Figure 3. An overlay user model.

system's information, as illustrated in Figure 3. Overlay modelling was introduced by Carr and Goldstein[48] in an intelligent tutoring system to model students playing an adventure game called "Hunt the Wumpus." This game has an expert module containing rule-based knowledge of the best way to play the game. As a student plays, the tutoring system draws conclusions about which of the expert's rules are held by the student, based on the student's actions. Thus, the model of the student is built by marking which domain knowledge the student has, so the student model is an overlay of the system model.

A variation on overlay modelling, called *differential* modelling, can draw further conclusions about the user. In overlay modelling, the information the system believes the user has is marked. Differential modelling also distinguishes information the user does not know from information for which no conclusion can be made by comparing the user's behavior with that of the system's domain expert. If the user does not use information the system would have used in the same situation, then it is assumed the user does not know this information. However, if neither the user nor the system use a piece of information in a given situation, no conclusion can be drawn. Systems that have used differential modelling include WEST[49] and GUIDON2.[8] A differential model is illustrated in Figure 4.

The main advantage of overlay modelling (and differential modelling) is that new knowledge does not need to be acquired for the user model—the user model is always merely a subset of the domain model. This advantage is also its drawback, since it seriously constrains what information the model can have about the user. Frequently user models are needed *because* the user's beliefs, goals, or plans are significantly different from what the system's domain model might have, in order to correct user misconceptions or instruct the user. Thus, overlay modelling is inadequate for such situations.

Perturbation models can represent information about the user beyond that of the system domain model, while maintaining a close link between the two. In a perturbation model (illustrated in Figure 5), the user model is assumed to be

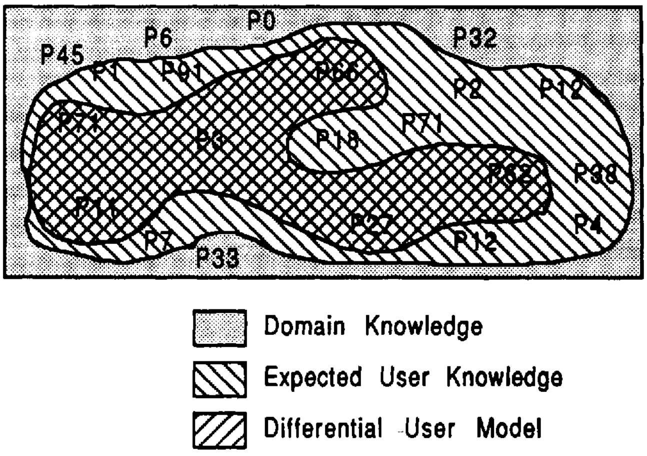

Figure 4. A differential user model.

similar to the domain model, differing only in certain perturbations to the domain model. Perturbation modelling is particularly attractive for modelling user plans, since a user plan will frequently be similar to a system plan, but not identical. Systems that use perturbation models include DEBUGGY, a system that modelled correct and buggy student subtraction skills;[7] PROUST, which identified Pascal programming errors;[22,23] and Sidner and Israel's system that recognizes speaker's intended meanings and plans.[38]

Perturbation models, since they are not limited solely to the system's domain knowledge, enable more robust and accurate user modelling. However, they also remove the limitations on the number of possible user models (since

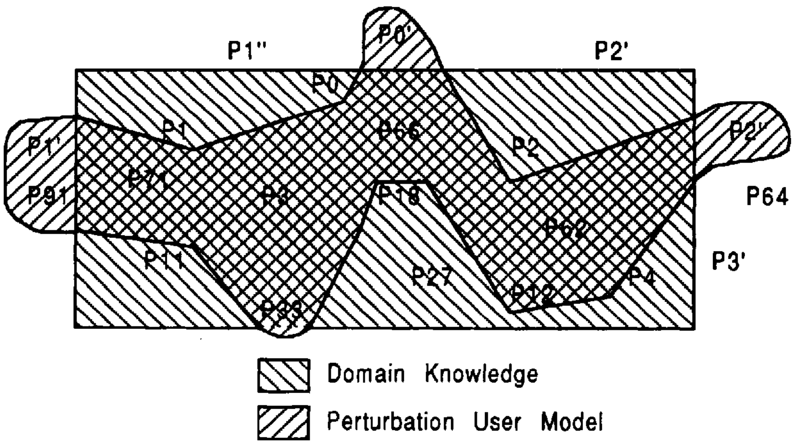

Figure 5. A perturbation user model.

any perturbation may be added), making user model acquisition more difficult. Thus, generic and stereotype modelling is frequently used to collect common features and reduce the space of user models that a system must search to find one that matches the user.

B. Representing Embedded Beliefs

A second representation feature peculiar to user modelling is the need (in some cases) to represent embedded beliefs, such as a user's beliefs about what another agent (such as the system) believes. For example, in normal conversation a speaker needs to reason about what the hearer believes about the speaker, and even what the hearer believes the speaker believes about the hearer. In common dialogue, the embedding may easily be three or four levels deep at times, while mutual beliefs require an infinitely deep embedding.

User's beliefs about other agents and mutual beliefs cause significant representational difficulties. Kobsa[50] lists three techniques that have been used to represent beliefs of other agents:

- The *syntactic* approach, where the beliefs of an agent are represented in terms of derivability in a first-order object-language theory of the agent.[40,51,52]
- The *semantic* approach, where knowledge and wants are represented by the accessibility relationships between possible worlds in a modal logic.[53–55]
- The *partition* approach, where beliefs and wants of agents are represented in separate structures that can be nested within each other to arbitrary depths.[50,56,57]

While the first two aproaches are primarily formal attempts, the partition approach has been implemented by Kobsa in the VIE-DPM system. VIE-DPM uses a KL-ONE-like semantic network to represent both generic and individual concepts. The individual concepts (and associated individualized roles) form elementary *situation descriptions*. Every agent modelled by the system (including the system itself) can be thought of as looking at this knowledge base from a particular point of view, or *context*. The context contains the *acceptance attitude* the agent has towards each individual concept and role in the knowledge base. An acceptance attitude can be either belief, disbelief or no belief.* An agent *A*'s beliefs about another agent *B* is formed by applying acceptance attitudes in *A*'s context to the acceptance attitudes of *B*. This technique can be applied as often as needed to build complex belief structures involving multiple agents, with special handling for infinite-reflexive beliefs.

* This is how acceptance attitudes were implemented in VIE-DPM. A wider range of values for the acceptance attitudes, such as a four-valued logic or numeric weights could easily be used instead.

C. Nonmonotonicity

A third representation issue unique to user modelling is the fact that conclusions made about the user sometimes need to be retracted. This nonmonotonicity may be due to erroneous conclusions that need to be retracted when correct information arrives, or may be due to changes in the beliefs, goals, or plans of the user over time.

The need to retract errant conclusions in a reasoning process is not unique to user modelling; truth maintenance systems (TMS's)[58,59] have been developed specifically to address this issue in general problem solving situations. User modelling, however, has features that distinguish it from conventional problem solving with a TMS. For example, in TMS problem solving, a set of available data is used by the problem solver to reason towards a conclusion, such as finding a diagnosis to explain the observed behavior of a malfunctioning electronic component. In user modelling, the input data (the user's behavior) is not always available to reason with, since new information is acquired during the interaction between system and user.

Another distinction about user modelling concerns the consistency of the knowledge base. A major component of Doyle's TMS[58] is its ability to ensure the consistency of the set of knowledge currently held (the *in* items in his terminology). In a user model, this may not be desirable, since users frequently hold inconsistent beliefs. User modelling needs a technique more like de Kleer's assumption-based TMS (ATMS),[59] which maintains consistency by indicating what assumptions sanction a particular conclusion. The important distinction between user model maintenance and standard truth maintenance is that a TMS enforces logical consistency, while user models cannot make that assumption. Users do not know all the logical consequences of their beliefs, thus are quite capable of holding beliefs that are inconsistent. To model people, a user model must be able to attribute such inconsistent beliefs to users. Thus, although truth maintenance systems may benefit the maintenance of a user model, the types of reasoning assumed must be significantly different.

D. Default Reasoning

Default reasoning[60] is a form of nonmonotonic reasoning, but merits special consideration. Default reasoning is a way of making inferences based on the lack of information, such as "In the absence of information to the contrary, if Tweety is a bird, then Tweety can fly." Subsequent information, such as the fact that Tweety is a penguin, can result in such default inferences being overridden.

Default reasoning is frequently used in user modelling, in the guise of generic or stereotypic user models. Such models make a set of assumptions about the user, assumptions that may be overridden by further information about the user. A problem with stereotype modelling is determining how to arbitrate between conflicting beliefs and conflicting stereotypes. Rich[61] and Finin[6] distinguish between definitional, or necessary, features of a stereotype, and those that are assumed by default. If a user is classified according to a stereotype $S1$, and new information about the user contradicts a default feature of $S1$, that feature may be overridden. However, if the new information is certain, and conflicts with a definitional feature of $S1$, then the user is misclassified, and the whole stereotype must be retracted. Conflicts between default information, or between stereotypes, are much more difficult to resolve—no clear methods have been developed yet.

In summary, user modelling shares much in common with general issues in knowledge representation, so progress there will benefit user model representation as well. However, some features of user modelling require techniques unique to it. Some of these techniques, such as overlay or perturbation modelling, and partitioning for representing embedded beliefs, have proven valuable while others, such as user model maintenance, require more work.

V. DESIGN CHARACTERISTICS FOR USER MODELS

The form and content of user models is quite varied; choosing to implement a user model for a particular system will involve making choices about what sort of model to implement. Associated with such choices are costs to implement them. These costs may consist of time and space requirements of the model and how they affect the system's performance, or they may be manpower costs to encode the knowledge of the user model. In many cases the use of a user model may not be feasible at all. This section explores issues important in choosing whether to implement a user model and if so, what kind of model to implement. The issues to be considered are:*

- Who bears the burden of responsibility for communication in the interaction?
- What is the penalty for error?
- How rich is the interaction space?
- How adaptable must the system be, and how quickly must it adapt?
- What type of interaction is required by the system?

Each of these issues is examined in the following subsections.

A. Responsibility

In any dialog, one or more of the participants takes the responsibility to ensure that the communication is successful. In most human dialogs, this burden is assumed by all participants, but not in all cases. In tutoring situations the teacher usually assumes most of the burden for ensuring both that he understands the student's question and that the student understands his response. An expert advisor also assumes much of this responsibility, although the advisee usually attempts to ensure that his problem is understood by the advisor.† The

* The first three issues are suggested by Sridharan in Ref. 62.

† This was observed by Pollack, Hirschberg, and Webber in Ref. 18.

concept of "responsibility for the dialogue" is closely related to the notion of dialogue initiative.

Which party must bear the burden of responsibility will significantly affect the user modelling requirements of an interactive system. If the user will assume responsibility for ensuring that the application understands his questions, and that the user understands the system responses, the user model can be much simpler. This is the case in traditional programs and most query systems. It is the user's responsibility to correctly phrase questions and interpret system responses. Even systems that provide cooperative responses to queries (such as those of Kaplan[63] and Mercer[64]) still leave the burden of responsibility with the user. Consequently, such systems have little need for goal or plan inference. Rather, they are more concerned with providing clear responses that do not mislead the user. Thus data base query systems require a user model that focuses primarily on the domain knowledge (and perhaps attitudes) of the user.

Systems that bear the burden of responsibility require a different type of user model. Such systems must be capable of identifying the goals and plans of the user. Thus, much knowledge must be acquired implicitly through the interaction with the user. These systems will have a very strong emphasis on short term modelling information. Intelligent tutoring systems are good examples of systems that bear most of the communication responsibility in an interaction, as are consultative expert systems, such as MYCIN.

Systems that share the burden of responsibility with the user will require the most complex user models. When the user assumes communication responsibility, the user model is relatively simple, while if the system assumes responsibility, it still has the advantage of directing the interaction, so that the user is not allowed to stray from the current topic of conversation. When responsibility is shared, however, the system must be able to recognize when the user wants to shift topics or alter the focus of the interaction. Thus, a user model for such a system will require an even richer representation of possible user goals and plans to be able to recognize when the user shifts away from the system's plan or goal. Expert systems that interact as peers with the user will have these requirements.

Current systems usually assume that the user will bear responsibility for the interaction (as in data base query systems), or else control the interaction themselves (as in expert systems). Systems that share responsibility with the user, for the most part, are still a research goal: Reichman[65] has analyzed human-human dialogues, and Sergot[66] has studied the architecture of interactive logic programming systems where the initiative of asking and answering queries can be mixed, but for the most part, systems capable of shared responsibility have not been implemented.

B. Penalty for Error

What if the user model is incorrect? How will this error influence the performance of the application system? When the accuracy of the system is critical, such that a wrong assumption about the user could have major negative consequences, much consideration must be given to the user model. For example, a machine translation system used to translate diplomatic messages sent between governments would have a high error penalty. The form of expression in such messages is very important. A translator would need a very good model of the sender to recognize the multiple meanings intended in the message. Failure to recognize one meaning in the translation, or incorrectly assuming another meaning to a sentence could cause serious problems for the system users.

A user model's penalty for error depends on the location of responsibility in an interaction. If the user assumes the responsibility for the communication content, the user model in the system need not be affected by the degree of penalty for error. However, if the system does assume responsibility to some degree, a high penalty for error implies the application must be very certain of its assumptions about the user. This may require a very large user model, and either sophisticated acquisition techniques or a large amount of explicitly encoded model information. If the penalty for error is small, the user modelling component need not be so sophisticated, for a wrong assumption can be corrected later in the interaction. Rich[1] notes that if the system produces a response the user does not understand or does not agree with, he will either ask specific questions about terms in the response or continue to dwell on the same issue. A user modelling system can recognize this action and modify its user model accordingly.

C. Richness of Interaction Space

The range of interaction a system is expected to handle greatly affects the type of user model required. If the possible user goals are very limited (such as meeting or boarding trains), a user model need not record much information about goals. The application in this case can simply check to see if the user's utterance matches any of the list of goals it recognizes and record the matching goal. When the range of interaction increases, more demands are placed on the user model. Inferring user plans is a typical example. The number of possible plans a user might have grows explosively as the complexity of the task increases. It is not possible to record all possible plans and simply search for a match. Instead, typical or likely plans are kept in the user model and these are searched to help identify user plans. "Buggy procedures" and "mal-rules" are examples of this technique.

The range of possible users also influences the type of user model. A generic model is suitable for a homogeneous class of users. As diversity among users increases, stereotype models or individual user models may be required.

D. Adaptability

Adaptability is closely tied to the richness of the interaction space and to the penalty for error: the greater the range of possible users, the more the system will be required to adapt. If the penalty for error is high as well, the acquisition abilities of the user model must be very good. The more adaptable the application must be, the greater the learning ability of the user modelling component must be.

Adaptability also involves the speed at which the system must adapt. Some interactions may require the system to deal with a wide range of users, but the user modeller has a relatively long time to develop a model of the individual. Such systems will have a low penalty for error. If the application must adapt very fast a large set of stereotypes will be useful, including the ability for the system to synthesize new, useful stereotypes when it recognizes the need. Such a user model will need to be concerned not only with modelling the current user, but also potential future users.

E. Type of Interaction

The type of interaction with the user will also affect the requirements for a user model. Wahlster and Kobsa[11] present a range of four types of man-machine interaction which have increasing requirements on the user modelling capabilities of the system:

(1) Simple question answering or biased consultation;
(2) Cooperative question answering;
(3) Cooperative consultation; and
(4) Biased consultation pretending objectivity.

Figure 6 shows these four types plus a final, very difficult category: Non-cooperative interaction. The following paragraphs take a short look at the user modelling requirements of each.

No explicit user model is required for simple interactions such as current data base query systems. If one were to be used it could be minimal, keeping track only of what the user knows about the domain itself. Biased consultation has similar requirements. No matter what the user says, the consultant will make the same recommendation. The only aid of the user model is in helping the system select information likely to sway the user.

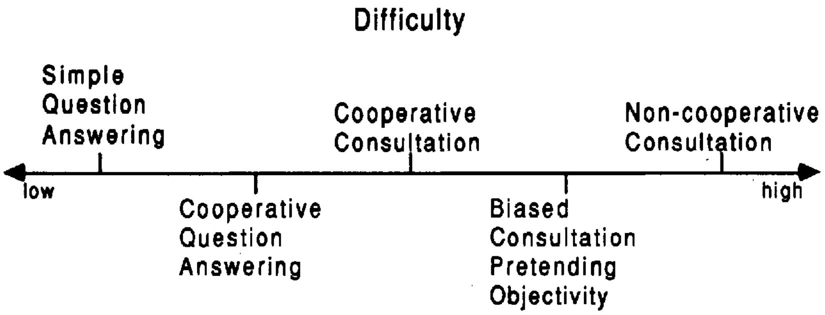
Difficulty
Simple Question Answering
Cooperative Consultation
Non-cooperative Consultation
low
high
Cooperative Question Answering
Biased Consultation Pretending Objectivity

Figure 6. Types of interaction.

Cooperative question answering requires the system to have some idea of the goals of the user. Typically, the range of goals the system can be expected to recognize will be quite limited, since the system is being used primarily as an information source. Such systems can employ a generic user model, since there will be little differentiation among users from the standpoint of the question answering system.

Cooperative consultation requires an extensive user model. As noted in Ref. 18, a consultation between an expert and the individual asking advice is like a negotiation. A consultation system must be able to recognize and understand a wide variety of user goals, further compounded by the fact that they may involve many misconceptions about facts of the domain of consultation. A good consultant should even be able to recognize analogies the user makes to other domains (as with consulting about text editors or foreign language learning[15,36,37]). Such consultations frequently involve extended interactions where much information about the user can be collected. In most cases, this information about the user should be retained, since it is likely further consultations will occur. Thus user models for cooperative consultation need to record all types of information about the user, and save this information in long term individual user models.

A biased consultation in which the system pretends objectivity (an electronic salesman) requires even more inferences about the user than cooperative consultation. Biased consultation requires a deep model of user attitudes, and how particular terms or concepts affect the attitude of the user. The system must have good models of what the user feels is cooperative conversation (since the system must appear objective), and of the user's model of the system (since the system must ensure that the user feels the system is objective).

Noncooperative interaction makes the acquisition of information about the user very difficult. Even witn cooperative interaction, much of the information assumed about the user is uncertain. If the user is not cooperating with the system, the possibility of the user lying (or withholding the truth) further complicates the acquisition of knowledge about the user. The system must be able to reason about the motivations of the user and be able to discern what information is likely to be untrue, and what information should not be influenced by the noncooperative goals or attitudes of the user. User models in such situations require very extensive knowledge about people in general, and categories of people in particular.

VI. AN EXAMPLE: INVESTMENT CONSULTING

This section illustrates the many user modelling issues discussed in the previous sections with an example of the particular user modelling requirements of a hypothetical investment advisory system. Individual users come to the system for advice on personal investments, or with questions about investing or the securities domain. The advisory application consists of an expert system with rule-based and factual knowledge about the domain. We will begin by looking at the types of information necessary in the user model, then examine the issues affecting choice of a user model. Finally we shall discuss what type of user model this situation requires.

A. Contents of the Model

What information will the user model need? Certainly the model will be needed to ascertain the goals of the user. The user may come to the system simply to get information ("What is a convertible security?"), to get advice on a simple decision ("Should I sell my stock now or wait for the dividends?"), to get very general advice ("I'll be retiring in three years, where should I be putting my money now?"), or to receive assurance or justification ("I put $25,000 into a market fund, was that okay?"). Pollack, Hirschberg, and Webber[18] list 11 types of goals a user might have in this situation. The examples listed above could be considered primary goals. As the interaction continues, new goals may develop, such as verifying that a recommendation is understood, or ensuring that a particular fact is properly considered. A user model for the consultant system will thus require a rich model of user goals, and must be able to recognize general goals and possible subgoals that develop during the course of the dialog. Some of these goals will be necessary for the performance of the underlying expert system, while others will be needed to enhance the interaction with the user.

The user model must also keep track of user plans. In many situations the user will not have a domain plan, rather the expert system will need to recommend one. On other occasions the advisor may be called upon to critique a user's plan. There are also many discourse strategies that the user may employ to achieve subgoals in the dialog; the user model will need to track when these strategies are employed by the user (or by the underlying system).

Preferences and attitudes may need to be modelled to some degree, to know the preferences or fears the user has with respect to particular types of investments. These preferences serve to constraint the range of possibilities the expert system must consider, since in most cases it should not attempt to override the user's preferences.

The user model will need to know what the user knows of the domain in order to communicate its recommendation clearly to the user, and to provide understandable explanations.[67] Furthermore, misconceptions held by the user often must be identified and corrected by the advisor, requiring knowledge of the user's beliefs about the domain. Knowledge of the user's beliefs about the system will be necessary if the user does not assume the system is always correct, in order to address user fears that particular recommendations are not well reasoned.

B. Modelling Issues

The investment advisor conducts a form of cooperative consultation. Some users may not be fully cooperative, but since the system is available for people who choose to use it, it is reasonable to expect that the users will cooperate with the system. Communication responsibility rests with both system and user. Since the determination of the goals of the dialog is achieved through a negotiation process between system and user, both parties must attempt to ensure that the communication is successful. Penalty for error is not too great; if the system misjudges the user it can expect the user to correct it, or else the user's behavior will give sufficient clues to allow the system to catch its mistake. Clarification dialogs may be required, but that is not a serious problem in this interaction.

The interaction space is very rich. Not only is the domain large, the user's goals and plans may be quite diverse as well. The system will need a good deal of knowledge to aid in identifying the type of user, and dealing with the user in a cooperative manner. Because of the range of possible interactions, the user model must be adaptable—the user model should very quickly recognize new goals and plans, since the conversation will falter without quick recognition. Adapting to changes in the knowledge and beliefs of the user should not be required as often, and should be less crucial in any case.

C. Choosing a Model

The user model information may be divided into two components, discourse related and domain related, each with different modelling requirements. The discourse related information in the model consists mostly of the goals and plans of the user. This is basically short term information that need not be retained beyond the end of the consultation. Pollack, Hirschberg, and Webber have identified a core classification for types of goals and plans in this setting, so a generic user model should be adequate to model this information. The model must definitely be dynamic. It will be used descriptively to report the current state of the user.

The domain related model is quite different, requiring information about what the user knows about the domain, plus information on user preferences and attitudes. All of this information is long term and specific to an individual. If users will use the system with moderate frequency, such information should be saved. Because of the great range in users, models of stereotypical users should be maintained as well. This will aid in inferring what knowledge the user has, and aid in inferring user plans and goals. The model will be used descriptively, but may also be used prescriptively when generating responses to the user. Acquisition of model information should be both explicit and implicit. The domain knowledge of the system itself is a good basis for modelling user domain knowledge, with stereotype information adding substantially to this. If a good basis of information is available, implicit knowledge acquisition should be much easier since the system will have many ways of verifying its assumptions.

A user model for the financial advisor will thus consist of two parts: a discourse user model containing short term, generic knowledge, and a domain user model containing long term information about individuals and classes of users via stereotypes.

VII. CONCLUSIONS

Sophisticated user models can serve many important functions in cooperative interactive systems—they are used to tailor the interaction to an individual user, to increase the system's cooperativeness, and to correct or even prevent misconceptions by the user. However, user modelling may not be appropriate for all interactive systems. Given the considerations discussed in Section V, and the current user modelling capabilities, some conclusions about when to use a user model can be made.

First, user models should only be used in situations where the range of interaction is sufficiently great that the user model can significantly affect the performance of the system. This does not preclude their use in more limited interactions, but the costs of implementing the user model can easily exceed the benefits that might be gained, particularly compared to other interaction techniques (such as menus) that are easier to implement and quite effective when the range of interaction is limited.

The fact that the user model will be used to alter the behavior of the system implies that the system will assume some degree of responsibility for ensuring the communication between user and system. This means the mode of interaction should at least be cooperative. Given the range of interaction types presented in Figure 6, cooperative question answering and cooperative consultation are appropriate types of interactions for using a user model. The more difficult forms of interaction, such as biased consultation pretending objectivity or noncooperative forms of interaction, are very difficult and at present have little practical use in the types of applications being built.

Finally, user models are currently viable only in situations where there is a low penalty for error. A high penalty for error demands very robust user models, requiring either extensive explicit coding of the user model, or sophisticated acquisition techniques. The human costs of coding a robust user model are very high, while sophisticated acquisition techniques will not be forthcoming soon. Thus in applications where the penalty for error is high, responsibility needs to remain on the shoulders of the user, with user modelling playing at most a secondary role.

A. General User Modelling Systems

Most of the work involving this kind of user models discussed in this article is at an early research stage. This research typically focuses on just one aspect of the overall user modelling problem, such as plan recognition or modelling multiple agents. Ultimately, intelligent human interfaces will need to address the task of implementing a complete, rich and general user modelling system.

A complete user modelling system has three functional components:

(1) A representation and maintenance component to manage the knowledge about the user;
(2) An acquisition component to add new knowledge to the user model (this includes both explicit and implicit acquisition techniques); and

(3) An access facilities component, to support and respond to the information needs of an interactive system.

Thus, user modelling should evolve in a manner similar to data bases or knowledge bases, becoming a centralized source of information and user modelling services for an application system.

Furthermore, the possibility of building *general* user models appears feasible. Our recent work has explored this issue in two ways. In GUMS (a General User Modelling System),[5,6] general user modelling is considered in terms of a knowledge-based environment containing a user modelling facility that an application can manipulate and query, focusing on the representation and maintenance aspects of user modelling. We have also considered a general user model to be an independent module,[46,47] with a defined set of facilities and methods for communicating with other system modules, focusing on implicit user model acquisition methods. In fact, a domain independent user modelling module appears to be feasible.

User modelling is not an easy task. Effective user modelling requires sophisticated knowledge representation, acquisition and reasoning abilities. However, suitable solutions for some of the significant problems exist, enabling further progress in user modelling research, as well as practical use of user models.

References

1. Elaine Rich, "Users as individuals: individualizing user models," *International Journal of Man-Machine Studies*, **18**:199–214 (1983).
2. J.W. Wallis and Edward H. Shortliffe, "Customizing Explanations Using Causal Knowledge," In *Rule-Based Expert Systems*, Bruce G. Buchanan and Edward H. Shortliffe (Eds.), Addison-Wesley, Reading, MA, 1984.
3. Elaine Rich, "User modelling via stereotypes," *Cognitive Science*, **3**, 329–354 (1979).
4. Katharina Morik and Claus-Rainer Rollinger, "The Real-Estate agent—modeling users by uncertain reasoning," *AI Magazine*, **6**, 44–52 (1985).
5. Tim Finin and David Drager, "GUMS$_1$: a general user modelling system," *Proceedings of the 1986 Conference of the Canadian Society for Computational Studies of Intelligence*, 24–30 (1986).
6. Tim Finin, "GUMS—a General User Modelling Shell," In *User Models in Dialog Systems*, Alfred Kobsa and Wolfgang Wahlster, (Eds.), Springer Verlag, Berlin—New York, 1988.
7. J.S. Brown and R.R. Burton, "Diagnostic models for procedural bugs in basic mathematical skills," *Cognitive Science*, **2**, 155–192 (1978).
8. Bob London and William J. Clancey, "Plan recognition strategies in student modelling: prediction and description," *Proceedings of the Second National Conference on Artificial Intelligence*, 335–338 (1982).
9. Brian J. Reiser, John R. Anderson, and Robert G. Farrell, "Dynamic student modelling in an intelligent tutor for Lisp programming," *Ninth International Conference on Artificial Intelligence*, 8–14 (1985).
10. Elliot M. Soloway, Beverly Woolf, Eric Rubin, and Paul Barth, "Meno II: An intelligent tutoring system for novice programmers," *Seventh International Conference on Artificial Intelligence*, 975–977 (1981).

11. Wolgang Wahlster and Alfred Kobsa, "Dialog-based user models," *Proceedings of the IEEE,* **74(7),** 1986.
12. Sandra Carberry, "Tracking user goals in an information seeking environment," *Proceedings of the Third National Conference on Artificial Intelligence,* 59–63 (1983).
13. Tim Finin, "Help and Advice in Task Oriented Systems," Department of Computer and Information Science, University of Pennsylvania, Technical Report MS-CIS-82-22, 1982.
14. Michael Genesereth, "The role of plans in automated consultation," *Sixth International Conference on Artificial Intelligence,* 311–319 (1979).
15. Ethel Schuster and Tim Finin, *Understanding Misconceptions,* Department of Computer and Information Science, University of Pennsylvania, Technical Report MS-CIS-83-12, 1983.
16. J. Shrager and Tim Finin, "An expert system that volunteers advice," *Proceedings of the Second National Conference on Artificial Intelligence,* 339–340 (1982).
17. A. Gershman, "Finding out what the user wants—steps toward an automated Yellow Pages assistant," *Seventh International Conference on Artificial Intelligence,* 423–425 (1981).
18. Martha A. Pollack, Julia Hirschberg, and Bonnie Webber, *User Participation in the Reasoning Processes of Expert Systems,* Department of Computer and Information Science, University of Pennsylvania, Technical Report MS-CIS-82-9, 1982.
19. Karen Sparck-Jones, *User Models and Expert Systems,* University of Cambridge, Cambridge, England, Technical Report 61, Computer Laboratory, 1984.
20. James F. Allen, Alan M. Frisch, and Diane J. Litman, "ARGOT: the Rochester dialogue system," *Proceedings of the Second National Conference on Artificial Intelligence,* 66–70 (1982).
21. Kathleen R. McKeown, "Tailoring explanations for the user," *Ninth International Conference on Artificial Intelligence,* 794–798 (1985).
22. W. Lewis Johnson and Elliot Soloway, "Intention-based diagnosis of programming errors," *Proceedings of the Fourth National Conference on Artificial Intelligence,* 162–168 (1984).
23. William Lewis Johnson, "Intention-Based Diagnosis of Errors in Novice Programs," PhD thesis, Department of Computer Science, Yale University, 1985.
24. William R. Murray, "Heuristic and formal methods in automatic program debugging," *Ninth International Conference on Artificial Intelligence,* 15–19 (1985).
25. Robert G. Farrell, John R. Anderson, and Brian J. Reiser, "An interactive computer based tutor for LISP," *Proceedings of the Fourth National Conference on Artificial Intelligence,* 106–109 (1984).
26. D.H. Sleeman and M.J. Smith, "Modelling student's problem solving," *Artificial Intelligence,* **16,** 171–187 (1981).
27. Michael R. Genesereth, "The Role of Plans in Intelligent Teaching Systems," In *Intelligent Tutoring Systems,* D. Sleeman and J.S. Brown (Eds.), Academic Press, New York, 1982, pp. 137–156.
28. James F. Allen and C. Raymond Perrault, "Analyzing intention in utterances," *Artificial Intelligence,* **15,** 143–178 (1980).
29. Sandra Carberry, "Modeling the user's plans and goals," *Computational Linguistics,* Special Issue on User Modelling, 1988.
30. A. Jameson, "Impression monitoring in evaluation-oriented dialog: the role of the listener's assumed expectations and values in the generation of informative statements," *Eighth International Conference on Artificial Intelligence,* 616–620 (1983).
31. Anthony Jameson "But What Will the Listener Think? Belief Ascription and Image Maintenance in Dialog," In *User Models in Dialog Systems,* Alfred Kobsa and Wolfgang Wahlster, Springer-Verlag, Berlin—New York, 1988.
32. William R. Swartout, "XPLAIN: a system for creating and explaining expert consulting programs," *Artificial Intelligence,* **21,** 285–325 (1983).

33. K.R. McKeown, *Text Generation—Using Discourse Strategies and Focus Constraints to Generate Natural Language Text,* Cambridge University Press, 1985.
34. Kathleen F. McCoy, *Correcting Object-Related Misconceptions*, Department of Computer and Information Science, University of Pennsylvania, Technical Report MS-CIS-85-57, 1985.
35. Karen Sparck-Jones, "Realism about User Modeling," In *User Models in Dialog Systems,* Alfred Kobsa and Wolfgang Wahlster (Eds.), Springer-Verlag, Berlin—New York, 1988.
36. Ethel Schuster, *VP^2: The Role of User Modelling in Correcting Errors in Second Language Learning,* Department of Computer and Information Science, University of Pennsylvania, Technical Report MS-CIS-84-66, 1984.
37. Ethel Schuster, "Grammars as user models," *Ninth International Conference on Artificial Intelligence*, 20–22 (1985).
38. Candace L. Sidner and David J. Israel, "Recognizing intended meaning and speakers' plans," *Seventh International Conference on Artificial Intelligence*, 203–208 (1981).
39. Alfred Kobsa, "Three steps in constructing mutual belief models from user assertions," *Proceedings of the Sixth European Conference on Artificial Intelligence*, 423–427 (1984).
40. Aravind K. Joshi, "Mutual Beliefs in Question Answering Systems," In *Mutual Belief,* N. Smith (Ed.), Academic Press, New York, 1982.
41. Jaime G. Carbonell, W. Mark Boggs, Michael L. Mauldin, and Peter G. Anick, "The XCALIBUR project: a natural language interface to expert systems," *Eighth International Conference on Artificial Intelligence*, 653–656 (1983).
42. Wolfgang Hoeppner, Thomas Christaller, Heinz Marburger, Katharina Morik, Bernhard Nebel, Mike O'Leary, and Wolfgang Wahlster, "Beyond domain independence: experience with the development of a German language access system to highly diverse background systems, *Eighth International Conference on Artificial Intelligence*, 588–594 (1983).
43. D.H. Sleeman, "UMFE: a user modelling front end subsystem," *International Journal of Man-Machine Studies*, **23**, 71–88 (1985).
44. William J. Clancey, "Tutoring Rules for Guiding a Case Method Dialogue," In *Intelligent Tutoring Systems,* D. Sleeman and J.S. Brown (Eds.), Academic Press, New York, 1982, pp. 201–226.
45. Jill Fain Lehman and Jaime G. Carbonell, "Learning the User's Language: A Step Towards Automated Creation of User Models," In *User Models in Dialog Systems,* Alfred Kobsa and Wolfgang Wahlster, (Eds.), Springer-Verlag, Berlin—New York, 1988.
46. Robert Kass, *Implicit Acquisition of User Models in Cooperative Advisory Systems,* Department of Computer and Information Science, University of Pennsylvania, Technical Report MS-CIS-87-05, 1987.
47. Robert Kass and Tim Finin, "Rules for the implicit acquisition of knowledge about the user," *Proceedings of the Sixth National Conference on Artificial Intelligence*, 295–300 (1987).
48. Brian Carr and Ira P. Goldstein, *Overlays: A Theory of Modelling for Computer Aided Instruction,* MIT Artificial Intelligence Laboratory, Cambridge, Massachusetts, Technical Report A.I. Memo 406, 1977.
49. J.S. Brown, R. Burton, M. Miller, J. DeKleer, S. Purcell, C. Hauseman, and R. Bobrow, *Steps Towards a Theoretical Foundation for Complex, Knowledge-Based CAI,* Technical Report 3135, Bolt, Beranek and Newman, 1975.
50. Alfred Kobsa, "Using situation descriptions and Russellian attitudes for representing beliefs and wants," *Ninth International Conference on Artificial Intelligence,* 513–515 (1985).
51. Kurt Konolige, "A deductive model of belief," *Eighth International Conference on Artificial Intelligence*, 377–381 (1983).

52. A. Joshi, Bonnie Webber, and Ralph Weischedel, "Living up to expectations: computing expert responses," *Proceedings of the Fourth National Conference on Artificial Intelligence,* 1984.
53. Robert C. Moore, "A Formal Theory of Knowledge and Action," In *Formal Theories of the Commonsense World,* R.C. Moore and J. Hobbs, (Eds.), Ablex Publishing, Norwood, NJ, 1984, pp. 319–358.
54. Joseph Y. Halpern and Yoram Moses, "A guide to the modal logics of knowledge and belief: preliminary draft," *Ninth International Conference on Artificial Intelligence,* 480–490 (1985).
55. Ronald Fagin and Joseph Y. Halpern, "Belief, awareness and limited reasoning: preliminary report," *Ninth International Conference on Artificial Intelligence,* 491–501 (1985).
56. Alfred Kobsa, "A taxonomy of beliefs and goals for user models in dialog systems," Unpublished paper from UM86, the International Workshop on User Modelling, Maria Laach, West Germany, 1986.
57. Y. Wilks and J. Bien, "Beliefs, points of view, and multiple environment," *Cognitive Science,* **7,** 95–119 (1983).
58. Jon Doyle, "A truth maintenance system," *Artificial Intelligence,* **12(3),** 231–272 (1979).
59. Johan de Kleer, "An assumption-based TMS," *Artificial Intelligence,* **28,** 127–162 (1986).
60. Raymond Reiter, "A logic for default reasoning," *Artificial Intelligence,* **13(1),** 81–132 (1980).
61. Elaine Rich, "Stereotypes and User Modelling," In *User Models in Dialog Systems,* Alfred Kobsa and Wolfgang Wahlster, (Eds.), Springer Verlag, Berlin—New York, 1988.
62. D. Sleeman, Doug Appelt, Kurt Konolige, Elaine Rich, N.S. Sridharan, and Bill Swartout, "User modelling panel," *Ninth International Conference on Artificial Intelligence,* 1298–1302 (1985).
63. S.J. Kaplan, "Cooperative responses from a portable natural language database query system," *Artificial Intelligence,* **19(2),** 165–188 (1982).
64. R. Mercer and R. Rosenberg, "Generating corrective answers by computing presuppositions of answers, not of questions," *Proceedings of the 1984 Conference of the Canadian Society for Computational Studies of Intelligence,* 16–19 (1984).
65. Rachel Reichman, "Plain-Speaking: A Theory and Grammar of Spontaneous Discourse," PhD thesis, Harvard University, 1981.
66. M. Sergot, "A Query-the-User Facility of Logic Programming," In *Integrated Interactive Computing Systems,* P. Degano and E. Sandewall, (Eds.), North-Holland, 1983, pp. 27–41.
67. Robert Kass and Tim Finin, "The Need for User Models in Generating Expert System Explanations," Department of Computer and Information Science, University opf Pennsylvania, Technical Report MS-CIS-87-83, 1987. Submitted to *International Journal of Expert Systems,* Special Issue on Natural Language Processing.