Int. J. Human–Computer Studies (1994) **40**, 31–62

A shell for developing non-monotonic user modeling systems

GIORGIO BRAJNIK AND CARLO TASSO

Laboratorio di Intelligenza Artificiale, Dipartimento di Matematica e Informatica, Università di Udine, Via Zanon 6, 33100 Udine, Italy

(Received 1 May 1992 and accepted in revised form 3 September 1993)

This paper first presents a general structured framework for user modeling, which includes a set of *basic user modeling purposes* exploited by a user modeling system when providing a set of services to other components of an application. At a higher level of abstraction such an application may perform a *generic user modeling task*, which results from an appropriate combination of some basic user modeling purposes. The central aim of the paper is to present, within the proposed framework, a flexible general-purpose shell, called *UMT* (*User Modeling Tool*), which supports the development of user modeling applications. UMT features a non-monotonic approach for performing the modeling activity: more specifically, it utilizes a modeling approach called *assumption-based user modeling*, which exploits a truth maintenance mechanism for maintaining the consistency of the user model. The modeling task is divided into two separate activities, one devoted to user classification and user model management, and the other devoted to consistency maintenance of the model. The modeling knowledge exploited by UMT is represented by means of stereotypes and production rules. UMT is capable of identifying, at any given moment during an interaction, all the possible alternative models which adequately describe the user and are internally consistent. The choice of the most plausible one among them is then performed using an explicit programmable preference criterion. UMT is also characterized by a very well defined and simple interface with the hosting application, and by a specialized development interface which supports the developer during the construction of specific applications. This paper includes an example application in the field of information-providing systems. UMT has been developed in Common LISP.

1. Introduction

User modeling has been recognized as an important and useful feature in several systems where the interaction with the user plays a crucial role (Kobsa & Wahlster, 1989; Kay & Quilici, 1991). User modeling enables flexible and profitable adaptation of the system behaviour to the specific characteristics and nuances of an individual user, improving in such a way the usability of the system, the economy of the interaction, and the capabilities of the system of understanding and processing user requests beyond the pure content of the user's utterances or commands. The application of user modeling techniques has been reported in many fields, such as dialog systems (Wahlster & Kobsa, 1989) and intelligent tutoring systems (Self, 1988). Another area where user modeling has been considered a fundamental requirement is that of information retrieval systems (Daniels, 1986; Belkin *et al.*, 1987; Brajnik, Guida & Tasso, 1987*a*, *b*; Belkin, 1988).

The design and development of systems based on user modeling constitutes a very ad-hoc and demanding process. Problems to be faced include:

1071–5819/94/010031 + 32$08.00/0

- extensive analysis of user population and identification of user features which are considered relevant in a given domain;
- design of appropriate mechanisms for acquiring, representing, and manipulating user information;
- design of effective methods for the exploitation of such user information during operation of the application system.

Moreover, when user modeling is approached with the knowledge-based system technology, other challenging problems emerge, related to the effort of eliciting knowledge about users, of representing it in an expressive and efficient way, and of identifying and constructing suitable inference engines capable of performing the user modeling activity.

For these reasons the idea of exploiting generic shells specifically devoted to user modeling has emerged, and a few systems have been developed along this line, such as GUMS (Finin, 1989), UM-tool (Brajnik, Guida & Tasso, 1990), BGP-MS (Kobsa, 1990) and UM (Kay, 1990, 1991), which can be integrated within a specific application in order to provide user modeling services. Even if no established and general theories underlying the user modeling task exist, all the above mentioned tools have been designed following some common idea. More specifically, all of them are based on an explicit information structure, called the user model, for storing the individual description of the current user of the system, and all of them exploit the idea of stereotypes (Rich, 1983) for representing different classes of users.

None of them, however, includes satisfactory mechanisms for approaching one of the typical features of the user modeling task, its non-monotonicity. In fact user models usually contain two main kinds of user information: observed facts and assumptions; assumed information may possibly be revised, if evidence against it is collected. The need for explicitly including non-monotonic capabilities within user modeling systems has been recognized by several authors (Finin, 1989; Rich, 1989; Kobsa, 1990; Brajnik, Guida & Tasso, 1990).

This paper first presents a general structured framework for user modeling, which includes a set of *basic user modeling purposes* to be provided by a user modeling system as a set of services to an external hosting application. The higher level reasoning tasks accomplished by the external application based on user modeling are further analysed and classified from the user modeling viewpoint, yielding a set of *generic user modeling tasks*. Secondly, the non-monotonic aspects of the user modeling process are outlined and motivated. Thirdly, within the proposed general framework, a new flexible general-purpose shell, called *UMT* (*User Modeling Tool*), is illustrated. UMT supports the design and development of user modeling applications which feature an original strategy, called *assumption-based user modeling*, for performing the modeling activity in a non-monotonic way. The modeling task is divided into two separate activities, one devoted to user classification and user model management, and the other devoted to consistency maintenance of the modeling process. Modeling knowledge is represented by means of stereotypes and production rules. UMT is capable of identifying, at any given moment during an interaction, all the possible alternative models which adequately describe the user and are internally consistent. The choice of the most plausible one

among them is then performed using an explicit preference criterion. UMT is also characterized by a very well defined and easy-to-use interface with the hosting application, and by a specialized development user interface which supports the knowledge engineering during the construction of specific applications.

This paper is organized in the following way. Section 2 presents a general framework for generic application systems exploiting user modeling capabilities. Section 3 discusses the main concepts underlying the non-monotonicity of the user modeling process and introduces the assumption-based user modeling approach. Section 4 illustrates the UMT shell, its architecture and mode of operation, while in Section 5 a case study on an information-providing system based on UMT is shown. Section 6 presents some comparisons with related work, and Section 7 concludes the paper.

2. User modeling in perspective

This section provides a conceptual description of user modeling (UM) and of its purposes. We start by analysing the concept of "user model".

The exploitation of user modeling capabilities within a software system may follow two alternative routes. The user model is *explicit* when it is directly represented in the system as a separate component, accessible during system operation. On the other hand it is *implicit* when it is hard-wired in the system in such a way that it cannot be recognized in a specific piece of code or collection of data: in a sense, it is dispersed in several points and hidden under the system code, such as in operating systems, editors, etc.

In this paper we will focus only on explicit user models, in order to concentrate on a wide class of interesting systems: flexible, adaptive, and capable of handling a wide spectrum of information about the user, and which may exploit complex acquisition and validation procedures.

In the literature several definitions of user model have been proposed which differ for its content, representation, and role. Our definition of user model is inspired by Kass and Finin (1988) which, in turn, refers to the well known knowledge representation hypothesis (Smith, 1982).

> A *user model* is a collection of explicit structural ingredients of a system that (a) we as external observers naturally take to represent a propositional account of the knowledge the system has of its user, and that (b) independent of such an external semantical attribution play a formal but causal and essential role in the behaviour that manifests that knowledge. (Kass & Finin, 1988: p. 6.)

Following Wahlster and Kobsa (1989), we define a *User Modeling System* as the part of a software system (henceforth called *UM host system*) whose function is: (i) to construct a user model; (ii) to store, update and delete data about the user which are supplied by other components of the host system (*UM observers*); (iii) to maintain the consistency of the model; and (iv) to supply other components of the host system (*UM consumers*) with information about the user. A *UM shell* is a generic domain-independent system whose purpose is to allow the development of user modeling systems.

Within a generic UM host system there are components which, from the UM viewpoint, operate as UM observers and others operating as UM consumers

(Orleant, 1991). Often these two functions are strongly interrelated and they are not carried out by two separate architectural components. This is the case, for instance, when some sort of feedback phenomenon takes place between the user and the host system. For example, suppose that the purpose of the application program is to filter, on the basis of a user model, the information that should be provided to the user. During presentation of selected information (i.e. when the host system is acting as UM consumer) the system takes into account the user's reactions (satisfaction, dissatisfaction, new interests, etc.) and these new data are included in the model. The host system is performing both the functions of a UM consumer and of a UM observer (when acquiring the user's reactions) and it is unfeasible to implement them with two different components. From a conceptual perspective, we prefer to clearly separate the two functions of UM consumer and UM observer, for their analysis can shed some light upon which user's aspects can be observed, which are to be inferred, and which can be directly used for supporting the host system.

The above definition of UM system implies that UM consumers and observers isolate the user from the UM system, with the advantage of restricting the range of problems that the UM system has to deal with. In fact, the host system which is observing the user's behaviour may be able to recognize inconsistencies which couldn't be recognized by the sole UM system because they depend on aspects of the behaviour which are not necessarily known to the UM system. For example, although the user may have said to the UM host that he or she is a computer expert (assertion included in the user model), the UM host which observes his or her behaviour (taking into account aspects which are not understandable by the UM system, such as repeatedly failing to remove a file from a directory) may find that he or she most probably is not an expert at all.

In order to better characterize the role of UM systems let us analyse their possible purposes, largely dependent on the task to be accomplished. A number of basic, relatively independent UM purposes have been identified in the literature (Kass & Finin, 1988; Self, 1988; Chin, 1989; Wahlster & Kobsa, 1989; Brajnik, Guida & Tasso, 1990; Cahour & Paris, 1991; Kay, 1991; Nwana, 1991). We propose the following list of *basic user modeling purposes,* i.e. possible contributions of the UM system to the overall operation of the host system:

1. to provide contextual knowledge for interpretation of the user's utterances during a dialogue (for example, the user's goals and plans);
2. to support in deciding what to present to the user (which concepts to present and to refer to, at what level of detail, etc.), either when the system provides information to or when it asks information of the user (Cahour & Paris, 1991);
3. to support in deciding how to present something (which phrase or discourse structure to adopt, which terms to use), either when providing or requesting information (Cahour & Paris, 1991);
4. to predict how the user will interpret the system's utterances [i.e. anticipation feedback (Wahlster & Kobsa, 1989)];
5. to understand the user's goals and plans (Carberry, 1988);
6. to assess the user's state of knowledge (in terms of concepts and relations which are known by the user, how well they are known, etc.);
7. to support in deciding whether to take the interaction initiative;

8. to diagnose the user's bugs (in the context of intelligent tutoring systems) by analysing the user's misbehaviour and identifying its possible causes, such as missing or erroneous knowledge;
9. to support in deciding how to correct the user's bugs and to extend the user's knowledge by adopting specific tutoring strategies;
10. to predict the user's likely response;
11. to provide a description of the user's characteristics (e.g. the user's needs) that could be used to assess the relevance of some information [for example, whether a reading could be interesting for the user (Rich, 1983)].

These basic UM purposes may be appropriately combined in the development of host systems capable of supporting more complex tasks. Several authors provided classifications of the tasks that may be accomplished by UM hosts (Kass & Finin, 1988; Kay, 1991; Cahour & Paris, 1991). We call *generic UM task* the activity, independent from the actual domain of application, carried out by a UM host and viewed from the UM perspective (i.e. with respect to the features which are relevant to UM). Figure 1 illustrates this general UM framework: the user's behaviour is observed by UM observers which provide input data to the UM system; the UM system supports, by means of some basic UM purposes, the operation of UM consumers which, from the UM point of view, is classified in terms of some generic UM task.

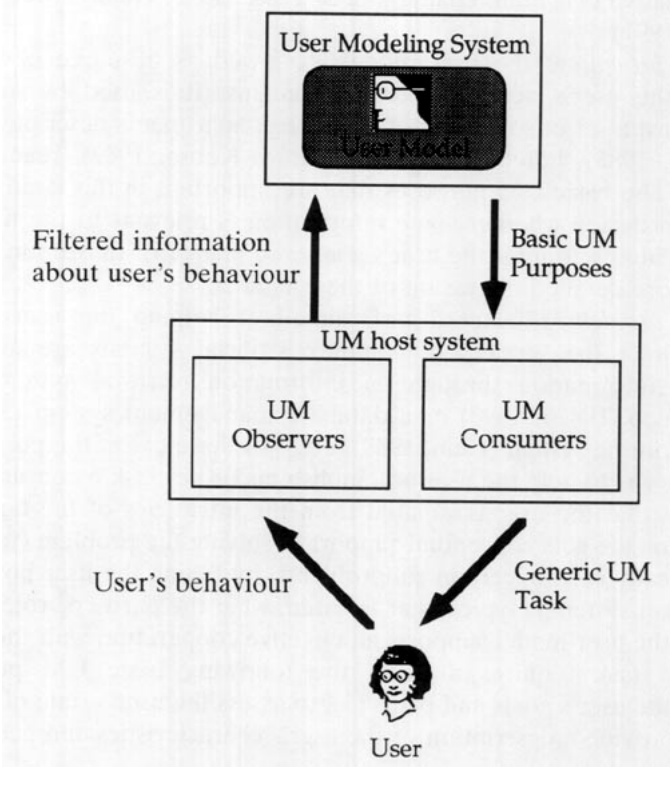

FIGURE 1. The general user modeling framework.

The following non-exhaustive list includes the most frequently reported generic UM tasks. We highlight the role of the user model as well as the basic UM purposes (numbers in brackets) that are essential to each task:

- *communication*: the user model is utilized to accomplish communication with the user. In particular the user model is relevant for interpreting the input from the user, and for generating the output to the user. Relevant basic UM purposes are: to interpret what the user provides (1), to infer user goals and plans (5), to decide what to present and how to present it (2, 3), to predict how the user will interpret the system's utterances (4), to decide whether to take the initiative (7).
- *explanation*: the UM system acts as a front-end between the user and a knowledge-based system. The role of the UM system is to tailor the knowledge-based system's responses to the user in terms of concepts which the UM system believes the user understands (see e.g. Swartout, 1983; Sleeman, 1985). This task requires at least the following basic UM purposes: to support in assessing the user's state of knowledge (6), in understanding the user's goals and plans (5), in deciding when to provide help information (7), and in deciding what to present to the user (2).
- *tutoring*: the purpose of the user model is to provide information that is relevant to the tutoring task. This requires the ability to understand the user's plans (5), to diagnose the user's behaviour (8), to correct and extend the user's state of knowledge (9), to predict the user's likely response (10), and to decide when to take the initiative (7) (Fum, Giangrandi & Tasso, 1990; Huang, McCalla, Greer & Neufeld, 1991).
- *information providing*: the user model is viewed as a source of knowledge concerning the user's needs in certain domains. It is used by matching its content against a set of some sort of offers [e.g. apartments descriptions (Morik & Rollinger, 1985), hotel rooms (Wahlster & Kobsa, 1989), readings (Rich, 1983), etc.]. The basic UM purposes that are important in this context concern the ability to decide whether some information is relevant to the user's needs (11) and the ability to infer the user's goals and plans (5). In Section 5.1 we will provide a more detailed discussion of these tasks.
- *cooperative problem solving*: a task aimed at helping the user in solving problems where the services of complex artificial systems are needed [e.g. looking for information through an information retrieval system (Brajnik, Guida & Tasso, 1987*b*, 1990) or a database management system (Kok, 1989), using an operating system (Chin, 1989), etc.]. In these cases, the purpose of the UM consumer is to help the user accomplish his or her task by ensuring, among other things, that the user is isolated from the intricacies of the target system and that he or she gets conceptual support in solving the problem (for example, by automatically solving certain subproblems, by telling the user how a session with the target system is typically articulated, what the partners' roles are, etc.). In this case the user model supports an effective cooperation with the user. This generic UM task requires at least the following basic UM purposes: to understand the user's goals and plans (5), to assess the user's state of knowledge (6), and to provide a description of the user's characteristics and needs (11).

A further way to characterize user models is by taking into account a number of

dimensions (Rich, 1983; Brajnik, Guida & Tasso, 1990; Kay, 1991). Four dimensions are important to our purpose:

Given vs. inferred user models—A user model is called *given* if it has been defined and coded at system design time and later stored for use during system operation. A given model is in a sense a read-only knowledge, which can be modified or updated only off-line from system operation. In a model which is given, only very limited updates, such as parameter adjusting, can occur automatically during system operation. A user model is called *inferred* if it is automatically constructed (by relying on some knowledge base) during system operation, without any direct external intervention by the system designer or the user.

Static vs. dynamic user models—A user model is called *static* if it is supposed to remain unchanged and fixed over the operation of the system it belongs to. A user model is called *dynamic* if it is continuously and incrementally refined, extended and updated during system operation in order to cope with new facts and evidences about the user.

Generic vs. individual user models—A user model is called *generic* if it is aimed at capturing common characteristics of the whole user population, focusing on the most frequent and shared features but discarding individual peculiarities. On the other hand, a user model is called *individual* if it tries to represent also the characteristics of each individual user in a given population, including those common to other users and those unique to each specific user.

Long vs. short-term user models—A *short-term* user model is created at the beginning of each session and it ceases to exist at the end of the interaction. A *long-term* one lasts for more than a single session.

3. User modeling and non-monotonic reasoning

3.1. THE BELIEF REVISION PROBLEM IN USER MODELING

Within the general framework described in the previous section, we will focus now on the non-monotonicity of user modeling. Several authors in the UM community have pointed out that this is an important aspect of UM systems which has not received, so far, an adequate treatment. The problem of implementing some sort of non-monotonic reasoning form in UM systems appears in almost every UM generic task and especially when UM is based on stereotypes (Rich, 1989). It was recognized in natural language interfaces (Kobsa, 1990), in intelligent tutoring systems (Huang, McCalla, Greer & Neufeld, 1991), in information-providing systems (Brajnik, Tasso & Vaccher, 1991), in cooperative problem solvers (Chin, 1989; Brajnik, Guida & Tasso, 1990), and in generic UM shells (Finin, 1989).

A common view of the UM process is a scenario with two interacting agents (the user and the UM host system), where the user model is viewed as a set of *beliefs* about the user held by the host system.

An important problem that must be faced by UM systems is then *belief revision*, i.e. deciding how to revise the available beliefs when facing an inconsistency caused

by newly acquired information. In the UM context the belief revision problem is due to three fundamental properties of UM which are typical of default reasoning scenarios (Doyle, 1979; Hanks & McDermott, 1987; Reiter, 1980).

First of all, *incompleteness*. The UM system is generally required to provide information about the user (goals, plans, needs, attitudes, etc.) while being in a situation of extremely incomplete knowledge. Incompleteness is due to two reasons: one is the narrow bandwidth of the interaction channel with the user (usually through screen, keyboard and possibly mouse) and the limited observational power of the UM system (usually unable to recognize visual communication clues, physical characteristics of the user, communication features, etc.). This leads to the consequence that the UM system will always have a very rough and poor representation of the user (if compared to human–human communication). The second reason for incompleteness (independent from the first one) is that when the user model is devoted to containing cognitive aspects about the user, these may not even be perceived by the user or, when perceived, he or she may be unable to express them in an adequate way (e.g. through a complete, accurate and precise statement of his or her needs). From the UM system viewpoint, these are (perceived or unperceived) unobservable user's aspects. In order for host systems to appropriately accomplish their tasks, UM systems must cope with incompleteness. The existence of unperceived unobservable aspects forces the UM system to try to extend its beliefs by taking advantage of any available knowledge about its population of users. When performing these extensions the UM system may recognize conflicts with already held beliefs, thus requiring belief revision.

In order to cope with the narrow interaction channel and the perceived but inexpressible aspects, the host system could engage the user in some sort of clarification dialogue. Unfortunately not everything can be always directly asked to the user. Designers of applications based on user modeling must find a compromise between the cost of asking something to the user and its payoff. For example, knowing the educational background of the user could be very useful for assessing the competence level of the user in certain domains, but directly asking him or her what his or her educational background is might be unpolite, distracting and unreliable. This leads to the second fundamental property of user modeling, namely *non-obtrusiveness*, a crucial requirement for UM systems, prerequisite for achieving a high degree of acceptability. Therefore, when it is not appropriate to ask for the needed information, the UM system must again extend its beliefs about the user by relying on some knowledge. During these extensions inconsistencies may arise between conflicting conclusions, requiring belief revision.

Finally, the third fundamental property is the *incrementality* of the UM process. Incrementality enables UM systems to take into account any new piece of user information as soon as it becomes available during the modeling process and to integrate it into the user model. Therefore this makes it possible for the (ideal) UM system (i) to gather successive formulations of those user's aspects which are badly perceived, producing increasingly better approximations (possibly requiring revision), and (ii) to use newly acquired information for validating those beliefs generated by the UM system when trying to extend its set of beliefs.

The solution of the belief revision problem affects the quality of the user modeling system and of the whole application. First, it affects the *acceptability* of the system

as a whole. Acceptability may be improved by an appropriate belief revision schema because it may help in reducing the number of obtrusive and distracting system interventions. Secondly, *robustness* of the system is directly enhanced by a belief revision mechanism which ensures consistency and possibly well-foundedness of beliefs. Thirdly, a general purpose and well designed belief revision mechanism may simplify the development effort of the UM system.

In general the belief revision problem may be decomposed into two subproblems, namely: (a) *consistency maintenance,* i.e. to decide what to do in order to restore consistency of the entire belief set (for example, a course of action could be simply to remove some of the conflicting beliefs); and (b) *multiple-configuration choice,* i.e. to decide which criteria to adopt in order to choose one belief configuration among many, equally acceptable, co-existing configurations which are mutually exclusive (for example, adopting a criterion that minimizes the number of beliefs that change status when moving from the current configuration to the next one).

3.2. NON-MONOTONIC USER MODELING BASED ON ASSUMPTIONS

We propose a UM system which performs belief revision in order to construct and maintain a user model. Our proposal (called *assumption-based user modeling*), that will be outlined at a conceptual level in this section, is based on the following standpoints.

Firstly, the user population may be described by a taxonomy of (possibly overlapping) classes and subclasses of users, obtained according to a set of independent classification criteria. It is assumed that each class shows a number of common properties usually shared by all members of the class and that it is possible to establish whether a user belongs to a given class.

Secondly, for each user interacting with the system it is possible to collect, during a session, individual information about him or her by observing his or her behaviour or by directly asking for some information. This individual information can be integrated into the user model with the information derived from the description of the class(es) of the taxonomy the user belongs to.

Finally, we focus on a non-obtrusive UM system, based on an incremental UM process, and a user model which is individual, dynamic, inferred, and long-term.

We propose to solve the belief revision problem by adopting a reason maintenance system (RMS) (Doyle, 1979; McAllester, 1982; De Kleer, 1986; McDermott, 1991) which is capable of recording the dependencies among beliefs and of tracing back a belief to a set of assumed beliefs supporting it. Thus the information stored in a user model (which is constituted by a set of elementary statements called *assertions*) is classified into three categories according to status:

- *premises,* i.e. known and undoubted facts about the user (e.g. his or her age) which will always be considered true. Premises are usually supplied to the UM system from the outside (e.g. told by user, or otherwise observed by some UM observer);
- *assumptions,* i.e. facts about the user which are believed in the absence of any evidence against them. In the case of the non-appropriateness of some assumptions (in particular where conflicts concerning those assumptions are revealed), one or more of these will be retracted. Assumptions may be

provided either from the outside of the UM system (e.g. something which is told by user) or by the UM system itself (e.g. assuming some default user property);

- *derived facts,* i.e. facts which are the consequence of some set of assertions and that are established by application of some inferential procedure; derived facts are produced by the UM system relying on some internal knowledge.

There are, in general, several mechanisms for generating assumptions: (i) acquisition procedures, in the case of assumptions externally provided to the host system (e.g. told by the user in response to some question posed by the system) and (ii) specific knowledge, in the case of assumptions directly produced within the UM system (e.g. the description of the properties of a taxonomic class) or by the UM observer (e.g. interpretation of the user's behaviour). These mechanisms producing assumptions (usually application dependent, and called *assumption sources*) may be used to provide some evidence on the validity of the assumptions they produce. In this way conflicts among assumptions may be resolved by also taking into account this information.

The second problem of belief revision (i.e. multiple-configuration choice) is solved by considering a number of partial choice criteria (such as preferring more specific assertions over more general ones, preferring more recently derived assertions, taking into consideration the preference associated to assumption sources, etc.) and appropriately combining these in order to obtain a global choice criterion among belief configurations. In particular, each assumption source represents a partial choice criterion among two belief configurations. According to the partial choice criteria based on assumption source S, a belief configuration is preferred if it features a larger number of assumptions generated by S.

Often these partial preference criteria lead to conflicts (i.e. a criterion may prefer a belief configuration A over B, while a second criterion may prefer B over A). As recently pointed out by Doyle and Wellman (1991), integrating many independent choice criteria in order to obtain a global criterion that satisfies a number of intuitive and desirable properties is a problem which, in general, has no solution. A global criterion may be obtained from the partial ones by making appropriate simplifying hypotheses, such as assuming that there is an order relationship among criteria in such a way that conflicts can always be resolved by favouring the criterion coming first. The solution we propose for user modeling follows this idea and orders statically all the partial choice criteria.

3.3. USING AN ATMS FOR NON-MONOTONIC USER MODELING

The RMS adopted for the solution of the belief revision problem is an assumption-based truth maintenance system (ATMS) (de Kleer, 1986). The main advantage of the ATMS over other RMSs is that it makes explicit all possible consistent belief configurations, making in such a way the exploitation of a global choice criterion easier and explicit.

In our approach to non-monotonic user modeling we divide the overall process into two separate activities: the first, called *model management,* concerns (i) classification of the user (i.e. identifying which class(es) the user belongs to and assuming, by default, some set of beliefs depending on his or her membership of a given class) and (ii) management of the user model (i.e. accessing the model, storing

new information, performing domain-dependent inferences on newly acquired information, and recognizing inconsistencies). The second activity, called *consistency management,* is solely concerned with the belief revision problem (i.e. devoted to restoring consistency in the model and to solving the multiple-configuration choice problem).

Consistency management aims at generating and maintaining *all* the possible user models resulting from current premises, assumptions and derivations. More precisely, a *possible user model* is a consistent set of assertions, including all known premises, a maximal set of known assumptions, and all the assertions derived from these premises and assumptions through notified inferences. In other words, using the ATMS terminology, each possible user model is a *context* and, at every moment of system operation, all possible user models share the same set of premises. In addition each possible user model may be associated to a corresponding set of assumptions (i.e. *environment*) that generates it.

The choice of a user model among all the alternative possible user models is performed within the consistency management activity. The *current user model* is a possible user model which is maximal according to a lexicographic partial ordering (called *plausibility ordering* of user models) induced from the ordered set of assumption sources (called *preference ordering* of assumption sources) and their number of occurrences.† Examples of such preferences are: "prefer observations over descriptions of classes of users", "prefer derivations from descriptions of more specific classes of users over more general ones", etc. Informally defined, model M1 is *more plausible* than model M2 if the number of occurrences of the most preferred assumption source in M1 is larger than that in M2. Accordingly, the *most plausible* model features the highest number of occurrences of the most preferable assumption sources.

In the next section we will illustrate how the assumption-based user modeling framework presented above has been implemented in the UMT system.

4. User modeling tool

UMT is a UM shell system used to develop UM systems within larger application systems.

Within UMT a user model is a description of the user in terms of a set of (domain-dependent) multivalued attributes. Each assertion about the user is represented by an attribute/value pair. As already introduced in the discussion of our standpoints, UMT extends its beliefs about the user by relying on a set of descriptions, called *stereotypes,* of the default traits obtained from a taxonomy of potential users. A stereotype is the description of the prototypical user of the class denoted by the stereotype.

UMT provides the following functions to its UM observers and consumers: construction, storage and retrieval of a model for each user of the application; extraction of information from the current user model; integration into the model of

† In general, even if the preference ordering is a total order, the plausibility ordering is a partial one. In fact, two user models may feature the same number of occurrences of the same assumption sources and still differ in the assumptions (and therefore derived facts) they include.

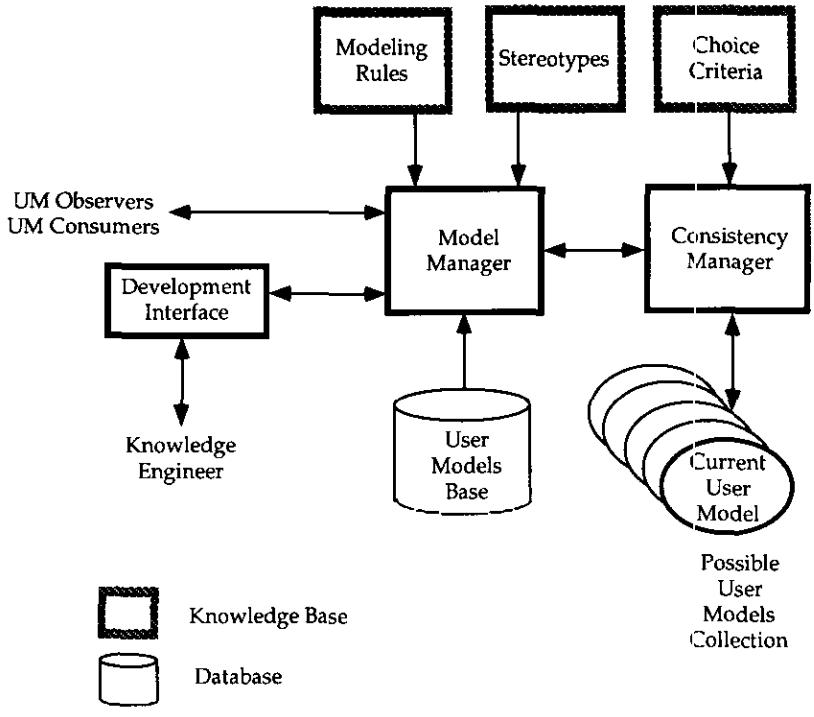

FIGURE 2. Overall architecture of UMT.

new data provided by UM observers and concerning the individual user; extension of the content of the user model by taking advantage of knowledge bases describing the user population; and maintenance of the consistency of the user model.

4.1. ARCHITECTURE OF UMT

The architecture of UMT is illustrated in Figure 2. It derives from viewing the user modeling activity as a combination of model management and consistency management.

UMT is constituted by the *Model Manager* module, carrying on the model management activity, and the *Consistency Manager* module, carrying on the consistency management activity and handling the possible user models. Another module, namely the *Development Interface,* is devoted to supporting the knowledge engineer during the construction of the knowledge bases that are needed for exploiting UMT within an application system.

More specifically, the Model Manager provides the following functions:

- it handles the interaction of UMT with UM hosts;
- it performs inferences according to the knowledge bases that are provided to it (in particular user classification). These inferences produce assertions (premises, assumptions, and derived facts) and dependencies among assertions (justifications, in ATMS terms). Both these results are communicated to the Consistency Manager;

- it recognizes inconsistencies among some set of assertions included in the user model and signals these to the Consistency Manager;
- it forwards to the Consistency Manager assertions (premises or assumptions) or inconsistencies that are produced by UM hosts;
- it manages the retrieval and storage operations of the user model at the beginning or end of the session.

The functions of the Consistency Manager are:

- it responds to requests concerning the content of the current user model;
- it receives from the Model Manager new assumptions, premises or derived facts that are appropriately integrated into all possible user models;
- it receives from the Model Manager notifications of inconsistencies (either recognized by the Model Manager or by UM observers) relating to some set of assertions of the current user model;
- it performs the multiple-belief configurations choice, according to appropriate choice criteria defined in the Choice Criteria Knowledge Base (see below). This choice selects a new current user model among the possible user models.

The operation of UMT is supported by the following three knowledge bases:

- the *Stereotypes Knowledge Base.* Stereotypes are organized in a multiple inheritance network through an IS-A relation defined by the inclusion relation holding between corresponding classes of users. The single root of the hierarchy is the *Generic User Stereotype,* which is a general description of any potential user and is used to define the set of user attributes (their names and the types and cardinality of their values). All other stereotypes are called *Class Stereotypes*, and are descriptions of particular classes of users. Definitions of attributes (i.e. names, types and cardinality) and their default values are inherited downward in the hierarchy. Thanks to the multiple inheritance mechanism adopted in UMT, many independent classification criteria are easily handled.

  Each stereotype is divided into two parts: a *trigger,* representing a sufficient membership condition of a specific user with respect to the class denoted by the stereotype, and the *defaults,* a list of attribute/values pairs representing typical traits of users belonging to the class. Each stereotype, during the operation of UMT, can be either active or non-active. Only active stereotypes may contribute with their defaults to the possible user models. A stereotype is *active* if and only if its trigger is satisfied or if it is the parent (through the IS-A relation) of an active stereotype. Not all defaults of an active stereotype need to be included in the possible user models. In fact, defaults leading to an inconsistent user model are excluded.
- the *Modeling Rules Base,* containing domain-specific knowledge concerning users. More specifically, this rule base contains *constraints* on the admissible values of a set of attributes. When a constraint is not satisfied an inconsistency is raised. The rule base also contains *inference rules* allowing the derivation of attribute values given that certain values of other attributes are present in the current user model.
- the *Choice Criteria Knowledge Base,* containing the definition of the criteria

that are used by the Consistency Manager for choosing the current user model out of the possible user models. In the current version of UMT this knowledge base consists of the definition of the preference relation between assumption sources. It contains a list of the possible assumption sources, which are totally ordered according to their preference.

These knowledge bases support the inferential process enabling the derivation of decision properties (i.e. information used by UM consumers) from observable aspects of the individual user (elicited by UM observers).

The Consistency Manager, during its operation, exploits the *Possible User Models Collection,* i.e. a representation of the set of all possible user models generated during the modeling process. They represent alternative, consistent and mutually exclusive representations of the current user as perceived, up to that moment, by UMT. Each user model of this set has the same structure as the Generic User Stereotype and is constituted by assertions which are:

- individual facts (e.g. the user's age) supplied to UMT by the UM host;
- defaults inherited from the Generic User Stereotype;
- defaults inherited from the active class stereotypes;
- results of inferences performed within UMT by applying modeling rules.

During operation of the Consistency Manager, the *Current User Model* is thoroughly distinguished. The *Current User Model* is the most plausible user model selected from the Possible User Models Collection according to the choice criteria defined in the Choice Criteria Knowledge Base.

Finally, UMT includes also the *User Models Base,* which stores the individual models of all the users that used the application in the past. These models are appropriately retrieved at the beginning of each interaction, and stored back at the end.

4.2. THE MODELING PROCESS IN UMT

The modeling process is accomplished by both modules of UMT and is structured in the following four activities:

PROLOGUE: performed at the beginning of the session and aimed at identifying the user and possibly retrieving his or her stored models from the User Models Base if he or she is already known.

ACCESSING THE MODEL: performed throughout the session and aimed at responding to requests issued by UM consumers and concerning the values of some user attribute in the current user model.

MODIFYING THE MODEL: performed throughout the session and aimed at integrating in the possible user models new data about the user as soon as they are provided by UM observers, or at managing the inconsistencies signalled to UMT.

EPILOGUE: performed at the end of the session and devoted to storing the user models into the User Models Base.

More specifically, a modification requiring the integration of a new datum into the Possible User Models Collection entails the following sequence of operations:

1. if the datum consists of an assumption or premise, then it is asserted by the Consistency Manager in all models of the Possible User Models Collection;
2. if, on the other hand, a contradiction is signalled by some UM observer or consumer to UMT, then the contradiction and its supporting assertions are notified to the Consistency Manager;
3. user classification is performed by the Model Manager, which first identifies active stereotypes by evaluating their triggers or following appropriate IS-A relations, and then instantiates them. Instantiation of a stereotype S consists, for each default A/V (where A/V is a single attribute/value pair) of S, of:
   3.1. assuming that the default A/V of S is believable (i.e. asserting an enabling assumption for the triple $\langle S, A, V\rangle$);
   3.2. considering as the assumption source of the enabling assumption for $\langle S, A, V\rangle$ the name of the stereotype S;
   3.3. representing A/V as a derived fact justified on the basis of (i) the fact that S is active and (ii) the enabling assumption for $\langle S, A, V\rangle$.

UMT ensures that after each modification of the possible user models a number of tasks are performed:

a) all applicable modeling rules are evaluated and possibly fired by the Model Manager (this also includes constraints on the possible user models);
b) the Consistency Manager resolves all known inconsistencies in each user model by updating the Possible User Models Collection;
c) all non-active stereotypes are checked against the Current User Model and are possibly activated and instantiated;
d) all and only consistent user models are maintained in the Possible User Models Collection by the Consistency Manager. Each of these user models include:
   (i) the set P of *all* known premises notified by the UM host;
   (ii) a set A of known assumptions;
   (iii) the set D of all derived facts recursively inferred from the union of A, P and D through all known justifications (yielded by active stereotypes or fired user modeling rules);
e) the Current User Model is always (one of) the most plausible model(s) among the Possible User Models.

Whenever an inconsistency among two or more assertions is discovered, the ATMS underlying the Consistency Manager identifies the (minimal) sets of assumptions (called *nogoods*) upon which the contradictory assertions depend. The Possible User Models Collection is then updated by removing the user models which include some nogood.

It is worth noting what may happen to an instantiated stereotype S: (i) either S is no longer active (i.e. there are no longer reasons for believing S's trigger and no descendants of S through IS-A are active), or (ii) some defaults $\{D_1, \ldots, D_k\}$ of S are in conflict with some part of the current user model. In the former case, the Consistency Manager automatically hides all the assertions which depend on S being active. In the latter case, single enabling assumptions among $\{\langle S, D_1\rangle, \ldots, \langle S, D_k\rangle\}$ might be retracted.

5. A case study in information-providing systems

5.1. INFORMATION-PROVIDING SYSTEMS

We developed a prototype of an information-providing system based on UMT. The major aim of this effort was to test, in a sufficiently rich domain, the design principles of UMT. Before going into some detail of the prototype application, in this subsection we will discuss information-providing systems (briefly introduced in Section 2) in more detail by outlining general characteristics and organizational features.

Information-providing systems (IPS) accomplish the *information-providing task* (IPT), which takes as input a description of a user's information need and provides as output some information that, at least in part, satisfies it.

To this end the IPS relies on a database of the required type of information (e.g. readings, apartments, hotel-rooms, tourist sites) and the IPT is organized around a database search process. From the UM perspective, however, we could also envision a more general view, including knowledge-based systems which exploit a knowledge base to *generate* a set of offers in the IPSs. In these cases the IPS would produce the offers instead of looking them up: however, taken from the UM perspective, the relevant aspects of this task remain unchanged.

From an abstract viewpoint the IPS is required to compute two important functions: it must be able to look-up (or generate) offers, and it must be able to compare each offer to the description of the user's needs in order to determine the appropriateness of the offer with respect to the needs. Let us now concentrate only on the comparison and let us assume that it is implemented through some partial matching algorithm that provides a ranking of (some of) the available offers.

We consider here the IPT carried out interactively by a user and an IPS. During the entire session, the user may provide data that he or she thinks are useful to (better) identify his or her needs, whereas the IPS tries to extend whenever possible its knowledge about the user and his or her needs. However, this scenario is not complete: in fact, the user might be unable to describe his or her needs to the system in a complete and precise way (for a number of reasons including: not knowing how the available offers are identified by the IPS, not knowing the variety of available offers' attributes, the system perceiving a need which differs from the "real" one due to communication difficulties, vocabulary mismatch, etc.). Thus it is quite common for the IPS to implement some kind of feedback, by acquiring user judgements on the proposed offers and taking into consideration this information for refining the description of user's needs. Usually, the system is capable of producing an appropriate answer only after several proposal/corrections have occurred.

5.2. TOURIST ADVISOR

In this section we illustrate a simple IPS developed with UMT, called *Tourist Advisor* (TA). The purpose of TA is to provide an hypothetical tourist in Rome with advice on what is worth visiting.

In particular, TA acquires some information about and from the user, manages it through UMT, provides suggestions together with some of the reasons supporting

them, and accepts criticism and observations from the user in order to tune its operation.

TA uses a database containing a multi-attribute description of tourist sites (such as the Colosseum, St. Peter's Basilica, etc.). The following is an example of part of the description of the Colosseum, which includes the following attributes: `type` (it is a monument), `documentation` (a brief textual description of the site used by TA when presenting this offer to its user), `architecture` (specifying that this offer is an example of classic and ancient architecture), `art` (similarly), `characteristics` (specifying that the site is impressive and majestic).

```
(OFFER Colosseum
  :TYPE '(monument)
  :Site 'Colosseum
  :Documentation
  "It is the most famous monument of Imperial Rome.
  Its real name is Anfiteatro Flavio. . . ."
  :Architecture '(classic ancient)
  :Art '(classic ancient)
  :Characteristics '(impressive majestic)
  . . .)
```

In general the operation of TA is organized in the following way. If the user is not yet known to TA then a preliminary interview takes place in order for TA to acquire some data directly from the user. Once a user model is made available by UMT, TA performs a best match search, based on the number of matching values between the user attributes and those of available tourist sites. The selected tourist site is shown to the user who can judge upon the relevance of the suggestion. Data provided by the user are transmitted to UMT where they are appropriately included in the user model. In particular, if the user agrees on the suggestion, these data are used to "reinforce" those assertions in the current user model that have supported that suggestion. On the other hand, if the user does not like the suggestion, inconsistencies that require a modification to the user model are signalled to UMT. In this way the current user model is updated to reflect user judgements.

Figure 3 illustrates the general architecture of TA. It includes UMT and the TA

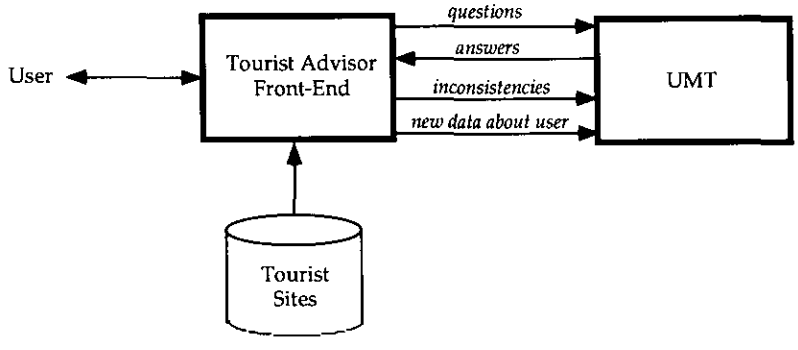

FIGURE 3. General architecture of the Tourist Advisor application.

Front-End, a module devoted to performing both the roles of UM observer and consumer. These roles include the capability of interacting with the user (through a window-based interface) and of accessing the Tourist Sites database.

5.3. KNOWLEDGE REPRESENTATION IN TOURIST ADVISOR

Let us briefly overview the knowledge bases used by UMT within the TA application.

The generic user stereotype used in TA defines a set of general user attributes and their possible values (such as: AGE which takes one value from the interval [1-100], NATIONALITY taking one value from the set {Italian, European, Japanese, Asiatic, . . .}, GENERAL-INTERESTS taking values from the set {SPORT, POLITICS, HISTORY, . . .}, etc.). The Generic User Stereotype of TA is:

```
(GUS
  (Name= :TYPE string)
  (Age= :TYPE (integer 1 100) :CARD 1 :DEFAULT 40)
  (Nationality= :TYPE
   (:ONE-OF (Italian European Asiatic African American
  . . .) :DEFAULT Italian)
  (General-Interests= :TYPE
   (:ONE-OF (Sport Politics History Society Literature
  Fiction Architecture Science Music. . . )))
  . . .
  . . .
  )
```

All other stereotypes have been obtained from the classification of the user population according to the following classification criteria and classes (within parentheses): age (YOUNG-MAN, ADULT, OLD-MAN), nationality (ITALIAN, . . .), educational degree (ELEMENTARY, MEDIUM, UNIVERSITY, HUMANITIES, SCIENTIFIC), artistic preferences (CLASSIC, MODERN, . . .), etc.

An example of class stereotype is the stereotype YOUNG-MAN, a specialization of the Generic User Stereotype, saying that usually young people love music and sport, have a relatively low budget, they like to visit open places (such as buildings and parks) but not museums, churches or cathedrals; they also like to go to music concerts and to exhibitions. This stereotype triggers when the user's age is below 35.

```
(STEREOTYPE Young-man
  (IS-A Generic-User-Stereotype)
  (:TRIGGER
   (age ?x) (:LISP (<?x 35)))
  (General-Interests= :DEFAULT (Sport Music))
  (Budget= :DEFAULT (Low))
  (Preferred-Masterpieces= :DEFAULT (Open-Places Squares Parks))
  (Non-Preferred-Masterpieces= :DEFAULT
   (Museums Churches Cathedrals))
  (Preferred-Events= :DEFAULT (Exhibitions Concerts))
```

```
(Preferred-Sports= :DEFAULT (Soccer)))
```

We also defined modeling rules for the TA application. One example is the following rule which expresses that when a user is known to be lazy, and to have a good financial budget, then it will also be the case that one of his preferred means of transport will be the taxi-cab. This is specified by the following construct:

```
(RULE (Personal-Defects= lazy)
      (Financial-Budget= ?x)
      (:LISP (member ?x '(medium high))))
      →
      Preferred-Means-of-Transport= taxi-cab))
```

User Modeling Rules may also represent constraints on the user model. For example, in the case of TA, a constraint is given by the fact that the same thing cannot be, simultaneously, liked and not liked by the same user, and this is expressed by the following constraint:

```
(INCONSISTENCY
  (Preferred-Masterpieces= ?x)
  (Non-Preferred-Masterpieces= ?x))
```

As already mentioned, assumption sources correspond to the different ways in which assumptions may be generated. In particular, each stereotype may be used for generating assumptions. Another assumption source within TA is constituted by the user when providing new information or when giving judgements about a previously proposed suggestion. The assumption sources and the preference ordering are defined with:

```
(SOURCES
  (PRELIMINARY-INTERVIEW)
  (FEEDBACK-FROM-USER)
  (DEFAULT-FROM-STEREOTYPE ITALIAN)
  (DEFAULT-FROM-STEREOTYPE STRANGER)
  (DEFAULT-FROM-STEREOTYPE YOUNG-MAN)
  . . .)
```

which means that the preferred derivation source is `PRELIMINARY-INTERVIEW`, the second preferred one is `FEEDBACK-FROM-USER`, and so on: we consider information provided by the user to be more reliable than that acquired from feedback, which is more reliable than that derived from any stereotype, and so on.

5.4. AN EXAMPLE SESSION

We illustrate now a brief example of the operation of UMT in the TA application. What follows is an excerpt from a real session held with TA.

We consider a situation in which the user is not yet known to TA and the preliminary interview is taking place. The user is looking for sites to visit in Rome and is willing to provide some data about himself and a brief description of his interests. The interaction is taking place with the user interface of TA, but we will

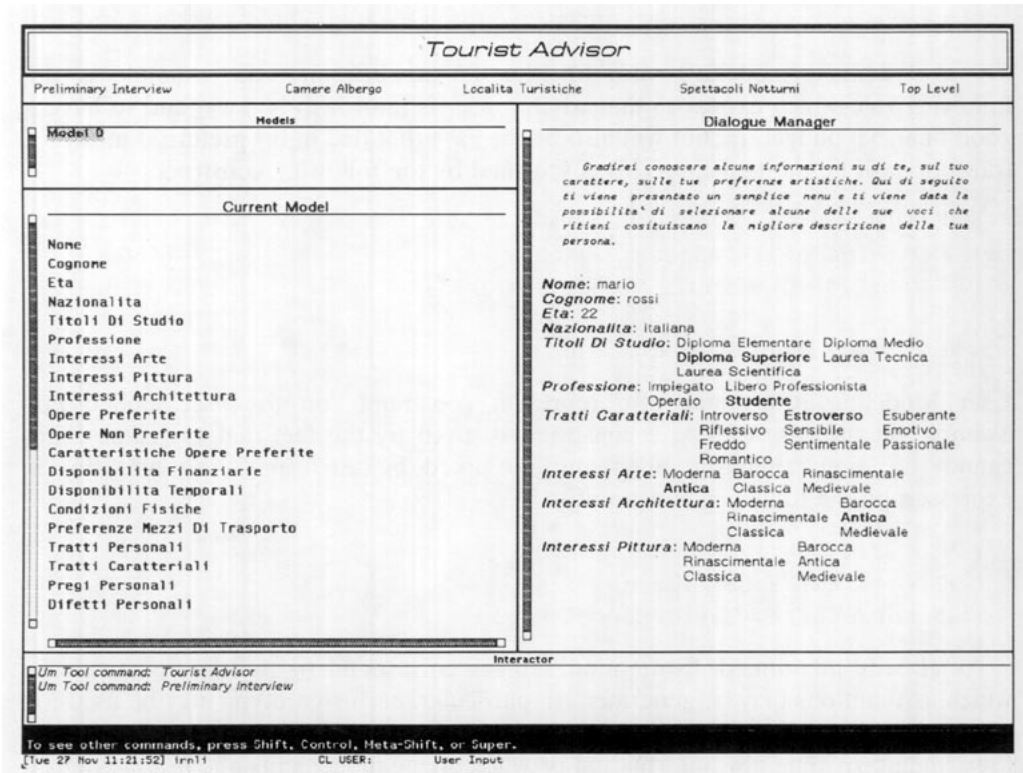

FIGURE 4. Preliminary interview.

also show the development interface of UMT in order to clarify some internal aspects of its operation. Figure 4 shows the user interface of TA: a screen divided into several panes. Pane CURRENT MODEL (left hand-side) displays all the empty slots since the user is not yet known to TA, whereas pane DIALOGUE MANAGER handles a form that may be appropriately filled in by the user (not all attributes must necessarily be provided).

Let us assume that the user provides very few data about himself: name (Mario Rossi), age (22), nationality (Italian), educational level (high-school degree), and job (student). TA treats these facts as premises, i.e. unretractable assertions. Other facts communicated by the user to TA, such as `personal-traits=extroverted`, `art-interests=ancient`, `architecture-interests=ancient` and `painting-interests=ancient`, are considered by TA as assumptions, since they could later be changed by the user as the session proceeds. These assumptions are labeled with the `preliminary interview` assumption source. UMT adds this information to the only possible user model being created so far (see pane MODELS in Figure 4, containing only one model, name MODEL-0). On the basis of these premises and assumptions UMT classifies the user. The classification process, carried out by the Model Manager (MM), considers all user modeling rules and stereotype triggers that can be instantiated given the above-mentioned data. The Consistency Manager (CM) takes each instantiated rule or trigger and generates appropriate justifications among ATMS nodes representing assertions. In the example, the trigger of

stereotype young-man is satisfied by the initial description of the user. As a consequence UMT asserts that the stereotype is active by creating a (derived) fact active(young-man) and a justification for it: justifies(age=22, active-(young-man)). Subsequently, for each of the stereotype defaults, appropriate enabling assumptions and justifications are asserted. This works as follows: for the default assertion general-interest=music the MM assumes that it is believable (i.e. the CM creates an assumption node representing the assertion believable (young-man, general-interest, music)). The assumption source for this assumption is set to (DEFAULT-FROM-STEREOTYPE YOUNG-MAN). Secondly, UMT generates the following justification for the derived fact general-interest=music:

```
justifies(active(young-man) AND
  believable(young-man, general-interest, music),
  general-interest=music).
```

At this point general-interest=music will be believed only in user models where the enabling assumption believable(young-man, general-interest, music) holds. Since the premise age=22 cannot be retracted, facts directly derived from it, such as active(young-man), also cannot be retracted and the stereotype cannot be later deactivated. The mechanism used by UMT to retract single default assertions of stereotypes is retracting the corresponding enabling assumption. Sometimes, however, the trigger of a stereotype S may depend on retractable assertions (i.e. assumptions or derived facts): in these cases the fact active(S) may also be retracted, as well as all the defaults of S.

Focusing again on the example, UMT processes the other stereotypes of the TA application against the set of all known assertions until quiescence occurs: no UM rules or triggers fire anymore.

During this process inconsistencies are possibly discovered and resolved by generating alternative user models. For example, the default budget=low from the stereotype student cannot be consistently believed together with preferred-activity=boutique-shopping derived from stereotype frivolous. This inconsistency is recognized by an appropriate UM rule saying that budget=low and preferred-activity=boutique-shopping are contradictory. The CM identifies the sets of assumptions supporting both assertions (the nogoods) and labels as inconsistent any other set of assumptions including them. In this example the nogood consists of believable (student, budget, low) and believable (frivolous, preferred-activity, boutique-shopping). UMT then generates alternative descriptions of the user which are self-consistent, but mutually incompatible. More specifically it generates a model containing believable(student, budget, low), but not believable(frivolous, preferred-activity, boutique-shopping) and another model containing the latter assumption, but not the former one. The starting model is discarded.

In the example other inconsistencies are found, eventually producing 6 different possible user models, all descendant from the initial MODEL-0.

User modeling rules are dealt with by a simpler machinery. For example the rule

```
(RULE (Physical-conditions=good)
```

```
(Financial-Budget=low)
→
(Preferred-Means-of-Transport=walking))
```

fires during the classification process (i.e. its condition part is satisfied by a pair of assertions both holding in one of the possible user models). UMT creates a justification linking the rule conclusion to its conditions (i.e. `justifies(physical-conditions=good & financial-budget=low, preferred-means-of-transport=walking)`).

By means of this classification process UMT tries to cope unobtrusively with the incompleteness of user modeling: a very brief initial user description is further and further enriched by guessing other aspects describing the user. Sometimes these guesses are to be retracted, causing the generation of alternative user descriptions.

Figure 5 illustrates the development interface of UMT, which has been invoked manually, for the purpose of this example. The `CURRENT MODEL` pane (left-hand side) shows the current user model, while the `STEREOTYPES` pane lists all TA stereotypes, including the currently active ones which are shadowed. The user is believed to be a young-man, Italian, with a medium-level education-degree, modern, and frivolous. In fact, most of the information present in the user model is derived from active stereotypes (e.g. `Architecture-interests=modern` derives

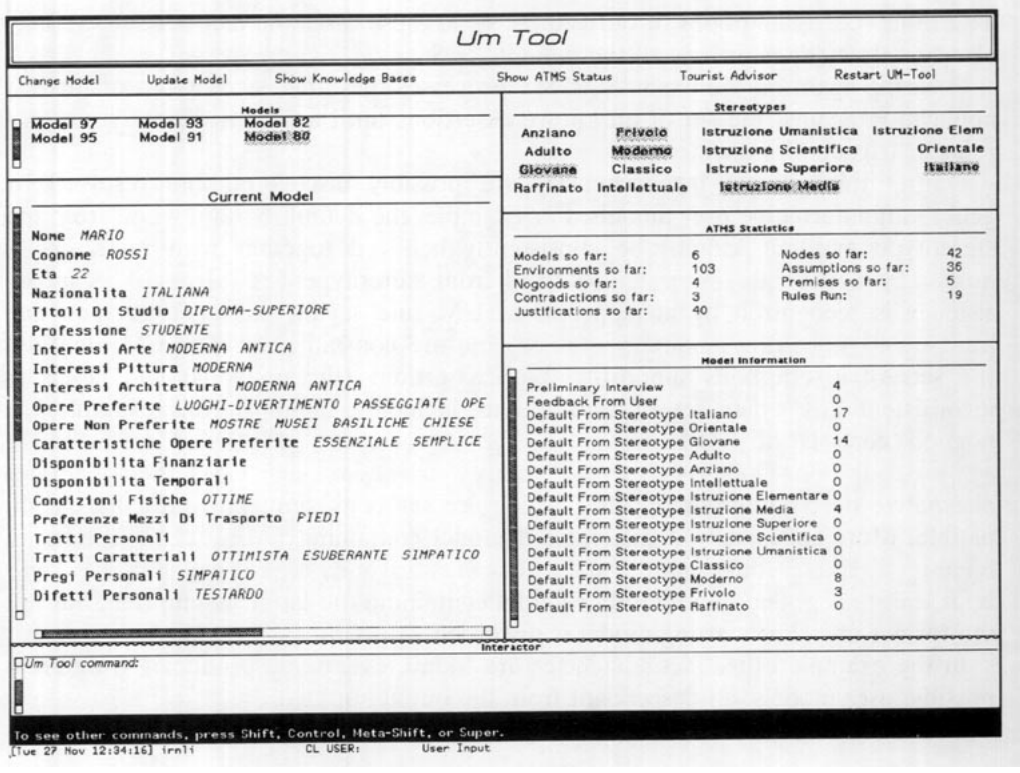

FIGURE 5. The most plausible user model.

from stereotype `Modern, Preferred-masterpieces=walking-around` from stereotype `Italian, Preferred-masterpieces=entertainment-sites` from stereotype `Frivolous`, etc.).

Pane `MODEL INFORMATION` (bottom right-hand side of the screen) shows the ordered list of assumption sources and their number of occurrences (with respect to the Current User Model). One can see that four assumptions are labeled with the `Preliminary Interview` assumption source, 17 assumptions with (`DEFAULT-FROM-STEREOTYPE ITALIAN`), etc. Finally, the upper left-hand side pane (`Models`) shows the Possible User Models and, shadowed, the Current User Model (namely `Model-80`). It is possible for the knowledge engineer to manually select another model as the new Current User Model. The effect of choosing `Model-93` as the Current User Model is shown in Figure 6. Note that the plausibility of this model is lower than the previous one, because in `Model-93` there are 13 occurrences of the fifth assumption source (`DEFAULT-FROM-STEREOTYPE YOUNG-MAN`) instead of 14 as in `Model-80`. Note also that the two models differ in their content: for example, in one model it is stated that the user prefers "simplicity in preferred masterpieces" (see `Caratteristiche-opere-preferite=semplice` in Figure 5), while in the other one the user prefers "elegance".

Resuming normal TA operation now, and resetting the Current User Model to `Model-80`, TA provides a first tentative answer to the user's need.

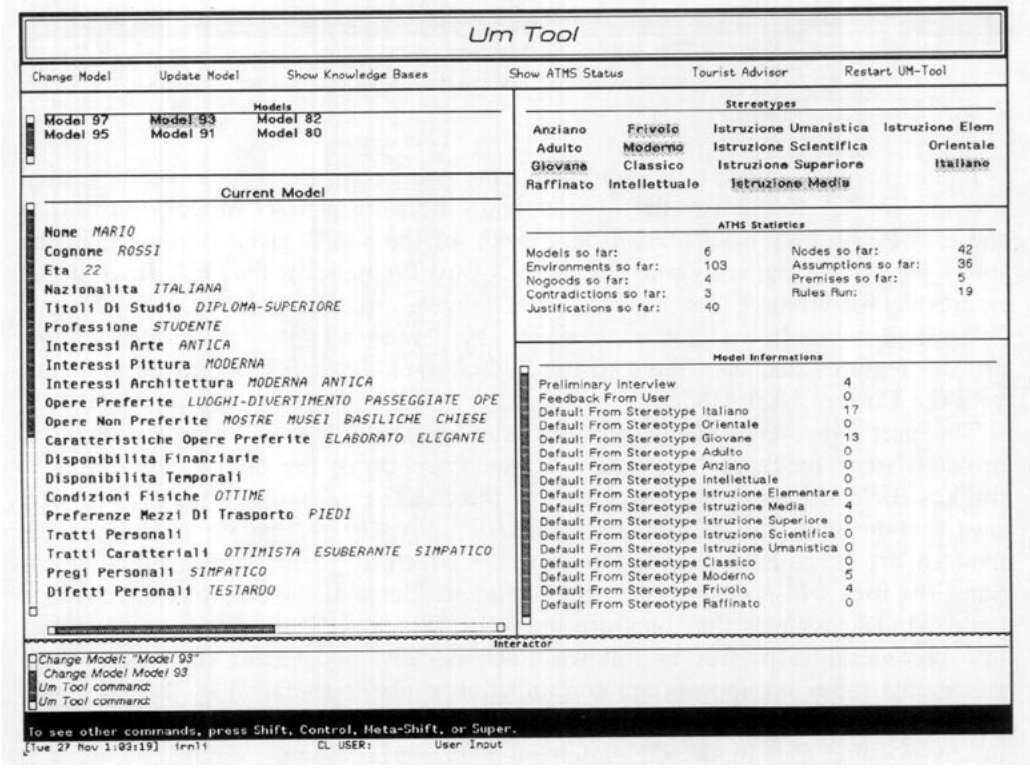

FIGURE 6. An alternative user model.

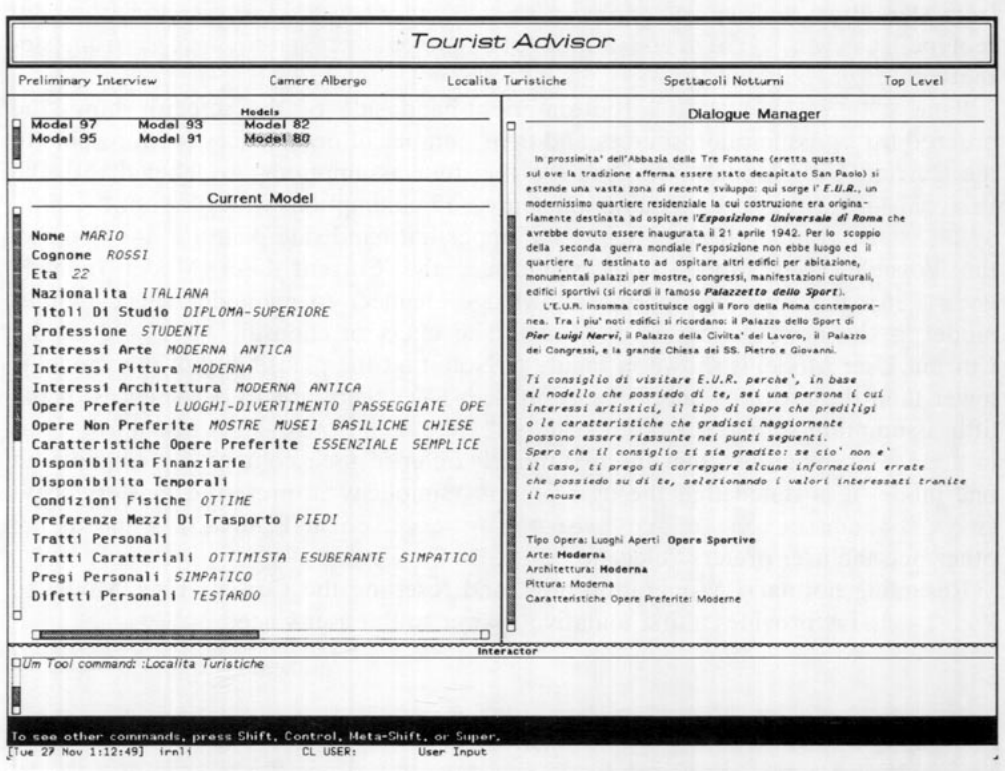

FIGURE 7. A suggestion provided by TA and the corresponding user feedback.

Figure 7 shows, in the right-hand side pane (`Dialogue Manager`), the suggestion provided by TA to the user that best matches the Current User Model (in this case the E.U.R. site). A brief textual description of the site is given, followed, in the lower part of the pane, by a list of reasons why TA believes that E.U.R. could be interesting for the user (e.g. `Masterpiece-type=(Sporting-site Open-place)`, `Art=(Modern)`, `Architecture=(Modern)`, `Painting=(Modern)`, etc.). These are the features that contribute to the highest ranking of E.U.R. in the matching with the Current User Model.

The user now reads the suggestion, and judges it not to be interesting. He provides some feedback by telling TA which aspects of the offer's description he dislikes (bold values in the lower part of the `Dialogue Manager` pane). The user says that he does not like sporting-sites (`Masterpiece-type=Sporting-site`), modern art (`Art=Modern`), and modern architecture (`Architecture=Modern`). Since the user said nothing on the remaining attributes of the site (e.g. `Painting=Modern`), TA assumes that they are implicitly accepted. This information is taken into consideration by TA as follows. Each negative judgement (i.e. user dislike statement) raises an appropriate contradiction which is signalled by TA to UMT. For example, when the user says that he does not like sporting-sites, TA asserts that in any possible user model the statement `Masterpiece-type=Sporting-site` is not valid, i.e. it leads to a contradiction. The CM traces this fact back to a nogood

constituted by `believable(Modern, Masterpiece-type, Sporting-site)` and treats this as a normal inconsistency, with the result that the assumption is dropped from the current model. On the other hand, positive implicit feedback (for example with `Painting=Modern`) does reinforce the current model. This is done by changing the assumption source of the enabling assumption for `Painting=Modern` from (`DEFAULT-FROM-STEREOTYPE MODERN`) to `Feedback-from-user`. Since the latter assumption source weighs more than others associated with a stereotype, this mechanism pushes all the models containing the enabling assumption for `Painting=Modern` higher in the plausibility ordering. On the basis of the revised possible user models a further classification takes place, followed by a reordering of the models.

The result of this activity in response to the user judgements is shown in Figure 8 (the UMT development interface): the current model has changed (it is now `Model-132`) as well as its content. Notice, for example, that now the user is believed not to be interested in modern art any more (according to his statement), he is (believed to be) interested also in classic architecture, and so on. In this particular example the set of active stereotypes did not change.

As we can see, with a minimum user effort, TA enriches incrementally its description of the user until (hopefully) it becomes sufficiently rich and accurate to enable a selection of the offers that satisfy his need.

Finally, a second suggestion is given to the user on the basis of the revised user

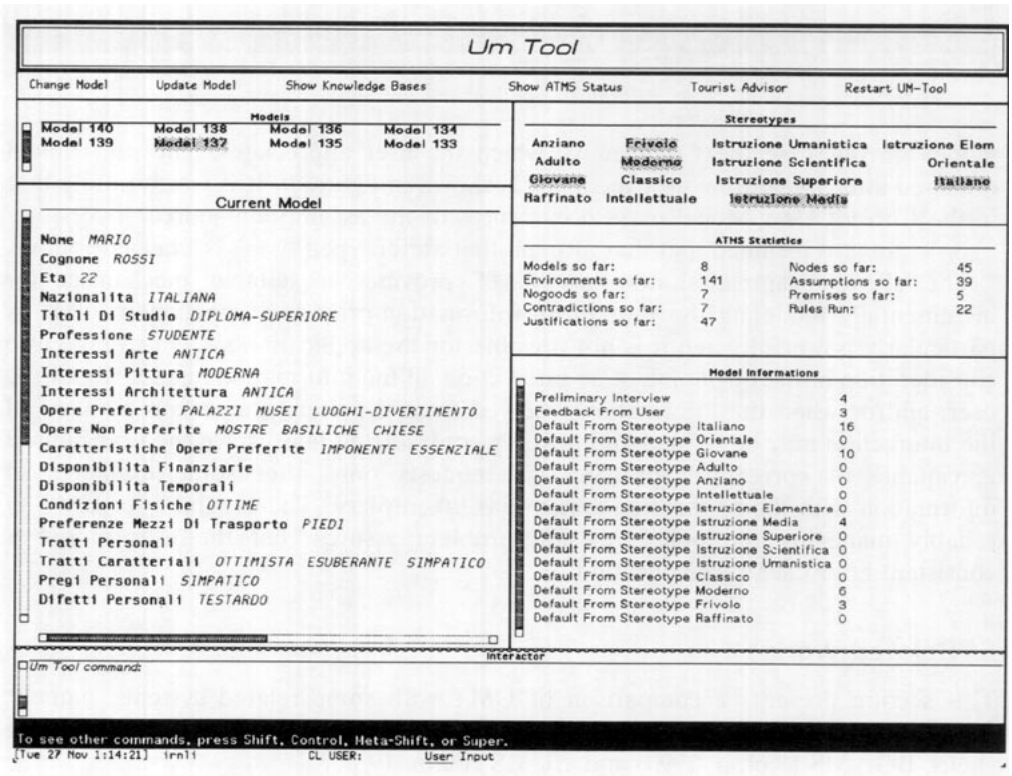

FIGURE 8. A revised user model.

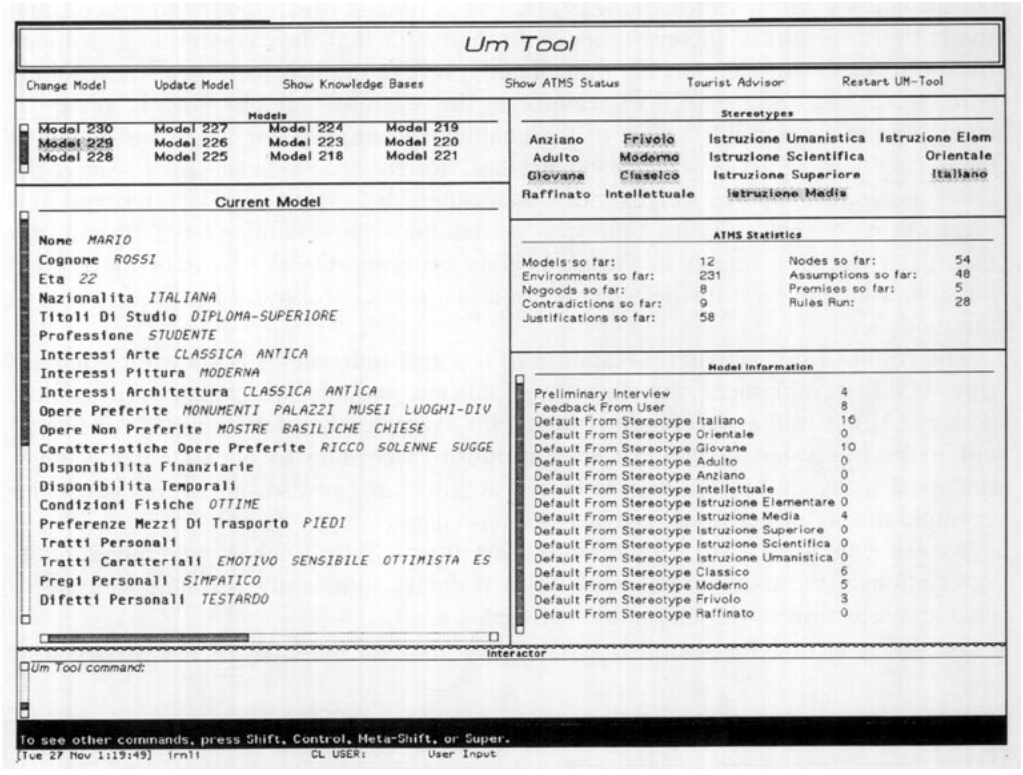

FIGURE 9. The final user model.

model (in this case the Colosseum), which the user appreciates. The user model obtained after integrating user feedback is shown in Figure 9. It is worth noting that in the Current User Model more occurrences of the assumption source `feedback-from-user` are included and that also another stereotype (`Classic`) became active.

The above example shows how UMT provides a suitable mechanism for incrementally modeling the user of an information-providing application. UMT is particularly powerful when it is not possible for the application system to take into consideration at the beginning of an interaction all the information needed about the user, and/or when the information received from the user in a certain moment of the interaction may be refined and possibly contradicted later, since the user himself is capable of correctly expressing his requests only after some exchange of information with the application system has taken place. The capability of UMT of reliably managing the belief revision problem assures that the user model is consistent after each reclassification.

6. Related work

This section presents a comparison of UMT with some related systems, namely: SMMS, a student modeling system (Haung *et al.*, 1991), and two user modeling shells, BGP-MS (Kobsa, 1990) and GUMS (Finin, 1989).

SMMS (Student Model Maintenance System) is a student modeling system (in

the domain of Lisp programming) which exploits stereotypes in a modeling process also based on belief revision. In particular two kinds of beliefs are handled by the system: (i) *deductive beliefs,* i.e. those representing observations, reasoning rules (such as "if the student knows the Lisp function "mapcar" then he knows also the function "car"), and derived beliefs; and (ii) *stereotypical beliefs,* i.e. those derived from stereotypes. Stereotypes mirror a hierarchy of domain concepts (e.g. Lisp general programming concepts, built-in Lisp functions, etc.) that might be known to the student. For each domain concept three stereotypes are defined, according to the student level of expertise in that concept: novice, average or expert. For each stereotype, a logical constraint is used as a trigger and a set of defaults provide indications of which other stereotypes might also be relevant. Two distinct belief revision processes are present. The first one, called *evolutionary belief revision,* is carried out only among deductive beliefs in such a way to minimize the number of changes that are required to restore consistency. In order to accomplish this, the belief revision problem is reduced to the (diagnostic) problem of finding the minimal set of assumptions which, when removed, also removes the inconsistency. This belief revision process is similar to the one proposed for diagnosis in Reiter (1987) and De Kleer and Williams (1987), and it exploits an ATMS. The second belief revision process, called *revolutionary,* concerns stereotypical beliefs only and aims at determining the set of stereotypes whose constraints are satisfied, and installing their defaults into the user model. It is performed by appropriately following the relationships among stereotypes defined by constraints and defaults. The student model consists of the currently-held set of deductive beliefs and the set of stereotypical beliefs derived from active stereotypes that are consistent with current deductive beliefs.

UMT and SMMS are similar:

- both use an ATMS as a basis for belief revision; SMMS uses it for evolutionary belief revision only, whereas UMT uses it as a general mechanism;
- in both systems the concept of a current set of beliefs is exploited and both systems explicitly choose the current set of beliefs among several candidate ones. In particular, while in UMT this choice is guided by the plausibility relation, in SMMS this choice seems to be based on priorities associated to assumptions;
- the (user or student) model may include in both cases individual data, default data derived from active stereotypes, and otherwise inferred data.

There are, however, differences worth mentioning. First of all, the student model in SMMS includes two kinds of beliefs which are kept separated: stereotypical and deductive ones. The same distinction holds also for the revision processes. In UMT, on the other hand, both revision processes are dealt with by the same mechanism, guaranteeing in such a way a higher level of uniformity, which makes easier the effort of the knowledge engineer. Moreover, although both approaches take advantage of stereotypes—consisting essentially of an activating condition and a set of default beliefs—and both approaches allow the activation of more than one stereotype at a time, SMMS, in contrast to UMT, does not permit overlapping stereotypes, in the sense that two distinct stereotypes may not include (conflicting) information about the same user attribute. In UMT, the exploitation of an

ATMS-based mechanism for the stereotypical revision process allows such limitations to be overcome: the case where a user is assigned to two classes corresponding to two stereotypes which have some conflicting values is handled in a straightforward way.

BGP-MS (Belief, Goal, Plan Management System) is a UM shell aimed at providing a set of integrated tools to the designers of application programs that need user modeling services. One application of BGP-MS is the XTRA program, a natural language interface to expert systems.

In particular, BGP-MS features a powerful representation language of conceptual knowledge of the user which is based on a hybrid knowledge representation language with *partitions* dedicated to represent nested beliefs (i.e. the system's beliefs, the system's beliefs of the user's beliefs, etc.). BGP-MS offers a functional interface for updating the model of the user as well as some basic domain-independent inference mechanisms.

As is the case for UMT, BGP-MS can be classified as a stereotype-based UM shell which provides a very flexible and customizable stereotype management utility. These facilities allow the definition of stereotypes and relationships among sets of stereotypes, such as: strict inclusion, where the contents of the stereotypes are ordered in a linear subset relationship (e.g. in the set {novice, average, expert}), arbitrary hierarchy graph, in which each stereotype may have more than one superordinate stereotype from which it inherits information, etc. Each stereotype provides a set of default beliefs represented in the underlying representation language. Several powerful inheritance mechanisms may be specified, as well as various activation and retraction conditions for each stereotype, in order to determine what stereotypes best fit the current user. Furthermore, the behaviour of the UM shell can be finely tuned by acting on a set of parameters controlling, for example, the maximum number of currently active stereotypes.

The user model consists of observations of the user's behaviour, of inferred data, and of stereotypical beliefs derived from active stereotypes. Through appropriate rules (which may, or may not, depend on the application domain), BGP-MS infers new beliefs from the concepts mentioned by the user. For example, if the user believes that concept C specializes concept B and that B specializes A, then GBP-MS infers that the user believes also that C specializes A. Periodically, after a certain (programmable) number of new beliefs have entered the user model, a reclassification process takes place, determining the stereotypes that should be activated and those that should be deactivated.

The main limitation of BGP-MS, in comparison with UMT, concerns the problem of resolving possible conflicts among predictions of different stereotypes or predictions that contradict information directly acquired from the user. The solution to this problem, suggested also by the author, is some belief revision procedure possibly based on an assumption-based truth maintenance system that guarantees the consistency of the model. All this has not yet been included in BGP-MS.

GUMS is aimed at providing a set of user modeling services to an application program that interacts directly with a user. In particular, GUMS accepts and stores new facts about the user, verifies if the new facts are consistent with previously acquired information, informs the application of any recognized inconsistency and answers queries posed by the application.

GUMS shares many characteristics with UMT: the overall architecture of the application, the exploitation of stereotypes and of default reasoning, and the same nature of the content of the user model. Stereotypes in GUMS are domain dependent and are organized in a strict hierarchy (i.e. each stereotype may have at most one parent stereotype). A stereotype is constituted by a set of *definite facts* and *definite rules* (which determine what is necessarily true of any individual classified under that stereotype) and a set of *default facts* and *default rules* (providing default information describing the user).

The user model in GUMS includes individual facts obtained through observations and an instance of exactly one stereotype. Whenever these facts conflict with the definite knowledge of the active stereotype, a reclassification takes place. Reclassification is performed in such a way as to move upward in the stereotype hierarchy to the least general stereotype that restores consistency in the user model; such a stereotype becomes the newly activated one.

Several differences exist between GUMS and UMT. First of all, GUMS assigns a user to only one class (only one stereotype can be active), and the way the revision process is organized (moving only towards a higher level stereotype) reduces the amount of stereotypical knowledge about a user applicable during the session. It seems more appropriate to assume (as we have done in UMT) that the new information about the user gathered during a session should allow a better classification of the user, taking into account a richer and more specific variety of information about him or her. Moreover, GUMS uses a failure-as-negation mechanism employed to maintain the model in a consistent state. This mechanism suffers from severe limitations because, as is also pointed out by the author, checking the consistency of a new fact with the currently-held assumptions is not at all sufficient to avoid subtler inconsistencies. Furthermore, no means are provided in GUMS for choosing an alternative description among those arising from the resolution of the inconsistency. Finally, from a lower level viewpoint, no caching mechanism is present to record inferences already made: GUMS must recompute them from scratch whenever they are applicable. As we have demonstrated in this paper, all these limitations are resolved at once in the assumption-based user modeling framework exploited in UMT.

7. Conclusions and future research

In this paper we have illustrated a general framework for user modeling and the organization of UMT, a general and flexible tool for building user modeling subsystems, which is centred on an assumption-based approach to user modeling. The proposed framework separates very clearly the user modeling component within a generic application system. The UMT shell can be directly utilized for the implementation of such a user modeling component, providing the following main advantages:

- the integration of stereotype-based user modeling with a belief revision system;
- the capability of reliably managing the consistency of the user model within a dynamic re-classification approach;

- the possibility of choosing the most plausible model among all the alternative ones according to a general, explicit, and customizable criterion;
- a very concise, simple and abstract interface of the module devoted to consistency maintenance.

These features also result in the simplicity with which a knowledge engineer can design and construct the user modeling knowledge bases (stereotypes and rules) through an appropriate development interface. In particular, the knowledge engineer is not asked to deal with mechanisms for handling inconsistencies, since UMT requires only the provision of knowledge about situations where such inconsistencies arise.

The main limitations of UMT are: (i) the combinatorial explosion of the number of alternative models, and (ii) the monotonic growth of the data structures internally handled by the Consistency Manager representing all believed assertions and all justifications among them. A specific research effort is planned in order to try to resolve both problems by adopting a heuristic criterion for pruning those models that are not likely to become very plausible.

Other future research openings will concern:

- The definition of a declarative language to specify the inheritance network of stereotypes and richer choice criteria.
- The development and experimentation of a user modeling subsystem based on UMT included in the FIRE prototype, a flexible environment for the development of intelligent information retrieval systems (Brajnik, Mastrodonato, Scaroni & Tasso, 1991). After a laboratory validation of such a user modeling subsystem, an experimentation of FIRE with real users will be carried out in order to evaluate the benefits offered by the inclusion of a user modeling component within an expert interface for on-line bibliographic databases.

UMT has been implemented in Common Lisp and Common Lisp Interface Manager. It was initially developed on a Symbolics Mac-Ivory, and later ported to other Common Lisp systems.

We are indebted to Antonio Vaccher for the many hours he spent with us in designing and implementing UMT and TA. Many thanks also to Alfred Kobsa for spontaneous comments on an early draft of this paper.

References

BELKIN, N. J., BORGMAN, C. L., BROOKS, H. M., BYLANDER, T., CROFT, W. B., DANIELS, P., DEERWESTER, S., FOX, E. A., INGWERSEN, P., RADA, R., SPARCK JONES, K., THOMPSON, R. & WALKER, D. (1987). Distributed expert based information systems: an interdisciplinary approach. *Information Processing & Management*, **23**(5).

BELKIN, N. J. (1988). On the nature and function of explanation in intelligent information retrieval. *Proceedings of the 11th ACM SIGIR International Conference on Research and Development in Information Retrieval*, pp. 135–145, Grenoble, France.

BRAJNIK, G., GUIDA, G. & TASSO, C. (1987*a*). Design and experimentation of IR-NLI: an intelligent user interface to bibliographic data bases. In L. KERSCHBERG, Ed. *Expert Data Base Systems*, pp. 151–162. Menlo Park, CA: Benjamin/Cummings.

BRAJNIK, G., GUIDA, G. & TASSO, C. (1987*b*). User modeling in intelligent information retrieval. *Information Processing & Management,* **23**(4), 305–320.

BRAJNIK, G., GUIDA, G. & TASSO, C. (1990). User modeling in expert man–machine interfaces: a case study in intelligent information retrieval. *IEEE Transactions on Systems, Man, and Cybernetics,* **20**(1), 166–185.

BRAJNIK, G., MASTRODONATO, L., SCARONI, C. & TASSO, C. (1991). FIRE: Un prototipo di ambiente di sviluppo per interfacce intelligenti e cooperative per l'accesso a banche di dati bibliografici. In B. FADINI, Ed. *Sistemi Informatici e Calcolo Parallelo,* pp. 258–266. Rome: Franco Angeli.

BRAJNIK, G., TASSO, C. & VACCHER, A. (1991). A flexible tool for assumption-based user modeling. *Proceedings of the 2nd Congress of the Italian Association for Artificial Intelligence,* pp. 445–449, Palermo, Italy.

CAHOUR, B. & PARIS, C. (1991). Role and use of user models. *Proceedings 12th IJCAI Workshop Agent Modelling for Intelligent Interaction,* pp. 73–79, Sydney, Australia.

CARBERRY, S. (1988). Modeling the user's plans and goals. *Computational Linguistics,* **14**(3), 23–37.

CHIN, D. (1989). KNOME: modeling what the user knows in UC. In A. KOBSA & W. WAHLSTER, Eds. *User Models in Dialog Systems,* pp. 74–107. Berlin: Springer Verlag.

DANIELS, P. J. (1986). The user modelling function of an intelligent interface for document retrieval systems. *Proceedings IRFIS 6: Intelligent Information Systems for the Information Society,* pp. 162–176. Amsterdam: North-Holland.

DEKLEER, J. (1986). An assumption-based truth maintenance system. *Artificial Intelligence,* **28,** 127–162.

DEKLEER, J. & WILLIAMS, B. (1987). Diagnosing multiple faults. *Artificial Intelligence,* **32,** 97–130.

DOYLE, J. (1979). A truth maintenance system. *Artificial Intelligence,* **12,** 231–272.

DOYLE, J. & WELLMAN, M. P. (1991). Impediments to universal preference-based default theories. *Artificial Intelligence,* **49,** 97–128.

FININ, T. W. (1989). GUMS—a general user modeling shell. In A. KOBSA & W. WAHLSTER, Eds. *User Models in Dialog Systems,* pp. 411–430. Berlin: Springer Verlag.

FUM, D., GIANGRANDI, P. & TASSO, C. (1990). Backward model tracing: an explanation-based approach for reconstructing student reasoning. *Proceedings of the AAAI-90 Eighth National Conference on Artificial Intelligence,* pp. 426–433, Boston, Massachusetts, USA.

HANKS, S. & MCDERMOTT, D. (1987). Nonmonotonic logic and temporal projection. *Artificial Intelligence,* **33,** 379–412.

HUANG, X., MCCALLA, G. I., GREER, J. E. & NEUFELD, E. (1991). Revising deductive knowledge and stereotypical knowledge in a student model. *User Modeling and User-adapted Interaction,* **1**(1), 87–115.

KASS, R. & FININ, T. (1988). Modeling the user in natural language systems. *Computational Linguistics,* **14**(3), 5–22.

KAY, J. (1990). UM: a user modelling toolkit. *Proceedings 2nd International User Modelling Workshop,* Honolulu, Hawaii, USA.

KAY, J. (1991). Generalised user modelling shells: a taxonomy. *Proceedings 12th IJCAI Workship Agent Modelling for Intelligent Interaction,* pp. 169–185, Sydney, Australia.

KAY, J. & QUILICI, A., Eds. (1991). *Proceedings 12th IJCAI Workshop Agent Modelling for Intelligent Interaction,* Sydney, Australia.

KOBSA, A. & WAHLSTER, W., Eds. (1989). *User Modeling in Dialog Systems.* Berlin: Springer Verlag.

KOBSA, A. (1990). Modeling the user's conceptual knowledge in BGP-MS, a user modeling shell system. *Computational Intelligence,* **6,** 193–208.

KOK, A. (1989). *User modelling in AI and DB systems.* Tech. report VF-n 111, University of Amsterdam, The Netherlands.

MCALLESTER, D. (1982). *Reasoning utility package user's manual.* AI Lab Report 667, MIT, Cambridge, MA, USA.

MCDERMOTT, D. (1991). A general framework for reason maintenance. *Artificial Intelligence,* **50,** 289–329.

MORIK, C. & ROLLINGER, C. (1985). The real estage agent—modeling users by uncertain reasoning. *The AI Magazine,* **6,** 44–52.

NWANA, H. (1991). User modelling and user adapted interaction in an intelligent tutoring system. *User Modeling and User-adapted Interaction,* **1**(1), 1–32.

ORLEANT, J. L. (1991). The doppelgänger user modeling system. *Proceedings 12th IJCAI Workshop Agent Modelling for Intelligent Interaction,* pp. 165–168, Sydney, Australia.

REITER, R. (1980). A logic for default reasoning. *Artificial Intelligence,* **13,** 81–132.

REITER, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence,* **32,** 57–95.

RICH, E. (1983). Users are individuals: individualizing user models. *International Journal of Man–Machine Studies,* **18,** 199–214.

RICH, E. (1989). Stereotypes and user modeling. In A. KOBSA & W. WAHLSTER, Eds. *User Models in Dialog Systems,* pp. 35–51. Berlin: Springer Verlag.

SELF, J. (1988). Student models: what use are they? In P. ERCOLI & R. LEWIS, Eds. *Artificial Intelligence Tools in Education,* pp. 73–86. Amsterdam: North Holland.

SLEEMAN, D. (1985). UMFE: a user modelling front-end subsystem. *International Journal of Man–Machine Studies,* **23,** 71–88.

SMITH, B. C. (1982). *Reflection and semantics in a procedural language.* Ph.D. Thesis and Technical Report MIT/LCS/TR-272.

SWARTOUT, W. (1983). XPLAIN: a system for creating and explaining expert consulting programs. *Artificial Intelligence,* **21,** 285–325.

WAHLSTER, W. & KOBSA, A. (1989). User models in dialog systems. In A. KOBSA & W. WAHLSTER, Eds. *User Models in Dialog Systems,* pp. 4–34. Berlin: Springer Verlag.

Paper accepted for publication by Associate Editor Professor E. H. Mamdani.