Pragmatic User Model Implementation in an Intelligent Help System

Baltasar Fernandez-Manjon

Alfredo Fernandez-Valmayor

Carmen Fernandez-Chamizo

The authors are at the Complutense University of Madrid. Their main research interests are educational software development, user modeling, intelligent help systems and multimedia applications. Address for correspondence: Escuela Superior de Informatica, Universidad Complutense de Madrid, Avda. Complutense s/n, 28040, Madrid, Spain. e-mail: bfmanjon@dia.ucm.es, valmayor@fis.ucm.es, cfernan@dia.ucm.es

Abstract: We present an intelligent help system for the Unix operating system called Aran. Aran is a passive, knowledge-based help system whose main goal is to help users deal with problems related to Unix operation. Aran gives the user access to different kinds of information, and it is designed to help the user accomplish a given task, while, at the same time, expanding the user's knowledge of the operating system. In order to accomplish its job Aran is endowed with an explicit representation of both domain specific knowledge and a pragmatic user model (UM) that takes into account the individual´s characteristics. In Aran, user modeling is dynamic and based on stereotypes. The UM is implemented with the same representation environment, Loom, used in the rest of knowledge base (KB) modules. This environment is based on a description logics formalism that facilitates the construction and maintenance of all the modules.

INTRODUCTION

As the complexity of software applications increases and computers are being used by all kind of individuals, the provision of on line help in complex software applications is becoming crucial. Today's computer users expect to be able to use an application with minimal or no training at all, and with no specialised computer background (Kearsley, 1988). These increased user expectations can not be fulfilled by improving system's interfaces only. In spite of the very positive impact on usability that the focus on friendly interfaces has already produced, we believe that additional help is required (Duffy *et al.*, 1992).

Intelligent Help Systems (IHS, also known as Intelligent Assistants) have been proposed as a way to improve the usability of complex software applications (Wasson and Akselsen, 1992; Winkels, 1992). In our project, we envisage this provision of help as a double goal process. First, as a short term objective, the IHS must enable the user to carry on with the task at hand, improving user performance. Second, the help offered by the IHS must expand the knowledge that the user already has of the application, so that the need for help will decrease in the long term. We consider this second goal an educational objective at least as important as the first one. This approach can be used not only to support software applications but in more general educational settings. We consider that IHS can integrate

smoothly the learning and working processes providing benefits such as a potential increase in users' motivation (Buenaga *et al.* 1995).

In order to provide assistance adapted to user's needs, a User Model (UM) is required (Kobsa and Whalster, 1989; Kok, 1991). In our view, techniques to adapt the assistance to the user's context must be at the core of an IHS (Jones and Virvou, 1991). That is, the provision of help appropriate to individual user's experience must be based on both knowledge about the domain and knowledge about the user. The pragmatic UM (i.e. simple but useful) will provide the IHS with all the relevant data necessary to improve its interaction with the user in complex domains like the Unix environment (Kay, 1994).

The peculiarities of the user assistance task have driven us to focus on stereotypes (Rich, 1989). Initially, stereotypes are automatically assigned to each individual UM. Later these stereotypes will be dynamically refined, updated and maintained during the session. It is also important that the UM can predict the user's knowledge level, based on partial information about that user (Chin, 1993).

The use of a standard description logics environment like Loom (MacGregor, 1991) simplifies the construction, integration and maintenance of the UM, because the same representation environment is used for all the knowledge base (KB) modules.

In this paper we will describe how we have applied these principles to the UM implementation of the IHS Aran. In the following section, we give an overview of Aran, the system that exemplifies our approach to user assistance in the Unix domain. In the third section, we present the UM integrated in Aran. Details of the definition of stereotypes, use and maintenance of this model and its implementation using Loom are also given. In the fourth section, we discuss some of the related work. The fifth section of this paper presents the conclusions of our project.

APPLICATION DOMAIN

Aran is an IHS for the UNIX operating system. When a user comes across a problem while using Unix, they can activate Aran, express their needs using a multimodal interface, and obtain the necessary information to overcome their problem. Initially, the main aim of Aran development was to demonstrate the feasibility of an IHS for a complex software domain. It was also conceived as a framework for investigating different aspects of user modeling and the provision of help.

Overview of Aran

Aran integrates different "standard" technologies to provide the help facilities. The aim is to simplify user access, selection and understanding of the information needed to overcome the user's current problem, while, at the same time, offering the user the possibility to expand his knowledge of the operating system. The integrated technologies are: hypertext, information retrieval (IR), formal concept analysis (FCA), user modeling, and domain modeling. Aran uses hypertext techniques as a tool to navigate through the Unix documentation and to interact with the explicit knowledge representation of Unix. The IR and FCA techniques simplify indexing and access to Unix standard documents. Adapting the help to the individual user's characteristics is done with the information stored in the UM. A more detailed description of Aran and of our approach to the IHS construction can be found elsewhere (Buenaga *et al.*, 1995; Fernandez-Manjon and Fernandez-Valmayor, 1997).

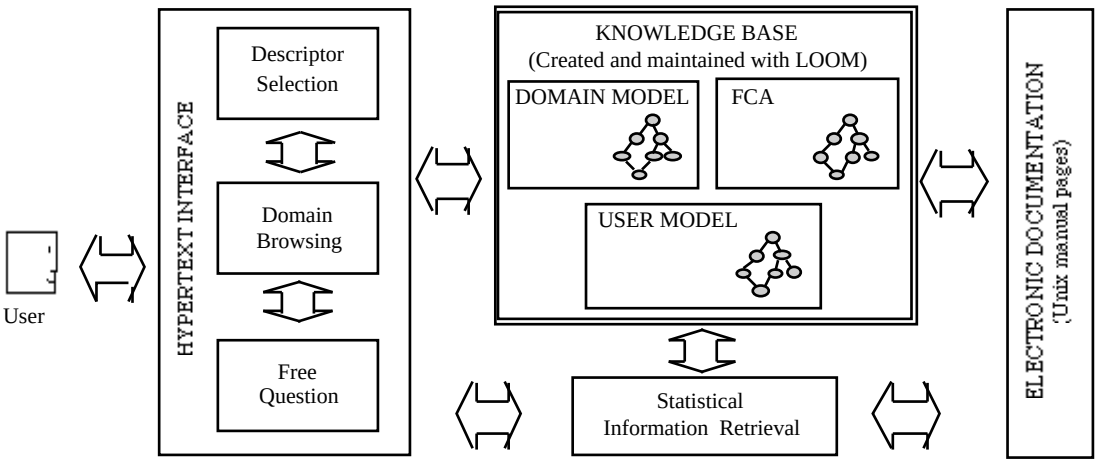

Figure 1: Structure of Aran

The core of Aran is a KB (made up of the domain model, the UM and the FCA module) where the different types of knowledge and information are represented, organised and maintained using the same knowledge representation and reasoning environment (Loom) (fig 1). The domain knowledge is a conceptual model of the Unix operating system that tries to reflect its information design. This model allows the organisation and indexing of different kinds of topics around the key concepts of the domain. We have used a representation similar to that of the Sinix Consultant (Hecking *et al.*, 1988). In the domain module of the KB, the world of Unix concepts is divided into entities, which correspond to Unix objects (e.g. file, process), and actions or operations which involve these objects (e.g. change , communicate with user). Higher-level concepts in the taxonomy reflect more general objects or actions. The individual Unix commands and objects are represented as instances of actions and entities, respectively. The command representation takes into account different aspects: syntactic (e.g. name, syntax), semantics (e.g. description text) and pragmatic/tutorial (e.g. related commands, prerequisite concepts, related concepts).

Other help systems present only *ad hoc* information to the user, but Aran reuses the complete documentation that is shipped, in electronic format (manual pages), within the operating system. This documentation is index in three different ways: a) knowledge-based indexing, where the documents are indexed using the concepts (mainly the actions) of the domain model; b) statistical free text indexing, where the documents are indexed by terms automatically extracted from the text; c) FCA indexing, where the documents are indexed with a set of descriptors (keywords).

Aran provides a hypertext graphical user interface that supports three different, but related, interaction modes. These three operating modes correspond to the three kinds of textual information indexing. In the *browsing mode*, menus and mouse-sensitive representations of the domain model are employed for accessing the domain information and documentation (fig 2). This direct interaction with the domain model will help the user to acquire a complete and accurate model of the Unix system. The *free question mode*, where the user makes requests for information using free text and obtains a ranked list of relevant documents (Araya, 1990). The *descriptor selection mode*, where the user chooses the descriptors incrementally from a list (provided by the FCA module) obtaining all the documents where those descriptors appear. If the user selects a document using the question mode or the descriptor selection mode and then switches to the browsing mode, the visualisation of the domain will be centered in the concepts that index this document.

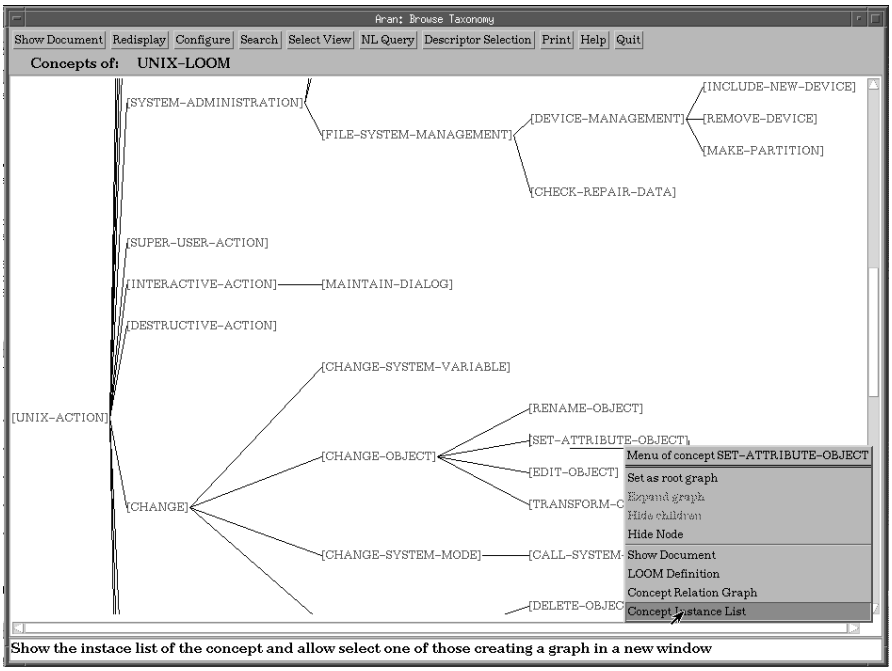

Figure 2: Aran browsing interface of the domain model providing different paths to access Unix information.

Adaptation to the individual user

As we have seen in the previous section, the KB contains and organises a variety of information about the Unix domain and the commands of the system. To meet our double goal of assistance, the information displayed by Aran should be different depending on the individual user's characteristics. The UM stores all the explicit assumptions and information about the user that may be relevant for helping the user to cope with all this complex information. Adaptation to user's needs is done by:

- Improving the free text interaction. Taking into account previous queries complemented with the feedback information provided by the user about the documents retrieved in previous searches, Aran is able to filter irrelevant documents (Araya, 1990).
- Ranking the retrieved information. According to this ranking, Aran first presents the retrieved documents that are new to the user, that is, never presented before and,inferred from the UM to be unknown documents. Next in the rank are documents assumed to be already known by the user and, finally, documents that have already been presented during the current session. This way, the UM serves Aran in deferring the presentation of repetitive information for a given user.
- Modifying the visualisation of the domain model. When the user accesses an instance (command), Aran aims at adapting it to their presumed conceptual knowledge. The UM allows different presentation strategies based upon domain elements known and unknown by the user. The visualisation of an instance is modified in the two following ways (fig 3):
    a) if the user is familiar with the command, the related examples (if any) are not shown;
    b) the command's prerequisite concepts and the command's related concepts that the user is already familiar with are not displayed.

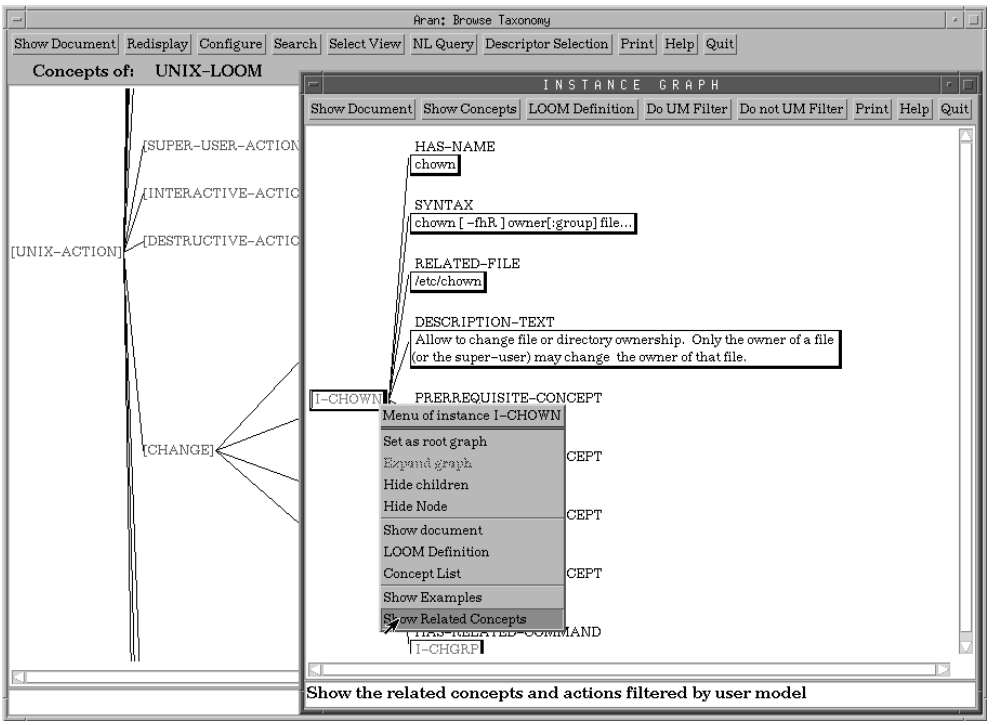

Figure 3: Example of information adaptation in the domain browser. Here, the examples and related concepts of the *chown* command were filtered but the user can recover this information.

As filtering can produce undesirable effects of information hiding, the user always has the possibility of recovering the filtered information (fig 3). The user can also switch off the automatic adaptation.

USER MODELING KNOWLEDGE

The peculiarities of the user assistance and adaptation tasks drive us to focus on automatic, rapid initial UM acquisition that will later be dynamically refined, updated and maintained during the session with the assistant (Chin, 1993). Also, it is important that the UM allows Aran to predict the user's knowledge based on partial information about the user. We have chosen a dynamic, individual UM based on stereotypes (Rich, 1989) because this approach permits an initial user classification that evolves individually as the user performs interactions.

The current version of Aran does not store the UM between sessions. Long-term user characteristics should be modeled with caution since an error in their acquisition will be more serious than in short-term characteristics.

Stereotypes

The *stereotype* UM is based in the identification of groups of users whose members are very likely to possess certain homogeneous application-relevant characteristics. For each stereotype, we identify (and formalise in Loom) a small number of key characteristics which allow the system to identify the user as belonging or not belonging to the corresponding user subgroup. These key characteristics will permit the creation of an individual UM as an instance of one or several stereotypes. The stereotypes are of

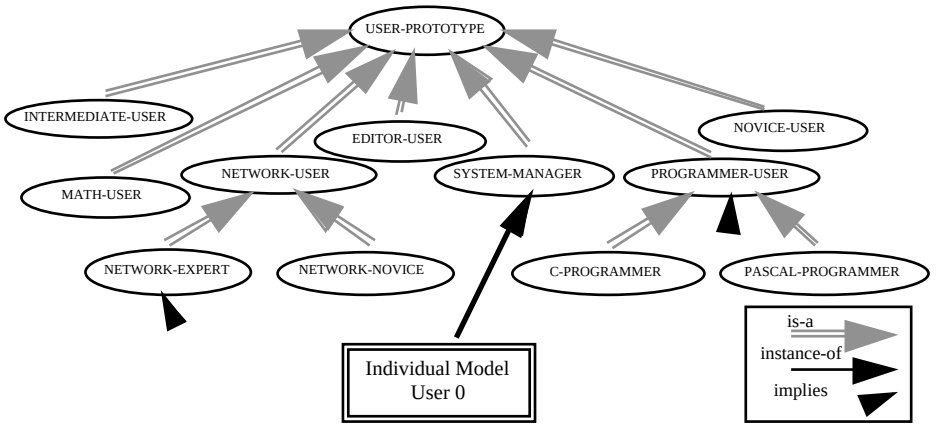

Figure 4: Stereotype hierarchy in the Unix domain.

particular importance at the beginning of a session, because at that point they can provide default knowledge that will be overridden when there is specific information obtained from the interaction with the user.

The stereotypes are classified hierarchically in a directed acyclic graph using Loom standard classification features (fig 4). These domain-oriented stereotypes try to represent users' experience and users' interests with respect to different context and subdomains of Unix. We do not claim this classification to be complete or even accurate, but it is enough to show the potential and characteristics of the user modeling done (also, it can be further refined without major modifications in Aran).
The root of the hierarchy is the stereotype *user-prototype*. This is the general stereotype which defines the knowledge that will be available for modeling any user in Aran. All other stereotypes include typical characteristics of users that specialise the contents inherited from more general stereotypes. In each stereotype, the assumptions that apply to any user of this subgroup are explicitly represented. In the hierarchy of stereotypes, the restrictions and relations between the stereotypes are also explicitly represented (and so automatically maintained by Loom). The stereotypes considered for a given criteria can be mutually disjoint or not. For example, a user can not be simultaneously classified as *networt-expert* and as *network-novice,* but the user can be classified as *c-programmer* and as *pascal-programer* at the same time.

Content of the individual user model

In the UM we distinguish three main different kinds of knowledge (fig 5):

a) initial information, the data directly acquired from the Unix system;
b) objective facts, the data dynamically acquired from the user interaction;
c) subjective facts, the assumptions drawn from the user interaction.

The UM has four slots for initial information: has-name, has-id, has-group and history-list. The first three slots store the user's identification data. The history-list stores the history of the previous command interaction between the user and Unix. The model has three slots for objective facts: term-list, accessed-components and accessed-concepts. The term-list stores the words obtained from previous queries and from the relevance feedback information. The accessed-concepts and accessed-components store respectively the domain concepts and documents directly accessed by the user. The model has four slots for assumptions representing the presupposed-(un)known-concepts and the presupposed-(un)known-

```
(defconcept User-Prototype
  "concept that groups the user stereotypes"
  :is-primitive
  (:and Unix-Thing
        (:the has-name Identifier)
        (:the has-id Number)
        (:the has-group Number)
        (:exactly 1 history-list)
        (:exactly 1 term-list)
        (:exactly 1 accessed-documents)
        (:all accessed-concepts Unix-Thing)
        (:all presupposed-known-commands Unix-Action)
        (:all presupposed-unknown-commands Unix-Action)
        (:all presupposed-known-concepts  Unix-Thing)
        (:all presupposed-unknown-concepts Unix-Thing)))
        (:all key-commands Unix-Action)
```

Figure 5: Definition of User-Prototype, the root concept of the stereotype hierarchy. This Loom template is specialised in the actual stereotype definition for each user subgroup.

commands of the domain representation. The key-commands slot stores special domain information used in stereotype assignment. Part of the assumptions and key-commands included in the individual UM are inherited from the stereotype(s) applied to this user.

Instantiation, acquisition and maintenance of the individual model

The initial UM acquisition is automatic and implicit, based on the information obtained from the Unix system. From these identification and history data, Aran creates an instance (i.e. an individual UM) and determines the initial stereotype(s) that apply. There are two possibilities for the initial stereotype assignment:

a) If the stereotype is fully defined (by the necessary and sufficient conditions) then this assignment is done automatically by the classification process. For example, in Unix, if the user has zero as identification number then this user is considered as system manager or superuser, so Aran creates an instance that is automatically classified by Loom as *system-manager*. Using Loom features Aran also implies automatically that he is a *network-expert* and a *programmer-user* (fig 4). The (simplified) Loom code is:

```
(defconcept System-Manager
    :is
    (:and User-Prototype
       (:filled-by has-id 0))
    :implies
       (:and Network-Expert Programmer-User))
```

b) If the stereotype is not defined by the necessary and sufficient conditions this assignment is based on the user's characteristics matching a given proportion of the stereotype. So this assignment is done by heuristics rules (programmed in Loom) that take into account the history list and the slots presupposed-known-commands and presupposed-unknown-commands. Also, if a user has a command in the history list that is specified as value of the key-commands slot of a stereotype, then this stereotype is assigned. For example, if a user has used the "cc" (C compiler) command they could be classified as *c-programmer*. At least, it is probable they will be interested in information about C library functions.

Dynamic acquisition of objective and subjective facts

Aran incrementally constructs an individual UM for each session. This is done mostly in an implicit way. From the interaction with the user, Aran obtains the objective data for the model. The user only has to indicate explicitly when they are changing the subject of the search in order to reset the term-list. From the user's actions, and using domain specific heuristics, Aran derives assumptions about the commands (and related documents) and concepts the user presumably does or does not know. Aran draws the two following primary assumptions about the user, based on his actions at the browsing interface:

a) If the user requests an example for a command, then he is assumed to be unfamiliar with this command.

b) If the user requests to see the command's related concepts previously filtered by the system, and he explicitly selects one of those concepts, then he is assumed to be unfamiliar with the selected concept.

These assumptions are stored in the individual UM.

Maintenance

Since the acquisition of new facts may produce inconsistencies in the individual UM, along with Loom classification process, Aran has procedures to check and maintain the coherence of the model. These procedures are programmed as Loom automatic rules that fire when there is a modification of the UM slot values.

If the inconsistency comes from the assignment of the user to a stereotype (by the inherited information), this assignment is retracted. If the inconsistency comes from incoherent information, the inaccurate data will be corrected. For example, if one concept is supposed known and unknown at the same time, the concept will be deleted from the known concepts.

The acquisition of new facts will produce also the re-evaluation of the assignment conditions of the stereotypes. This assignment is based on the user's characteristics matching a given proportion of the stereotype (as previously discussed in the initial model acquisition). Aran's dynamic behaviour takes into account changes in the user interests and knowledge, changing the appropriate stereotypes assignment accordingly.

RELATED WORK

Nowadays, research on adaptive systems is a very active field and it is widely accepted that this adaptivity requires an explicit UM to be embedded in the system (Kok, 1991). In the IHS domain, methods for adapting the behaviour of the system have been investigated in a number of different projects (Wasson and Akselsen, 1992). Several projects are on the Unix domain (Wilensky *et al.*, 1988; Winkels, 1992; Hecking *et al*., 1988) but in all of them the stress is on the adaptation of the natural language interaction. By contrast, in Aran the UM is used for modifying the visualisation of the domain model and the document retrieval and presentation. To some extent our approach is similar to the proposed by Hohl *et al.* (1996) but using domain-oriented stereotypes.

The formal representation and reasoning used for the UM component of Aran is related to the work done in user modeling shell systems (Kass and Finin, 1991; Kobsa and Pohl, 1995). Our model offers some of the possibilities offered by these shells (e.g. arbitrary stereotype hierarchy, UM coherence maintenance) combined with IR techniques. The

integration of IR techniques is also related to previous approaches on user relevance feedback for refining the formulation of user queries (Araya, 1990), but we extend them by integrating the relevance information with a more general UM.

CONCLUSIONS

Until now, only a limited evaluation of our system, in our own university environment, has been done. So the first positive feedback that we get from users must be tested with more extensive and systematic experiments. The conclusions we can offer in this phase of the project are mainly related to the development process of the system based on the formative evaluation realised in parallel with a small group of colleagues.

One of the strengths of Aran is to take advantage of a standard representation environment (Loom) to facilitate IHS construction instead of using *ad hoc* tools. Along with the taxonomic-based reasoning, Loom features an object-oriented assertional language which is dynamically truth maintained. Also, Loom integrates the terminological capabilities with two paradigms for behavioural specification: object oriented programming, and rule based programming. These characteristics of the Loom environment simplify the integration of different techniques in the IHS, and the construction and management of the KB (in particular the dynamic maintenance of the UM).

The pragmatic UM is the other strength of our system. It has been demonstrated that with limited data and assumptions about user knowledge, Aran can offer appropriate information for many of the practical problems that the user encounters when using Unix.

ACKNOWLEDGEMENTS

This work has been supported by the Spanish Commitee of Science & Technology (TIC94-0187 and TIC96-2486-CE)

REFERENCES

Araya J (1990) *Interactive Query Formulation and Feedback Experiments in Information Retrieval* Unpublished doctoral dissertation, Cornell University, USA.

Buenaga M *et al.* (1995) Information Overload At The Information Age *in* Collis B and Davies G (eds) *Innovating adult learning with innovative technologies* Elsevier Science, Amsterdam.

Chin D N (1993) Acquiring User Models *Artificial Intelligence Review* **7** (3-4) 185-197.

Duffy T M *et al.* (1992) *On Line Help Design and Evaluation* Ablex Publishing Corporation, Norwood, New Jersey.

Fernandez-Manjon B and Fernandez-Valmayor A (1997) Building Educational Tools based on Formal Concept Analysis *in* Proceedings of the IFIP WG 3.3 Working Conference on Human-Computer Interaction and Educational Tools, Sozopol, Bulgaria (to appear).

Hecking M *et al.* (1988) *The SINIX Consultant - A Progress Report* Memo Nr.28, KI-Labor, Informatik IV, Universitaet des Saarlandes, Saarbruecken, Germany.

Hohl H *et al.* (1996) Hypadapter: An Adaptive Hypertext System for Exploratory Learning and Programming *User Modeling and User-Adapted Interaction* **6** (2-3) 131-156.

Jones J and Virvou M (1991) User Modelling and Advice Giving in Intelligent Help Systems for Unix *Information and Software Technology* **33** (2) 121-133.

Kass R and Finin T (1991) General User Modeling: A Facility to Support Intelligent Interaction *in* Sullivan J and Tyler S (eds) *Intelligent User Interfaces* ACM Press, New York.

Kay J (1994) Lies, Damned Lies And Stereotypes: Pragmatic Approximations Of Users *in* Proceedings of the Fourth International Conference on User Modeling, The MITRE Corporation, Hyannis, MA.

Kearsley G (1988) *Online Help Systems: Design and Implementation* Ablex, Norwood, New Jersey.

Kobsa A and Pohl W (1995) The User Modeling Shell System BGP-MS *User Modeling and User-Adapted Interaction* **4** (2) 59-106.

Kobsa A and Whalster W (1989) (eds) *User Modelling in Dialog Systems* Springer, Berlin.

Kok A J (1991) A Review and Synthesis of User Modelling in Intelligent Systems *The Knowledge Engineering Review* **6** (1) 21-47.

MacGregor R (1991) The Evolving Technology of Classification-based Knowledge Representation Systems *in* Sowa J F (ed) *Principles of Semantic Networks* Morgan Kaufmann, San Mateo, California.

Rich E (1989) Stereotypes and User Modelling *in* Kobsa A and Wahlster W (eds) *User Models in Dialog Systems* Springer, Berlin.

Wasson B and Akselsen S (1992) An Overview of On-line Assistance: from On-line Documentation to Intelligent Help and Training *The Knowledge Engineering Review* **7** (4) 289-322.

Wilensky R *et al.* (1988) The Berkeley UNIX Consultant Project *Computational Linguistics* **14** (4) 35-84.

Winkels R (1992) *Explorations in Intelligent Tutoring and Help* IOS Press, Amsterdam.