An Application-Independent Intelligent User Support System Exploiting Action-Sequence Based User Modelling

L. Miguel Encarnação[1] and Stanislav L. Stoev[2]

[1] Fraunhofer Center for Research in Computer Graphics (CRCG), Inc., Providence RI, USA
[2] Department of Computer Science, University of Tübingen, Germany

Abstract. Many software systems' usability suffers from their complexity, usually caused by the market-driven trend to bundle a huge amount of features, which are supposed to increase the product's attractiveness. This attempt, however, more often than not leads to software with poor usability characteristics, therefore requiring an extensive amount of initial effort for the users to become familiar with the system. One way to overcome this problem is by providing user-adapted usage support.

In this paper we present an experimental system for intelligent user support, which has been developed under the aspect of portability. Focusing on this goal, the system supports a variety of user- and task-modeling approaches and is independent from the hosting software application environment, thus being ready to integrate with existing and new applications. The different user-modeling approaches have been empirically evaluated and compared in a medical software application embedding our system. The results of this evaluation are briefly described in the report.

1 Introduction and Related Work

The main drawback of many software systems is their complexity, usually caused by the market-driven trend to bundle all features, which should make the product more attractive even for demanding expert users. This attempt however, often leads to software with poor usability, which not only require a large amount of initial effort for the users to become familiar with the systems. It also keeps many non-professional and even professional users from accepting new, or switching over to better software products.

Several approaches offering solutions to this problem have been described in the literature. To the most popular belong *tutoring systems*, whose basic idea is to support the user in learning the use of the application system. A second branch includes the research towards intelligent user interfaces, which has been conducted for quite a long time now. This work was driven by researchers and developers from the most distinct areas of e.g., computer science, artificial intelligence (AI), human factors, and psychology (Sullivan and Tyler, 1991, Maybury, 1993, Gray et al., 1993, Schneider-Hufschmidt et al., 1993), yet only few compelling application systems have been developed (Marks et al., 1997).

In many cases the intelligent user support has had a negative influence on the performance of the application system itself, or on the user's performance at the interface to the application system (e.g., (Woods, 1993)). In other cases, stand-alone application systems were developed as platforms for AI methods and techniques in order to demonstrate the applicability of the latter, yet condemning reasonable intelligent user support to sole academic existence. Other approaches attempt to exploit the knowledge put into graphical user interfaces at design time to automatically

generate intelligent user support (Sukaviriya and Foley, 1990, Lonczewski, 1997). Here, one of the problems is that up until now only very few real-world applications have been developed on the basis of such user-interface development systems. We believe that the most promising approach is the intelligent user adapted support, which has to be flexible enough to allow an easy integration into new and existing applications.

Unlike most of the work referenced above, the ORIMUHS[1] system presented in this paper has been conceptually developed and implemented as an extension for new and existing graphical user interfaces. It offers a fair amount of intelligent user support by realizing as many components of intelligent user interfaces as possible such as; multi-modal communication (Neal and Shapiro, 1991), dynamic presentation (Castells et al., 1997), and adaptability (Dieterich et al., 1993). Yet it also strives to provide a small application programmers' interface (API) for application-independent integration with reasonably little effort, one that is flexible enough to support future enhancements towards new intelligent components or more sophisticated AI-based user support. Therefore, our approach allows the integration in real-world application systems, providing a means of introducing the different aspects of user modelling to market and industry – thus increasing its acceptance. On the other hand, it supports further research on intelligent user support – its impact and its shortcomings – in multiple 'real-world' environments, thus allowing comparative evaluations on different applications systems from various application areas.

In the next section we describe the basic concepts of our system, introducing the information we dispose of, the architecture, and the dependencies of the developed components. In Section 3 we present the user modelling methods used to evaluate the protocolled data and we outline the methods' application and usage in Section 4. Section 5 reports on results of the user modelling process, applying the techniques described in Section 3. The final section summarizes the attributes of the presented user support system.

2   System Overview

Keeping in mind the initial goal of developing an easy-to-integrate system, we attempted to define an 'as-slim-as-possible' interface between the hosting software application and our user support components. This effort resulted in the protocolling of the most basic trackable units: the user interactions with the GUI which we named *actions*. In doing so, the integration of our user support in an application system then consists of the following steps:

Application System:
- registering of the actions to be tracked.
  - adding a function notifying the HCIS.
  - assigning an ID to each action.

ORIMUHS:
- building up the action-context-concept hierarchy.
  - which unit contains which subunits.
  - assigning a LOD to each unit.
  - define for each unit the 'successful completed' criteria.
- defining correspondences between working context/concept and user support (i.e. action ID – HTML-page).
- recording the actions of the preclassified users to build the action-graphs.

---

[1] ORIMUHS stands for **O**bject-**o**riented **I**ntelligent **Mu**ltimedia **H**elp **S**ystem.

We implemented the protocoling of actions by adding a function to the GUI-elements we are interested in (i.e. buttons or menu items). This functionality is hosted by the *Action Translation Module (ATM)* depicted in Figure 1. Depending on the design and implementation the application GUI, a recompilation of the application system can be limited and is sometimes even dispensable.

Once the GUI-elements are 'registered' by adding the server-notifying function, there is no need to manipulate further application code. All of the following steps, like action protocoling, action sequence evaluation, and user modelling are performed by the server (*HCIS*[2]), thus not affecting the application's performance.

The HCIS is an application-independent server, communicating with the clients (the application systems) through a TCP/IP connection. It can therefore be used in a local area network (LAN) (cf. Figure 1) to serve an arbitrary number of clients simultaneously. Although the use in wide-area networks (WAN) is generally possible, security considerations make the limitation to LANs sound reasonable. The main task of the HCIS is to keep track of the performed actions and evaluate them when requested based on the evaluation methods discussed in Section 3.
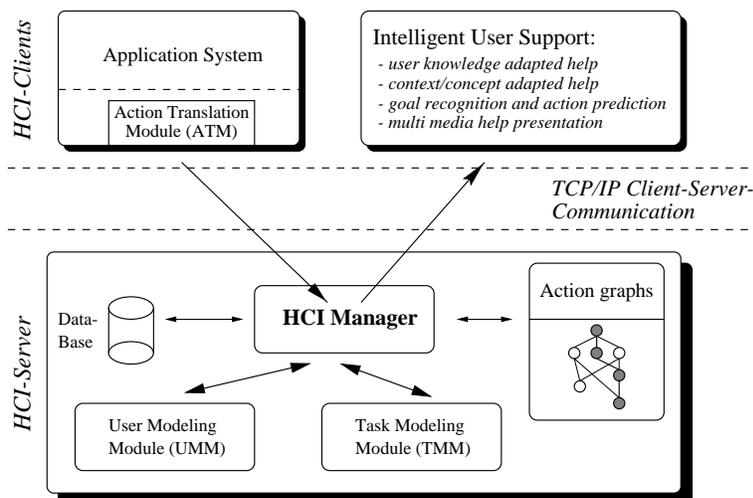
Figure 1. The architecture of the ORIMUHS system.

When a user starts working with the served application, HCIS uses his/her ID to load the available user data if the ID is unknown to the system. Otherwise a new user profile is created and initialized with default values[3]. After this initial 'handshake' with the application, the server keeps track of every interaction of interest (registered action) performed by the user (as depicted in Figure 2 of section 3.2).

The HCIS manages the action-sequence dependencies in graph-like structures, called *action graphs* (Encarnação, 1995). Such action graphs where the nodes are actions and the edges rep-

---

[2] **H**uman **C**omputer **I**nteraction **S**erver.

[3] We use a default user model for initialization and did not ask the users for rating themselves, because in most of the cases they cannot judge their own knowledge correct, thus misleading the help system.

resent connections building the action sequences, contain interactions performed by users with known levels of expertise. In this sense the graphs aim to represent the behaviour of users with similar application knowledge.

2.1 The Action Graphs

For a precise determination of the user's application knowledge a reference base is needed. Such reference model is created by protocoling the actions of users with a-priori known levels of expertise. Such models are coded in the action graphs. During the first information-gathering phase the action graphs are created adding the GUI-interactions performed by the pre-classified users to the corresponding action graph, or updating the weights of existing connections.

Once the reference action graphs have been built, the user's interactions with the application system are compared with the graphs to determine whether the user fits into a particular class of expertise or not. In this second phase the HCIS computes the expertise class of a connected user after every performed action by determining where she/he best fits in. This computation is performed for each session of the user with the application. The result is the class used to determine the adequate user support when requested, affecting the help's level of detail and presentation media. The process of updating the user-knowledge value is depicted in Figure 2 of Section 3.2.

2.2 Overlay Models

Most of today's commercial application systems are used for more than one purpose, out of which a single user usually only exploits a few. Hence, adequate user stereotypes cannot be applied to the whole application system, but have to be related to certain application purposes of the system which in return require appropriate internal representations. We therefore introduced an *overlay model* (Carr and Goldstein, 1977) paradigm to our system (Encarnação and Stoev, 1996). First, we defined lower-level units containing actions with similar meaning to the application called *contexts*. Contexts are low-level user *goals* (i.e. 'File I/O', 'Model Part', and 'Define Lights' for our CAD application system). On a higher level of abstraction, we introduced *concept* units, containing contexts as subunits. Concepts can be considered as representing the *purposes* for which an application system can be used for (i.e. 'Construction', 'Demonstration', and 'Analysis' in a CAD-System).

In contrast to traditional overlay models which only differentiate between known or unknown knowledge units, ORIMUHS' overlay modelling is able to assign levels of expertise of the corresponding user to the knowledge units. These levels of expertise are part of the user model and are computed utilizing the methods described in the next section.

3 User Modelling Strategies

Before describing the knowledge updating strategies, we introduce the user specific data the HCIS disposes of. Here we distinguish again between context- or concept-oriented data and information on the global user knowledge:

Global User Information:	***Working-Context Specific***
• overall system knowledge:	***User Information:***
○ statistically calculated value.	• unit related knowledge.
○ probabilistically calculated value.	• number of visits to unit.
• number of sessions with the application.	• number of all performed **Help, Cancel,**
• number of all performed **Help, Cancel,**	and **Undo** when being in this unit.
and **Undo** actions.	

The user modelling process itself, similar to the simulation of a learning process, is a dynamic process depending on the humans' learning capabilities. In order to respond to this dynamic, the user knowledge model should be updated frequently enough to represent the correct level of expertise. In our user support system a user's knowledge model is therefore updated every time he/she performs an action. This update process is depicted in Figure 2.
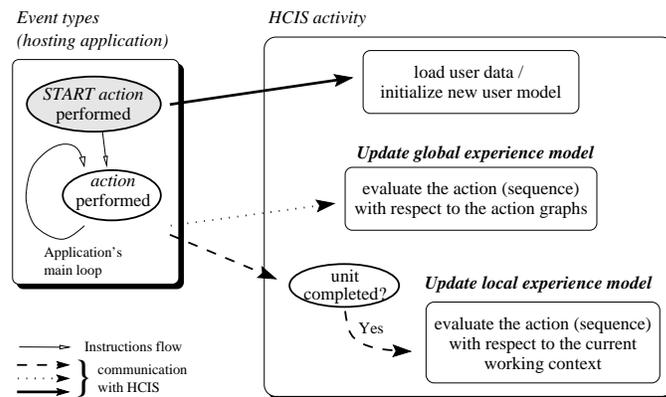
Figure 2. Updating the *global* (dotted line) and the *local* (dashed line) user expertise.

In ORIMUHS we realized two different strategies for modelling the user's knowledge in order to consider the different situations in which user support might be required or requested (Encarnação, 1997):

The *global user modelling* determines a user-expertise value related to the whole application system. It takes into account the action sequences performed by users with similar system knowledge (coded in the action-graph for this user class).

In contrast to this approach, the local strategy is based on the overlay model described above. It calculates an expertise model with respect to the given context and concept, thus reflecting not the overall system knowledge, but the expertise in particular *working contexts* (this refers to both the *contexts* and the *concepts*).

Depending on the application situation in which a user's level of expertise is computed and evaluated in order to provide user-adapted support, either one of the approaches might be feasible or a combination of both.

3.1 The Global User Model

For the calculation of the user's global expertise model we initially use two action-graph evaluation strategies. The first is the *statistical* evaluation which (statistically) computes the transmission loads between subsequent actions with respect to alternative sequences, using the action

graphs as a reference base. In order to determine statistically the value u_i, representing the update of the user's expertise after performing the action a_i, we introduce the system's *confidence* in executing action a_j followed by action a_i (Encarnação, 1997, Encarnação and Stoev, 1996). This approach, based on comparison of the total number of a_i's visits with the total number of (a_j, a_i) visits, has been extended to the case where the action a_i is performed after action a_j, 'going through' action a_k. Thus we (statistically) retrieve a measure of the 'local fitting' of this action sequence in an action graph.

The *probability*-based approach is the more global one, which computes the probabilities that certain (sequences of) actions are executed. The values computed in this manner are not calculated with respect to possible alternatives, but as nearly absolute measurement. Similar to the confidence by the statistical evaluation, we introduce the probability of performing action a_i after action a_j (or $p(a_i|a_j)$). Again similar to the other approach, we extended this probability to $p(a_i, a_j, a_k)$, which denotes the execution's probability for the action sequence (a_j, a_k, a_i) (Encarnação, 1997,Encarnação and Stoev, 1996).

Both strategies can be employed to update the user's overall level of affiliation with respect to an action graph, yet should be applied according to the situation out of which the user-support need evolved: The statistical is most applicable to situations where the user has been lost in his local context, whereas the probabilistic approach is applicable to task user-support that helps to plan certain tasks.

After performing the evaluation step with all available action graphs, we obtain a suitable method for assigning a level of expertise (and thus an expertise class) to a user.

For both evaluations three actions are considered; with only two actions it would be difficult to determine the user's current working context. On the other hand, more than three actions could cause the calculations to be too expensive for the server. The server would then not be able to respond to help requests in real time when serving several applications.

3.2   The Local User Model

When considering a user model, we are often not interested in the user's overall application system knowledge, but in expertise values related to the current working context. Typically, when the help system offers user support, it should take into account the user's knowledge in the concrete situation. Otherwise the offered information might be worthless or even confusing for the user.

In order to get such information we introduce new values to the user model which represent the user's level of expertise with respect to each context or concept. These values are updated every time the user completes his/her work in that working context/concept (or short *unit*) (as shown in Figure 2).

Note that an additional unit attribute is needed in order to update the local user expertise: the unit's level of difficulty (*LOD*). This LOD is part of the unit's attributes, which consist of subunit IDs and LOD, and have to be defined by the help system's administrator.

For the update of the local user model we take into account both the current user's expertise in this context/concept, and the unit's LOD. In addition, we need to know about the user's *success* on the work in this unit before recalculating the local user's expertise. This parameter ($\in [0, 1]$) is calculated in consideration of the number of performed *Undo*, *Cancel*, and *Help* actions since entering the current unit, related to the number of subunits of the processed unit.

ORIMUHS was conceptually designed as an experimental system which supports the integration and comparative evaluation of different user modelling approaches, two of which we implemented so far:

Similar to Chin (1989), where a fuzzy logic based uncertainty management method is utilized to incrementally update the assessment for the user's expertise, we use *fuzzy rules* like the following one:

IF user is a *Beginner* AND he/she completed a *simple*[4] working context,

THEN it now seems *more likely* that she/he is an advanced user,

to update the user's expertise in the working context.

The second approach that we implemented in ORIMUHS is based on Bayesian networks. It utilizes the same input data as the Fuzzy-logic based approach, namely the current user model, the unit's level of difficulty, and the user's success on the unit's completion. The update of the user's expertise level with this method is a straight forward bottom-up evidence propagation in a Bayesian network with three nodes (ABIS, 1997). A comparison of similar user modelling strategies can be found in Jameson (1995).

4 Applying the User Model

We believe that even excellent user modelling results might lead to inappropriate and therefore confusing user support, if they cannot be applied adequately. For this reason we integrated most of the modelling strategies discussed above in the *Hypermedia Help Navigator* (*HyHN*) shown in Figure 3.

When help is requested by the user, or the system ascertains than the user needs help[5], HCIS performs following steps:

- determine the current context(s)/concept(s)[6] (using the last three performed actions);
- determine the user's expertise class (again using the last three performed actions);
- display the help assigned to the determined working context and user expertise level with the appropriate media.

Since ORIMUHS does not support the automatic generation of hypermedia documents, we developed this separate navigation interface, which is directly connected to the hypermedia user support. HyHN communicates with the HCI-Server to retrieve its information on the user model and working context (Encarnação, 1997). These user data determine how detailed the provided help should be and which kind of help presentation is appropriate for a user with the given application and working context knowledge. Additionally the user model is used in combination with the 'best fitting' action-graph to calculate a set of alternative actions (Figure 3, list box 'Alternatives'), belonging to the currently determined context/concept. This is part of the goal recognition and action prediction module described in Stoev and Encarnação (1998).

To overcome inappropriate user-modelling results, the HyHN allows the end-user to switch to the categories (*variants*) of hypermedia documents with the preferred level of detail by using

---

[4] A simple mapping is used to convert the working context's LOD to a linguistic variable.

[5] When the performed Undo and Cancel actions are protocolled, HCIS can decide to offer the user the appropriate help, which can be rejected by the user.

[6] Sometimes the systems determines more than one possible context/concept.
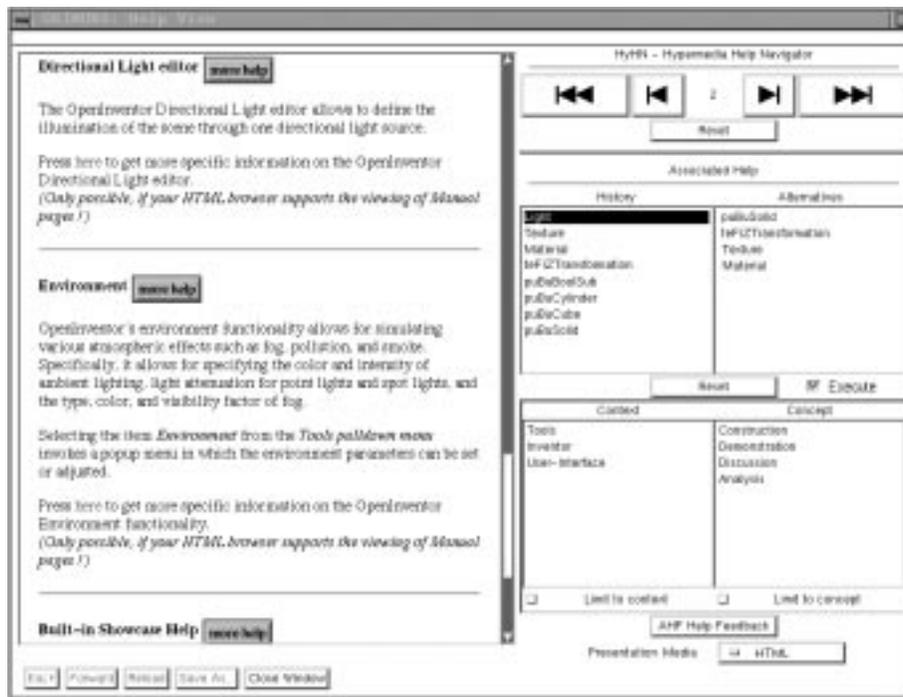
Figure 3. The *Hypertext Help Navigator*.

the arrow buttons in the upper row. In case of inappropriate goal recognition results, the correct current context and concept can be selected in the lists in the lower right corner (displayed are only units containing the last performed/selected action). To switch between the presentation media, the 'Presentation Media' button can be used which allows additional flexibility of the system with respect to the user's preferences.

The two lower selection lists display the current low-level goals (contexts) and concepts of the current (or selected) working context. They enable the end-user to retrieve less detailed, or more global information on the current (or selected) action's low-level goal or concept. In order to provide a consistent view of the entities of all realized levels of human-computer dialogue, we installed additional toggle buttons to allow the user to limit displayed sub-entities with respect to their affiliation to selected entities.

In addition to these features of HyHN and if the action which the user wants to execute next is contained in the listed set of 'alternative actions', it can be executed by double-clicking the desired item. This feature is based on the neighbourhood information embedded in the action graphs and the corresponding statistically or probabilistically computed connection weights dependent on the evaluation strategy the user or system administrator chooses to use.

The selection of alternative actions makes an indirect interaction with the application possible and helps out of (well known) situations where the users know *what* to do, but do not know *how* to do it and *where* the appropriate button or menu item is located. Based on a simple callback mechanism, this feature does not impact the portability of the system in any way, yet provides an effective tool for user support.

5 Results

The system described above is completely implemented and integrated with two software applications, *Arcade* and the *MEDStation*. The first application, *Arcade*, is a CAD-modeller (Stork and Anderson, 1995) and is an application with broad functionality, thus being important for the integration of our user support system.

The second, the *MEDStation* (Grunert et al., 1995), is developed for *CT* and *x-ray* image evaluation and diagnostic, case study and documentation, and patient data management. Through this base functionality the application system is addressing a broad community of clinical clients. This emphasizes the need for a system offering adequate user support.

To get a measure of the usability and quality of our system, we let 20 test users solve three given problems with the software applications *MEDStation*. We evaluated their interactions and then asked for a rating of the offered user support. In the first stage we used the performed actions to determine the users' levels of expertise and compared them with the users' assessment of their own levels of knowledge, i.e. affiliation with a certain user class (which could either way be quite unprecise). In 80% of the cases the system's calculated experience class corresponded to the user's self assessment. In the other 20% ORIMUHS could not clearly assign the user to the correct knowledge class (although the user belonged more often than not to the correct expertise class).

The users rated ORIMUHS' usability indirectly. We attempted to find out, whether or not our system can help users to 'explore' and apply the broad functionality of the hosting software system. In this case 65% of the users (most of them *Beginner*s) stated that the *MEDStation* was not transparent enough in the first session (solving a given problem without using ORIMUHS), however being more satisfied with the *MEDStation*'s functionality at the end of the third session (with integrated intelligent user support). 35% of the users, most of which were *Advanced* or *Expert* users, found the offered user support satisfactory.

6 Conclusion

In this paper we presented a user support system which is a very useful companion for users with different application system knowledge.

Empirical user studies that were conducted within a medical application system strongly indicate that the provided user support was adapted to a satisfactory degree to all users; beginners, advanced, and even expert users. This in return gives us empirical evidence that the applied user modelling strategies produce results adequate enough, to proceed with our approach. Broader user studies will be needed to prove these initial results.

The cost-efficient integration with new and existing applications makes ORIMUHS even more appealing for commercial software products which often suffer from a lack of system transparency, a shortcoming that could be neutralized by exploiting ORIMUHS' adaptive user support.

Furthermore, we believe that our system is a useful platform for testing new methods from research areas like: user modelling, automatic help page generation, help-presentation, plan recognition, and goal prediction. Keeping this fact in mind and considering the effortless portability of ORIMUHS, seems to make the presented system interesting as a plug-in-like component for commercial software applications.

References

VEW AG. (1996). *Proc. of the 4. GI Workshop "Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen" (ABIS'96), Dortmund, Germany*.

Universität des Saarlandes. (1997). *Proc. of the 5. GI Workshop "Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen" (ABIS'97), Saarbrücken, Germany*.

Carr, B., and Goldstein, I. (1977). Overlays: A theory of modeling for computer aided instruction. Technical report, AI Memo 406, MIT.

Castells, P., Szekely, P., and Salcher, E. (1997). Declarative models of presentation. In *Moore et al. (1997)*, 137–144.

Chin, D. (1989). KNOME: Modeling what the User Knows in UC. In *Kobsa and Wahlster (1989)*. Springer-Verlag. 74–107.

Dieterich, H., Malinowski, U., Kühme, T., and Schneider-Hufschmidt, M. (1993). State of the art in adaptive user interfaces. In *Schneider-Hufschmidt et al. (1993)*. North-Holland. 13–48.

Encarnação, L. M., and Stoev, S. (1996). An overlay model for adaptive high-level user support in ORIMUHS. In *ABIS (1996)*.

Encarnação, L. M. (1995). Adaptivity in graphical user interfaces: An experimental framework. *Computers & Graphics* 19(6):873–884.

Encarnação, L. M. (1997). *Concept and realisation of intelligent user support in interactive graphics applications*. Ph.D. Dissertation, Eberhard-Karls-Universität Tübingen, Fakultät für Informatik.

Gray, W. D., Hefley, W. E., and Murray, D., eds. (1993). *Proc. of the 1993 International Workshop on Intelligent User Interfaces, Orlando, FL*. ACM Press.

Grunert, T., Fechter, J., Stuhldreier, G., Ehricke, H.-H., Skalej, M., Kolb, R., and Huppert, P. (1995). A PACS Workstation with integrated CASE tool and 3D-Endosonography application. In *Computer Assisted Radiology CAR 95*, 293–298. Springer.

Jameson, A. (1995). Numerical uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User–Adapted Interaction* 5(3-4):193–251.

Kobsa, A., and Wahlster, W., eds. (1989). *User Models in Dialog Systems*. New York: Springer-Verlag.

Lonczewski, F. (1997). Providing user support for interactive applications with FUSE. In *Moore et al. (1997)*, 253–254.

Marks, J., Birnbaum, L., Horvitz, E., Kurlander, D., Lieberman, H., and Roth, S. (1997). Compelling intelligent user interfaces: "how much ai is enough?". Panel in Moore et al. (1997) (position statements).

Maybury, M. T. (1993). *Intelligent Multimedia Interfaces*. AAAI Press / The MIT Press.

Moore, J., Edmonds, E., and Puerta, A., eds. (1997). Orlando, FL: ACM SIGART/SIGCHI.

Neal, J., and Shapiro, S. (1991). Intelligent multi-media interface technology. In *Sullivan and Tyler (1991)*. ACM Press. 11–43.

Schneider-Hufschmidt, M., Kühme, T., and Malinowski, U. (1993). *Adaptive User Interfaces: Principles and Practice*. North-Holland.

Stoev, S., and Encarnação, L. M. (1998). A navigation tool for ORIMUHS based on goal recognition and action prediction. In *Report 6. GI Workshop "Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen" (ABIS'98), Erlangen*. FORWISS.

Stork, A., and Anderson, B. (1995). 3D Interfaces in a Distributed Modelling Environment – 3D Devices, Interaction and Visualization Techniques. In Fellner, D., ed., *1995 Int. Workshop on Modeling, Virtual Worlds, and Distributed Graphics (MVD'95)*, 83–92. infix. URL: http://www.igd.fhg.de/~stork/-papers/mvd95/mvd95.html.

Sukaviriya, P., and Foley, J. D. (1990). Coupling a UI framework with automatic generation of context–sensitive animated help. *Proc. of the SIGGRAPH 1990 Symposium on User Interface Software and Technology (UIST'90)* 152–166.

Sullivan, J. W., and Tyler, S. W., eds. (1991). *Intelligent User Interfaces*. ACM Press.

Woods, D. D. (1993). The price of flexibility. In *Gray et al. (1993)*, 19–25.