ELSEVIER

Information and Software Technology 40 (1998) 93–103

INFORMATION AND SOFTWARE TECHNOLOGY

Active help found beneficial in wizard of oz study

J. S. Davis*

Department of Management, 101 Sirrine Hall, Clemson University, Clemson, South Carolina 29634, USA

Received 21 July 1997; received in revised form 29 December 1997; accepted 15 February 1998

Abstract

Emerging AI techniques will make possible new intelligent help features, such as 'active help' (interrupting the user when appropriate) and 'back channel' communication (allowing users to send a message to a help system). An investigation was conducted to determine whether these features would be useful. In the experiment, users of a job scheduling application liked active help and performed better with it than did people without it. However, having a back channel was not particularly valuable to participants. Since developing this feature requires a lot of work, one should find evidence that it will be worth the effort before proceeding. © 1998 Elsevier Science B.V.

Keywords: Active help; Passive help; On-line help; User model; Artificial intelligence

1. Introduction

Although a number of researchers have applied artificial intelligence to on-line help systems, it is not clear what the benefits would be to users. Before one undertakes research and development toward building an advanced intelligent help system, it would be useful to know what features would be most valuable to users.

This paper presents results of an investigation of the usefulness of two features that AI could provide: active (unsolicited) help and a back channel for user communication with a help system. The paper is organized as follows. First the paper reviews the literature on user modeling, because it is an important foundation for active help. Then it discusses key issues and prior work pertaining to active help, the back channel, and simulation of a help system (to evaluate software features before developing them). Finally, it describes an experiment to evaluate usefulness of active help and the back channel.

2. User modeling

System knowledge about the user provides a basis for customizing system responses to make them more relevant to user needs [17] and helps formulate effective advice. (Indeed, computer users in Hill's [32] experimental study failed to productively follow half of the advisory messages provided by a human, primarily because the human made wrong assumptions about the users' knowledge.) Therefore many researchers have included some type of user modeling in their prototype systems. Chin [14] has provided a classification of user modeling approaches. The model may be user-specified (the simplest scheme) or system-derived. (Examples of both kinds of systems will be mentioned later.) Some systems seek to know just the user's knowledge or capabilities. Others go further and try to determine the user's goals or tasks for the current session at the computer. Knowledge of user goals provides a basis for a system to provide specific guidance, that has been shown to be more effective for novices than general directions [11].

2.1. User-specified modeling

There are two ways for a system to gather information to build a user model: user-specified ('tell me about yourself') and system-derived ('I'll watch what you do and learn about you that way'). The former is the simpler approach. A system has the user tell it what it needs to know and then reacts accordingly. For example, Mozeico [47] developed a tutorial and help for a graphics package that provided four different levels of interaction, depending on the sophistication of the user. The user directly selected the proper level. In future systems it may be practical to have the user plug in a 'smart card' containing a user profile, if such cards become commonplace [48].

* Corresponding author. Address: Department of Management, P.O. Box 341305, 101 Sirrine Hall, Clemson University, Clemson, South Carolina, 29634-1305, USA. Tel.: 864-656-3768, fax: 864-656-2015, e-mail: davis@clemson.edu

PII S0950-5849(98)00030-5

2.2. System-derived modeling

Rather than relying on direct self-description by the user, a number of researchers have recognized the potential value of a help system that derives from the user's actions an awareness of the user's task or expertise and they have begun to lay a foundation of ideas and theory for developing such a system [21,55,57,66,67,70,72,74,76,80].

Inferring the user model may be preferable to asking the user directly for a description of his/her background and goals, not only to save the user the trouble but also to capture a more accurate picture. Allen and Perrault [1] and Wilensky et al. [77] explained that a human's goal must often be inferred from an 'indirect speech act'. In Pollack's [55] experiments, human experts responding to requests for help on how to perform a certain action in an e-mail system often did not respond with advice on how to achieve the requested action; instead, they provided an answer pertaining to an inferred user goal. Pollack suggested that the experts' responses were more effective that way. Other researchers agree that many systems would need to infer the model from user actions, rather than from a user's statement, especially if the model is to include the user's goals or intentions [61,74]. As Norcio and Stanley [49] and Chin [13] have pointed out, humans often state intentions indirectly, for example, 'Do you know the time'? A listener usually assumes the person's goal is to find out the time, and so responds with the time rather than saying 'yes'.

Most current systems that derive a user model do it by straightforward analysis of user input. Rich's [60] system employs stereotypes. It characterizes users and suggests novels they would like to read. User responses to system-generated questions were used to determine which stereotype was most appropriate. Greenberg and Witten [28] set up an adaptive on-line telephone directory that keeps track of how frequently numbers are retrieved by each user, and adjusts the access mechanism to make frequently accessed numbers easier to look up. Beeson [5] developed an expert system for people learning mathematics. It monitors student performance to detect which arithmetic operators are understood by the user, and then tailors its explanations by omitting steps associated with the known operators. Or-Bach and Bar-on's [50] system is similar, but as the student progresses it develops a hierarchy of concepts that the student understands, and assigns scores for the strength of understanding. Finin [24] designed a general user modeling system based on long term user interaction with various applications. Each application selects an initial user stereotype, updates that selection as it discovers new facts about the user, and feeds information to a centralized model maintenance system. Chin's [12] help system classifies UNIX commands by degree of difficulty. User mistakes with, or questions about, commands in certain classes are used to determine user expertise (one of four levels). Siuru [69] described a tutoring system for orbital mechanics that derived a model of the student (a level of knowledge) from a 'misconception analysis' of his/her interaction with the system. Vaubel and Gettys [73] found that expertise of users of word processors could be inferred by examining frequencies of commands and requests for on-line help. Kass and Finin [38] employed 'user model acquisition rules' to help determine which stereotype most resembles each user. In their application, those rules were determined from an analysis of 100 dialogues between a human expert and people seeking advice on financial investments. The tutoring system for programmers reported by Corbett et al. [16] and by Corbett and Anderson [15] considers the accuracy of student responses to determine the probability that a student has learned a certain rule. The system knows in advance the limited number of alternatives available to the student. It forms a rudimentary model of student misconception by tracking deviations from canned scripts.

Some techniques for deriving a user model are more sophisticated than just gathering statistics or comparing user actions to canned scripts. Selker's [68] COACH system was designed to mimic the kind of advice a coach provides. It adapts a user model by observing the user and employing frames, facts, rules and learning techniques. Benyon and Murray [6] designed an adaptive interface that develops a user model with attributes at task level (tasks completed), logical level (errors made, average task completion time), personal profile level (experience with certain interfaces, frequency of computer use, cognitive level, spatial ability), and 'student model' level (knowledge of the domain).

Bredeweg and Winkels [7] built a system that incorporates a qualitative reasoning approach to help determine user awareness or misconceptions. Their GARP system is based on Breuker and Wielinga's [8] framework for modeling problem-solving expertise, consisting of four components: strategic knowledge, task knowledge, inference knowledge, and domain knowledge, but it really addresses only the last two. During a session, the system uses built-in domain and inference knowledge to construct one or more models that are consistent with user behavior. If the system cannot construct such a model that is valid, a user mistake is evident. The user might lack a necessary item of domain knowledge, or might substitute an item that does not function correctly in the current context. The latter case constitutes a user misconception. GARP develops hypothetical models of the user's situation by selecting various items of domain knowledge from its inventory, and trying to assemble models whose predictions are the same as those of the user's situation. If one of the resulting models is missing an item of knowledge that would make it valid, then it represents a hypothesis that the user is lacking knowledge. If one of the resulting models could be derived from a valid model by substituting an item of knowledge, then it represents a hypothesis that the user has a misconception. Each hypothesis provides a basis for a message of advice that could be given to the user.

2.3. Determining user intentions

The various approaches to user modeling, reviewed above, provide a foundation for developing an intelligent help system but do not go far enough toward task awareness. To provide effective advice, a help system must know what the user is trying to do. According to Chin [13], the central problem in building an intelligent agent is how to determine an appropriate goal for the agent, that is based on the user goal. Some researchers have developed experimental systems that identify a user's goal and keep track of progress toward the goal by monitoring user actions. For example, Wolz and Kaiser [81] designed a two-model approach to determine what advice would be appropriate. The domain model is a hierarchy of goals. Each goal is linked to plans for satisfying it or to a function that achieves it. Employing a similar structure, the user model is based on monitoring user actions. It keeps a history of what goals the user has had and what plans have been used to accomplish them. The system can compare the two models to determine the appropriate advice content and format (introduce, remain, clarify, or elucidate). In addition to a 'domain model' and a 'user model' (like Wolz and Kaiser's), Benyon and Murray [6] recommend maintaining an 'interaction model' that manages the relationship between the other models. It would store details of the user–system dialogue and would be capable of inferring things about the user such as intentions.

Rauterberg [58] proposed representing a mental model of the user as a state transition matrix. Sequences of user actions correspond to transitions in the matrix. He developed software that detects user actions, constructs the matrix, and helps analyze the matrix. For example, by comparing it with predefined state transitions corresponding to correct solutions to a task, the software can measure complexity of behavior and can infer the user's level of expertise.

Jones et al.'s [37] and Jones and Mitchell's [36] systems determine user intentions by first constructing a domain-specific 'operator function model' that describes possible user tasks (Payne et al. [53] suggested how those tasks may be analyzed and represented). Then their system uses it while watching user actions to develop a user model on a blackboard. Both the operator function model and the blackboard are structured like Wolz and Kaiser's [81] models. Posting an item on the blackboard (linking an observed action to a task) is called 'intent referencing'.

Mitrovic and Djordjevic-Kajan [45] employed both explicit and implicit information about the user to construct a model. Explicit information comes from user answers to questions asked by the system. Implicit information comes from system observation of user actions, including requests for help, typing errors, and mistakes in using the system. The user model is constructed by a learning algorithm implemented with production rules.

Sadek [63] presented a formal theory of intention that could be employed in systems that develop user models. One type of intention is the intention to do certain actions, but she suggested that the most useful type of intention to model is the intention to achieve a certain state (goal). User actions can be employed as indicators of user intentions and of progress toward a goal.

The aforementioned research attests to the significant interest in system-derived modeling. Researchers have applied some of those user modeling concepts to experimental systems [12,42,46,56,61,77]. Most of those systems construct a user model from a detailed history of user responses. Future systems may take advantage of more subtle clues such as eye movements [71].

Developing a user model gives a help system a basis for providing more meaningful advice, especially in situations where the user needs help but has not asked for it.

3. Active help

Most current systems provide only passive help (when explicitly requested by the user). Reichman-Adar [59] pointed out the potential value of a system that could act as a smart assistant by interrupting the user when appropriate, providing what we call active help. Several experimental systems suggest that active help could be practical in the near future.

As part of a tutoring system, Derry et al. [20] classified typical user errors in solving mathematical word problems and wrote rules for active help, such as:

If *subject constructs bad schema but subgoal is good,* ***then*** *interrupt with following message 'You are trying to obtain (name of subgoal set), which is good. But you don't seem to know how to compute (name of subgoal set). Think about a combine relation in which (name of known set) is a part set and (name of subgoal set) is the whole set'.*

Winkels [78] designed EUROHELP, a shell for building intelligent help systems for interactive computer programs. It could aid in extending systems like the aforementioned GARP. Given a known or supposed intention of the user the system puts together a sequence of actions that fulfils that intention (taking into account the displayed user domain knowledge), and compares the generated plan with the perceived plan of the user. Differences can be interpreted as lack of knowledge or as a user misconception. That interpretation serves as a basis for providing active help.

Some intelligent tutoring systems compare user actions to predetermined scripts to determine where students are going wrong and thus provide a basis for active help. Hecking's [31] help system for UNIX examines user commands, detects apparent inefficiencies, and recommends a better approach. For example, if the user gives two successive commands to move files to the same directory, the system suggests how to do it in a single command. Similarly, Siuru's [69] system detects user misunderstandings and develops an explanation of why they erred. Prager et al.

[56] designed a help facility for a command-oriented system that derived the user goal and issued active help if an error is detected. Likewise, Senay [65] developed a help system for IBM VM/CMS based on a fuzzy command grammar. If the user issues an incorrect command, the system generates a number of possible interpretations for the user's consideration. Wang and Kushniruk's [75] UNIX Tutor diagnoses each user command and gives an appropriate response if it is incorrect—a primitive kind of active help. Jokinen's [35] cooperative interface to Yellow Pages accounted for context and pragmatics of a human–computer dialogue to help generate appropriate system responses. Owei and Higa [52] built an interface to a database that returns the system interpretation of each query in natural language to the user for validation. Mikulecky and Bodi [44] built active help for an editor. It compares user actions with stored scenarios to detect mistakes or inefficient technique. Hill and Johnson's [33] tutoring system detects when the student is at an impasse (the student's action has failed or cannot achieve its intended purpose in the current system state) and then provides advice. Thus, the active help system avoids annoying the student with unwanted interruptions (the student's progress has already been interrupted by the impasse). Other interesting systems have been reported [3,15,51].

The aforementioned research efforts suggest that developing a substantial active help facility may be achievable in the near future, but they do not tell how much such a system would benefit users. The work of other researchers has hinted that active help may be useful. Fischer et al. [25] have discussed and developed active and passive help systems for UNIX, but they did not report experiments to compare them. Their active system intervenes when it notices the user performing in an inefficient way, such as using more keystrokes than necessary for a command, or using several commands when one would do. Winkels and Brueker [79] evaluated a simulated active help ('Intelligent Computer Coach') for students performing physiotheraputic diagnosis. 80% of the users said that when they made a mistake they found the information provided by the help system to be clear and useful, but there was no control group. Students liked having an active-help-equipped geometry tutoring system in the classroom, built by Schofield et al. [64], but passive help was also available in the system and was not separately evaluated. Since we were unaware of any studies that compared usefulness of active to passive help on an empirical basis, we decided such a study would be worthwhile.

Any system capable of providing active help will probably have to have knowledge about the user goals. The system must know what the user is trying to do in order to provide effective advice. Therefore, in the experiment reported later in this paper, we employed a human-assisted system that was knowledgeable about the user goal. Testing a help system that has knowledge of user intentions was not unrealistic, because several of the research projects mentioned in the previous section show progress toward such a system.

4. Back channel to help system

Besides active help, one other AI application to help systems has gathered some support but has not been adequately evaluated. It could be worthwhile to investigate the usefulness of a natural language 'back channel' that allows users to send a message to the help system, as recommended by Gwei and Foxley [30]. Such communication could facilitate the help system's determination of the user goal, because users tend to volunteer that information. More than two-thirds of the user questions in Pilkington's [54] study provided information about user goals, as did additional non-question utterances. Without prompting, subjects in Hill's [32] experiment mentioned their goal in 90% of the messages requesting advice on the next step. At least in interpersonal communication, having a 'back-channel' for responses can improve the quality of information received, even if it is just a brief indication of how the listener is reacting to what is being said [39]. However a severely constrained back channel may not be entirely satisfactory in the context of online help. Users of a help system for UNIX mail that provided a limited, menu-based back channel complained about the difficulty of expressing their needs with the menus provided [54].

It only makes sense to explore the usefulness of a back channel if there is some chance of implementing one. Difficulties in developing a general purpose natural language processor are well known. Young et al. [82] found that the natural language document retrieval queries at a library often had problems such as errors in spelling, incorrect punctuation, and improper use of conjunctions. Similar grammar problems showed up in a test of a help system for people doing statistical analysis tasks [29]. Deciphering ambiguities can be difficult.

In spite of the difficulties, a natural language back channel applicable to a specific domain may be feasible in the near future. For example, Rumpel and Krost [62] pointed out that a natural language interface to a power system is achievable because the domain restricts the 'number of notions' that must be handled and it avoids ambiguity in terms. Guindon [29] noticed that users of an online help system tend to use more formal and less fragmentary language than when they know they are typing to another human. This tendency may make user responses easier to parse. Liddy et al. [40] employed a sub-language approach to develop an effective natural language processor for an expert system that processed user comments on life insurance applications. Burton [10] developed techniques for understanding free text user responses that allow for spelling errors and the use of abbreviations, acronyms, synonyms, antonyms, and synonymous multi-word terms. In light of progress like the aforementioned, it seems not too futuristic to explore a natural language back channel in a help system as we have done in the work reported here. Although a back channel has been used in several research projects [29,32,54], we are unaware of any that have really evaluated a back channel.

5. Simulating a help system

To determine how users like to be helped, one might try analyzing interaction between humans. Such studies have limited value, because human advice-giving often occurs in a different context than on-line intelligent help. Human advice is often given in an office in an off-line fashion, and the advice-giver cannot directly observe the advice-receiver's actions before and after getting help. It would be much more realistic to experiment with working versions of proposed help features, but a 'chicken or the egg' problem must be resolved: how can one evaluate intelligent help features before they exist? One promising, low cost alternative is to simulate intelligent help using a secret human consultant. Murray [48] and Greef and Breuker [27] considered this 'Wizard of Oz' (WOZ) technique valid for evaluating user interface features. It has been used in several research projects [29,32,54]. We employed that approach in the study reported here.

One of the challenges in setting up a WOZ experiment is to make a human-operated help system realistic [18]. In the research reported here, we hoped to investigate the value of a system that could be built with artificial intelligence technology in the near future. It is not realistic to expect a system with human capabilities, having common sense and knowledge spanning a broad domain. So a simulation by a human tends to have too much capability [26,41]. Johnstone et al. [34] compared behavior of users having a computer advisor vs. their behavior with a human advisor (in a WOZ situation). They noticed that there were more messages exchanged with the human advisor. They believed the additional messages were caused by the human advisor's use of cues for 'grounding' (establishing mutual understanding), that is beyond the research of artificial intelligence in the near future. Amalberti et al. [2] found similar differences in user behavior when talking with a computer vs. talking with a human. When talking with a computer they made fewer exchanges and offered fewer justifications for their results (because there was no grounding being achieved).

The tendency for human help providers to exercise too much of their talent makes it difficult to maintain consistency among subjects and to account separately for help on how to use the software and help on how to do the task. For practical reasons, WOZ requires one human advisor per subject session. If subjects participate simultaneously, precautions must be taken to achieve consistency among multiple human help providers. If subjects are handled one at a time, it is easier to control variation but still care must be taken to provide, in effect, the same help system to each subject.

Pilkington [54] compensated for the 'excessive human intelligence' problem by restricting what the human help-provider could see. While the user was working with the UNIX vi editor, the human help-provider had a display of the user's screen but the words were scrambled to prevent the help-provider from understanding the text that the user was editing and using this knowledge in responses. In our experiment, we attempted to constrain intelligence and to maintain consistency among subjects by preparing in advance a set of scripts containing responses for situations we could anticipate.

6. An experiment with help features

To gather insight on users' preferences for help, we wanted to explore a situation where users would likely need help. We decided to expose subjects to a typical but unfamiliar business application with minimal prior explanation. We selected a manufacturing scheduling system that represents a schedule as a Gantt chart [19]. It was written for the IBM PC in C under AIX with Motif. Jobs are represented as blocks on a time line. Each job is characterized by its setup time, processing time, due date, and costs for earliness and lateness. Several measures of schedule quality are continuously updated and displayed: total cost, tardiness, flow time and utilization. The user task involved rescheduling to minimize cost. The schedule can be modified by rearranging jobs using the mouse. Generally, schedule cost is reduced by placing jobs such that they finish as close as possible to their due dates. A sample screen display is shown in Fig. 1. In the lower part of the display, each job may be moved along its time line. In the time line, the beginning (left side) of the white space signifies the job release date and the end of the white space indicates its due date. The job symbols having two shades of color represent jobs having setup time in addition to processing time. A typical move is shown by the curved arrow near the middle of the display. That move would put job CN0004 as close as possible to its due date (without pushing the rightmost job, CN0006, to the right, making it tardy).

Four different versions of on-line help were employed, based on two levels each of 'type' and 'channel'. The type was active or passive. Active help was unsolicited; it was provided when the help system detected a user problem. Passive help was provided only on request (by the user pressing the F1 key). The channel was 'back channel' if the user had a means of composing English text messages to the help system, otherwise it was 'no back channel'.

6.1. Participants and procedure

Subjects were thirty-two graduate students enrolled in a Management Information Systems course. They participated voluntarily and were awarded extra credit. None of them had seen the scheduling system before. Subjects were randomly divided into four groups. Each group of eight people worked with one of the four versions of help: passive/no back channel, passive/back channel, active/no back channel, and active/back channel.

Before the exercise began, subjects were briefed on the exercise and provided with a one-page introduction to the

system. We explained that we were evaluating approaches to on-line help, and we described how the help system would work. Those who would be provided active help were told that an experimental help system might volunteer suggestions if it detected a user error. They were not told that the help system involved a human consultant. Because the exercise required a human consultant and special hardware, we handled only one subject at a time. When the subject sat down at the computer, the application and help system were already active. Subjects accomplished the task described on a handout, and at the end of the exercise completed a questionnaire. They were allowed up to 45 minutes for the task.

6.2. The help system

Users could request context-sensitive help by pressing the F1 key. Those having a back channel could also type a message to the help system in a pop-up window. Active help, if available, appeared in a pop-up window on the lower part of the screen and stayed on the screen after the first message. (The pop-up window was always in the window with current focus. That is the best place for it according to experiments by McLellan et al. [43]). Users could scroll that window to look at earlier messages. The help system was task-aware, so it could provide advice on how to do the task as well as on how to use the software. In this regard it was similar to those explored by several other researchers such as Wolz and Kaiser [81].

The help system was implemented by connecting two personal computers running X-windows. Subjects worked at one computer while the other, located in another room, was operated by a human consultant who simulated the intelligent help. The subjects' machine was connected to an extra monitor in the consultant's room to allow the consultant to observe the session. To save time in composing messages, to lend consistency, and to limit the amount of human intelligence applied by the help system, a number of help messages were composed in advance of the experiment. The pre-written messages were used whenever one was relevant. Also to lend consistency, the author trained two graduate students who operated the help system, and directly supervised their activity during each session. Active help, when available, was issued when the help-provider detected an error or when the user seemed to be at an impasse. An impase was indicated by a prolonged period of no activity or of non-productive activity. In both active and passive modes, the system helped resolve problems in doing the task and difficulties in using the system.

6.3. Task and typical user problems

The task required the user to manipulate jobs of two schedules to achieve for each schedule a cost of less than $300. User difficulties were of two types: problems in performing the task and problems in using the system. A typical type of task-related problem was moving jobs in the wrong way, so as to increase schedule cost. (Since jobs had

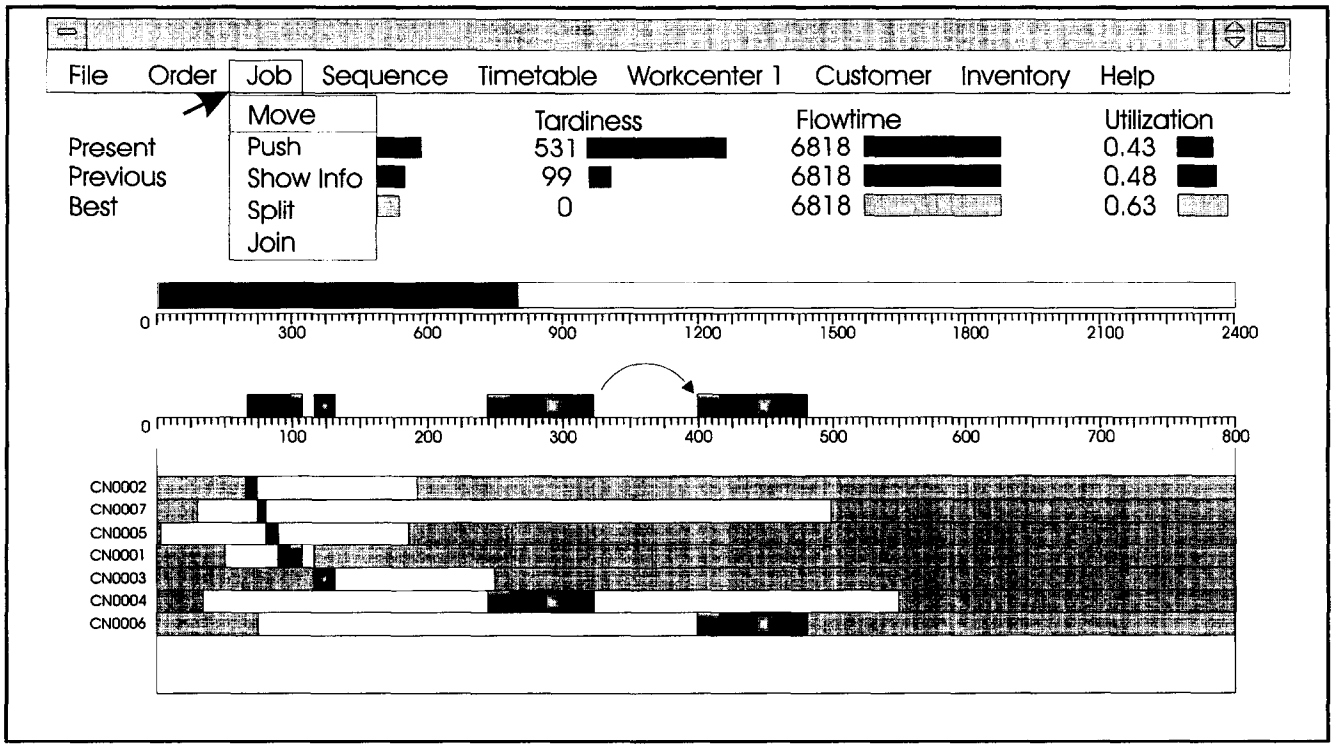

Fig. 1. Example of screen interaction (moving a job).

penalties for earliness and tardiness, the key to improving a schedule was to move jobs as close as possible to their due dates.) Here is a specific task-related problem and the associated help message. Problem: Moving a late job to make it even later, for no apparent reason. Help message: The jobs must be moved closer to their due dates, that are to the far right of each open space (white area).

A common system-related problem was attempting to move a job in the 'display only' part of the Gantt chart that shows a separate time line for each job (a job can be moved only along the single time line containing all jobs). Here is a specific system-related problem and the associated help message. Problem: Attempt to move a job from the upper 'all jobs' time line to the individual job time lines below. Help message: Move jobs along the upper 'all jobs' time line, not to the individual job time lines below.

6.4. Experimental results and discussion

To determine the significance of various differences among mean values obtained during this experiment, we employed the statistical method called analysis of variance. That method assumes that random samples are drawn from normally distributed populations with equal variances, but minor departures from these assumptions do not materially affect the results of the analysis. This experiment was a 2x2 factorial with 8 replications. These were 4 samples, each containing 8 observations and each defined by the type of help subjects used: passive/no back channel, passive/back channel, active/no back channel, and active/back channel.

The experimental procedure worked smoothly, although a few of the sessions were briefly interrupted by the application locking up. Since people equipped with the passive help system only received a help message when they requested it, one might expect that they would receive fewer messages than the active group. With our samples of computer users, the means for number of messages were 3.7 for passive and 6.9 for active help. We tested the null hypothesis, that there is no significant difference among the means, by calculating $F = 6.7$, the ratio of the variance between samples to the variance within samples, and then consulting a table of the F distribution. To use this table, we noted the number of degrees of freedom between groups is $2 - 1 = 1$. The number of degrees of freedom within each sample, each containing 8 observations, is $8 - 1 = 7$, and since there were 4 samples, the number of degrees of freedom within samples is $4*7 = 28$. We wrote $F(1,28) = 6.7$, consulted a table, and found that we could reject the null hypothesis at the 0.025 level of significance (written $p < 0.025$), meaning there is less than a 2.5% chance of getting an F ratio of 6.7 or larger if there really is no difference in the mean number of messages in the entire population of potential users of this computer system. Therefore, we rejected the null hypothesis and concluded that people equipped with the passive help system received significantly fewer messages than the active group. There was no significant interaction effect. For the group with passive help there were significantly fewer ($p < 0.05$) system-provided help messages concerning how to do the task (Table 1). That result is consistent with the interpretation that a task-aware active help facility tends to recognize more task-oriented problems (such that a help message is appropriate) than would a user who is involved in a new task.

The mean time to perform the task was 30 minutes. The range was 12 to 43 minutes for those who successfully completed the exercise. Five people did not accomplish the requirement to achieve schedules less than $300. Those participants were excluded from the statistical analysis for the completion time variable. The completion time includes time spent typing requests for help. This overall time is of primary interest and is the only time we measured. (If using a help facility reduces the time to actually perform a task but increases time overall, it is of little value.) However we agree with an anonymous reviewer that it could be valuable for research purposes to account separately for time spent using the help facility, and we shall do so in future work.

There was a significant difference by type of help (active or passive) in the time taken ($F(1,23) = 6.2$, $p < 0.025$) (we could reject the null hypothesis that there was no significant difference between means). Those with active help finished the tasks about 9 minutes faster as shown in Table 1. (This global effect of the time factor is about the same whether it is calculated as an equally weighted average of the two cell averages or as a weighted (by the frequencies of the cells) average of the two averages.) This finding is intuitive but not obvious without empirical evidence. Reasons why active help might not save time include: it may disrupt the user's productive train of thought, it may not be followed efficiently and effectively (as Hill [32] found) and it may not say what the user needs to know, as Doane et al. [22] found in many cases. There was no significant interaction effect between type and channel (back channel or no back channel), as shown in Fig. 2.

Having the back channel did not significantly reduce overall time to complete the task, and the interaction effect was not significant. We can't be sure why the back channel wasn't more beneficial. Possibly the actual task time was reduced if one excludes the time consumed by typing requests for help, that we did not measure. Or, perhaps the

Table 1
Mean scores with active and passive help

		Help Messages ($n = 32$)		
Type of Help	Minutes to Complete** ($n = 27$)	Total*	How to Use System	How to Do Task*
Active	24.4	6.9	4.4	2.5
Passive	33.2	3.7	2.6	1.1

* difference significant at $p < 0.05$
** difference significant at $p < 0.025$

back channel was not beneficial because there just weren't many situations during the exercise that made it particularly valuable to be able to compose a message to the help system. It will be interesting to seek an explanation in future research.

According to the questionnaire, those who had active help available found it very helpful. The mean response to the question 'How did you like the help system prompting for help without asking?' was 4.0 on a scale anchored at 0 for 'very annoying' and at 5 for 'very helpful'. The mean response to the question 'Did the unrequested help enable you to better perform the tasks?' was 0.9, where 1 represented 'yes' and 0 represented 'no'.

From questionnaire responses alone, we can't conclude that active help was preferable to passive. Subjects liked whichever help system they had, reinforcing the intuition that any type of help system is better than none at all. To the question 'How helpful were the help messages?' the active group responded 4.3 on a scale anchored at 0 for 'Not helpful at all' and at 5 for 'Very helpful', and the passive group responded 4.1. The interaction effect was not significant.

Placing our findings in context of previous studies is somewhat difficult because other studies were not intended to explore the same factors (active vs. passive, and back channel vs. no back channel). However, other research lends some support to our finding that active help improved task performance. Doane et al.'s [22] active/no back channel system provided a series of prewritten help messages, one at a time, until an error in typing a UNIX command was corrected.

On average, more people got correct commands as the number of help messages increased. Eberts et al. [23] compared user performance in composing UNIX commands under three conditions: active/no back channel help, off line help, and no help. Users of the active/no back channel system achieved a higher frequency of efficient UNIX commands. Our finding that users liked having passive help, even though it did not reduce the time to perform a task, is consistent with findings of other researchers. Subjects in Guindon's [29] test of a passive/back channel system apparently liked it since they used it extensively (although the author did not report a user evaluation). Users of Hill's [32] passive/back channel system followed advice effectively and efficiently in more than half the cases.

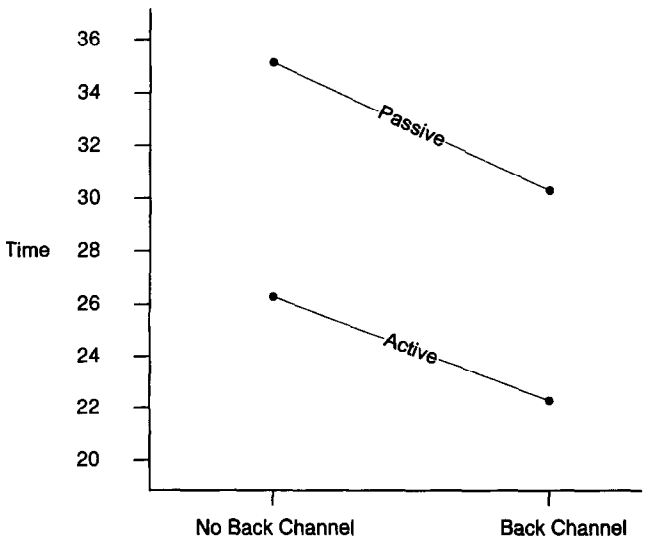

Fig. 2. Mean completion times as a function of type of help (active or passive) and channel (back channel or no back channel).

After the experiment, we asked on a questionnaire 'What suggestions do you have for improving the help system?' There were two main themes: improve the slow response time and add an attention-getting signal that a help message has been sent. Slow response was also the main complaint in the Wizard of Oz experiments of Brunner et al. [9] and in Pilkington's [54] experiments with a help system for UNIX mail. As in our experiment, subjects in Brunner's experiment communicated with a human advice-giver with written messages via computers. Below are some subjects' suggestions for improvement.I didn't [always] know that there was a help message for me. I think it would be better if it beeps before giving a message. [The active help messages appeared in a non-screen window, but might go unnoticed.]-Make help messages disappear when the operator returns to the task. [The small help window stayed on the screen.]I would have been more helped with interrupting messages. [This came from a subject in the Passive/No Back Channel group. It is an unsolicited expression of support foractive help.]

7. Concluding remarks

It seems worthwhile to pursue development of AI interfaces that provide active help. At least in this experiment, users liked active help and performed better with it than did people without it. If active help is provided, it should be accompanied by some signal to get the attention of the user. Also, it should be provided without delay, on a real-time basis.

Confidence in the value of active help would be increased by additional experiments that place realistic limits on the type of help provided by the human expert, as Andry et al. [4] and Pilkington [54] attempted to do. Since in our experiment some help messages were composed on the fly by the human expert, we cannot be sure that all of them were consistent with what we could reasonably expect from artificial intelligence. Also it would be interesting to explore separately the value of active help on 1) how to use the software, and 2) how to do the task. In our experiment both types of help were provided as needed and there is no way to compare the usefulness of each.

Our finding that a back channel was not particularly valuable does not support placing a high research priority on developing a facility for natural language user input in

help systems. Since this feature requires a lot of work to develop, before proceeding one should have some fairly strong evidence that it will be worth the effort. Our results are not conclusive, however. Experiments in other situations might make a case for the back channel. It would be interesting to evaluate a streamlined system with a back channel. In the experiment reported here, using the back channel incurred the effort and delay of typing. Of course, there was also some delay in the response. One way to cut down on the delays of typing and responding would be to allow spoken communication with the help system.

Acknowledgements

John Lewis and Patty Weaver contributed to this research during their participation in the National Science Foundation 'Research Experiences for Undergraduates' program, Grants No. CR-8804393 and IRI-8900354. Steve Cantrell and V. Sridharan provided helpful suggestions on an earlier version of this article. The Defense Logistics Agency supported development of the software used in the experiment. Chris Jarvis and Jack Peck supported this project as co-directors of Clemson Apparel Research.

References

[1] J. J. Allen, C. R. Perrault, Analyzing intention in utterances, Artificial Intelligence 15 (1980) 143–178.

[2] R. Amalberti, N. Carbonell, P. Falzon, User representations of computer systems in human–computer speech interaction, International J. of Man–Machine Studies 38 (1993) 47–56.

[3] J. R. Anderson, Intelligent tutoring and high school mathematics, in: C. Frasson, G. Gauthier, G. I. McCalla (Eds.), Intelligent Tutoring Systems, Springer–Verlag, Germany, 1992, pp. 1–10.

[4] F. Andry, E. Bilange, F. Charpentier, K. Choukri, M. Pnamale, S. Soundoplatoff, Computerised simulation tools for the design of an oral dialogue system, ESPRIT 90: Proceedings of the annual ESPRIT Conference, Brussels, 12–15 November 1990, pp. 342–354.

[5] M. J. Beeson, The user model in MATHPERT: an expert system for learning mathematics, artificial intelligence and education in: D. Bierman, J. Breuker, J. Sandberg (Eds.), Proceedings of the 4th. International Conference on AI and Education, IOS, Amsterdam, 24–26 May 1989, pp. 9–14.

[6] D. Benyon, D. Murray, Applying user modeling to human–computer interaction design, Artificial Intelligence Review 7 (1993) 199–225.

[7] B. Bredeweg, R. Winkels, Student modeling through qualitative reasoning, in: J. E. Greer, G. I. McCalla (Eds.), Proceedings of the NATO Advanced Research Workshop, St. Adele, Quebec, Canada, 4–8 May 1991, Springer–Verlag, Berlin, 1994, pp. 63–97.

[8] J. A. Breuker, B. J. Wielinga, Model driven knowledge acquisition, in: P. Buida, G. Tasso (Eds.), Topics in the Design of Expert Systems, North Holland, Amsterdam, 1989, pp. 265–296.

[9] H. Brunner, G. Whittemore, K. Ferrara, J. Hsu, An assessment of written/interactive dialogue for information retrieval applications, Human–Computer Interaction 7 (1992) 197–249.

[10] A. Burton, Type in your answer: free text input in computer based learning, International Journal of Computers in Adult Education and Training 5 (1995) 28–48.

[11] R. Catrambone, Specific versus general procedures in instructions, Human Computer Interaction 3 (1990) 49–93.

[12] D. N. Chin, Knome: Modeling what the user knows in UC, in: A. Kobsa, W. Wahlster (Eds.), User Models in Dialog Systems, Springer–Verlag, Berlin, 1989, pp. 74–107.

[13] D. N. Chin, Intelligent interfaces as agents, in: J. W. Sullivan, W. T. Sherman (Eds.), Intelligent User Interfaces, ACM Press, New York, 1991, pp. 177–206.

[14] D. N. Chin, Acquiring user models, Artificial Intelligence Review 7 (1993) 185–197.

[15] A. T. Corbett, J. R. Anderson, Student modeling and mastery learning in a computer-based programming tutor, in: C. Frasson, G. Gauthier, G. I. McCalla (Eds.), Intelligent Tutoring Systems, Springer–Verlag, Germany, 1992, pp. 413–420.

[16] A. T. Corbett, J. R. Anderson, E. G. Patterson, Student Modeling and Tutoring Flexibility in the Lisp Intelligent Tutoring System, in: C. Frasson, G. Gauthier, G. I. McCalla (Eds.), Intelligent Tutoring Systems: At the Crossroad of Artificial Intelligence and Education, Ablex Publishing Corporation, Norwood, 1990, pp. 83–106.

[17] B. Cox, Communicating conceptual integrity in distributed systems through intelligent assistance, OMEGA International Journal of Management Science 22 (1994) 113–122.

[18] N. Dahlbäck, A. Jönsson, L. Ahrenberg, Wizard of Oz studies—why and how, Knowledge-Based Systems 6 (1993) 258–266.

[19] J. S. Davis, J. J. Kanet, Production Scheduling: An Interactive Graphical Approach, The Journal of Systems and Software 38 (1997) 155–163.

[20] S. J. Derry, L. W. Hawkes, U. Ziegler, T. Diefenbach, Characterizing the problem solver: a system for on-line error detection, in: D. Bierman, J. Breuker, J. Sandberg (Eds.), Proceedings of the 4th. International Conference on AI and Education, IOS, Amsterdam, 4–26 May 1989, pp. 86–91.

[21] M. C. Desmarais, A. Maluf, User-expertise modeling with empirically derived + − probabilistic implication networks, User Modeling and User-Adapted Interaction 5 (1996) 283–315.

[22] S. M. Doane, D. S. McNamara, W. Kintsch, P. G. Polson, D. M. Clawson, Prompt comprehension in UNIX command production, Memory & Cognition 20 (1992) 327–343.

[23] R. Eberts, L. Villegas, C. Phillips, C. Eberts, Using neural net modeling for user assistance in HCI tasks, International Journal of Human–Computer Interaction 4 (1992) 59–77.

[24] T. W. Finin, GUMS: A general user modeling shell, in: A. Kobsa, W. Wahlster (Eds.), User models in Dialog Systems, Springer–Verlag, New York, 1989, pp. 411–430.

[25] G. Fischer, A. Lemke, T. Schwab, Knowledge-based help systems, in: L. Borman, B. Curtis (Eds.), Proceedings of CHI '85 Human Factors in Computer Systems, San Francisco, 14–17 April 1985, ACM Press, New York, pp. 161–167.

[26] N. Fraser, G. Gilbert, Simulating speech systems, Computer Speech and Language 5 (1991) 81–99.

[27] H. P. Greef, J. A. Breuker, Analysing system–user cooperation in KADS, Knowledge Acquisition 4 (1992) 89–108.

[28] S. Greenberg, L. H. Witten, Adaptive personalized interfaces—a question of viability, Behavior and Information Technology 4 (1985) 31–45.

[29] R. Guindon, Users request help from advisory systems with simple and restricted language: Effects of real-time constraints and limited shared context, Human–Computer Interaction 6 (1991) 47–75.

[30] G. M. Gwei, E. Foxley, Towards a consultative on-line help system, International Journal of Man–Machine Studies 32 (1990) 363–383.

[31] M. Hecking, How to use plan recognition to improve the abilities of the intelligent help system SINIX consultant, in: H. J. Bullinger, B. Shackel (Eds.), Human–Computer Interaction—INTERACT '87, Elsevier Science Publishers, North-Holland, 1987.

[32] W. H. Hill, A Wizard of Oz study of advice giving and following, Human–Computer Interaction 8 (1993) 57–81.

[33] R. W. Hill, W. L. Johnson, Situated plan attribution for intelligent tutoring, Proceedings of the Twelfth National Conference on Artificial Intelligence, Menlo Park, CA, 1994, pp. 499–505.

[34] A. Johnstone, U. Berry, T. Nguyen, There was a long pause: influencing turn-taking behavior in human–human and human–computer spoken dialogues, International Journal of Human–Computer Studies 42 (1995) 383–411.

[35] K. Jokinen, Reasoning about coherent and cooperative system responses, Proceedings of the Fourth European Natural Language Generation Workshop, Pisa, Italy, 28–30 April 1993, pp. 168–187.

[36] P. M. Jones, C. M. Mitchell, Model-based communicative acts, International Journal of Human–Computer Studies 1 (1994) 527–551.

[37] P. M. Jones, C. M. Mitchell, K. S. Rubin, Validation of intent inferencing by a model-based operator's associate, International Journal of Man–Machine Studies 33 (1992) 177–202.

[38] R. Kass, T. Finin, General user modeling: A facility to support intelligent interaction, in: J. W. Sullivan, W. T. Sherman (Eds.), Intelligent User Interfaces, ACM Press, New York, 1991, pp. 111–156.

[39] R. M. Krause, C. M. Garlock, P. D. Bricker, L. E. Mcmahon, The role of audible and visible back-channel responses in interpersonal communication, J. Pers. Soc. Psychol. 7 (1977) 523–529.

[40] E. D. Liddy, C. L. Jorgensen, E. E. Sibert, E. S. Yu, A sublanguage approach to natural language processing for an expert system, Information Processing and Management 29 (1993) 633–645.

[41] D. Maulsby, S. Greenberg, R. Mander, Prototyping an intelligent agent through Wizard of Oz, Proceedings of INTERCHI '93, pp. 277–284.

[42] K. R. Mckeown, M. Wish, K. Matthews, Tailoring explanations for the user, Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 1985, pp. 794–798.

[43] S. G. McLellan, A. W. Roesler, A. L. Elliot, The effect of advice message location on user performance, IEEE Transactions on Professional Communication 39 (1996) 43–48.

[44] P. Mikulecky, V. Bodi, An active artificial advisor, Proceedings of the Sixth International Conference on Artificial Intelligence and Information Control Systems of Robots, Slomenice, Slovakia, 12–16 September 1994, pp. 339–342.

[45] A. Mitrovic, S. Djordjevic-Kajan, Interactive reconstructive student modeling: a machine-learning approach, International Journal of Human–Computer Interaction 7 (1995) 385–401.

[46] I. Monarch, J. Carbonell, CoalSORT: A knowledge-based interface, IEEE Expert 2 (1987) 39–53.

[47] H. Mozeico, A human/computer interface to accommodate user learning stages, Communications of the ACM 25 (1982) 100–104.

[48] D. M. Murray, Modeling for adaptivity, Proceedings of the 8th. Interdisciplinary Workshop on Informatics and Psychology, Scharding, Austria, North Holland, Amsterdam, May 1989.

[49] A. F. Norcio, J. Stanley, Adaptive human–computer interfaces, Naval Research Laboratory Report 9148, September 30 1988.

[50] R. Or-Bach, E. Bar-On, PROBIT—Developing a diagnostic intelligent tutor, in: D. Bierman, J. Breuker, J. Sandberg (Eds.), Artificial Intelligence and Education, Proceedings of the 4th. International Conference on AI and Education, IOS, Amsterdam, 24–26 May 1989, pp. 185–192.

[51] R. Or-Bach, E. Bar-On, Why should an ITS bother with students' explanations?, in: C. Frasson, G. Gauthier, G. I. McCalla (Eds.), Intelligent Tutoring Systems, Springer–Verlag, Germany, 1992, pp. 372–381.

[52] V. Owei, K. Higa, A paradigm for natural language explanation of database queries: a semantic data model approach, Journal of Database Management 5 (1994) 18–30.

[53] D. Payne, M. Cohen, R. Pastore, Computer-based task representation: a methodology for improving system design, Interacting With Computers 4 (1992) 267–288.

[54] R. M. Pilkington, Question-answering for intelligent on-line help: The process of intelligent responding, Cognitive Science 16 (1992) 455–489.

[55] M. E. Pollack, Information sought and information provided: An empirical study of user/expert dialogues, Proceedings of CHI '85: Human Factors in Computing Systems, 1985, pp. 155–159.

[56] J. M. Prager, D. M. Lamberti, D. L. Gardner, S. R. Balzac, REASON: An intelligent user assistant for interactive environments, IBM Systems Journal 29 (1990) 141–164.

[57] E. L. Ragnemalm, Student diagnosis in practice; bridging a gap, User Modeling and User-Adapted Interaction 5 (1996) 93–116.

[58] M. Rauterberg, AMME: An automatic mental model evaluation to analyse user behaviour traced in a finite, discrete state space, Ergonomics 36 (1993) 1369–1380.

[59] R. Reichman-Adar, Communication paradigms for a window system, in: D. A. Norman, D. W. Draper (Eds.), User Centered System Design: New Perspectives in Human–Computer Interactions, Lawrence Erbaum Associates, Hlllsdale, NJ, 1986, pp. 285–313.

[60] E. Rich, User modeling via sterotypes, Cognitive Science 3 (1979) 329–354.

[61] E. Rich, Users are individuals: individualizing user models, International Journal of Man–Machine Studies 18 (1983) 199–214.

[62] D. Rumpel, G. Krost, Natural language interface and database issues in applying expert systems to power systems, Proceedings of the IEEE 8 (1992) 758–764.

[63] M. D. Sadek, A study in the logic of intention, principles of knowledge representation and reasoning: Proceedings of the 3rd. International Conference 'KR 92', 1992, pp. 462–473.

[64] J. W. Schofield, R. Eurich-Fulcer, C. L. Britt, Teachers, computer tutors and teaching: the artificial intelligent tutor as an agent for classroom change, American Educational Research Journal 3 (1994) 597–607.

[65] H. Senay, Fuzzy command grammars for intelligent interface design, IEEE Transactions of Systems, Man, and Cybernetics 22 (1992) 1124–1131.

[66] J. A. Self, Bypassing the intractable problem of student modeling, in: C. Frasson, G. Gauthier (Eds.), Intelligent Tutoring Systems: At the Crossroad of Artificial Intelligence and Education, Ablex Publishing Corporation, Norwood, 1990, pp. 107–123.

[67] J. A. Self, Cognitive diagnosis for tutoring systems, Proceedings of 10th. European Conference on Artificial Intelligence, Vienna, Austria, 3–7 August 1992, pp. 699–703.

[68] T. Selker, Cognitive adaptive computer help (COACH), in: D. Bierman, J. Breuker, J. Sandberg (Eds.), Proceedings of the 4th. International Conference on AI and Education, IOS, Amsterdam, 24–26 May 1989, pp. 245–251.

[69] W. D. Siuru Jr., Challenger: A domain-independent intelligent tutoring system, IEEE Expert 4 (1989) 77.

[70] P. Sukaviriya, Dynamic construction of animated help from application context, Proceedings of the ACM SIGGRAPH Symposium on User Interface Design, Banff, Alberta, Canada, 17–19 October 1988, pp. 190–202.

[71] P. Suppes, Three current tutoring systems and future needs, in: C. Frasson, G. Gauthier (Eds.), Intelligent Tutoring Systems: At the Crossroad of Artificial Intelligence and Education, Ablex Publishing Corporation, Norwood, 1990, pp. 251–265.

[72] L. Quinn, D. M. Russell, Intelligent interfaces: user models and planner, Proceedings of CHI'86: Human Factors in Computing Systems, Boston, MA, USA, 13–17 April 1986, pp. 314–320.

[73] K. P. Vaubel, C. F. Gettys, Inferring user expertise for adaptive interfaces, Human–Computer Interaction 3 (1990) 95–117.

[74] W. Wahlster, A. Kobsa, Dialogue-based user models, Proceedings of the IEEE 74 (1986) 948–960.

[75] H. Wang, A. Kushniruk, The UNIX tutor, in: C. Frasson, G. Gauthier, G. I. McCalla (Eds.), Intelligent Tutoring Systems, Springer–Verlag, Germany, 1992, pp. 317–324.

[76] G. I. Webb, M. Kuzmycz, Feature based modeling: a methodology for producing coherent, consistent, dynamically changing models of agents' competencies, User Modeling and User-Adapted Interaction 5 (1996) 117–150.

[77] R. Wilensky, Y. Arens, D. Chin, Talking to Unix in English: an overview of UC, Communications of the ACM 27 (1984) 574–593.

[78] R. Winkels, A new framework for describing and designing intelligent tutoring systems, in: V. Marik, O. Stephankova, Z. Zdrahal (Eds.), Artificial Intelligence in Higher Education, Lecture Notes in Computer Science, Vol. 451, Springer–Verlag, Berlin, 1990, pp. 230–243.

[79] R. Winkels, J. Breuker, Modeling expertise for educational purposes, in: C. Frasson, G. Gauthier, G. I. McCalla (Eds.), Proceedings of ITS-92, Montreal, Springer–Verlag, Berlin, 1992, pp. 633–642.

[80] U. Wolz, Providing opportunistic enrichment in customized on-line assistance, Proceedings of the 1993 International Workshop on Intelligent User Interfaces, Orlando, Florida, pp. 167–174.

[81] U. Wolz, G. E. Kaiser, A discourse-based consultant for interactive environments, Proceedings on Fourth Conference on Artificial Intelligence Applications, San Diego, CA, USA, 14–18 March 1988, pp. 23–33.

[82] C. W. Young, C. M. Eastman, R. L. Oakman, An analysis of ill-formed input in natural language queries to document retrieval systems, Information Processing and Management 27 (1991) 615–622.