Artificial Intelligence Review **14:** 5–22, 2000.
Stephen J. Hegner et al. (eds.), Intelligent help systems for UNIX.
© 2000 *Kluwer Academic Publishers. Printed in the Netherlands.*

An Intelligent Human-Computer Interface for Provision of On-Line Help

JENNIFER JERRAMS-SMITH
Department of Information Systems, University of Portsmouth, University House, Winston Churchill Avenue, Portsmouth, Hants, PO1 2UP, UK
(E-mail: jenny.jerrams-smith@port.ac.uk)

Abstract. Some user interfaces, such as that of Unix,[1] are difficult for novices to use, and this paper suggests a possible solution to such problems. The results of a study of Unix users enabled the development of a taxonomy of error types so that users' errors can be classified. This information is encapsulated as production rules within a knowledge base and forms the basis for the design and development of an intelligent interface to Unix. The prototype makes inferences about users' mental models and uses these to select appropriate tutorial advice. Performance of users of the prototype intelligent interface was compared with that of users of the usual Unix interface. The prototype users were found to make fewer errors, exhibit fewer misconceptions and take less time to complete a standard set of tasks.

Keywords: user models, intelligent interface, knowledge based system, adaptivity, Unix

1. General Interface Problems and Possible Solutions

Many reports indicate that computer applications are difficult to use and that users have insufficient expertise to benefit from their full range of functions (Hayes et al. 1981; Lang et al. 1981; Nickerson 1981; Hayes and Szekely 1983). More problematic, there is evidence that the application's response to an error may be confusing (Mishra et al. 1984) or even positively unhelpful and misleading, thus causing additional errors (Hanson et al. 1984).

In particular, the Unix operating system is seen as being difficult for novices to learn and use, as indicated by the interest in the provision of improvements such as COUSIN-Unix (Hayes and Szekely 1983), and the "natural language" interfaces of the Unix Computer Consultant (Douglass and Hegner 1982), the Unix Consultant (Wilensky et al. 1984), and the Unix File Orienter (Mishra et al. 1984).

Many of the current suggestions for remedying users' problems with interfaces focus on studying users (Ramsay and Attwood 1979; Hollnagel 1983; Norman 1983). Other guidelines recently suggested for interface development indicate that the interface should facilitate error handling but allow freedom of expression, and should be easy to learn but appealing to experienced users (Shneiderman 1979); should provide feedback and a sense of presence (DuBoulay et al. 1981; Ehrenreich 1981); should provide not just answers, but answers which help users to achieve their goals (James 1981); should protect users from their own errors (James 1981) both to prevent serious consequences and because errors made in the first few attempts at a new activity tend to become ingrained and should enable commands to be chosen by recognition rather than recall (Bailey 1982).

Associated with recent work on difficulties with interaction, there are numerous recommendations that an effective method for enabling users to make the best use of a system is by including a tutorial element within the system (James 1981; Tagg 1981; Relles and Price 1981; Bailey 1982; Scapin 1981), instead of (or in addition to) the provision of a stand-alone tutor such as the Unix tutor (Irgon and Martin 1984). Further, exploration-based learning is believed to be educationally preferable to that of instruction-based learning (Carroll 1984; Kamouri et al. 1986).

The author provides a solution to the problems users have with Unix and other applications. Unix was chosen as an example application because it is an operating system whose use is becoming more widespread, but which causes serious problems for novice users because of the enigmatic nature of its responses to their commands.

The solution involves a study of Unix users, leading to the design and development of an additional intelligent interface to Unix, which includes a knowledge based system (KBS) and which also incorporates the current recommendations for interface design and development within the interface, as outlined above. This interface includes a tutorial component and is able to make inferences about users' mental models so that inaccuracies can be remediated as soon as they are detected, which relates the work to that in the field of intelligent interfaces (Innocent 1982) and also to tutoring systems such as that by (Self 1974; Clancey 1986; Brown and Burton 1978; Burton and Brown 1979; Sleeman and Hendley 1979; Sleeman and Smith 1981; Finin 1983; and Johnson and Soloway 1984). In particular, the solution makes use of previous work on the acquisition of mental models by novices: Bayman and Mayer (1983) found that novices' mental models (of BASIC) are often inaccurate, while Sleeman (1983) proposed the concept of malrules (the wrong rules which children use when they learn algebra).

All user commands pass through the intelligent interface before they reach the application. This transparent interface monitors the user's interaction with the application, but users are unaware of it except when the interface has inferred that tutorial help is required or that the user's action could have harmful consequences. In this event, the interface immediately provides tutorial information at the moment when it is most needed and therefore of most value to the user. This enables "minimalist training", as advocated by Carroll (1984), so that users can work in experimental mode while at the same time it traps their errors and helps them to develop a correct mental model of the application.

For such a new type of interface there were few guidelines available for its development. Since possession of a correct mental model of the application cannot be assumed for novice users (Ramsay and Attwood 1979), it was decided to extend Norman's (1983) suggestion that some rules for the design of the interface for a specific application might be derived from analysis of human error. The author therefore investigated user errors in order to discover if they indicate what help users need, and where the user's mental model is inaccurate, and therefore what remedial tutoring is required.

A method was devised which enabled the development of a taxonomy of error types so that errors could be classified. Subsequent investigation indicated the misconceptions (inaccuracies in the mental model) which were associated with some of the errors, and hence the remediation requirements. An investigation of the differences between errors made by experts and novices was also made. Fewer of these mistakes were expected to be caused by inaccuracies within the expert user's mental model, since the accuracy of the mental model depends on the level of the user's expertise (Carey 1983).

The resultant information was encapsulated as production rules in the knowledge base within the intelligent interface.

2. An Empirical Study of Unix Users

2.1. *Method of study*

A study was made of 55 novice users of Unix. These were first year undergraduates in software engineering, who were familiar with other operating systems. Observations were made over a four week period by logging all input to the Unix shell and simultaneously collecting verbal protocols from the subjects.

2.2. *Data*

Figure 1 shows a sequence of commands given by one of the subjects. It also has the error categories into which each of the incorrect commands was subsequently classified.

Figure 2 provides typical examples of verbal protocols provided by the subjects.

Command	Error category (see Table 1)
elp	5
help	–
passwd	1
robins	10
help passwd	–
passwd	1
logot	5
logout	–
passwd	1
help passwd	–
⟨cat/manual/summary/passwd⟩	1
passwd	1
YYYYYYY	10
login	7
login	7
passpasswd	14
passwd help	7 + 2
help passwd	–
passwd (v)	7
passwd (u)	7
help passwd	–
passwd	1
aaaaaa	10
aaaaaaa	10
hoqrr	10
uoou	10
⟨cat/manual/summary/passwd⟩	1
login	7
cat/manual/summary/passwd	1
help passwd	–
⟨cat/manual/summary/passwd	1
why not	13
help goto	13
help passwd	–
help passwd	–
help login	–
/etc/passwd	13 + 11
⟨car/etc/passwd	1

Figure 1. Typical log of user commands their errors and categories.

2.3. *Error analysis and classification*

Error analysis was carried out by an expert user of Unix who was also experienced in teaching novices how to use Unix, so that a taxonomy could be developed and errors classified in terms of this taxonomy.

Errors were considered to be not only those actions which would produce a system error message, but also included commands which, although valid,

June 14, 1983. 15.31.
Just dabbling to see if I can do anything.
It would appear that @ doesn't work and removing underlines.
Don't know why that is.
Just put in my password.
Think I'll leave it and come back after I've looked at the literature.
June 15th, 1983. 13.23.
Objective today to try and understand what this thing does.
Since yesterday I've failed to do anything.
Actually understand some of the text handling . . . files and things.
Couldn't manage to write to terminal G.
Don't know why that was.
Well, so far we have achieved nothing again today.
So I'm leaving off for today.
June 16th, 1983. 13.32.
Well, I've now got half an hour on this machine.
We'll call it junk.txt (using the ned screen editor).
We appear to have – what about ˆF?
If we can find the right spot in our Unix papers we can actually do something.
Totally stuck on the literature.
Well, we can honestly say it's hopeless finding anything in this manual.
(Consults friends. One explains how to use the em line editor).
Well, I think I'll start again.
End of session three (note of despair or disgust in his voice).

Figure 2. Typical verbal protocols.

make inefficient use of the system or do not produce the intended result. Errors which were caused by misconceptions were differentiated from errors caused by mistyping.

2.4. *The meaning of errors – the differentiation between mistypes and misunderstanding*

Examples of command sequences which indicate inaccuracies in the user's mental model were derived from the error analysis.

Mistyping categories were also derived from the error analysis. The following gives additional guidelines to those provided by Damerau (1964) for the formulation of the rules indicating that mistyping has occurred. These were derived by analysis of the logs generated by the subjects' interaction with Unix. It is very important to detect mistyping and to differentiate it

Table 1. Errors and percentages for the first 2 weeks and the second 2 weeks

Error	Category	1st	%1st	2nd	%2nd	Total
1	Unable to give command	40	29	14	3	54
2	Failed try for help	17	12	39	9	46
3	Wrong mode or directory	7	5	18	4	25
4	Using previous knowledge	21	15	32	7	53
5	Mistype	20	14	77	18	97
6	Prevented	7	5	10	2	17
7	Misunderstanding	5	3	5	1	10
8	Inefficient use	6	4	55	13	61
9	Misread documents	5	3	11	2	16
10	Unknown	0	0	57	13	57
11	Misconception	0	0	45	10	45
12	Obscenities	0	0	2	0.5	2
13	Guesses	0	0	2	0.5	2
14	Loss of attention	0	0	26	6	26
15	Forgot own file name	0	0	3	0.5	3
16	Easier option available	0	0	8	1	8
17	Known to be wrong	0	0	9	2	9
18	Forgotten	0	0	3	0.5	3
19	Documentation errors	7	5	0	0	7

from misunderstanding so that experienced users in particular are not given unnecessary advice (Number of occurrences are shown in brackets).

1. Omission of one character – this includes the omission of space between arguments in the case of an experienced user. (12)
2. Substitution of another character – next one to the right or left on the QWERTY keyboard. (8)
3. Repetition of one character. (16)
4. Repetition of two characters. (2)
5. Inversion of two characters. (4)
6. Extra characters at the start – this could be caused by incomplete deletion of incorrect characters. (17)
7. Repetition of a previous character instead of the correct one – example nror for nroff. (4)

Command	Interpretation
em	does not know the !⟨shell command⟩
SC	facility when in em
em	
SC	
SC f > f2	does not understand >
lf f2	and pipes
rm f1	does not know that all
rm f2	files could be supported
rm f3	on one command line
cat f|lf	cat is redundant
create f	create is redundant since
em f	em automatically creates
SC f1	use of same command could indicate lack of
SC f2	knowledge of pattern match such as *
SC f3	

Figure 3. Typical command sequences indicating inaccuracy in the user's mental model. (SC = any Shell Command, em = Editor, lf = List File, create = Create File).

3. Development of an Intelligent Transparent Interface for Unix

Since this was an unusual application, the production of a prototype is an important part of the process of development and gave useful indications for the final design. The prototype, and the final version, were written in FRANZ LISP on a VAX 11/730. During prototype development the code remained uncompiled and underwent continual modification. The prototype contained an intelligent spell-checker in addition to the components for recognition and remediation of other types of errors.

3.1. *Rule based inferencing*

A knowledge-based component was developed for inclusion within the prototype interface, and was based on an expert system suggested by Winston and Horn (1981). The knowledge base consisted of a set of production rules, and the control mechanism allowed goal-directed search or pattern-directed search. The knowledge base consists of modules of knowledge in the form of production rule sets grouped according to function and accessed when required for solving particular parts of a problem. A production rule expresses knowledge in the form of a conditional part followed by some action. A simple example might be:

```
IF
(command has failed) (intended command has been deduced)
(deduced command = passwd) (user is at novice level)
THEN
(explain how to set the password)
```

The inference engine of the KBS uses facts deposited in the database to produce new facts or actions. It matches the conditionals of the rules with the facts in the database, and causes the required action when a match occurs. Thus deduction is carried out by forward chaining only. Forward chaining has previously been shown to be useful for modelling cognitive processes and for solving problems which require very broad but shallow knowledge.

An explanatory facility was developed, similar to that of MYCIN (Shortliffe 1976), which enabled the user to ask for justification: how a fact was derived or why a fact was required. The output to the user was in pseudo natural language, which was aided by coding in LISP so that chosen parts of a rule were directly displayed on the screen. Pseudo-probabilities (confidence factors) were provided which were similar to those of MYCIN.

It became clear from the experience of using the experimental KBS that it was more difficult than had been anticipated to decide on the values of confidence factors, and they were therefore omitted from later versions. As indicated above, it was also realised that the backward chaining mechanism (and therefore the "why" explanations) would also be un-necessary for this application and it was decide to omit them too.

The "how" explanations allow users to follow the reasoning strategy of the KBS and was used during the testing and modification stage of development to ensure that the interface correctly deduced the user's intentions.

3.2. *User models*

The user model consists of two components: the generalised or static model and the dynamic or specific model. The static (generalised) model is derived from error and protocol analysis (held within the rules of the knowledge base), from published information on the use of Unix, from command usage frequency information and from likely previous knowledge of other Operating Systems, such as Multics, CP/M or Tops-20 (held as a list of possible substitutes for each command).

It is well known that people will attempt to apply their previous knowledge when attempting a new task, and numerous examples are visible in the user logs. However, as the logs indicate, this can sometimes hinder the successful completion of the new task.

The Dynamic User Model includes the user's model of the system, the user's behaviour patterns, the user's expertise level and the trace of the user's behaviour for the current session. The record of previous activity is held on property lists: attributes for each input line include original command, deduced command, intended command, original arguments, and deduced arguments, intended arguments. Sequences which may indicate the occurrence of inefficient use are maintained, and the frequency of usage of commands and files is also recorded. In future versions, as part of the planning operation, the Dynamic User Model should also maintain possible sequences which might indicate a command which is likely to be used next, or a file which is currently being operated upon (example: edit file1 cat file1 lf file1).

3.3. *Pre-testing*

As soon as the prototype was complete and had been tested to eliminate all apparent errors in the code, pre-testing was carried out in order to design the test-evaluation stage of development. The pre-test subjects answered questions which helped to validate the rules of the knowledge base. The interface with the completed interpretation module was pre-tested on a small group of expert users of Unix (one research associate, two research students). They were not constrained to use any specific operations, but asked to use it in whatever way they liked. The explanations facility of the KBS was used as a development tool which enables the subject to ask how each deduction was made and work back through a chain of reasoning. The rules were shown to the user in a natural language version; a property list holds the English language description of each function. The subjects indicated whether they believed each deduction to be true or false.

Pre-testing revealed that additional rules and hypotheses were required. For example, if “cc fred” were given as input, “fred” is a valid file, but not a valid argument for “cc”. However “fred” would be a valid argument for “bc” or “wc” and this is discovered by the interface. The final hypothesis is that the input cannot be interpreted because the relevant rules state that it is only likely to be “cc” if no other command has matching arguments. Thus a rule is required which will choose between the two possibilities or offer both. Currently the interface attempts to choose the most likely and rules out all others.

The final pre-test was carried out using a standard sequence of operations which were completed by a single test subject (a computer expert but new to Unix). The tester followed a standard sequence of frequently used operations such as display a file on-screen, make a hard copy of a file, find which files are present in the current directory, delete a file, make a copy of a file, use the editor, use the text-formatter. By the end of this stage the LISP coding had been compiled in order to improve speed.

3.4. *Action of interface for the final pre-test*

The following is an ordered list of the actions carried out by the intelligent interface for the final pre-test.

1. Production rules and Unix data are read in.
2. Menu, file-list and prompt simulator are output.
3. A command line from the user is read in.
4. A scanner divides this string of characters into tokens (example: divides “fred&” into “fred” and “&”).
5. Each token is checked for the types it might fit (examples: username, string, filename, file in current directory, valid command, digits, switch).
6. Each token is checked to discover if it could be a mistyped command or a mistyped file in the current directory.
7. The arguments are matched against each of the possible commands (examples: ps = ps, pr, ls; cap = cmp, cat, cal; ll = ln, l, ls; cp = cc, cp, cmp, cd).
8. The production rules are used to determine the user’s intention. This results in a list of hypotheses about the user’s intentions.
9. The explanations of the KBS are shown to the subject to discover the validity of the rules. Subjects indicate the truth/falsity of rules for their situation.
10. The command is passed to Unix to show its response.
11. The subject is asked for comment on the Unix response.
12. A record is kept within the interface of user commands, and of the user’s response to the rules and to Unix.

3.5. *Modification of the rule base after pre-testing*

New rules were easily added to the rule set. Such rules were formulated after pre-testing on the basis of information provided by the user in response to the deductions offered by the interface. Rules were also modified and added after consultation with users during the full test. Ideally, the new rules would be added by the interface, but there was insufficient time to automate this activity.

3.6. *Evaluation of the interface*

Although usability studies were not in common use when this research was conducted, the transparent interface was subsequently evaluated by tests with users in order to discover how effective it is for Unix and also to discover which aspects might be generally applicable to other systems which are difficult for novices to learn and use.

For this between-subjects study; 13 joint honours undergraduates in Mathematics and Computer Science were divided randomly into an experimental group (7 subjects) and a control group (6 subjects). Each subject in the experimental group was provided with a version of the intelligent interface, which was removed as soon as the study was complete. Testing involved completing a standard sequence of operations to be attempted in a given order and involving frequently-used Unix commands, an improved version of that used for the pre-test since, for instance, the pre-test indicated that the operation to find on-line help should be successfully completed before any other command is attempted.

Logs were recorded for subjects and in addition a record was made of all the analyses and interpretations which the intelligent interface carried out for each input line plus the responses made by the subjects to the questions provided by the tutoring module.

The following summarises the results. Further details are in Jerrams-Smith (1989).

1. Fewer mistakes were made by the experimental group.
2. The control group made many more repeats of identical errors.
3. Most members of the experimental group completed the sequence; most of the control group did not.
4. Fewer commands were given by members of the experimental group.
5. More misconceptions were found in the control group than in the experimental group.

Protocol analysis was carried out in which the subjects attempted to describe their intentions and activities while they carried out the standard sequence, and this was recorded on tape. However, in most cases there was a greater emphasis on debriefing because many users found it extremely difficult to talk about what they were doing while also trying to learn new and complex tasks. This proved to be a fruitful exercise because it indicated important aspects of the intelligent interface which were not otherwise obvious. These included:

1. The directory listing provided as part of the intelligent interface provided feedback: it was used to verify that commands had acted as expected.

2. There was evidence of a wrong mental model in use, which was not detectable from the log: one subject thought that files had to be created before using the ‘mv’ or ‘cp’ commands.
3. There was evidence that the interface corrected misconceptions about Unix as shown in the following extract from the protocol transcript:

   “Edited ‘test3’ using vi editor and now using the spell command to check for spelling mistakes in ‘test3’. (pause) 5 spelling mistakes recorded and these are going to be put into a file ‘test.sp’ using piping function. (pause) Pipe can’t be used since I’m trying to put it into a file so I’ll use the greater-than sign. (pause)
   Fine. That works.”

4. It was possible to gain some idea of the subjects’ impression of the interface in action, as indicated in the following:

   questioner: “Did you actually get anything out of the tutorial comments that you got back? Can you make a comparison between the INFO information and the tutorial comments?”
   subject 1: “Yes. The tutorial – it aims at the problem more closely. The INFO is general information, but the tutorial has found out where you’ve got a problem and actually aims for that – that’s pretty useful.”

The results indicate that members of the control group gave many more commands and showed more misconceptions than the members of the experimental group. The mode of activity may be different for learners where help is not easily available. They may adopt a more experimental approach and hence use many more commands to effect the same results. There is an indication that misconceptions are corrected by the intelligent interface as soon as they appear and so are not repeated, as tends to occur in the control group.

4. Recent Developments

Research on adaptive (intelligent) interfaces, which respond to meet the needs of the individual user, is still in progress (Jerrams-Smith 1989), and a good case has recently been made for the provision of various kinds of adaptive system (Benyon and Murray 1993; Elsom-Cook 1993). The essential component of such adaptive systems is a user model (also referred to within tutoring systems as a student model) which stores the system’s current beliefs about the attributes of a specific user of the system.

In addition, recent research has identified a number of potential problems endemic to hypermedia which could ultimately restrict its usefulness but which might be solved by the provision of adaptivity. For instance, Waterworth (1992) suggests that there remain significant unresolved usability issues, while Whalley (1993) discusses a potential problem associated with hypertext when used within an educational context: the fragmented nature of a hypertext document means that the most natural way to study hypertext is by browsing or exploring the various paths of the document. In some instances, however, the browsing activity may be an inappropriate method of learning, for example, when an author is trying to develop a series of ideas within a particular context or framework.

Further problems which need to be addressed were identified by Conklin (1987), who describes two main difficulties associated with reading hypertext documents (which apply equally to hypermedia): disorientation and cognitive overhead. Disorientation or 'getting lost in space' can occur if the user is uncertain as to his/her location on the hypertext network. Arguably this problem could also exist in traditional linear text documents, however, in this case the reader is limited to searching either earlier or later in the text. Because hypertext offers more dimensions in which the user can move, the likelihood of a user becoming lost is increased, especially in a large network.

The problem of cognitive overhead occurs when the user is presented with a large number of choices about which links to follow. These directional decisions are absent in a traditional linear text document (or in a film or TV programme) when the author has already made the choices for ordering of material. With hypertext (and also with hypermedia), the moment a link is encountered, the reader must decide whether or not to follow the link. If many links are encountered the reader may become distracted, leading to what Conklin (1987) refers to as information 'myopia'.

Adaptive systems may provide a solution to such problems and the current interest in intelligent and adaptive multimedia/hypermedia systems is indicated by recent work which involves an investigation into the development of intelligent multimedia systems, such as Jerrams-Smith (1991), Bocker et al. (1990), Hendley et al. (1993), Edwards et al. (1995) and Perez et al. (1995).

The author is currently engaged in a research programme developing a number of adaptive hypermedia systems in which a flexible and detailed user model is constructed and applied. These systems include:

- HyperLearner (an adaptive multimedia authoring/tutoring system; see 4.1 below for details)
- Telecare Companion (a prototype adaptive hypermedia system which is part of the British Telecom 'Telecare' programme; see 4.2 below for details)
- Adaptive Hypermedia Memory Remediator (for use with persons with various memory disorders)

- Adaptive Navigation Guidance for the World Wide Web; see 4.3 below for details.

These systems provide vehicles with which to identify the variables responsible for individual user differences and to adapt the system to accommodate user requirements in terms of attributes such as level of domain knowledge, personality, preferences, information processing styles, goals and tasks, roles within an organisation. Work currently in progress indicates that the following variables are important in enhancing the usability of the system.

A variable which appears to be fundamental to adaptivity is Field Dependence-Independence (FD-I) (Witkin et al. 1972). FD-I refers to a fundamental individual difference in information processing. The field dependent user is easily influenced by information in his/her environment and tends to incorporate environmental information non-discriminately. In contrast, the field independent user tends to be influenced by internally generated cues and is more discriminating in the use of environmental information. Adaptive systems could take advantage of user differences in cognitive style to present information in a form consistent with the user's field dependency. Such design features could have value for tutoring applications because field-dependent students' cognitive style appears to be less suited to hypermedia instructional systems than does that of field-independent students because they are less able to create a framework with which to connect the information (Witkin et al. 1977).

A further variable which may be of importance to adaptivity is that of Locus of control (Rotter 1966). Two types of user can be distinguished: internals and externals. Those with an internal locus of control regard outcomes as the result of their own efforts, whereas those with an external locus of control regard outcomes as the result of factors beyond their influence. While a system which removes control from the user may be compatible with an external user, its usability is reduced for internal locus users. Indeed, when internals are required to use restrictive systems they often ignore instructions and some ultimately cease to use the system. The adaptive systems currently in development will identify the user's locus of control using a brief modified version of the I-E scale (Rotter 1966) and subsequently modify the interactive mode to accommodate the user's locus of control.

A variety of further user differences are being investigated as the basis of adaptivity. A more sophisticated form of adaptivity is also to be investigated: the identification of variables of particular relevance to the user and variables which are irrelevant to the user. In this way the system can focus on the factors which are important in the system's usability and ignore those which are not.

4.1. *The HyperLearner project*

Combining hypermedia technology with adaptive tutoring provides a tutoring system which can adapt the sequencing of material and can also adapt the presentation of material to suit the potentially diverse needs and abilities of individual users. In addition, adaptive hypermedia tutoring systems are well suited to an exploratory approach to learning, and therefore encourage the active learning which has long been advocated (Piaget 1973).

The author has recently developed HyperLearner, a prototype hypermedia authoring system which has been used to help tutors to build tutorials of course material. The first phase of the project investigated the issues involved in the development of a framework/authoring system for adaptive hypermedia tutoring systems. The long-term aim is the delivery of such a system, incorporating the results of the ongoing investigations. Current trends indicate that in the future, working, and learning, are likely to become increasingly home-based and the provision of systems which enable effective distance learning is therefore a topic of considerable importance.

The HyperLearner prototypes learn about the student user and therefore adapt the interaction to suit the individual. The aim of the HyperLearner project is primarily to help the tutor, and thus to help the student to learn about a specific domain, but the prototypes have also been used to help students to learn about some of the fundamental issues connected with intelligent hypermedia tutoring systems (Jerrams-Smith 1995).

4.2. *The Telecare Companion*

The Telecare Companion prototype (John et al. 1998) developed for British Telecom supports the provision of Telecare (telecommunications for disabled and elderly people) by applying adaptive features identified during the HyperLearner project. The Telecare system is designed to form a community network that will provide primary care to the elderly, housebound, disabled or otherwise disadvantaged living in the community. The Telecare Companion investigates the provision of adaptive support for users to access information stored in hypermedia networks, including the Internet, and communicate using a video-phone and e-mail.

4.3. *Adaptive Navigation Guidance for the World Wide Web*

The Computer Aided Internet Navigation project (CAIN) (Lamas et al. 1996) addresses the problem of disorientation, cognitive overhead and information overload. It provides weak hypertext linearisation so that users follow an individual and ordered sequence of selected web pages. It combines techniques of user modelling with data modelling in order to select and order pages.

In conclusion, the systems currently under development are providing important insights into the design and production of adaptive hypermedia systems. While a fully adaptive hypermedia system has yet to be developed, the present research programme has already made important inroads into the design and construction of such a system. It should be stressed that the present research is ongoing and the author anticipates that more sophisticated systems will emerge.

Note

[1] Unix is a trademark of X/Open, Inc.

References

Bailey, R. W. (1982). *Human Performance Engineering: A Guide for System Designers*.
Bayman, P. & Mayer, R. E. (1983). A Diagnosis of Beginning Programmers' Misconceptions of BASIC Programming Statements. *Communications of the ACM* **26**(9): 677–679.
Benyon, D. & Murray, D. (1993). Applying User Modelling to Human-Computer Interaction Design. *Artificial Intelligence Review* **7**: 199–225.
Bocker, H-D., Hohl, H. & Schwab, T. (1990). HYPADAPTER – Individualizing Hypertext. In Diaper, D. (ed.) *Human-Computer Interaction – INTERACT '90*, 931–936. B.V. (North Holland).
Brown, J. S. & Burton, P. R. (1978). A Paradigmatic Example of an Artificially Intelligent Instruction System. *International Journal of Man-Machine Studies* **10**: 323–339.
Burton, R. R. & Brown, S. (1979). An Investigation of Computer Coaching for Informal Learning Activities. *International Journal of Man-Machine Studies* **11**: 5–24.
Carey, T. (1983). User Differences in Interface Design. *IEEE Computer* **15**: 125–129.
Carroll, J. M. (1984). Minimalist Training. *Datamation* **30**(18): 125 et seq.
Clancey, W. J. (1986). *Qualitative Student Models*, 86–15. Stanford Knowledge Systems Laboratory: CA.
Conklin, J. (1987). Hypertext: An Introduction and Survey. *Computer* **20**(9): 17–41.
Damerau, F. J. (1964). A Technique for Computer Detection and Correction of Spelling Errors. *Communications of the Association for Computing Machinery* **7**(3): 254–258.
Douglass, R. J. & Hegner, S. J. (1982). An Expert Consultant for the Unix Operating System: Bridging the Gap Between the User and Command Language Semantics. *Proceedings of the 4th Conference of Canadian Society for Computational Studies of Intelligence*, 119–127.
DuBoulay, B., O'Shea, T. & Monk, T. (1981). The Black Box Inside the Glass Box: Presenting Computing Concepts to Novices. *International Journal of Man-Machine Studies* **14**(3): 237–249.
Edwards, M., Powell, H. & Palmer-Brown, D. (1995). A Hypermedia-based Tutoring and Knowledge Engineering System. *In Proceedings of Educational Multimedia and Hypermedia 1995*. Carlottesville, VA: Association for the Advancement of Computing in Education (AACE).
Ehrenreich, S. L. (1981). Query Languages: Design Recommendations Derived from Human Factors Literature. *Human Factors* **23**: 241–247.

Elsom-Cook, M. (1993). Student Modelling in Intelligent Tutoring Systems. *Artificial Intelligence Review* **7**: 227–240.

Finin, T. W. (1983). Providing Help and Advice in Task Oriented Systems. In Proceedings of *The Eighth International Joint Conference on Artificial Intelligence*, 176–178. Karlsruhe, FRG.

Hanson, S. J., Kraut, R. E. & Farber, J. M. (1984). Interface Design and Multivariate Analysis of Unix Command Use. *Association for Computing Machinery Transactions on Office Information Systems* **2**: 42–57.

Hayes, P. J., Ball, J. E. & Ready, R. (1981). Breaking the Man-Machine Communication Barrier. *Institute of Electrical and Electronic Engineers Computer* **14**: 19–30.

Hayes, P. J. & Szekely, P. A. (1983) Graceful Interaction Through the COUSIN Interface. *International Journal of Man-Machine Studies* **19**(3): 285–305.

Hendley, R. J., Whittington, C. D. & Juraschek, N. (1993). Hypermedia Generation from Domain Representation. *Computer Education* **20**(1): 127–132.

Hollnagel, E. (1983) What We Do Not Know About Man-Machine Systems. *International Journal of Man-Machine Studies* **18**(2): 135–143.

Innocent, P. R. (1982). Towards Self-Adaptive Interface Systems. *International Journal of Man-Machine Studies* **16**(3): 287–299.

Irgon, A. E. & Martin, J. C. (1984). CLYDE: A Unix tutor. *In Proceedings of the US–Japan Conference on Human Computer Interaction*. Hawaii.

James, E. B. (1981). The User Interface: How We May Compute. In Coombs, M. J. & Alty, J. L. (eds.) *Computing Skills and the User Interface*.

Jerrams-Smith, J. (1989). An attempt to Incorporate Expertise About Users into an Intelligent Interface for Unix. *International Journal of Man-Machine Studies* **31**: 269–292.

Jerrams-Smith, J. (1991). Report on the PIE Project (Personalised Interaction for End-Users). *Confidential Report, Philips Research Laboratories* (UK).

Jerrams-Smith, J. (1995). Combining Multimedia, Hypermedia and Artificial Intelligence to Support four Aspects of Learning. In Proceedings of *Educational Multimedia and Hypermedia*. Charlottesville, VA, USA: Association for the Advancement of Computing in Education.

John, D., Jerrams-Smith, J., Heathcote, D. & Boucouvalas, A. (1998). The Telecare Companion – an Adaptive Interface for Telemedicine. *Proceedings of the 1st International Symposium on Communication Systems*. Sheffield Hallam University.

Johnson, W. L. & Soloway, E. (1984). Intention-Based Diagnosis of Programming Errors. In Proceedings of *The National Conference on Artificial Intelligence*, 162–168. Austin, TX: AAAI Press.

Kamouri, A. L., Kamouri, J. & Smith, K. H. (1986). Training by Exploration: Facilitating Procedural Knowledge Through Analogical Reasoning. *International Journal of Man-Machine Studies* **24**: 171–192.

Lang, T., Lang, K. & Auld, R. (1981). A Longitudinal-Study of Computer-User Behavior in a Batch Environment. *International Journal of Man-Machine Studies* **14**(3): 251–268.

Lamas, D. R., Jerrams-Smith, J. & Gouveia, F. R. (1996). Computer Aided Information Navigation: Project Description. *Proceedings of Webnet '96*.

Mishra, P., Trojan, B., Burke, R. & Douglass, S. A. (1984). A Quasi-Natural Language Interface for Unix. In Salvendy, G. (ed.) *Human Computer Interaction.*

Nickerson, R. S. (1981). Why Interactive Computer Systems Are Sometimes Not Used by People Who Might Benefit from Them. *International Journal of Man-Machine Studies* **15**: 469–483.

Norman, D. A. (1983). Design Rules Based on Analyses of Human Error. *Communications of the Association for Computing Machinery* **26**: 254–258.

Perez, T. A., Lopisteguy, P., Gutierrez, J. & Usandizaga, I. (1995). HyperTutor: From Hypermedia to Intelligent Adaptive Hypermedia. In *Proceedings of Educational Multimedia and Hypermedia*. Charlottesville, VA: Association for the Advancement of Computing in Education (AACE).

Piaget, J. (1973). *Memory and Intelligence*. Rutledge and Kegan Paul: Cambridge, MA.

Ramsay, H. R. & Attwood, M. E. (1979). SAI-79-111-DEN (NTIS:ADA 075 679), *Human Factors in Computer Systems: A Review of the Literature*. Science Applications, Inc.: Englewood.

Relles, N. & Price, L. A. (1981). A User Interface for Online Assistance. In Proceedings of *The Fifth International Conference on Software Engineering*. San Diego, California: IEEE Computer Society.

Rotter, J. B. (1966). Generalised Expectancies for Internal versus External Control of Reinforcement. *Psychological Monographs* **80**(609).

Scapin, D. L. (1981). Computer Commands in Restricted Natural Language: Some Aspects of Memory and Experience. *Human Factors* **23**(3): 365–375.

Self, J. A. (1974). Student Models in Computer Aided Instruction. *International Journal of Man-Machine Studies* **6**: 261–176.

Shneiderman, B. (1979). Human Factors Experiments in Designing Interactive Systems. *Institute of Electrical and Electronics Engineers Computer* **12**: 9–19.

Shortliffe, E. H. (1976). *Computer-Based Medical Consultations: MYCIN*. Elsevier: New York, NY.

Sleeman, D. H. (1983). Intelligent Tutoring Systems and Student Modelling. Presented at *AISB Conference on AI and Education*. Exeter, UK.

Sleeman, D. H. & Hendley, R. J. (1979). ACE: A System Which Analyses Complex Explanations. *International Journal of Man-Machine Studies* **11**.

Sleeman D. H. & Smith, M. J. (1981). Modelling Students' Problem Solving. *Artificial Intelligence* **16**: 171–188.

Tagg, S. K. (1981). The User Interface of the Data-Analysis Package: Some Lines of Development. *International Journal of Man-Machine Studies* **14**(3): 297–316.

Waterworth, J. A. (1992). *Multimedia Interaction with Computers*. Ellis Horwood.

Whalley, P. (1993). An Alternative Rhetoric for Hypertext. In McKnight, C., Dillon, A. & Richardson, J. (eds.) *Hypertext: A Psychological Perspective*. Ellis Horwood.

Wilensky, R., Arens, Y. & Chin, D. (1984). Talking to Unix in English: an Overview of UC. *Communications of the ACM* **27**(6): 574–593.

Winston, P. H. & Horn, B. K. P. (1981). *LISP*. Addison-Wesley: Reading, MA.

Witkin, H. A., Lewis, H. B., Hertzman, M., Machover, K., Meissner, P. B. & Wapner, S. (1972). *Personality Through Perception: An Experimental and Clinical Study*. Greenwood Press.

Witkin, H. A., Moore, C. A., Goodenough, D. R. & Cox, P. W. (1977). Field-Dependent and Field-Independent Cognitive Styles and Their Educational Implications. *Review of Educational Research* **47**: 1–64.