Action Planning: The Role of Prompts in UNIX Command Production

Stephanie M. Doane, Danielle S. McNamara, Walter Kintsch, Peter G. Polson, R. Gary Dungca, Deborah M. Clawson

Institute of Cognitive Science and Department of Psychology
Campus Box 345
University of Colorado, Boulder, CO 80309-0345
sdoane@boulder.colorado.edu

Abstract

Our goal is to provide empirical support for assumptions of the Doane, Kintsch, & Polson (1989;1990) construction-integration model for generating complex commands in UNIX. In so doing we designed a methodology that may be used to examine the assumptions of other cognitive models. The planning task studied was the generation of complex sequences of UNIX commands. The sequences were novel, and as such could not be recalled from memory. We asked users whose UNIX experience varied to produce complex UNIX commands, and then provided help prompts when the commands they produced were erroneous. The help prompts were designed to assist the subjects with both knowledge and processes which our UNIX modeling efforts have suggested were lacking in less expert users. There are two major findings. First, it appears that experts respond to different prompts than do novices. Expert performance is helped by the presentation of abstract information, while novice and intermediate performance is modified by presentation of concrete information. Second, while presentation of specific prompts aids the less expert, it does not appear to be sufficient information to obtain optimal performance. To do this, the less expert subjects require information about the ordering of the items in a command. Our analyses suggest that information about the ordering of prompts helps the less expert with memory load problems in a manner consistent with skill acquisition theories.

Theoretical Background

The goal of this study is to provide empirical support for assumptions about user knowledge of UNIX and user memory processes which are proposed in the Doane, Kintsch, & Polson (1989;1990) UNICOM construction-integration model for generating complex commands in UNIX. In so doing, we present a general methodology that may used to examine the complex knowledge and process assumptions made by other cognitive models.

Previous empirical work gave UNIX users at varied levels of expertise textual descriptions to produce legal UNIX commands that required the redirection of standard input and output (i.e., "composite" commands) (Doane, Pellegrino, & Klatzky, 1990). Their data suggest that novices and intermediates have knowledge of the elements of the system; that is, they can successfully produce the single and multiple commands that make up a composite. They could not, however, put these elements together using pipes and/or other redirection symbols to produce the composite commands. The symbols that enable input/output redirection are fundamental design features of UNIX, and these features are taught in elementary computer science courses. Doane et al (1990) demonstrate that these features can only be used reliably after extensive experience (e.g., experts had, on the average, 5 years of experience with UNIX).

UNICOM (Doane, Kintsch, & Polson, 1989;1990), was developed to determine why the Doane et al. 1990 users had such difficulties generating composite commands. To facilitate discussion of the UNICOM knowledge base, we will use the command "nroff -ms ATT2>ATT1" as an example. This command formats the contents of the file ATT2 using the utility nroff and the -ms macro package, and then stores the formatted contents of ATT2 in the file ATT1. To correctly produce composite action plans like this, the model must first identify the items that must be used to produce the command, and then it must sequence these items using the properly placed redirection symbol to construct an action plan. Thus, the Doane et al (1989;1990) model assumed that expert users use a two-stage process to complete action plans and that this process requires knowledge of both items and order. To complete the first phase, the model requires four different types of item knowledge to produce composites. UNICOM must know two syntax specific types of knowledge (a) command syntax (e.g., the nroff command and the -ms flag) and (b) I/O redirection syntax (e.g., the ">" redirection symbol). The model must also have two general types of knowledge that are syntax independent. The model must know (c) general facts about the redirection of input and output (e.g., that redirection of input an output can occur between commands). This is separate from the syntax specific knowledge of I/O redirection symbols. That is, some users appear to know that redirection can occur, and not know the specific syntax. UNICOM must also have (d) general facts about command redirection (e.g., that the output of nroff can be redirected to a file). The latter form of knowledge is necessary, because UNIX commands are not consistent in their I/O structure (e.g., an "ls" cannot take input from another command, but "sort" can).

In the second phase of action planning, the model has to sequence items properly and then keep track of the

intermediate stages of its action plan in order to produce a successful command. For example, to produce the command "sort file|head|lpr", the model had to first select the items to be used (sort,|,|,lpr,head,file) and then determine the order of the items and keep track of which command was first, second, and third, and what the output of the first command was, where its output was redirected, and so on. Doane et al (1989;1990) assumed that the mechanisms involved in integrating and ordering the items to produce a composite would produce serious memory load problems. In addition, it was assumed that experts have knowledge structures which decrease this memory load, whereas novices do not.

It is important to our research that explanation of the item knowledge and ordering processes fits in a more global context than simply UNIX command production. There are reasons to believe that this is the case. First, UNIX composites have characteristics that are important for understanding problem solving and skill acquisition in general. The literature on expertise and skill acquisition (e.g., Chi, Feltovich, & Glaser, 1981; Larkin, 1983) suggests that less expert individuals may know the separate facts in a domain, but not be able to use them productively in the sense described by Wertheimer (1982/1945). Thus, this issue is not unique to the UNIX composite problem. Understanding how individuals utilize textual instructions to develop their factual knowledge into effective problem solving procedures is an important skill acquisition issue in general. Second, we are analyzing the knowledge and processes required to chain elements together in the context of a theory of comprehension that has been used to explain algebra story problem comprehension (Kintsch, 1988), and solution of simple computing tasks (Mannes & Kintsch, in press). Thus, rather than developing an idiosyncratic knowledge analysis, we are performing our research in the context of a general architecture of cognition. As such, we contribute to the goal suggested by Newell (1987) to develop a "unified theory of cognition".

Experiment

The goals of this study, then, are to determine more precisely what users at different levels of expertise know about UNIX, what information is lacking when users produce erroneous commands, and what information (i.e., prompt contents) helps. The experiment uses a prompting paradigm to assess the knowledge and processes of users at various levels of expertise. We assume that users have different amounts of the required four types of knowledge and that displaying the prompts that help with each respective type of knowledge will impact subsequent user performance. We hold similar assumptions about the process knowledge (i.e., keeping track of intermediate results).

Method

Subjects. Twenty-two computer science and electrical engineering majors with between six months and 6 years experience with UNIX were paid $20 for their participation. Novices had less than 1.25 years of experience with UNIX and no operating systems courses; intermediates had between 1.25 and 3.0 years experience with UNIX and some had operating systems courses; and experts had greater than three years experience with UNIX and all had taken an operating systems course.

Task Description:

Format the text in ATT2 using the -ms macro package and store the formatted version in ATT1

Prompts:

Prompt 1. You will need to use the following command
One that will format the contents of a file using the -ms macro package

Prompt 2 You will need to use this command
nroff -ms will format the contents of a file using the -ms macro package

Prompt 3 You will need to use a special symbol that redirects command output to a file

Prompt 4 You will need to use the arrow symbol ">" that redirects output from a command to a file

Prompt 5 You will need to use the arrow symbol ">" and the command nroff -ms

Prompt 6 You'll need to use an nroff -ms on ATT2 (which will output the formatted contents of ATT2), and you'll need to redirect this output as input to ATT1

Prompt 7 You will need to use exactly the following command elements (though not necessarily in this order):
>, nroff -ms

Prompt 8 You'll need to use the command nroff -ms followed by the arrow symbol ">"

Prompt 9 The correct production is
nroff -ms ATT2>ATT1
Please enter this production now

Table 1. Example of task description and prompts for problem nroff -ms ATT2> ATT1.

Apparatus and Materials. All tasks were performed on a Macintosh with large-screen monitor. The stimuli were task statements, a fixed directory of file names and a series of help prompts, all displayed on the screen, as well as three "error cards" presented by the experimenter. Subjects responded by typing at the keyboard a command or series of commands which would accomplish a given task and then using the mouse to "click" on a displayed button.

The task instructions described actions that could best be accomplished by combining two or three commands through the use of redirection (i.e., composite problems). An example is the task shown in Table 1 for which the command sequence would be "nroff -ms ATT2>ATT1." A fixed directory listing of file names was displayed on the screen at all times.

For incorrect responses, there was a series of help prompts designed to address specific deficits in the subjects' UNIX knowledge. These prompts were displayed on the screen one at a time in a fixed order regardless of the type of error that the subject had made. The help prompts were developed to address the most probable causes of error, as suggested by the UNICOM model of UNIX command production. There are seven different areas in which the help prompts could assist the subjects, and these are best described by referring to the example in Table 1.

Finally, we used error cards, describing the type of error made by the subject on an incorrect attempt. The three types were "Illegal," "Different: Legal but does something other than what the problem asks for," and "Keystrokes; legal and does what the problem asks for but uses more than the minimum possible keystrokes." (This last error category was intended to force subjects to use composite commands rather than a series of commands with temporary files in order to be correct.)

Procedure. Subjects were run individually for a single two-hour session. The experiment was made up of three sections: a questionnaire, an orientation, and completion of the actual UNIX problems.

Composite Command Problems. The subjects were given a series of 21 composite command problems. Together, the problems used 10 different utilities (e.g., lpr, cmp, cat) and 4 different input/output (I/O) redirection symbols (e.g., <, |, >>, >). In this activity, subjects were presented with a task statement on the computer screen, such as that shown in Table 1 and asked to type a command or series of commands that would accomplish the task. For each composite problem, the task statement appeared first along with the directory listing of all file names used in the experiment. The subjects would then type their first attempt at a command or command series to accomplish the task. After the subject clicked the button image labelled "DONE", the program would evaluate the subject's attempt and give feedback of "Correct" or "Try again". If the subject's answer was not correct, the experimenter evaluated the subject's attempt and handed the subject the appropriate error card. After reading the error card, the subject clicked another button image; the program then erased the subject's attempt and revealed the first help prompt.

Once the help prompt was revealed, the subject tried again to give a command sequence that would accomplish the task. If incorrect, the subject received another error card and was shown another help prompt. Throughout a given task, the task statement, directory listing, and all previous prompts remained visible; only the subject's previous attempts and error cards were removed. If at any time the subjects felt completely stymied, they were able to type "no help" rather than a solution attempt in order to get the next help prompt. The procedure of subject attempt, error card, help prompt continued until the subject typed the correct response. When the subject typed the required command sequence, the program displayed a "CORRECT!" screen and then revealed the next composite task instructions.

Results

Scoring Correct Command Productions. Productions were scored as correct by the Mac computer if they matched the syntax of the idealized command (spaces were not parsed). Thus, a subject had to produce the command that required the least number of keystrokes (i.e., subjects could not substitute "sort>file1; head file1>file2" for the command "sort file1|head>file2"). Problems were broken into two groups, where problems 1-4 constituted the first group where most of the learning took place, and problems 5-21 made up the second, more stable group. In this paper we report only the data from the first group of problems.

Correct Productions as a Function of Prompt. Figure 1(a) shows the cumulative percentage of correct composite productions at each prompt level for the three expertise groups. Experts have the highest percentage overall, followed by the less expert groups. We expect experts to exceed novice performance. Prompts have a differential influence on correcting performance for the three expertise groups. For example, the change in percent correct performance from Prompt 3 to Prompt 4 is very small for experts, leading to the inference that Prompt 4 provided little or no new information for them. Conversely, the same change between prompts 3 and 4 for the less expert groups is relatively large, suggesting that this prompt does provide them with some new information. To analyze this effect, we study the four types of knowledge at each prompt level in detail.

Scoring of Knowledge. Each of the 21 problems given to subjects required certain amount of the four types of knowledge. For example, the problem described in Table 1 requires two command syntax facts: nroff -ms as a command name, and that nroff takes a file argument. It requires knowledge of one I/O redirection syntax fact, that ">" redirects output from a command to a file. Conceptual I/O knowledge required is that redirection of input and output can occur, and that I/O redirection can occur from commands to files. The required command redirection knowledge includes the fact that nroff output can be redirected to a file. Answers for each task were scored for the percentage of each type of knowledge displayed by

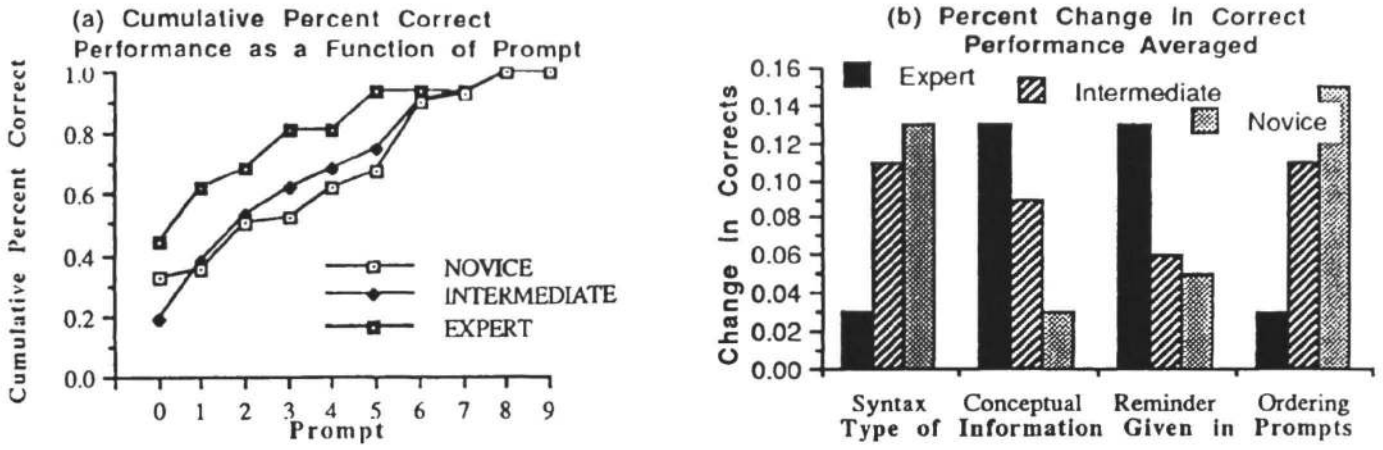

Figure 1. (a) Cumulative percent correct composite answers as a function of prompt; (b) Percent change in correct composite answers averaged across certain prompts.

each subject at each prompt level. For example, if a command requires two command syntax facts, and the attempt at prompt 0 (before any prompts) shows evidence of only one of these facts, then the subjects is credited for having 50% of the required command syntax knowledge for that task. When a production was correct, all knowledge types were at 100% levels.

Knowledge Analyses. Figures 2 (a) - 2 (d) show the mean knowledge scores for the three expertise groups for prompts 0-9. The arrow markers signal the reader what prompt first provided subjects with information relevant to the knowledge type displayed in the graph. For example, in Figure 2(a), Prompt 2 is the first prompt that describes all of the command syntax knowledge required to complete the task (see Table 1 for an example of all prompt types described in this section). To determine whether amount of relevant knowledge was influenced by expertise and prompt, these data were subjected to a 3 x 4 x 10 analysis of variance (ANOVA) with group (expert, intermediate, novice) as a between-subjects variable, and knowledge type and prompt as within-subjects variables. [In all cases, Greenhouse-Geisser degrees of freedom are used for within-subject variables.] This analysis resulted in a marginal effect of level, $F(2,19)= 3.24$, $p < .06$. Experts possess significantly higher knowledge scores than do the other two groups. There was also a main effect of knowledge type, $F (3,57) = 24.52$, $p<.01$, with the command redirection conceptual knowledge showing the lowest scores overall and the command syntax knowledge showing the highest scores overall. We did not obtain a knowledge type by level interaction, $F < 2$, seemingly due to the expert group ceiling effect. A main effect of prompt exists, $F (9,171) = 29.04$, $p < .01$, and a prompt by level interaction, $F (18,171) = 3.74$, $p < .01$. Performance improves as users receive more prompts, though the amount of improvement is different for experts than for novices. Experts appear to reach ceiling performance more quickly than do novices. Prompts influence the four types of knowledge differently, as evidenced by knowledge by prompt interaction, $F (27,513) = 8.12$, $p < .01$.

It is interesting to look at the three expertise groups separately to determine how the prompts influence their knowledge scores. Looking at Figures 2 (a-d), it appears that less expert group's knowledge scores differ from the expert's. To determine this, each of the three groups' knowledge scores were subjected to separate 4 x 10 ANOVAs with the knowledge types and the prompts as within-subject variables. The novices and intermediates show a main effect of knowledge type (Novice $F (3,27) = 21.99$, $p < .01$; Intermediate, $F (3,21) = 13.02$, $p < .01$) and a main effect of prompt (Novice, $F (9,81) = 26.82$, $p < .01$; Intermediate, $F (9,63)= 18.65$, $p < .01$). Both groups show different levels of the four knowledge types, and an improvement in performance with the prompts. They also each show a knowledge type by prompt interaction (Novice, $F (27,243) = 6.44$, $p < .01$; Intermediate, $F (27,189) = 4.84$, $p < .01$). This suggests that the influence of the prompts on the four knowledge types is not equivalent for these two groups. The experts also show a main effect of knowledge, $F (3,9) = 5.94$, $p < .05$, and prompt, $F (9,27) = 6.90$, $p < .02$. Their data do not show a knowledge type by prompt interaction ($F < 3$). The results may reflect expert ceiling effects. Their knowledge level is so high to begin with that there is minimal variance in their data.

The arrow markers on Figures 2(a-d) show some of the most interesting interactions, especially when compared with the percent correct performance shown in Figure 1(a). Figure 2(a) suggests that for novices and intermediates, presentation of Prompt 2, which helps with command syntax (see Table 1 for an example), improves command syntax knowledge and percent correct performance (see Figure 1(a)). This is less the case for the experts. Figure 2(b) shows groups with improved I/O conceptual knowledge scores for their attempts following presentation of Prompt 3, which gives this information. However, Figure 1(a) suggests that while the experts can use conceptual I/O information to increase their chances for a correct production, the novices cannot. This leads to the inference that the novices can use conceptual I/O

information to make a subsequent attempt that demonstrates greater I/O conceptual knowledge, but they are still lacking other information which would allow them to produce a correct command. Figure 2(c) shows that Prompt 4 appears to assist the novice group I/O syntax knowledge, and that the same trend holds to a lesser degree for the intermediates. Figure 1(a) suggests that this prompt increases the percent correct performance for the less expert groups. The experts, however, find this information useless. Figure 2(d) and Figure 1(a) show similar trends in the the change in command redirection knowledge for less expert groups following Prompt 6. This prompt gives both command redirection information and ordering information. Note that experts find this information useless.

A summary of the trends found in correct performance as a function of prompt is found in Figure 1(b). This figure shows the percent change in correct performance averaged together for the first prompts that provide syntax knowledge (both I/O and command syntax, Prompts 2 & 4), for the conceptual I/O prompt (Prompt 3), Prompt 5, which is the first reminder of the items required for the production, and the command redirection prompts, which also give ordering information (Prompts 6 & 8). Overall, there appears to be a strong interaction, with novices and experts showing very different changes in performance as a function of exposure to different types of prompts, and intermediates falling somewhere between the other two groups.

It appears that novices and intermediates need help with both command and I/O syntax, but that this is not a problem for the experts. Experts get help from an abstract statement of I/O conceptual knowledge (see Table 1 for an example),while this statement is practically worthless to the less expert groups. This finding is in line with the expertise literature (e.g., Chi et al., 1981) which suggests that experts can process more abstract information than can novices. Certainly Prompt 3 is more abstract that Prompts 2 and 4. Experts also successfully utilize a reminder prompt that lists all of the command and I/O elements together in one prompt. Novices and intermediates don't seem to utilize this information. Anecdotally, it is our view that the experts that have not solved a problem correctly by this point have often represented the problem with one command reversed or dropped, and this prompt reminds them that they have forgotten an element, and they can use this information to make a correct production. The less expert subjects seem to have more problems at this point than just knowing the elements, they can't put them together with just a list of the items. Their response to the ordering prompts (6 & 8) make this clear. These are the first prompts that tell users

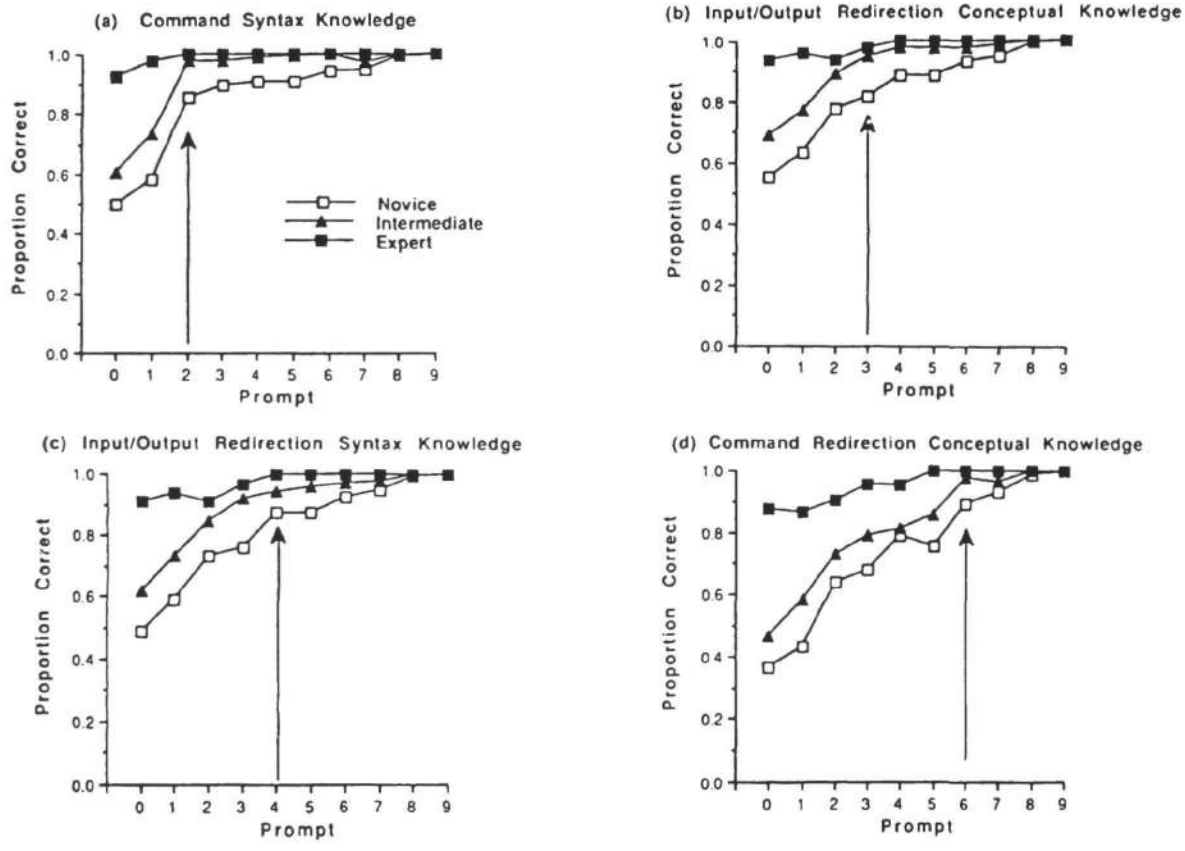

Figure 2. (a) Command syntax knowledge; (b) I/O conceptual knowledge; (c) I/O redirection syntax knowledge; and (d) Command redirection conceptual knowledge as a function of prompt.

what order they must use for the commands. For example, that first nroff -ms must be performed on ATT2 and then that this output must be redirected to the file ATT1, and so on. This ordering information helps the less expert groups quite a bit; it does almost nothing for the experts.

Discussion

Our findings indicate that users vary markedly in their response to very different classes of prompt as a function of expertise. The results suggest that novices and intermediates lack both specific syntax knowledge and more general conceptual redirection knowledge. If prompted with specific syntax knowledge, novices and intermediates can increase their chances of producing a correct composite. However, novices and intermediates cannot utilize our more abstract prompt (Prompt 3) which provides conceptual redirection information knowledge that they need (see Figure 2(b)) to increase their correct performance (see Figure 1(a)). Our experts, in contrast, can use this information effectively to increase their performance. This finding is consistent with the literature on expertise which suggests that experts can utilize abstract information, while novices cannot (e.g., Chi et. al, 1981).

Novices and intermediates seem to need help retrieving the elements that go into making a composite; but for many of them, this is not enough. They also need help ordering the elements -- or constructing an action plan. When they receive a prompt that helps them order the items, their performance improves. There is evidence that ordering the elements taxes their working memory. They delete and substitute many more symbols than do experts, and these memory errors decrease markedly once ordering information has been given (Doane et al. 1991). Thus, the hypothesis suggested by the UNICOM construction-integration model was supported. Retrieving the elements is only part of the problem in composing a composite. To chain these elements together is another aspect of the problem, and this requires relating the pieces of knowledge in an ordered fashion and tracking the intermediate results.

While these data provide support for the assumptions of the Doane et. al (1989;1990) model, more direct support is provided by extending our modeling work to account for these prompt data. We are currently building models of individuals in this study, and running simulations by giving UNICOM prompts, and evaluating prompt activation within the individual's network. We hope to determine whether the overlap between the knowledge in a prompt and the individual network of knowledge dictates the usefulness of the prompt. This work will have clear implications computer aided instruction and for system design.

Acknowledgements

This research was supported by grant IRI-8722792 from the National Science Foundation.

References

Anderson, J. R., & Jeffries, R. 1985. Novice LISP errors: Undetected losses of information from working memory. *Human-Computer Interaction, 1*, 133-161.

Chi, M. T. H., Feltovich, P. J., & Glaser, R. 1981. Categorization and representation of physics problems by experts and novices. *Cognitive Science, 5*, 121-152.

Doane, S. M., McNamara, D. S., Kintsch, W. Polson, P. G., Clawson, D. M., Dungca, R. G. 1991. *Prompt Comprehension in UNIX Command Production.* Technical Report 91-4. Institute of Cognitive Science, University of Colorado, Boulder.

Doane, S. M., Kintsch,, W. & Polson, P. 1989. Action Planning: Producing UNIX commands. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society.* (pp. 458-465). Ann Arbor, Michigan: Lawrence Erlbaum Associates.

Doane, S. M., & Kintsch, W., & Polson, P. G. 1990. *Understanding UNIX Commands: What Experts Must Know.* Technical. Report. 90-1. Institute of Cognitive Science, University of Colorado, Boulder.

Doane, S. M., Pellegrino, J. W., & Klatzky, R. L. 1990. Expertise in a computer operating system: Conceptualization and performance. *Human-Computer Interaction, 5*, 267-304.

Kintsch, W. 1988. The use of knowledge in discourse processing: A construction-integration model. *Psychological Review, 95*, 163-182.

Larkin, J. H. 1983. The role of problem representation in physics. In D. Gentner & A. Stevens (Eds.), *Mental models* (pp. 75-98). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Mannes, S. M. & Kintsch, W. in press. Routine computing tasks; Planning as understanding. *Cognitive Science.*
Murdock, B. B. 1974. Human memory: Theory and data. Lawrence Erlbaum Associates.

Newell, A. 1987. Unified theories of cognition. *The 1987 William James Lectures.*

van Dijk, T. A. & Kintsch, W. 1983. *Strategies of discourse comprehension.* New York: Academic Press.
Wertheimer, M. 1982/1945. *Productive thinking.* Chicago, IL: University of Chicago Press.