Journal of the Ergonomics Society of Korea
Vol.22, No.3 pp.93-111, August 2003

시스템 디자인을 위한 유닉스 사용성의 인지적 분석

Cognitive Analysis of User Interactions with UNIX and its Application to System Design

손영우*, 이지선**, 육형민***

ABSTRACT

This research extends a general theory of cognition to address cognitive constraints on complex command production and allows us to make system design recommendations. The research described in this paper addresses the cognitive origins of problems user have producing sequence-dependent command strings while interacting with the UNIX operating system. We describe an empirical and theoretical analysis of user difficulties, and then show how our analyses lead to design recommendations. In addition, we summarize results from testing the impact of our design recommendations on system usability

Keyword: Cognitive User Model, Human-Computer Interaction, UNIX Operating System, User Interface Design

* 연세대학교 심리학과
주소: 120-729 서울시 서대문구 신촌동 연세대학교
전화: 02)2123-2444
E-mail: ysohn@yonsei.ac.kr
** 연세대학교 심리학과
*** 연세대학교 인지과학 협동과정

1. INTRODUCTION

There may be many determinants to the problems that users have with the complex systems. Research on cognitive models of users have provided a great amount of insight into the loci of such problems (Bovair et al., 1988; Kieras & Polson, 1985; Kitajima & Polson, 1992, 1995; Mannes & Doane, 1991). In our laboratory, models of user interactions with UNIX have been profitable for determining difficulties users have with this and other complex systems (Doane et al., 1990, 1992, 1994, 2000).

UNIX is a widely used system, and this renders the present research of general interest. From a theoretical point of view, the UNIX operating system is interesting because of characteristics of its command structure. A property of the UNIX command structure is that complex commands can be created by concatenating simple commands with the use of advanced features that redirect command input and output (e.g., pipes). This allows users to "connect" commands that will together execute a customized sequence-dependent series of operations. This customization is difficult for users to accomplish, even when they know the elements (e.g., command names, redirection symbols) that will accomplish the sequence of operations (Doane et al., 1990). That is, even when the user has the requisite knowledge of command elements, they still have difficulty chaining the elements together.

1.1 Research Problem

To examine user performance as a function of command complexity, previous research classified commands into one of two main types. The types reflected the number and nature of relationships among the component sub commands (Doane et al., 1990). One type of command is single commands which can be expressed by "Do x" where x refers to a single subcommand (e.g., sort file). The second type of commands are composite commands which can be expressed as "Do x, then Do y, then Do z," where x, y, and z are unique single commands performed once and coordinated using pipes, redirection symbols, or both (e.g., sort file|head|lpr). Thus, composite commands are sequence dependent and can be correctly performed

only in a certain order because the output from one subcommand acts as input to the next.

In Doane et al. (1996), UNIX users at different levels of expertise were provided with textual descriptions and were asked to produce legal UNIX commands that require the redirection of input and output (i.e., composite commands). The findings suggested that the users could successfully produce the single commands that make up a composite. However, many of the less expert users could not put these single commands together by using pipes and/or other redirection symbols to produce the composite commands. In fact, these features were reliably used only by subjects with an average of 4 years of experience with UNIX. The symbols that enable input/output (I/O) redirection are fundamental design features of UNIX. The results are surprising because all subjects were computer science and electrical engineering majors, and these features were taught in their elementary computer science courses.

1.2 Research Goal

While the previous research demonstrated systematic performance deficits, it did not clarify their cause. The goal of this paper is to provide practical solutions for the problems by reviewing a research program used to determine the cause of the performance deficits. For this purpose, we will report only those aspects of the research program critical for testing usability of UNIX and proposing design solutions. This work makes both theoretical and applied contributions by addressing the following three questions:

- What are the cognitive loci of novice users' poor performance in generating sequence-dependent UNIX commands?
- What is the theoretical account of the cognitive processes underlying the complex command production?
- What are the practical solutions for improving the novice users' performance?

The theoretical contribution is to diagnose problems that computer users encounter when producing complex commands on the basis of theory of cognition. This is accomplished by reviewing empirical data on user

performance in an interactive training context and examining the architecture of a computational model based on the construction-integration theory of comprehension (Kintsch, 1988; Kitajima et al., 1995). We specify how comprehension-based cognitive processes are central to understanding how knowledge and processing demands influence user command production performance. The theoretical analyses of user problems also provide insight into the practical solution for the problems. The applied contribution is to propose actual remedies that can be directly applied to the HCI environment.

1.3 Cognitive Loci of User Performance Deficits

Two plausible explanations have been proposed for the user performance deficits in the sequence-dependent command production. One explanation for this pattern of performance is that a user must know a significant number of facts (i.e., command syntax knowledge, I/O syntax knowledge, conceptual knowledge of I/O redirection, and conceptual knowledge of command redirection) to produce a composite, and that these knowledge prerequisites are the locus of user problems (Doane et al., 1990). Another explanation is that in addition to the knowledge prerequisites, users must keep track of the temporary state of information flowing between commands. The resulting load on working memory may be significant, and this may be the locus of user difficulties (Doane et al., 1990). In the present case, we can explain the memory demands best through an example. To produce the composite "sort file1|head|lpr," users must generate "sort" on "file1", then keep in memory the result of performing the "sort" (the sorted contents of "file1"), then generate a "head" on the sorted contents of file1, then keep in memory the result of performing the "head" (the first ten sorted lines of "file1"), and then generate "lpr," where the "lpr" prints the first ten sorted lines of "file1.". These elements must be kept in memory because the interface does not provide any information about the intervening results. This memory explanation is consistent with other findings which express the role of working memory in novice performance in complex tasks such as production of LISP code (Anderson & Jeffries, 1985).

Doane et al. (1992) performed empirical study which suggest that the four types of knowledge described above are indeed required to produce a composite command. However, these studies did not provide a clear determination of whether novice

users' difficulties in generating composite UNIX commands resulted only from knowledge deficits or from additional working memory load problems. This problem motivated empirical studies which examine the working memory load problems for the sake of producing complex UNIX commands (Sohn & Doane, 1997). We summarize only those aspects of the studies relevant for the present work [for more detailed descriptions see (Lewis et al., 1990)].

2. EXPERIMENT1

2.1 Prompting Study

To determine the role of knowledge deficit and working memory load in complex command production, Sohn and Doane (1997) used a modification of the prompting paradigm developed by Doane et al. (1992). Subjects were serially prompted with the four types of requisite knowledge (i.e., command syntax, I/O concepts, I/O syntax, and command redirection) and then presented with ordering prompts. We created two forms of the ordering prompt, where both forms provided the same ordering information, but did so in ways that either reduced or increased working memory demands. The working memory demands posed by each form were manipulated by modifying the direction of problem solving reasoning used in the prompt. Previous research suggests that reasoning in a "forward" direction from the initial state of a problem towards the goal reduces working memory load, while reasoning "backward" from the goal to the initial problem state increases working memory load (Chi et al., 1981; Larkin et al., 1980). Based on these findings, we manipulated the direction of reasoning in the ordering prompt forms to be either "forward" or "backward". Then, we ensured that the two forms of ordering prompts differentially affect working memory load in a prior experiment [see Experiment 1 in (Sohn & Doane, 1997) for details].

If knowledge deficits alone are the locus of composite production errors, then only presentation of knowledge prompts should be required for correct performance. If subjects require further assistance with ordering the elements of a composite, then they will be presented with either the forward or backward form of the ordering prompt. If the reason subjects

require assistance ordering elements of a composite is related to working memory demands, then only the forward prompt form should facilitate performance. Given the changes in working memory demands as a function of expertise, this effect should be most pronounced for novices (Ericsson & Kintsch, 1995). If the assistance provided by the ordering prompt is unrelated to working memory demands, then there should be no difference in performance as a function of ordering prompt form.

2.2 Method

2.2.1 Subjects

Twenty-three undergraduate computer science and electrical engineering majors with 6 months to 9 years of experience with UNIX were paid $20 for their participation. A preexperimental questionnaire was administered to determine the subjects' experience with UNIX and other operating systems, their history of computer usage, computer training, and self-assessment of their computer skills. Novices (N = 9) had between 6 months and 1.25 years of experience with UNIX and no operating systems courses; intermediates (N= 9) had 1.25 to less than 4 years of experience with UNIX and some had operating systems courses; and experts (N = 5) had 4 or more years of experience with UNIX and all had taken an operating systems course.

2.2.2 Apparatus and Material

All tasks were performed on a Macintosh computer using a Hyper Card program. The stimuli were task statements, a fixed directory of file names and a series of help prompts displayed on the screen. Subjects typed a command or a series of commands at the keyboard to accomplish a given task and then used the mouse to hit on a displayed button to let the program evaluate their answers. The task instructions described actions that could be accomplished by combining two or three commands with the use of redirection (i.e., composite problems). The 21 composite problems presented by these task instructions required the use of 10 different utilities (e.g., "lpr," "cat") and four different I/O redirection symbols (e.g., ">," "|," "<," ">>"). An example is the task shown in Figure 1 for which the command sequence would be "nroff -ms ATT2>ATT1." For erroneous responses,

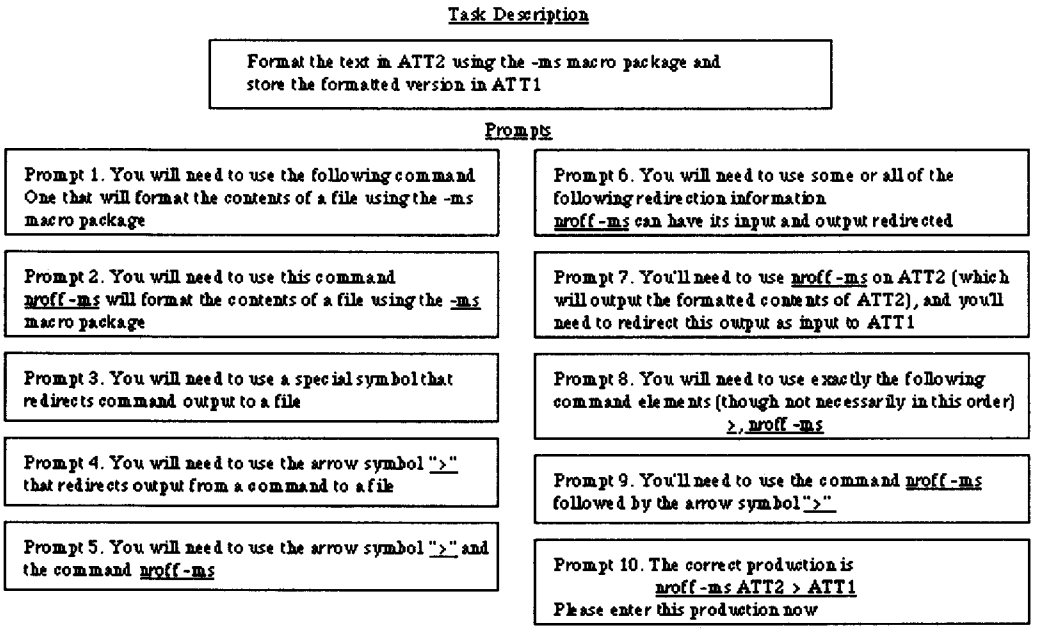

Figure 1. Examples of Task Description and Prompts for the Problem "nroff -ms ATT2>ATT1" in the forward ordering condition.

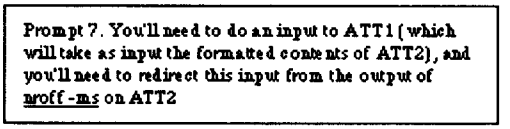

Figure 2. Example of Prompt 7 for the Problem "nroff -ms ATT2>ATT1" in the backward ordering condition.

there were series of help prompts designed to address specific types of 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 knowledge and memory prerequisites. These are best described by referring to the example in Figure 1. Prior to Prompt 7, subjects have been prompted with knowledge of command syntax (Prompt 2), I/O concepts (Prompt 3), I/O syntax (Prompt 4), and command redirection (Prompt 6). Prompt 7 is the first prompt that provides ordering assistance in either the forward (Prompt 7 in Figure 1) or backward (Prompt 7 in Figure 2) forms.

2.2.3 Procedure

The subjects were run individually

for a single 2-hour session. For each problem, the subjects were asked to type their attempt at a command or a series of commands to accomplish the task and then clicked on the display button labeled "DONE" to cause the program to evaluate their attempt as "Correct" or "Try again". If the subject's answer was not correct, the program erased the subject's attempt and revealed the first help prompt. The procedure of subject attempt and help prompt was repeated until the subject typed the correct answer, and then the next task instruction appeared. Throughout a given task, the task statement, directory listing, and all previous prompts remained visible.

2.3 Results and Discussion

The data were analyzed by grouping the problems together according to percentage of new knowledge required for solution to compare all relevant effects of expertise and prompt. A fact in a problem was considered new knowledge if it had not been used in a previous problem by the subject in the experiment. As evidenced by Doane et al. (1992), the percent of new knowledge was a significant predictor of correct production performance. We split the problems into three groups, those requiring 0% new knowledge (8 problems), those requiring 1%-59% new knowledge (10 problems), and those requiring 60%-100% new knowledge (3 problems). The 60%-100% new knowledge results are reported here. The other percent new knowledge problems resulted in similar, but attenuated effects.

The analyses of percent correct performance showed that while prompting knowledge increases performance, it is not sufficient to ensure successful composite production. Novice and intermediate users did not reach the expert level of performance when they were assisted only with the four types of requisite knowledge. It is important to discuss changes in percent correct performance following presentation of the ordering prompt to determine the role of working memory demands. As summarized in Table 1, the forward form increases percent correct performance particularly for novices, whereas the backward form has minimal impact. The results support the hypothesis that working memory demands are also the locus of performance deficits rather than knowledge deficits per se. Only the

forward form of the ordering prompt facilitates performance, and the effect of prompt form was most pronounced for novices, which is consistent with the hypothesis that expertise-related performance deficits are related to working memory demands [c.f., (Ericsson, K. A., & Kintsch, W., 1995)].

Table 1. Effects of ordering prompts on percent correct performance (Change in percent correct performance from Prompt 6 to Prompt 7)

Prompt	Novices *M(SD)*	Intermediates *M(SD)*
Forward	46.7 (18.2)	13.4 (18.3)
Backward	8.4 (16.7)	8.3 (16.7)

Note. Experts are not included because they showed a ceiling effect prior to presentation of Prompt 7.

3. EXPERIMENT 2

3.1 Support System Study

To demonstrate the role of requisite knowledge and working memory load in the more practical HCI context, we designed a graphical interface intended to support users with the knowledge and working memory demands of composite production tasks. We manipulated the amount of support provided and measured the effects on performance using a modified "Training Wheels" approach outlined by Carroll and McKendree (Carroll & McKendree, 1987; Sohn & Doane, 1997). Specifically, we designed and tested a graphical interface that provides two levels of support for sequence-dependent command production (e.g., see Figures 3 and 4). The support levels correspond to the knowledge and memory prerequisites for successful composite production discussed previously in the prompting study. We devised a version that provides relevant knowledge (knowledge; see Figure 3), and another that provides knowledge and working memory (knowledge and working memory; see Figure 4). We will briefly summarize the method and results of this study, which are relevant to the goal of this paper (Sohn & Doane, 1997).

3.1.1 Knowledge Support System

Looking at Figure 3, support was provided via shape-coding of composite elements at the bottom of the screen. The oval, diamond and rectangle shapes depict commands, redirection symbols, and files, respectively. The

oval "frames" ("a" and "b") contain shape-coded representations of command syntax knowledge that depict the standard input and output for a specific command, separated by an arrow. For example, the oval composite element "a"contains two rectangles that abstractly represent the contents of an unidentified file before and after being processed by the "tail" command. The diamond frames ("c" and "d") contain shape-coded representations of redirection syntax and redirection conceptual knowledge that depict a specific non-standard input/output redirection process. For example, the diamond labeled "c"depicts output from an unidentified command (oval on the left) being redirected via a pipe (a cylindrical shape representing a "|" in the middle) as input to another command (the oval on the right). The rectangle frames ("e" and "f") represent files used in the composite. Subjects in the knowledge condition indicated the correct order of composite elements using the underlined spaces displayed at the middle of the screen.

3.1.2 Knowledge & Working Memory System

Unique to this system is additional support for ordering elements and keeping track of intermediate results. The shape-coded elements shown at the bottom of Figure 4 depict the sequence-dependent state of file content. For example, rectangular element "f" now contains an abstract representation of the contents of file PROP1. This serves to make the abstract representation of file contents in the command element "b" (oval) more useful in constraining command order, as does the addition of the PROP1 file label. Element "b" also shows the "sorted PROP1" as an intermediate result, and this is echoed on the left side of element "a." And, the redirection symbols (diamond elements "c" and "d") explicitly show command redirection order. Ordering support was also provided by shape-coding the "slots" in the middle of the screen. The shape coding served to constrain the number of possible placements of composite elements in the correct sequence.

3.2 Method

3.2.1 Subjects

Nineteen undergraduate computer science and electrical engineering

majors with 6 months to 7 years of experience with UNIX were paid $10 for their participation. Subjects were classified as novice or expert following the criteria used in Doane et al. (1990). Novices (N = 11) averaged 1.6 years UNIX experience and had not taken any operating systems courses. Experts (N = 8) averaged 4.5 years UNIX experience and all had taken at least one operating systems course. Subjects from each expertise group were randomly assigned to the knowledge or knowledge & working memory support condition.

3.2.2 Apparatus and Materials

All tasks were administered on a Macintosh computer. As shown in Figures 3 and 4, each set of stimuli was composed of task instructions presented at the top of a screen, graphic representations of composite elements at the bottom of a screen, and either spaces or shape-coded slots indicating the correct sequence of composite elements in the middle of the screen. Ten problems were presented one at a time, and were displayed until subjects entered a composite command and hit the "return" key, at which time the next problem screen was presented.

3.2.3 Procedure

Subjects were tested individually in a session lasting approximately 15 minutes. For each problem screen, subjects were asked to enter letters corresponding to the correct sequence of composite elements (e.g., "A", "B", "C") using the slots provided in the middle of the screen, and then hit the "return" key to proceed to the next problem. They were allowed to use the "delete" key to make corrections prior to entering "return." Subjects reviewed on-screen instructions and completed two practice problems before starting the experiment. Following each practice trial, subjects were shown the correct sequence of elements. This feedback was not provided during the experimental trials.

3.3 Results and Discussion

Responses were scored as correct if participants entered the correct sequence of composite elements. As shown in Table 2, experts showed superior performance overall as compared with novices. Expert performance does not differ as a function of support level, though this is probably due to a performance ceiling effect. In contrast, novice

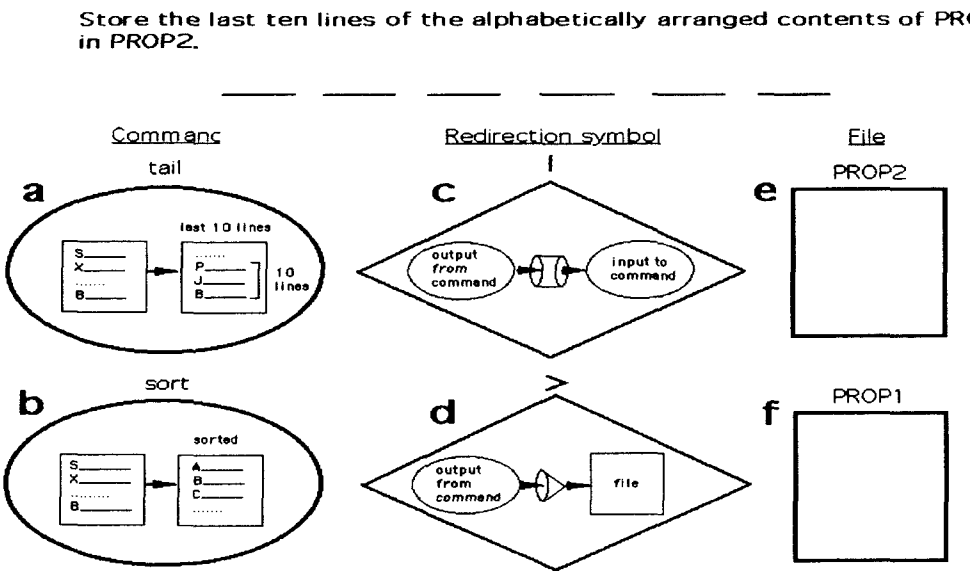

Figure 3. Examples of task description, slots, and graphical UNIX elements for the problem "sort PROP1 | tail>PROP2" in the knowledge system

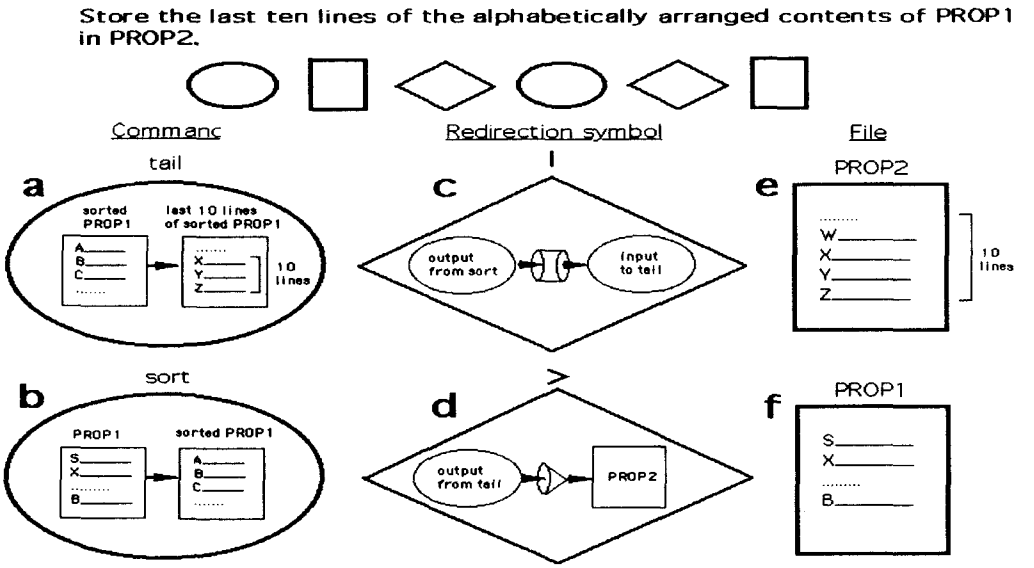

Figure 4. Examples of task description, slots, and graphical UNIX elements for the problem "sort PROP1 | tail> PROP2" in the knowledge & working memory support system

performance increases with increasing levels of support. These results support our findings from the prompting study, suggesting that assisting novices with required knowledge is not sufficient to obtain correct performance; they must receive additional assistance that reduce working memory demands. While the present system is in the prototype stages, our results suggest that the design concept facilitates novice performance on complex tasks.

Table 2. Mean percent correct productions for novice and expert groups as a function of support system

Support system	Novices *M(SD)*	Intermediates *M(SD)*
Knowledge	36.0 (13.4)	90.0 (8.2)
Knowledge & Working memory	78.3 (22.0)	92.5 (5.0)

4. GENERAL DISCUSSION

4.1 Theoretical Account of Cognitive Processes

In the previous section, we discussed empirical evidence from the prompting and support system studies to understand cognitive difficulties computer users encounter when they solve sequence-dependent command production problems. The challenge for any psychological theory of human-computer interaction is to provide a formal explanation and predictions for user performance. Toward this end, we developed a comprehension-based model of UNIX command production (UNICOM) (Doane et al., 1994, 2000). In UNICOM, knowledge of the current problem solving situation is associated with existing background knowledge, and knowledge activation is constrained based on relevance to the problem at hand. By simulating actual user performance on the composite command production in the prompting context, we specified the underlying mechanisms of knowledge and memory processes that the previous empirical studies were unable to address (Doane et al., 1994, 2000).

The predictive validity of UNICOM was tested by simulating 22 individual UNIX users whose UNIX experience varied in the prompting context previously

described. In this modeling, individual users' performance was analyzed to identify their initial knowledge base, which represents the knowledge they displayed without prompting. Using this knowledge base, we then "gave" the model the same prompts that the user was given when it executes an unsuccessful action plan. Then we ran the model again so that the incoming prompt instructions can activate knowledge to attempt to solve the problem again. Comparisons of model and the human empirical data resulted in a high degree of agreement, validating the ability of UNICOM to predict user response to prompting. For the sake of brevity, we summarize only the theoretical aspects of the UNICOM model relevant to the goal of this paper [for more detailed descriptions of the other aspects see (Doane et al., 2000)].

4.1.1 UNICOM Construction /Integration Model

The theoretical foundation of our modeling rests on Kintsch's construction-integration theory of comprehension (Kintsch, 1988). Specifically, Kintsch's construction- integration theory presumes that low-level associations between contextual information (e.g., task instructions) and long-term memory are constructed and used to constrain knowledge activation via a constraint-based integration process. The resulting pattern of context-sensitive knowledge activations is referred to as a "situation model" and represents the current state of comprehension.

Kintsch's theory of comprehension suggests that successful problem solving requires building an effective model of the problem situation (Kintsch, 1988). The construction of this situation model is assumed to be central to skilled performance because it dictates how knowledge is activated and used throughout problem-solving episodes (Ericsson & Kintsch, 1995). The importance of the situation model has been evidenced for understanding human -computer interaction skills (Doane et al., 1994, 2000; Kitajima & Polson, 1992, 1995; Mannes et al., 1991). Kintsch theorizes that situation models are built via a two-stage process that involves "constructing"an associated knowledge base and then "integrating" knowledge activation on the basis of relevance to the current problem situation (Kintsch, 1988).

Similarity-based associations between incoming information (e.g., descriptions of the problem) and existing background knowledge (e.g., known facts about the problem domain) are used to constrain knowledge activation, resulting in context-sensitive activation values for all knowledge retrieved from memory or accessed in the environment. Collectively, this integrated knowledge is referred to as the situation model.

4.1.2 Modeling UNIX User Performance

In order to build a situation model effective for producing correct composite command action plans, UNICOM required the four types of UNIX knowledge. In addition to the knowledge components, UNICOM also used a working memory buffer that retrieved, compared, and stored composite elements throughout the problem solving process. To model working memory constraints, prompted knowledge items required to produce a composite command were retained in the memory buffer on the basis of their activation. The model retained the currently prompted knowledge items and the fixed number (4 in the present model) of the most activated previously prompted items in the buffer. Four was an approximation of working memory capacity that served as a constant across all simulations. Because the activation of all knowledge items was constrained by their relevance to the current task context, we were simulating context-sensitive working memory limitations.

In this theoretical framework, the ability to simulate context-sensitive knowledge activation is most important for modeling user performance. In the prompting context, producing commands requires comprehending the interactive situation and then activating knowledge that is relevant for solution. Specifically, we propose that comprehending instructions to produce commands results from associating brief instructions with retrieved knowledge, and then integrating this associative knowledge to develop a situation model that guides subsequent performance. In UNICOM, instructional text and the current state of the operating system serve as cues for activation of the relevant knowledge and for organizing this knowledge to produce an action sequence.

4.2 Practical Solutions

In the previous section, we described the theoretical aspects of a comprehension-based model of UNIX problem solving to understand knowledge and memory processes involved in the complex command production. To summarize our findings, novice UNIX users have difficulty chaining sequence-dependent operations together even when they display knowledge of the component commands; working memory demands are the additional locus of coordinative performance limitations. The comprehension-based approach suggests that working memory is constrained on the basis of context-sensitive knowledge activation that dictates retrieval of relevant knowledge from memory and guides an action sequence.

Our work provides several practical solutions for the problems that users have with a complex computer system. First, the design concept used in the support system study will facilitate novice performance on complex tasks. Given our findings, it is interesting to note that existing graphical UNIX interfaces do not support composite production. It is perhaps worthwhile to consider incorporating some of the features from the support system used in our study into future UNIX interfaces, though the present system is in the prototype stages (Sohn & Doane, 1997). For example, systems might provide users with an option to enter a composite production "mode," where the interface would graphically display intermediate results as a composite was entered. The amount of composite support provided might vary as a function of user performance using a "training wheels" approach (Carroll & McKendree, 1987).

Second, the prompting method can be essentially used as a simplistic tutoring system. The prompt order and contents were based on both longitudinal studies of UNIX skill acquisition and the UNICOM comprehension-based analysis (Doane et al., 1990, 1992). The empirical studies showed that UNIX users acquire command syntax knowledge first, and then learn about redirection. Given this, prompts can be used to help novice users with command syntax knowledge first, followed by redirection knowledge and ordering prompts. Our findings suggest that effectiveness of this tutoring system would improve if explicit tutoring of coordinative knowledge is provided, and particularly if it is in a form that reduces student working memory demands.

Third, the comprehension-based model, UNICOM, can be used to predict user performance. We have shown how a comprehension-based theory of learning accounts for a significant amount of user performance when learning from technical instructions (Doane et al., 1994, 2000). Using a model based on the construction-integration theory of comprehension, we developed individual knowledge bases based on a small subset of user performance data. The initial knowledge bases were used by UNICOM to "participate" in the UNIX prompting study, and allowed us to predict significant aspects of user performance. This modeling work has implications for computer-aided tutoring as well. Tutoring systems have already used models to assess the untutored expertise of a student (Anderson, 1988; VanLehn, 1988). Using the present techniques, instructions chosen by the tutor for presentation could be optimized for a particular student model in a context-sensitive manner. If, for example, a student model had to choose among three instructional options at a given point during training, it could use our methods to determine which of the three options would receive the highest activation given the current assessment of student knowledge.

Finally, our findings also suggest a specific design recommendation for providing coordinative task support. For example, the redirection structure for UNIX utilities is variable (e.g., the input of "sort" can be redirected but this is not true for "ls"). The cognitive consequence of this design feature is that users must recall the relevant redirection structure for each command before incorporating it into a novel composite command. If the demand placed on memory by this feature were eliminated by designing consistent command redirection structures, or by making structures visible, user performance would improve.

REFERENCES

Anderson, J. R. (1988). The expert module. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of intelligent tutoring systems*. (pp. 21-54) .Hillsdale, NJ: Erlbaum.

Anderson, J. R., & Jeffries, R. (1985). Novice LISP errors: Undetected losses of information form working memory. *Human-Computer*

Interaction, 1 133-161.

Bovair, S., Kieras, D.E., & Polson, P.G. (1988).The acquisition and performance of text-editing skill production-system analysis, (Technical Report No. 28), University of Michigan, Ann Arbor.

Carroll, J. M., & McKendree, J. (1987, January). Interface design issues for advice-giving expert system *Communications* of the ACM, 30, 14-31.

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

Doane, S. M., McNamara, D. S., Kintsch, W., Polson, P. G., & Clawson, D.(1992). Prompt comprehension in UNIX comm -and production. *Memory and Cognition* 20 (4), 327-343.

Doane, S. M., Sohn, Y. W., Adams, D., & McNamara, D. S. (1994). Learning fro instruction: A comprehension-based approach. In *Proceedings of the 16th Annual Conference of the Cognitive Science Society*, 254-259. Atlanta, GA:Erlbaum.

Doane, S. M., & Sohn, Y. W., McNamara, D. S., & Adams, D. (2000). Compre-hension-based skill acquisition. *Cognitive Science*, 24,1-52

Draper, S.W. (1985). The nature of expertise in UNIX. In *Proceedings of INTERACT'84 IFIP Conference on Human-Computer Interaction*, 465-471, New York: Elsevier North-Holland.

Ericsson, K. A., & Kintsch, W. (1995). Longterm working memory. *Psychological Review*, 102, 211 -245.

Kieras, D.E. & Polson, P.G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man -Machine Studies*, 22, 365-394.

Kintsch, W. (1988). The use of knowledge in discourse processing: A construction integration model. *Psychological Review*, 97, 163-182.

Kitajima, M. & Polson, P.G. (1992). A computational model of skilled use of a graphical user interface. *CHI 1992 Proceedings*,241-249.

Kitajima, M. & Polson, P.G. (1995). A comprehension-based model of correct performance and errors in skilled, display-based, human- computer inter-action. *Interna- tional Journal of Human- Computer Studies*, 43, 65-99.

Larkin, J., McDermott, J., Simon, D. P. & Simon, H. A. (1980). Expert and nov-ice performance in solving physics problems. *Science*, 208 (20).1335-1342.

Lewis, C., Polson, P., Wharton, C., & Rieman, J. (1990). Testing a Walk-through Methodology for Theory-Based Design of Walk -Up-and -Use Interface. *CHI 1990 Proceedings*,235-242.

Mannes, S. M., & Doane, S. M. (1991). A hybrid model of script generation: Or getting the best of both worlds. *Connection Science*, 3(1), 61-87.

Sohn, Y. W., & Doane, S. M. (1997). Cognitive constraints on computer problem solving skills. *Journal of Experimental Psychology:Applied*, 3, 288-312.

VanLehn, K. (1988). Student modeling. In M. C. Polson & J. J. Richardson(Eds.), *Foundations of intelligent tutoring systems*. (pp. 55-76). Hillsdale, NJ: rlbaum.

저자 소개

◈ 손영우

고려대학교 경영학과 학사
University of Missouri at Columbia 사회학 석사
University of Illinois at Urbana-Champaign 심리학 박사
University of Connecticut at Storrs 심리학과 조교수
현재 연세대학교 심리학과 조교수
관심분야: 인간공학, 항공심리, 인지심리

◈ 이지선

연세대학교 심리학, 영문학 학사
연세대학교 심리학과 석사과정
관심분야: 인간공학, 인지심리

◈ 육형민

연세대학교 식품영양학과 이학사
연세대학교 인지과학협동과정 석사과정
관심분야 : 인간공학, 인간-컴퓨터 상호관계,인지심리

논문접수일 (Date Received): 2003/06/02
논문게재승인일(Date Accepted): 2003/07/07