Int. J. Man–Machine Studies (1987) **26**, 649–660

Cognitive processing differences between novice and expert computer programmers

ALLAN G. BATESON, RALPH A. ALEXANDER AND MARTIN D. MURPHY
Towson State University, Towson, Maryland 21204

(Received 15 May 1986 and accepted in revised form 20 March 1987)

Research on cognitive processing differences between novice and expert groups has recently begun to focus on applied areas like computer programming. An often-used research paradigm has measured subjects' syntactic memory, their ability to recall briefly presented computer programs. This study demonstrates that expert programmers use semantic memory and high-level plan knowledge to direct their programming activities. Fifty subjects were divided into novice and expert groups based on the number of programming courses taken. Four tests were developed to measure syntactic memory, semantic memory, tactical skill, and strategic skill. Experts performance was superior on all tests. Additionally, the best set of predictors of programmer expertise was semantic memory, tactical skill, and syntactic memory. Results from this and subsequent research may have implications for areas such as selection and training.

Research on cognitive processing differences between novices and experts has typically focused on such areas as chess (Chase & Simon, 1973; Cleveland, 1907; deGroot, 1965), go (Reitman, 1976), and physics (Chi, Teltovich & Glaser, 1981; Larkin, McDermott, Simon & Simon, 1980). These topical areas are well suited to the study of novice/expert differences for knowledge and problem-solving abilities due to the substantial range of observed skill and experience differences.

Recently, research on novice/expert differences has begun to focus on more applied areas. In a study which measured the memorization/recall ability for circuit diagrams, Egan and Schwartz (1979) found that skilled electronic technicians recalled elements of the diagrams in functional chunks. Computer programming is another applied area in which research on cognitive processes is relatively new. Research on programming has dealt mostly with issues such as how to identify candidates for programming jobs and on improvements in program writing through comments, documentation and modularization. As a result, little theoretical research has been done (Adelson, 1981).

The approach to research on programming clearly parallels the research on cognitive processes for chess. Studies on chess initially focused on the ability of master chess players to reproduce 90% of midgame board positions, with 20 pieces, after only 5 s of study (deGroot, 1965). Chase and Simon (1973) replicated deGroot's findings, revealing that experts form larger chunks than less-experienced players. The superior recall of the experts was not found to be related to above average memory capacity, suggesting that the chunks were organized hierarchically (Chase & Simon, 1973).

Requests for reprints should be sent to Allan Bateson, Department of Psychology, Towson State University, Towson, MD 21204, U.S.A.

0020-7373/87/060649 + 12$03.00/0

Subsequent research has focused on the ability of chess players to reconstruct briefly presented, meaningful and random board positions. The typical finding is that experts perform better on recall of meaningful positions but no differently from novices when positions are random (Charness, 1976; Chase & Simon, 1973; Lane & Robertson, 1979; Simon & Chase, 1973).

Applying the memorization/recall paradigm of the chess research to computer programming, skilled programmers recalled significantly more of meaningful programs than novices and, consistent with the chess results, performed no differently from novices on random programs (Shneiderman, 1976; Shneiderman & Mayer, 1979). One explanation for experts' programming skill suggests that superior recall results from the ability to organize information into chunks based on the larger number of procedural patterns held in memory by experts (Adelson, 1981; Shneiderman, 1976; Shneiderman & Mayer, 1979). With experience, frequently encountered information is hierarchically organized into increasingly larger chunks, incorporated into memory, and indexed to enable quick, accurate recall of the appropriate chunks (Atwood & Ramsey, 1978; Curtis, 1982). As a result, more-experienced programmers form more-efficient representations in memory.

A second explanation in the programming literature suggests that experts also seem to use a higher-level knowledge to understand problems while novices tend to focus on the specific statements employed in a program (Adelson, 1981; Atwood & Ramsey, 1978; Green, 1980; Shneiderman, 1976, 1980; Shneiderman & Mayer, 1979; Soloway, Ehrlich, Bonar & Greenspan, 1982; Wiedenbeck, 1985). Wiedenbeck (1985) suggests that experts are able to write and understand, with little allocation of attention and with few errors, various well-practiced programming subtasks. Such subtasks may include recognition of syntactic errors and the writing and understanding of well-practiced program subroutines and subcomponents. One result is that the programmer should be able to focus on high-level problem-solving. Both Atwood and Ramsey (1978) and Shneiderman (1976) proposed that information is stored in memory in predominantly semantic form, i.e. in terms of its meaning relative to the programmer's prior store of information, rather than in literal, syntactic form. Adelson (1981) provided some empirical support for the proposition that experts organize program information semantically. The author found that subjective organization of program statements presented in a multi-trial free-recall format was according to program membership, semantically organized into meaningful program segments. Novices organized the lines of code by syntactic category such as statement type.

Consistent with the second explanation, Shneiderman and Mayer (1979) proposed a model of long-term memory consisting of both semantic and syntactic components. Semantic memory is conceptualized as being relatively independent of the syntax for particular programming languages. Semantic knowledge is abstracted through continuing experience in problem-solving and instruction and is stored as general, meaningful sets of information. The learner must be able to anchor new information within existing semantic knowledge (Shneiderman & Mayer, 1979; Tulving, 1972). Syntactic memory includes information such as details concerning types of statements within particular programming languages, the format of iteration, valid character sets, and the names of library functions (Shneiderman & Mayer, 1979). In the context of learning a programming language, syntactic learning initially

encompasses single statements and progresses to collections of statements. The collection of statements may come to be perceived as patterns, such as a subroutine to find the average of a series of numbers. The processes involved in memory for syntactic information are congruent with the first explanation for novice/expert differences involving memory for procedural patterns. Little meaningful learning is taking place. The focus is predominantly on the rote memorization of subprograms and program statements.

Mynatt (1984) provided empirical support for the model in a study of the effect of semantic complexity on program comprehension. Program modules that were more cognitively complex resulted in decreased delayed vs immediate recall and were less accurately hand-executed than semantically simpler procedures. Increased complexity results in greater requirements for semantic encoding and less subsequent accuracy. The information must first be integrated with existing semantic information. Decreased recall results when syntactic information such as variable names is forgotten.

Finally, a third explanation for programmer skill is provided by Soloway *et al.* (1982), who suggest that experts use more than just knowledge of syntax and semantics of programming when they write programs to solve problems. Experts may use high-level plan knowledge to direct their programming activities. A plan is a procedure in which the key elements of a process have been abstracted and represented explicitly. When presented with a new problem, the expert retrieves plans from a knowledge base which proved useful in similar situations and incorporates them into the solution. Plans are classified into three categories: (a) strategic plans specify a global strategy used in an algorithm; (b) tactical plans specify a local strategy for solving a problem; and (c) implementation plans specify language-dependent techniques for realizing strategic and tactical plans. For example, to read an array and compute a running average (strategic plan), you might choose a looping strategy (tactical plan) and write the code using a FORTRAN DO loop with a running total and counter. Soloway *et al.* (1982) suggest that experience facilitates the choice of appropriate strategies and tactics. Thus, it seems that research should focus, not only on how experts differ from novices on organization of and recall from memory, but also on what the expert must know in order to select and utilize the appropriate plan.

A recent study of memory for chess parallels the conceptualization of Soloway *et al.* (1982). Specifically, the factors of semantic knowledge, episodic memory, positional judgment, and tactical skill were investigated in relation to chess skill. Pfau (unpubl.) found semantic knowledge to be the main cognitive factor which separates novices from experts. In addition, semantic knowledge, positional judgment, and tactical skill were better predictors of skill than was episodic memory.

The purpose of this study is to demonstrate the importance of semantic memory for skill in computer programming. Much of the literature on differences in the organization of memory between novice and expert programmers has focused on syntactic memory and chunk size for recall tasks. Programmers, however, define skill in different terms. Programmers discuss skill in terms of comprehension, the ability to modify and debug programs, knowledge of programming languages and data structures, and imagination in solving problems and in writing code. The

present study has focused on operationalizing the knowledge and problem solving components which programmers say are important for skill in programming. These components reflect the hierarchical organization and intellectually demanding learning of semantic memory.

We propose that the relationships of tests designed to measure semantic memory, syntactic memory, strategic skill, and tactical skill will result in two principal findings: (1) the importance of semantic memory over syntactic memory in distinguishing between novice and expert programmers; and (2) semantic knowledge, tactical skill, and strategic skill as the best set of predictors of programmer skill. Both hypotheses are based on the expectation that conceptually anchored information resulting from meaningful learning will be more representative of skill than will syntactic memory. Syntactic memory should be least important for determining skill because both novices and experts can be expected to possess basic knowledge of syntax.

Additional hypotheses are: (3) strategic skill and tactical skill should be related more highly to semantic memory than to syntactic memory; and (4) the knowledge and problem-solving components of semantic memory should show the strongest relationships to strategic skill and tactical skill, respectively.

Method

SUBJECTS

Subjects were 50 undergraduate and graduate student volunteers at a large midwestern university, ranging in age from 18 to 38 ($\bar{X} = 23$, S.D. = 3·46). Thirty-seven subjects were male, 17 were female. The sample consisted mainly of computer science majors ($n = 37$). The minimum requirement for participation was enrollment in the introductory Fortran course (subjects were recruited after the twelfth week). Other subjects were recruited by the experimenter out of more advanced computer-science courses. Participants were not paid for taking part in the study.

Questionnaire data were used to divide subjects into two groups, novices and experts. Shneiderman (1976) categorized subjects according to experience into novice (enrolled in an introductory programming course), intermediate (second or third course), and expert. Soloway *et al.* (1982) found that novices and intermediates chose different looping strategies than did experts, suggesting that novices and intermediates may be grouped together. Two groups of subjects were therefore used, novices and experts. The novice group ($n = 20$) consisted of students who had completed no more than three programming courses ($\bar{X} = 2·10$, S.D. = 0·79). The expert group ($n = 30$) included all subjects who had completed more than three programming courses ($X = 9·10$, S.D. = 3·97, range of from 4 to 20 courses taken).

TESTS

Syntactic memory

Four, short computer programs were adapted or written in Fortran IV to be approximately equal in length and complexity. Three of the programs consisted of 21 lines of code, the fourth had 20 lines. The statements for two of the programs

were scrambled, using a random number table so that the programs were no longer executable. The programs were typed on separate sheets of paper. Subjects were asked to reproduce one meaningful and one randomized program.† For each program, the subject was given a 3-min viewing period. At the end of this time the subject turned the paper face down and immediately began recall, writing the answers on standard Fortran coding forms. Subjects were permitted 4 min to recall the program and were instructed to reconstruct the program exactly as it had been presented. Each reconstructed program was scored for serial recall. Statements were scored as correct if they were listed in serial order and contained only minor syntax errors such as misspelled variable names.

Strategic skill

Three problem statements were written or adapted from available programming literature. Each statement provided the necessary information to enable subjects to write an algorithm, in either pseudo-code or outline form, of a proposed solution. For each step in their answers, subjects were also asked to include a brief statement explaining why that step was necessary. Subjects were given an example in the form of a sample problem and solution as the second page in the test booklet immediately following written instructions. While no strict time limit was enforced, the instructions stated that subjects should allow only about 10 min for each of the three problems. The problems were scored for correctness of solution, efficiency, and clarity of the algorithm. Emphasis was placed on the high-level planning demonstrated in participants' algorithms. Two experts independently scored each problem. Each problem was rated on a five-point Likert scale for specific, *a priori*, criteria for each of the above factors. Ratings were then summed for the three factors for the three problems to determine a total score for each subject. The scores for the two raters were averaged to determine a final, composite score for each subject (maximum possible = 45).

Tactical skill

Six problems were constructed or adapted from available literature. The problems required subjects to write either short programs or program segments and in some cases also to explain the function of a specific program segment. For example, one problem asked subjects to write a function subprogram LOGICAL FUNCTION SEARCH (A, N, KEY) where A is an integer array, N is the size of the array, and KEY is an integer number. The LOGICAL function will return .TRUE. if KEY is found in the current array, otherwise return .FALSE.. Subjects were given the six problems, each on a separate sheet of paper, and allowed enough time to provide written answers to all problems. The instructions did suggest, however, that subjects should allow only about 30 min to complete all of the problems. The tactical skill problems were scored by the experimenter according to specific, *a priori* criteria. Each problem was rated, on a five-point Likert scale, for completeness and

† Two meaningful and two random programs were used for the syntactic memory part of the study. The programs were counterbalanced into each of eight possible combinations of one random and meaningful program. Results were subjected to an ANOVA which showed there were no significant differences in recall among the combinations, $F(7, 42) = 1{\cdot}03$, $p = 0{\cdot}43$.

correctness of the solution. Emphasis was placed on correct execution of plans already provided to participants such as the example given above. The ratings for each of the six problems were summed to obtain a total score (maximum possible = 30).

Semantic memory

A 42-item multiple-choice test was constructed in consultation with the computer-science department. The questions were written to be of varying difficulty. All items consisted of five alternative answers and were conceptually organized into two subtests measuring problem-solving ability (18 items) and knowledge of Fortran, data structures, and general programming principles (24 items). The score was determined as the total number correct. The instructions stated that the test should take approximately 1 h, although subjects were allowed enough time to answer all items.

Procedure

Testing was conducted in groups of from one to five participants. Subjects began the session by completing a questionnaire requesting demographic information as well as information concerning computer-science and programming courses taken, occupation (if computer/programming related), and other outside experience (e.g. computer clubs, user groups).

All subjects completed the tests in the following order: syntactic memory, strategic skill, tactical skill, and semantic memory. Subjects read an orientation sheet, prior to beginning the session, which explained the purpose of the study and included an approximate time frame for the completion of all of the tests. At the conclusion of the strictly timed memorization/recall measure, subjects were given a folder containing the remaining tests. They were allowed to work at their own pace, completing all measures in the previously stated order. Average time for completion of all tests was approximately 2·5 h.

Results

Table 1 provides the means, standard deviations, and t test results for the novice and expert groups on each of the four experimental measures. Experts scored significantly higher than novices on all of the tests.

Measures of internal consistency reliability were computed for the measures of both tactical skill ($\alpha = 0{\cdot}81$, Cronbach's alpha) and semantic memory ($\alpha = 0{\cdot}85$). Interrater reliability was computed for the measure of strategic skill ($r = 0{\cdot}80$).

Scores for the measure of syntactic memory were analysed as a 2×2 ANOVA with skill group (novice/expert) and program type (random/ordered) as factors. Experts' scores were significantly higher than novices, $F(1, 47) = 12{\cdot}12$, $p < 0{\cdot}001$. The proportion of statements recalled was higher for the ordered vs the random programs, $F(1, 47) = 10{\cdot}35$, $p < 0{\cdot}01$. A significant skill × program interaction was expected because previous research has found that experts recall significantly more of ordered programs than novices but nonsignificant skill group differences on recall of random statements. The interaction was not significant, however ($F < 1$).

t tests were performed to determine how the results of the present study compare

TABLE 1
Descriptive statistics and t tests for novice/expert differences on the experimental measures

Variable	*n*	$\bar{X}$	S.D.	df	*t* value
Memory composite					
Novice	20	0·16†	0·14		
Expert	30	0·26†	0·11	48	3·07‡
Knowledge					
Novice	20	10·20	4·25		
Expert	30	15·43	3·55	48	4·72†
Problem-solving					
Novice	20	7·05	4·02		
Expert	30	11·10	2·26	48	3·15†
Tactical skill					
Novice	20	10·30	7·20		
Expert	30	18·80	4·17	48	5·28†
Strategic skill					
Novice	19	23·37	6·86		
Expert	30	30·03	4·22	48	4·22§

NB, These are one-tailed tests.
† Values are proportions.
‡ $p < 0·01$.
§ $p < 0·001$.

with previous research. Group means are provided in Table 2. Within the program factor, experts recalled a significantly higher proportion of both the random [$t(48) = 2·97$, $p < 0·01$] and the ordered [$t(48) = 3·16$, $p < 0·01$] programs than did novices (one-tailed tests). For the skill factor, experts' recall was superior on the ordered vs random programs, $t(29) = 1·89$, $p < 0·05$. For novices, while the mean for ordered recall was higher than that for the random programs, the result was not statistically significant, $t(19) = 1·57$, $p < 0·10$.

Our results differ from previous research only in that expert performance on recall of the random programs did not decline to the novice level. Meaning should facilitate recall for the experts due to their increased knowledge and experience. The performance of the experts remained superior to that of novices for recall of both random and ordered programs, however.

TABLE 2
Means for memory test used in composite score

Skill group	*n*	Program type	$\bar{X}$†
Novice	20	Random	0·34
		Ordered	0·41
Expert	30	Random	0·46
		Ordered	0·54

† Proportion of total possible.

TABLE 3
Correlation matrix of skill and experimental measures

Variable	1	2	3	4	5	6
(1) Skill						
(2) Syntactic memory	0·40†					
(3) Semantic memory	0·57‡	0·39†				
(4) Knowledge	0·56‡	0·32†	0·93‡			
(5) Problem-solving	0·55‡	0·48‡	0·91‡	0·72‡		
(6) Tactical skill	0·61‡	0·43†	0·76‡	0·68‡	0·78‡	
(7) Strategic skill	0·52‡	0·53‡	0·75‡	0·69‡	0·72‡	0·73‡

NB, $n = 50$; $n = 49$ for strategic skill.
† $p < 0·01$.
‡ $p < 0·001$.

Hypothesis 1 stated that semantic memory should be important in distinguishing between skill groups. The correlation of semantic memory with skill (number of programming courses) is significant, $r = 0·57$, $p < 0·001$, as can be seen from Table 3. In addition, a stepwise regression showed semantic memory to be a better predictor of skill than syntactic memory, $F(1, 47) = 23·60$, $p < 0·001$, $R^2 = 0·33$. Syntactic memory did not significantly aid in predicting programmer skill.

Hypothesis 2 was analysed by a maximum R-square regression to determine the best set of predictors of skill. The best set proved to be semantic memory, tactical skill, and syntactic memory, $F(3, 45) = 9·98$, $p < 0·001$, $R^2 = 0·40$. Strategic skill, counter to our hypothesis, did not significantly add to the prediction of programmer skill.

Hotelling's t test for correlated correlations was used to test hypothesis 3 (Guilford & Fruchter, 1978, p. 164). Two separate tests were necessary. The first test determined the tactical skill-semantic memory ($r = 0·76$, $p < 0·001$) correlation to be significantly greater than the correlation of tactical skill-syntactic memory ($r = 0·43$, $p < 0·01$), t $(47) = 2·75$, $p < 0·01$ (one-tailed test). The second t test showed the correlation of strategic skill-semantic memory ($r = 0·75$, $p < 0·001$) to be significantly greater than that of strategic skill-syntactic memory ($r = 0·53$, $p < 0·001$), t $(47) = 2·25$, $p < 0·05$. The results of both tests support the hypothesis that both strategic skill and tactical skill are more highly related to semantic memory than to syntactic memory and may tap the meaningful learning and acquired knowledge of the programmer rather than simply memory for the syntax of a programming language.

Hypothesis 4 was examined, again using the t test for correlated correlations. Relationships of interest were between strategic skill and knowledge and between tactical skill and problem-solving. The subtests of the test for semantic memory were both highly correlated with strategic skill and tactical skill. The test results do not support our hypothesis. There were no significant differences in the correlations of knowledge and problem solving with strategic skill, $t(47) = 1·56$, $p < 0·10$, nor of the two subtests with tactical skill, $t(47) = 0·95$, $p < 0·10$. In addition, stepwise regression analysis showed both problem solving abilities and general knowledge to be important for the prediction of both strategic skill [$F(2, 46) = 31·07$, $p < 0·001$,

$R^2 = 0{\cdot}57$] and tactical skill [$F(2, 46) = 41{\cdot}78$, $p < 0{\cdot}001$, $R^2 = 0{\cdot}61$]. Syntactic memory also aided the prediction of strategic skill, $F(3, 45) = 25{\cdot}53$, $p < 0{\cdot}001$, $R^2 = 0{\cdot}63$.

Discussion

The focus of this study was threefold: (a) to demonstrate semantic knowledge to be the principal cognitive factor in skill for computer programming; (b) to determine the best set of predictors of skill from among our measures; and (c) to investigate the relationship of the memorization/recall measure of syntactic memory, both to skill and to the other measures of interest in this study.

Semantic memory was shown to be important for programmer skill. Semantic memory was also shown to be a better predictor of skill than was syntactic memory. The importance of this finding lies in contrast to the use of the memorization/recall task as the predominant measure found in the literature for assessing programmer skill.

Semantic memory, tactical skill, and syntactic memory provided the best three-factor model for predicting programmer skill. Syntactic memory replaced strategic skill in our hypothesized model, due perhaps to the strong intercorrelations between the variables. Syntactic memory was included to account for unique variance not accounted for by either semantic memory or tactical skill. This interpretation is reasonable in light of the correlation between strategic skill and syntactic memory ($r = 0{\cdot}53$, $p < 0{\cdot}001$), and that syntactic memory, rather than strategic skill, was included in the regression model to predict programmer skill.

We found that the high-level plan conceptualizations of strategic and tactical skill (Soloway *et al.*, 1982) were most strongly related to the semantic memory component of Shneiderman and Mayer (1979). Plan knowledge thus seems more dependent on meaning derived through acquired knowledge and problem-solving than on memory for language syntax. Subsequent research should focus on the organization of expert knowledge and on how semantic information is accessed to develop the high-level plans experts use to solve problems.

It is also interesting to note the importance of knowledge and problem-solving abilities for high-level planning for both strategy and tactics. From a review of the literature, we expected problem-solving and syntax to be more important for tactics than for strategy because tactical plans involve more immediate solutions than strategic plans. Our results suggest the memory component to be more important for strategic plans.

Some caution should be used in interpreting the implications of the syntactic, semantic, strategic, and tactical measures. While great care was taken in developing and refining the separate tests, they were not validated. Additional research operationalizing these components is needed before we can determine whether the tests are sturdy measures of the labels applied.

The memorization/recall task was intended as a baseline measure by which to compare our results with similar research on memory for programming (Shneiderman, 1976), memory for chess (Chase & Simon, 1973; Pfau, 1983), and memory for script processing (Bower, Black, & Turner, 1979). Past research found that performance on the recall task for ordered information is superior to that for

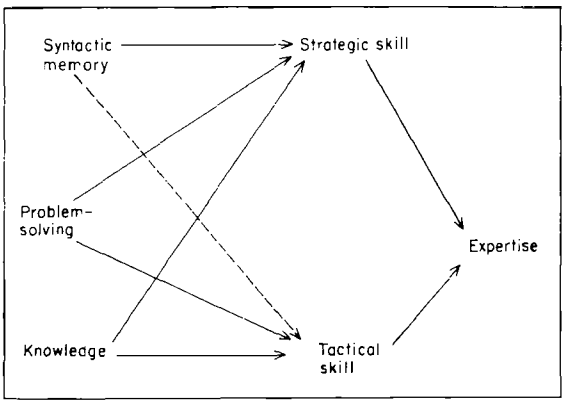

FIG. 1. Proposed cognitive processing model (problem-solving and knowledge comprise the semantic memory component).

random information. More specific to programming, Shneiderman (1976) found that experts performed better than both novices and intermediates on recall of the structured programs. Shneiderman also found that experts performed similar to novices on the random programs.

In the present study, consistent with prior research, experts' performance on ordered programs was superior to recall of the random programs. Also, novices' recall of the ordered programs was not significantly different from recall of the random programs. Contrary to previous findings, experts in the present study performed significantly better than novices on recall of both the ordered and random programs. A possible explanation is that too much was allowed for memorization (3 min). With greater familiarity of the Fortran language than the novice group, the experts may have had time to chunk the random programs. While the time limits were the same as those used by Shneiderman in his research, the programs used in the present study contained longer statements, perhaps increasing the difficulty. Research on chess (Chase & Simon, 1973) allowed only 5 s to view a board of 20 pieces.

Figure 1 provides a tentative model which integrates the variables investigated in this study. Solid lines suggest direct relationships. The broken line between syntactic memory and tactical skill was added because of the correlation between the variables ($r = 0{\cdot}43$, $p < 0{\cdot}01$). This model will serve as the basis for future research.

In summary, analogies are often made between chess and computer programming regarding the organization and recall of information in memory. We have shown, consistent with previous research for chess, that semantic memory is an important cognitive factor in the determination of skill for computer programming. Research on comprehension and memory for programming could profitably focus on such factors as the best way to learn both first and subsequent programming languages, and in identifying the components of expert skill. Shneiderman (1980) suggested, and DiPersio, Isbister, and Shneiderman (1980) provided empirical support, that employers might profitably use a memorization/recall task as a means of assessing skill for performing complex cognitive tasks. The results of the present study suggest

that a measure of semantic memory would be a better method by which to assess skill and would add face validity. Mynatt's (1984) findings suggest that varying the cognitive complexity of the semantic measure would aid in the detection of skill differences. The use of some combination of both methods may prove optimal.

The importance of this research for training stems from the desire to improve on the presentation–memorization–recall process of learning. Material to be learned is best presented in a manner which facilitates organization in and subsequent recall from permanent memory. Wexley and Lathan (1981) state that training should focus on the match between teaching and learning to facilitate transfer to the job. If we can improve on our ability to teach topics in computer science, such as programming languages, the result may well be better programmers.

This research is based on the first author's masters thesis.

We gratefully acknowledge the assistance of Christy De Vader, Jim McElvey, and the referees for comments on an earlier draft of this article.

References

ADELSON, B. (1981). Problem solving and the development of abstract categories in programming languages. *Memory and Cognition,* **9,** 422–433.

ATWOOD, M. E. & RAMSEY, H. R. (1978). Cognitive structures in the comprehension and memory for computer programs: an investigation of computer programming debugging. *Army Research Institute Technical Report TR-78-A210,* Science Applications, Englewood, CO.

BOWER, G. H., BLACK, J. B. & TURNER, T. J. (1979). Scripts in memory for text. *Cognitive Psychology,* **11,** 177–220.

CHARNESS, N. (1976). Memory for chess positions: Resistance to interference. *Journal of Experimental Psychology: Human Learning and Memory,* **2,** 641–653.

CHASE, W. G. & SIMON, H. A. (1973). Perception in chess. *Cognitive Psychology,* **4,** 55–81.

CHI, M., FELTOVICH, P. & GLASER, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science,* **5,** 121–152.

CLEVELAND, A. A. (1907). The psychology of chess and of learning to play it. *American Journal of Psychology,* **18,** 269–308.

CURTIS, B. (1982). A review of human factors research on programming languages and specifications. *Proceedings: Human Factors in Computer Systems.*

DEGROOT, A. D. (1965). *Thought and Choice in Chess.* The Hague: Mouton.

DIPERSIO, T., ISBISTER, D. & SHNEIDERMAN, B. (1980). An experiment using memorization/reconstruction as a measure of programmer ability. *International Journal of Man–Machine Studies,* **13,** 339–354.

EGAN, D. E. & SCHWARTZ, B. J. (1979). Chunking in recall of symbolic drawings. *Memory and Cognition,* **7,** 149–158.

GREEN, T. G. R. (1980). Programming as a cognitive activity. In SMITH, H. T. & GREEN, T. G. R. Eds. *Human Interaction with Computers.* London: Academic Press.

GUILFORD, J. P. & FRUCHTER, B. (1978). *Fundamental Statistics in Psychology and Education.* New York: McGraw Hill, Inc.

LANE, D. M. & ROBERTSON, L. (1979). The generality of the levels of processing hypothesis: an application to memory for chess positions. *Memory and Cognition,* **7,** 253–256.

LARKIN, J., MCDERMOTT, J., SIMON, D. & SIMON, H. (1980). Expert and novice performance in solving physics problems. *Science,* **4,** 317–345.

MCKEITHEN, K. B., REITMAN, S. C., RUETER, H. H. & HIRTLE, S. C. (1981). Knowledge organization and skill differences in computer programmers. *Cognitive Psychology,* **13,** 307–325.

MYNATT, B. T. (1984). The effect of semantic complexity on the comprehension of program modules. *International Journal of Man–Machine Studies,* **21,** 91–103.

REITMAN, J. S. (1976). Skilled perception in go: deducing memory structures from inter-response times. *Cognitive Psychology,* **8,** 336–356.

SHNEIDERMAN, B. (1976). Exploratory experiments in programmer behavior. *International Journal of Computer and Information Sciences,* **5,** 123–143.

SHNEIDERMAN, B. & MAYER, R. E. (1979). Syntactic/semantic interactions in programmer behavior: a model and experimental results. *International Journal of Computer and Information Sciences,* **8,** 219–238.

SHNEIDERMAN, B. (1980). *Software Psychology: Human Factors in Computer and Information Systems.* Cambridge, Massachusetts: Winthrop Publishers, Inc.

SIMON, H. A. & CHASE, W. G. (1973). Skill in chess. *American Scientist,* **61,** 394–403.

SOLOWAY, E., EHRLICH, K., BONAR, J. & GREENSPAN, J. (1982). What do novices know about programming? In BADRE, A. & SCHNEIDERMAN, B. Eds. *Directions in Human/Computer Interaction.* New York: Ablex Publishing Corp.

TULVING, E. (1972). Episodic and semantic memory. In TULVING, E. Ed., *Organization in Memory.* New York: Academic Press.

WEXLEY, K. & LATHAM, G. (1981). *Developing and Training Human Resources in Organizations.* Glenview, Illinois: Scott, Foresman & Co.

WIEDENBECK, S. (1985). Novice/expert differences in programming skills. *International Journal of Man–Machine Studies,* **23,** 383–390.