The Psychology of How Novices Learn Computer Programming

RICHARD E. MAYER

Department of Psychology, University of California, Santa Barbara, California 93106

This paper examines the current state of knowledge concerning how to increase the novice's understanding of computers and computer programming. In particular, it reviews how advances in cognitive and educational psychology may be applied to problems in teaching nonprogrammers how to use computers. Two major instructional techniques are reviewed: providing a concrete model of the computer and encouraging the learners to actively put technical information into their own words.

Keywords and Phrases: psychology, programming, novices, instruction, man-machine interface

CR Categories: 1.5, 4.2

INTRODUCTION

This paper focuses on the question, "What have we learned about how to increase the novice's understanding of computers and computer programming?" In particular, it reviews ideas from cognitive and educational psychology that are related to the problem of how to teach nonprogrammers to use computers. Since people who are not professional programmers will have to learn how to interact with computers, an important issue concerns how to foster meaningful learning of computer concepts by novices.

Meaningful learning is viewed as a process in which the learner connects new material with knowledge that already exists in memory [BRAN79]. The existing knowledge in memory has been called a "schema" and the process of connecting new information to it has been called "assimilation." However, there is not yet agreement concerning the specific mechanisms that are involved in "assimilation to schema" [ANDE77, AUSU77, BART32, KINT74, MINS75, RUME75, SCHA77, THOR77].

Figure 1 provides a general framework for a discussion of the process of meaningful learning (or assimilation to schema) of technical information [MAYE75a, MAYE79a]. In the figure the human cognitive system is broken down into

- *short-term memory*—a temporary and limited capacity store for holding and manipulating information;
- *long-term memory*—a permanent, organized, and unlimited store of existing knowledge.

New technical information enters the human cognitive system from the outside and must go through the following steps for meaningful learning to occur:

(1) *Reception.* First the learner must pay attention to the incoming information so that it reaches short-term memory (as indicated by arrow a).
(2) *Availability.* Second, the learner must possess appropriate prerequisite concepts in long-term memory to use in assimilating the new information (as indicated by point b).

CONTENTS

INTRODUCTION
1. UNDERSTANDING OF TECHNICAL INFORMATION BY NOVICES
   1.1 Definitions
   1.2 Distinction Between Understanding and Rote Learning
2. DO CONCRETE MODELS AID MEANINGFUL LEARNING OF COMPUTER PROGRAMMING?
   2.1 Statement of the Problem
   2.2 *Concrete Models in Mathematics Learning*
   2.3 Models, Titles, and Advance Organizers in Text
   2.4 Concrete Models in Computer Programming
   2.5 Effects of Models on Transfer Performance
   2.6 Locus of the Effects of Models
   2.7 Effects of Models on Recall Performance
   2.8 Effects of Models on Transfer and Recall Using a Different Language
   2.9 Ability
   2.10 Text Organization
   2.11 Conclusion
3. DOES STUDENT ELABORATION ACTIVITY AID MEANINGFUL LEARNING?
   3.1 Statement of the Problem
   3.2 Putting It in Your Own Words
   3.3 Effects of Model Elaboration on Transfer Performance
   3.4 Effects of Comparative Elaboration on Transfer Performance
   3.5 Effects of Model and Comparative Elaboration on Recall
   3.6 Effects of Note Taking on Transfer and Recall Performance
   3.7 *Conclusion*
4. UNDERSTANDING COMPUTER PROGRAMMING
   4.1 Understanding a Statement
   4.2 Understanding a Program
5. SUMMARY
ACKNOWLEDGMENTS
REFERENCES

(3) *Activation.* Finally, the learner must actively use this prerequisite knowledge during learning so that the new material may be connected with it (as indicated by arrow c from long-term memory to short-term memory).

Thus, in the course of meaningful learning, the learner must come into contact with the new material (by bringing it into short-term memory), then must search long-term memory for what Ausubel [AUSU68] calls "appropriate anchoring ideas" or "ideational scaffolding," and then must transfer those ideas to short-term memory so they can be combined with new incoming information. If any of these conditions is not met, meaningful learning cannot occur; and the learner will be forced to memorize each piece of new information by rote as a separate item to be added to memory. The techniques reviewed here are aimed at ensuring that the availability and activation conditions are likely to be met.

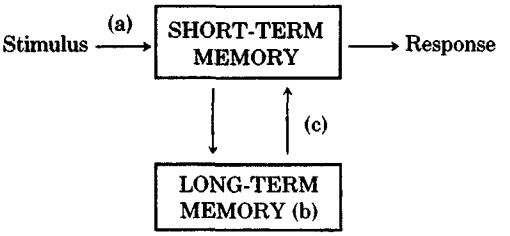

FIGURE 1. Some information processing components of meaningful learning. Condition (a) is transfer of new information from outside to short-term memory. Condition (b) is availability of assimilative context in long-term memory. Condition (c) is activation and transfer of old knowledge from long-term memory to short-term memory.

The goal of this paper is to explore techniques for increasing the novice's understanding of computer programming by exploring techniques that activate the "appropriate anchoring ideas." Two techniques reviewed are (1) providing a familiar concrete model of the computer and (2) encouraging learners to put technical information into their own words. Each technique is an attempt to foster the process by which familiar existing knowledge is connected with new incoming technical information. For each technique a brief rationale is presented, examples of research are given, and an evaluative summary is offered.

1. UNDERSTANDING OF TECHNICAL INFORMATION BY NOVICES

1.1 Definitions

For our present purposes *understanding* is defined as the ability to use learned information in problem-solving tasks that are different from what was explicitly taught.

Thus understanding is manifested in the user's ability to transfer learning to new situations. *Novices* are defined as users who have had little or no previous experience with computers, who do not intend to become professional programmers, and who thus lack specific knowledge of computer programming.

1.2 Distinction Between Understanding and Rote Learning

The Gestalt psychologists [WERT59, KATO42, KOHL25] distinguished between two ways of learning how to solve problems—rote learning versus understanding. With respect to mathematics learning, for example, a distinction often is made between "getting the right answer" and "understanding what you are doing."

In a classic example Wertheimer suggests that there are two basic ways to teach a child how to find the area of a parallelogram [WERT59]. One method involves dropping a perpendicular line, measuring the height of the perpendicular, measuring the length of the base, and calculating area by use of the formula, Area = Height × Base. Wertheimer calls this the "rote learning" or "senseless" method, because the student simply memorizes a formula and a procedure. The other method calls for the student to explore the parallelogram visually until he sees that it is possible to cut a triangle from one end, put it on the other end, and form a rectangle. Since the student already knows how to find the area of a rectangle, the problem is solved. Wertheimer calls this method "structural understanding" or "meaningful apprehension of relations," since the learner has gained insight into the structure of parallelograms.

According to Wertheimer, if you give a test involving parallelograms like the one used during instruction, both groups of children will perform well. However, if you give a transfer test that involves unusual parallelograms, the rote learners will say "We haven't had this yet," while the understanders will be able to derive answers. Thus the payoff for understanding comes not in direct application of the newly learned material, but rather in transfer to new situations. This example suggests that when creative use of new technical information is the goal, it is important to use methods that foster understanding.

2. DO CONCRETE MODELS AID MEANINGFUL LEARNING OF COMPUTER PROGRAMMING?

2.1 Statement of the Problem

Since novices lack domain-specific knowledge, one technique for improving their understanding of new technical information is to provide them with a framework that can be used for incorporating new information. This technique is aimed at ensuring availability of knowledge in long-term memory (see Figure 1). The present section explores the effects of concrete models on people's understanding and learning of new technical information such as computer programming. The major research questions concern how concrete models influence the learning process and how to choose an effective model.

2.2 Concrete Models in Mathematics Learning

One technique for providing the appropriate prerequisite knowledge is the use of familiar, concrete models. For example, Brownell and Moser [BROW49] taught third graders how to use a subtraction algorithm, employing two different methods. One group of several hundred children was taught by means of concrete objects like bundles of sticks. For these children, concepts like "borrowing" and "place value" were described in terms of rearranging bundles of sticks into groups of tens. The other group was taught in a "purely mechanical rote fashion"; these children were explicitly given the rules for subtraction at the start and provided with plenty of "hands on" experience in executing the procedures on standard two-digit subtraction problems. Although both groups of students learned to perform equally well on standard two-digit subtraction problems, the students who learned with bundles of sticks performed better on tests involving transfer problems (e.g., more complicated subtraction problems).

In current instructional practice, *manip-*

ulatives, such as coins or sticks or blocks, are used in mathematics teaching in order to make computational procedures more concrete [WEAV72, RESN80]. In a careful set of interviews with children who were learning to subtract, Resnick and Ford [RESN80] noted that the children often invented a concrete model to help them understand the procedure. Since computer programming shares many of the characteristics of computational procedures in mathematics, it seems possible that the use of manipulatives in computer programming might be as successful as in mathematics.

2.3 Models, Titles, and Advance Organizers in Text

There is also encouraging evidence that similar techniques may be used to increase the meaningfulness of technical information presented in text. For example, Bransford and Johnson presented the following passage to subjects:

> The procedure is actually quite simple. First you arrange items into different groups. Of course, one pile may be sufficient depending on how much there is to do. If you have to go somewhere else due to lack of facilities, that is the next step; otherwise, you are pretty well set. It is important not to overdo things. In the short run this may not seem important, but complications can easily arise. A mistake can be expensive as well. At first, the whole procedure will seem complicated. Soon, however, it will become just another facet of life. It is difficult to foresee any end to the necessity for this task in the immediate future, but then, one never can tell. After the procedure is completed one arranges the materials into different groups again. Then they can be put into their appropriate places. Eventually they will be used once more and the whole cycle will have to be repeated. However, this is part of life [BRAN72, p. 721].

Subjects who read this passage without a title rated it low in comprehensibility (2.3 on a 7-point scale) and recalled an average of only 2.8 out of 18 ideas from the passage. However, some subjects were given a description of the topic—washing clothes—before the passage. These subjects rated the passage much higher in comprehensibility (4.5 on a 7-point scale) and recalled more than twice as much information (5.8 idea units out of 18). In addition, a third group was given the washing clothes topic after the passage was presented. However, this group performed at about the same low level as the subjects who were given no topic (rating the passage at 2.1 in comprehension and recalling an average of 2.7 idea units). Similar studies [BRAN72, DOOL71, DOOL72] also found that students' recall of ambiguous and technical passages was enhanced when an organizing title, diagram, or sentence was given prior to reading. However these techniques did not have the same facilitating effect when presented after the student had read the passage. These results suggest that the learner must have an appropriate assimilative set available at the time of learning. Even though the same total amount of information may be presented, the students' ability to recall and use the information in the passage is much higher when the clarifying title or picture is given before rather than after reading.

Ausubel [AUSU68] has argued that learning of new technical prose may be enhanced by providing an *advance organizer*—a short expository introduction, presented prior to the text, containing no specific content from the text, but providing the general concepts and ideas that can be used to subsume the information in the text. The first advance organizer studies conducted by Ausubel and his colleagues in the early 1960s [AUSU60, AUSU63, AUSU68] provided some support for this assertion. For example, in a typical study [AUSU60] 120 college students read a 2500-word text on metallurgy after reading either a 500-word advance organizer that presented the underlying framework for the information or a control 500-word historical passage. The advance organizer presented the abstract principles involved in the text. On a reading comprehension posttest covering the basic information in the passage, the advance organizer group performed significantly better than the control group, with scores of 47 percent correct versus 40 percent correct, respectively.

More recently, reviews of the advance organizer literature reveal that advance organizers tend to have their strongest effects in situations where learners are unlikely to already possess useful prerequisite concepts—namely, for technical or unfamiliar material, for "low-ability" or inexperienced students, and for a test involving transfer

to new situations [MAYE79a, MAYE79b]. For example, to study the effects of advance organizers on different kinds of materials, Lesh [LESH76] asked 48 college students to watch a four-hour videotape on finite geometry. An organizer that gave concrete examples and models was provided either before or after instruction. The instructional lesson was organized either in an order of increasing difficulty (hierarchical order) or in an order that repeated key concepts and related new material to previous material (spiral order). Results of a standard posttest indicated that the advance organizer group outperformed the postorganizer group for the hierarchical unit, but the difference was much less for the spiral unit. Similar treatment × material interactions were obtained using social studies lessons [SCHU75] and mathematics lessons [GROT68]. Similarly, Raye [RAYE73] reported that the title biasing effects obtained by Bransford and Johnson [BRAN72] with the washing clothes passage were eliminated when the passage was made more concrete and familiar. Thus there is consistent evidence that organizers have stronger effects for unfamiliar, abstract information than for familiar, concrete information.

In a study investigating the effects of advance organizers on students with high and low ability (or knowledge), physics material was taught to high school students [WEST76]. Advance organizers consisting of concrete models tended to improve test performance of low-ability students but had a much smaller effect for high-ability subjects. Similar group × ability interactions were obtained by several other researchers [RING71, FITZ63, AUSU62, AUSU61, AUSU63, AUSU77, SMIT69]. Thus there is evidence that advance organizers have a stronger effect on low-knowledge or low-ability learners as compared with high-knowledge or high-ability learners.

Finally, in studies involving transfer tests (i.e., problems that are different from those in instruction), there is consistent evidence that advance organizers have a stronger effect on transfer performance than on simple retention. For example, this pattern was obtained with material on mathematical topology [SCAN67], number bases [GROT68, MAYE77], and an imaginary science [MERR66].

Many of the apparent conflicts in the advance organizer literature [BARN75, LAWT77] can be accounted for by the idea that advance organizers find a way of connecting new information with existing knowledge—organizers are not needed for familiar material, experienced learners, or when the test does not involve transfer.

While there is at present no foolproof procedure for generating useful advance organizers, a careful review of the existing literature suggests the following guidelines [MAYE79a]: (1) The organizer should allow the reader to generate all or some of the logical relations in the text. (2) The organizer should provide a means of relating the information in the text to existing knowledge. (3) The organizer should be familiar to the learners. (4) The organizer should encourage the learner to use prerequisite knowledge that the learner would not normally have used. To date, advance organizers have been most effectively used in mathematics and science topics [MAYE79a].

Royer and his colleagues [ROYE75, ROYE76] have demonstrated that concrete models may serve as effective advance organizers in learning new scientific information. In their studies, subjects read two passages, such as a passage on electrical conductivity followed by a passage on heat flow. For some subjects the first passage contained several concrete analogies, such as electrical conduction being described as a chain of falling dominoes. For other subjects the first passage presented the same information in abstract form without any concrete analogies. Reading of the second passage was facilitated if students had been given concrete models in the first passage (e.g., recall of the information in the second passage was about twice that of control groups). Apparently, the models presented in the first passage could be used by learners during the reading of the second passage to help relate the technical terms to familiar concepts.

Similarly, White and Mayer [WHIT80] analyzed physics textbooks to determine how concrete models were used. For example, many textbooks explain Ohm's law

by describing water flowing in pipes, or a boy pushing a heavy load up an inclined street, or electron flow through a circuit. Recent results [MAYE81] show that when concrete analogies are embedded in a technical text, novices tend to perform best on recalling these familiar models and tend to recognize the information adjacent to the model in the text.

2.4 Concrete Models in Computer Programming

In previous sections research was presented concerning the role of manipulatives in mathematics instruction, titles and pictures in remembering ambiguous passages, and advance organizers and models in science text. In each case there was evidence that these techniques serve to provide the learner with appropriate anchoring knowledge that is required for comprehension of new technical information. The present section focuses on research related specifically to computer programming.

Du Boulay and his colleagues [DUBO76, DUBO80] have provided a concrete model for teaching LOGO to children. The model consists of a conceptual LOGO machine with concrete memory locations, switches, and work space, which allow the learner to "work" the machine.

Du Boulay and his colleagues have argued that there are two basic approaches to learning to interact with a computer. The first approach could be called the *black box approach*. In this approach the user develops the attitude that the computer is a black box—you put in commands and data and out comes the answer as if by magic. The mechanisms by which the computer operates are hidden from the user, and the user is likely to assume that computers are just not understandable. Such users are likely to memorize algorithms that "work;" that is, that generate the desired answers. However such users are not able to relate the commands to an understanding of what goes on inside the black box.

The second approach is what can be called the *glass box approach*. In this approach the user attempts to understand what is going on inside the computer. Each command results in some change in the computer and these changes can be described and understood. The level of description need not—indeed should not—be at the "blood and guts" level. Users do not need to become electronics experts. There is an appropriate level of description that Mayer [MAYE79c] refers to as the "transaction level." Similarly, du Boulay et al. [DUBO80] offer two important properties for making hidden operations of a language clearer to a novice: (1) *simplicity*—there should be a "small number of parts that interact in ways that can be easily understood"; (2) *visibility*—novices should be able to view "selected parts and processes" of the model "in action." The LOGO model appears to fit these specifications because it is a simple, familiar model of the computer operations involved in LOGO; in short, it allows the user to develop intuitions about what goes on inside the computer for each line of code. Unfortunately, however, du Boulay and his colleagues have not provided empirical tests concerning whether the LOGO machine model actually influences the problem-solving performance of new learners as compared with traditional methods that emphasize only hands-on experiences.

2.5 Effects of Models on Transfer Performance

In order to provide some information concerning the effects of concrete models on learning computer programming, a series of studies was conducted [MAYE75b]. In the studies, subjects were either given a concrete model of the computer, or they were not given such a model. Then subjects read a manual on a BASIC-like language and took a transfer test on the material.

Method. Figure 2 shows the model of the computer that was used to explain elementary BASIC-like statements to novices. The model provides concrete analogies for four major functional units of the computer: (1) Input is represented as a ticket window at which data are lined up waiting to be processed and placed in the finished pile after being processed; (2) output is represented as a message note pad with one message written per line; (3) memory is represented as an erasable scoreboard in which there is

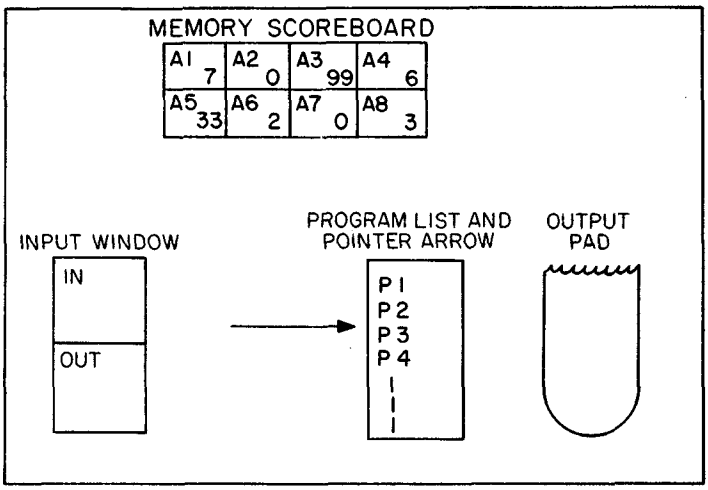

FIGURE 2. A concrete model of the computer for a BASIC-like language [MAYE76].

DESCRIPTION OF MODEL PROVIDED TO SUBJECTS

The figure above represents a simple computer system which you will learn about in this experiment. The computer is made up of three main parts: (1) INPUT and OUTPUT WINDOWS, which allow communication between the computer's memory and the outside world; (2) MEMORY SCOREBOARD, which stores information in the computer; and (3) PROGRAM LIST and POINTER ARROW, which tell the computer what to do and what order to go in. Each of these three parts will now be explained.

Input and Output Window. Notice that to the far left is an input window divided into two parts. A pile of computer cards with numbers punched into them can be put in the left part of the window; as the computer finishes processing each card, it puts the card on the right side of the input window. Thus when the computer needs to find the next data card, it takes the top card on the left side of the input window; when it is done with the card, it puts it on the right side.

On the far right is the output window. This is where printed messages (in this case only numbers can be printed) from the computer's memory to the outside world appear. Each line on the printout is a new message (i.e., a new number).

Thus the computer can store in memory a number that is on a card entered through the input window, or it can print out what it has in memory onto a printout at the output window. The statements which put the input and output windows to work are READ and WRITE statements, and each will be explained later on.

Memory Scoreboard. Inside the computer is a large scoreboard called MEMORY. Notice that it is divided into eight spaces with room for one score (one number) in each space. Also notice that each space is labeled with a name—A1, A2, A3, A4, A5, A6, A7, A8. These labels or names for each space are called "addresses" and each of the eight addresses always has some number indicated in its space. For example, in our figure A1 shows a score of 81 and A2 has the number 17.

It is possible to change the score in any of the eight spaces; for example, the score in box A1 can be changed to 0, and you will learn how to change scores in memory later on when we discuss EQUALS statements and CALCULATION statements.

Program List and Pointer Arrow. Inside the computer to the right of the MEMORY is a place to put a list of things to do called PROGRAM LIST and an arrow which indicates what step in the list the computer should work on.

Notice that each line in the PROGRAM LIST has a number so that the first line is called P1, the second step is P2, and so on. When a program is inserted in the step, the indicator arrow will point to the first line (P1); when the first step is finished, the arrow will go to the next step on the list (P2); and so on down the list. You will learn how to control the order of steps later on when the IF statement, GOTO statement, and STOP statement are discussed.

TABLE 1. SEVEN STATEMENTS USED IN BASIC-LIKE INSTRUCTIONAL BOOKLET

Name	*Example*
READ	P1 READ (A1)
WRITE	P2 WRITE (A1)
EQUALS	P3 A1 = 88
CALCULATE	P4 A1 = A1 + 12
GOTO	P6 GO TO P1
IF	P5 IF (A1 = 100) GO TO P9
STOP	P9 STOP

natural destructive read-in and nondestructive read-out; (4) executive control is represented as a recipe or shopping list with a pointer arrow to indicate the line being executed. This model is similar to du Boulay's model of the LOGO machine in the way it makes the basic operations of the computer visible to the learner. A two- by four-foot diagram containing these parts and a brief one-page description were provided to subjects in the model group (see Figure 2), but no model was given to the control group.

All subjects then were given a ten-page manual that described seven statements modified from BASIC and FORTRAN (see Table 1). For each statement the manual presented the statement, provided the grammar rules for the statement (e.g., definitions of legal address names), and gave an example of the statement as it might occur in a line of a program. Subjects in both groups were given the same manual to read at their own rates, which averaged 20 to 30 minutes.

Following the reading the same test was given to all subjects. The test consisted of six types of problems: (1) *Generate-statement* problems gave a problem in English and required a one-statement program as the solution; (2) *generate-nonloop* problems gave a problem in English and required a short nonlooping program for solution; (3) *generate-looping* problems gave a problem in English and required a looping program for solution; (4) *interpret-statement* problems gave a single-statement program and asked the student to describe what the computer would do; (5) *interpret-nonloop* gave a nonlooping program and asked for a description of what the computer would do; (6) *interpret-looping* problems gave a looping program and required a description of what the computer would do. Examples of the six problems are given in Table 2.

Results. The proportion of correct responses by type of problem for each of the treatment groups is given in Table 3. As can be seen, the control group performs as well or better on problems that are very much like the material in the instructional manual, for example, generate-statement and generate-nonloop. However, on problems that require moderate amounts of transfer[1]—for example, generate-loop and the shorter interpret problems—the model group excels. Both groups do poorly on the very complex interpret-looping programs. The difference in the pattern of performance is consistent with earlier results in other domains in which models enhance transfer performance but not simple retention of presented material. Apparently the model provided an assimilative context in which novices could relate new technical information in the booklet to a familiar analogy. This learning process resulted in a learning outcome that supported some transfer.

2.6 Locus of the Effect of Models

One problem with the foregoing study is that the model subjects received more information than the controls. However, assimilation theory (see Introduction) predicts that presenting a model prior to learning will enhance learning because it provides a meaningful context, but presenting the model after the text will not enhance learning because students will have already encoded the material in a rote way. In further studies [MAYE76] subjects read the same BASIC-like manual, but some subjects were shown a concrete model of the

[1] Transfer problems are problems that are different from those given in the text but can be solved using the information provided. Since information about how to generate single statements and simple programs is given, these two kinds of problems are not transfer problems. Since looping is not mentioned explicitly, problems that require the generation of a looping program are transfer problems. Similarly, since interpretation of programs is not dealt with explicitly, interpretation problems are transfer problems in this study. However, looping interpretation may require much more transfer than the others, since it is most different from this discussion.

TABLE 2. EXAMPLES OF SIX TYPES OF TEST PROBLEMS FOR A BASIC-LIKE LANGUAGE

Generation-Statement	*Interpretation-Statement*
Given a number in memory space A5, write a statement to change that number to zero	A5 = 0
Generation-Nonloop	*Interpretation-Nonloop*
Given a card with a number on it is input, write a program to print out its square	P1 READ (A1) P2 A1 = A1 * A1 P3 WRITE (A1) P4 STOP
Generation-Looping	*Interpretation-Looping*
Given a pile of data cards is input, write a program to print out each number and stop when it gets to card with 88 on it	P1 READ (A1) P2 IF(A1 = 88) GO TO P5 P3 WRITE (A1) P4 GO TO P1 P5 STOP

TABLE 3. PROPORTION OF CORRECT ANSWERS ON TRANSFER TEST BY TYPE OF PROBLEM FOR MODEL AND CONTROL GROUPS[a]

Group	*Generation*			*Interpretation*		
	State-ment	*Nonloop*	*Looping*	*State-ment*	*Nonloop*	*Looping*
Model	.63	.37	.30	.62	.62	.09
Control	.67	.52	.12	.42	.32	.12

[a] Adapted from MAYE75b.
Note. 20 subjects per group; interaction between group and problem type, $p < .05$.

computer before reading while others were shown the same model after reading the manual. Thus subjects in the before group (i.e., those who received the model first) were able to use the model while encoding the material in the text, but the after group (i.e., receiving the model last) was not.

Method. The booklet, model, and test were similar to those used in the previous experiment. The before group received the model first, then the booklet, and then the test. The after group received the booklet first, then the model, and then the test.

Results. The proportion of correct answers by type of problem for the two groups is given in Table 4. As can be seen, the after group (like the controls in the previous study) excelled on retention-like problems (i.e., generation-statement and generation-nonloop), but the before subjects excelled on problems requiring creative transfer to new situations (i.e., generation-loop, interpretation-statement, interpretation-nonloop). Thus these results provide further support for the claim that subjects who use a concrete model during learning develop learning outcomes that support broader transfer. As predicted, the locus of the effect is before rather than after instruction.

2.7 Effects of Models on Recall Performance

The foregoing studies used transfer tests as a measure of what is learned under different instructional techniques. Another technique involves asking subjects to try to write down all they can remember about certain statements. In a follow-up study [MAYE80b] subjects read the same manual and were given the model either before or after reading as in the previous study. However, as a test, subjects were asked to recall all they could about portions of the manual.

Method. The same booklet and model were used as in the previous experiments, with some minor modifications. The before group received the model, then the manual, then the recall test; the after group received the manual, then the model, and then the recall test.

TABLE 4. PROPORTION OF CORRECT ANSWERS ON TRANSFER TEST BY TYPE OF PROBLEM FOR BEFORE AND AFTER GROUPS[a]

Group	*Generation*			*Interpretation*		
	State-ment	*Non-loop*	*Looping*	*State-ment*	*Non-loop*	*Looping*
Before	.57	.50	.20	.47	.63	.17
After	.77	.63	.13	.27	.40	.17

[a] Adapted from MAYE76.
Note. 20 subjects per group; interaction between group and problem type, $p < .05$.

TABLE 5. EXAMPLE OF CONCEPTUAL, FORMAT, AND TECHNICAL IDEA UNITS

Type	*Idea Unit*
Technical	READ is one kind of statement
Format	The format is READ ( )
Format	An address name goes in the parentheses
Conceptual	An address name is a space in the computer's memory
Conceptual	There are eight memory spaces
Technical	The spaces are called A1, A2 ...
Technical	An example is READ (A2)
Conceptual	First the computer checks the number from the top data card
Conceptual	Then that number is stored in space A2
Conceptual	The previous number in A2 is destroyed
Conceptual	Then the data card is sent out of the computer
Conceptual	This reduces the pile of data cards by one
Conceptual	Then go on to the next statements

Results. In order to analyze the recall protocols, the information in the manual was broken down into "idea units." Each idea unit expressed one major idea or action. There were three kinds of idea units in the manual: (1) *conceptual idea units* related to the internal operation of the computer, (2) *technical idea units* that gave examples of code, and (3) *format idea units* that gave grammar rules. Table 5 gives examples of each type of idea unit.

Table 6 shows the average number of correctly recalled idea units from each category for the two groups. As can be seen, the before group recalled more conceptual information, while the after group recalled more technical and format information. This pattern is consistent with the idea that good retention requires recall of specific code, whereas good transfer requires understanding of conceptual ideas. Also, Table 6 shows that the before group included more intrusions about the model and about other idea units from other sections of the booklet, thus suggesting that they integrated the information better. For example, an intrusion is "An address is a slot in the memory scoreboard." The after group, however, included more vague summaries and connectives which served as "filler." For example, a connective is "And that's how READ statements work." Thus, as with the transfer test, subjects given the model before learning showed evidence of more integrated and conceptual learning of technical information.

2.8 Effects of Models on Transfer and Recall Using a Different Language

Although the above results are consistent and were obtained in a long series of studies, their generality is limited by the fact that just one type of language was used. Thus a follow-up study [MAYE80a] was conducted using a file management language based on SEQUEL [GOUL74, REIS77]. The goal of this study was to determine whether the results from previous studies generalize to a new domain.

Method. Subjects read a manual that presented the file management language. For one group of subjects, the model group, the manual began with discussion of a concrete model and related each statement to the model (see Figure 3), but no model was given to the control group. The manuals were informationally equivalent. Each page of the booklet presented one of the eight statements shown in Table 7, along with examples of how the statement fit into a program. Figure 3 presents the concrete model that was used: Long-term memory is

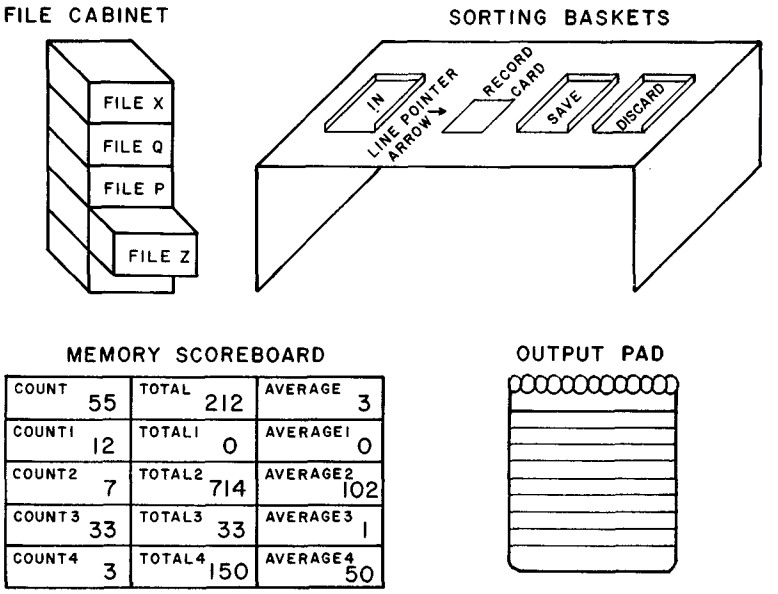

FIGURE 3. A concrete model of the computer for a file management language.

DESCRIPTION OF MODEL PROVIDED TO SUBJECTS

The computer is capable of three main functions: sorting record cards into sorting baskets, remembering numbers on its memory scoreboard, and outputting information to the world through its message pad.

To understand the sorting function of the computer, you could think of an office worker sitting at a desk with three sorting baskets, a line pointer arrow, and file cabinet with many drawers. Each drawer of the file cabinet contains a different set of records; the name of the file is indicated on each drawer. If the worker needs all the records in a particular file, all the worker needs to do is open that drawer and take out all the records. To avoid mixups, the clerk can take out all the records of only one file at a time; if the clerk needs to bring records from a certain file drawer to his desk, first all the records from all other files must be put back in their proper drawers. Thus a worker may have all the records for only *one* file on his desk at a time. These could be placed in the "in-basket" which is on the left side of the clerk's desk—it thus contains all of the to-be-processed record cards, waiting for the office clerk to look at them. In the middle of the desk is a work area with a line pointer arrow; the clerk may place only one card in the work area at a time, and the pointer arrow points to just one line at a time. To the right are two more baskets—the "save basket" and the "discard basket." If a record card passes the clerk's inspection, it is placed on top of the pile of cards in the "save basket"; but if it fails, it is placed in the top of the pile of cards in the "discard basket." The procedure the office worker uses is to take the top card from the "in-basket," place it in the work area with a pointer arrow aimed at one line, and on the basis of inspection of this line move that card to either the "save" or "discard basket." The worker continues until all of the records in the "in-basket" have been processed so that the "in-basket" is empty and the "save" and "discard baskets" contain all the records; then the worker may sometimes be asked to take the pile in either the "save" or the "discard basket" and put it in the "in-basket" for further processing.

To understand the memory function of the computer, think of a memory scoreboard. The scoreboard consists of 15 rectangular spaces like a classroom blackboard, divided into 15 spaces. Each space has a label, such as COUNT2, and each space has one number (of any length) in it. The office worker may count all the records that have been stored in the save basket, and this number could be stored in one of the spaces on the scoreboard. When a new number is stored in a space on the scoreboard, the old number is erased. However, when the office worker copies a number from one of the memory spaces onto the output pad, the number is not erased.

To understand the output function of the computer, think of a telephone message pad. To communicate with the outside world, the computer can write one piece of information on each line of the pad. If it fills all the lines on one page, it will just turn to the next page and begin with the top line. The office worker may write down two kinds of information on the output pad: a number may be copied from one of the spaces on the scoreboard onto the pad (but this does not alter the number on the scoreboard), or information that is on each card in the save basket can be copied onto the output pad.

TABLE 6. AVERAGE NUMBER OF RECALLED IDEA UNITS FOR THE BEFORE AND AFTER GROUPS[a]

Group	*Idea Units*			*Intrusions*		
	Technical	*Format*	*Conceptual*	*Inappropriate*	*Appropriate*	*Model*
	(14)	(12)	(35)			
Before	5.0	1.9	6.6	1.5	1.3	3.1
After	6.0	2.9	4.9	2.5	.8	.5

[a] Adapted from MAYE80b.
Note. 30 subjects per group; interaction between group and problem type, $p < .05$. Numbers in parentheses indicate total possible.

TABLE 7. EIGHT STATEMENTS USED IN FILE MANAGEMENT LANGUAGE BOOKLET

Name	*Example*
FROM	FROM *AUTOMOBILE*
FOR	FOR *WEIGHT* IS CALLED *3000 OR MORE*
AND FOR	AND FOR *COLOR* IS CALLED *GREEN*
OR FOR	OR FOR *MAKE* IS CALLED *FORD*
LIST	LIST *NAME*
COUNT	COUNT
TOTAL	TOTAL *CURRENT VALUE*
LET	LET *TOTAL* ÷ *COUNT* BE CALLED *AVERAGE*

represented as a file cabinet; the sorting function is represented as an in-, out-, and save basket; temporary memory is represented as an erasable scoreboard; executive control is represented as a list and pointer arrow; output is represented as a message pad. The entire model was presented on a two- by three-foot diagram in order to enhance the learner's ability to visualize the system.

After reading the manual, all subjects took the same 20-item test. Problems varied in complexity from generating or interpreting a sort-1 program (with very few operations) to a compute-2 program (with many different statements integrated into one large program). Table 8 lists the five different kinds of programs used.

Results. Table 9 gives the proportion of correct answers by type of problem for the two treatment groups. As can be seen, the control group performed as well as the model group on very simple problems like those in the manual, but the model group excelled on longer problems that require creatively integrating all of the statements in the booklet. Thus, as in the studies with BASIC-like materials, a familiar model serves to enhance performance on creative transfer when it is presented prior to technical instruction.

2.9 Ability

The pattern of results described above tended to be strongest for low-ability subjects [MAYE75b] where ability is defined in terms of SAT mathematics scores. For example, for low-ability subjects the advance organizer increased transfer test performance (55 percent correct) as compared with the control group (45 percent correct), but for high-ability learners the advance organizer group performed more poorly than the control group (55 percent versus 62 percent correct, respectively). Apparently high-ability learners already possess their own useful "models" for thinking about how a computer works, but low-ability students are more likely to lack useful prerequisite knowledge.

2.10 Text Organization

The pattern of results described above also tended to be strongest when material was poorly organized [MAYE78]. For example, the BASIC-like manual was presented either in its original order or in a random order. In the random order, presentation order of paragraphs was randomized. For the randomized version of the manual the advance organizer group performed better on a transfer test than a control group (41 percent versus 31 percent correct, respectively), but for the logical version of the manual the advance organizer group performed as well as but did not outperform

TABLE 8. EXAMPLES OF TEST PROBLEMS FOR A FILE MANAGEMENT LANGUAGE[a]

Sort 1 List the owners' names for all cars weighing 3000 pounds or more.	FROM *AUTOMOBILE* FOR *WEIGHT* IS CALLED *3000 OR MORE* LIST *NAME*
Sort 2 List the owners' names for all model green Fords.	FROM *AUTOMOBILE* FOR *YEAR* IS CALLED *1976 OR MORE* AND FOR *COLOR* IS CALLED *GREEN* AND FOR *MAKE* IS CALLED *FORD* LIST *NAME*
Count How many cars are registered in Santa Barbara County?	FROM *AUTOMOBILE* FOR *HOME COUNTY* IS CALLED *SANTA BARBARA* COUNT LIST *COUNT*
Compute 1 What is the average current value of all cars?	FROM *AUTOMOBILE* COUNT TOTAL *CURRENT VALUE* LET *TOTAL* ÷ *COUNT* BE CALLED *AVERAGE* LIST *AVERAGE*
Compute 2 What percentage of 1977 cars are Chevrolets?	FROM *AUTOMOBILE* FOR *YEAR* IS CALLED *1977* COUNT LET THIS BE CALLED *COUNT 1* AND FOR *MAKE* IS CALLED *CHEVROLET* COUNT LET THIS BE CALLED *COUNT 2* LET *COUNT 2* ÷ *COUNT 1* BE CALLED *AVERAGE* LIST AVERAGE

[a] From MAYE80a.

the control group (36 percent versus 44 percent correct, respectively). Apparently the model is more useful when material is poorly structured because it helps the reader to hold the information together.

2.11 Conclusion

These results provide clear and consistent evidence that a concrete model can have a strong effect on the encoding and use of new technical information by novices. These results provide empirical support to the claims that allowing novices to "see the works" allows them to encode information in a more coherent and useful way [DUBO76, DUBO80]. When appropriate models are used, the learner seems to be able to assimilate each new statement to his or her image of the computer system. Thus one straightforward implication is: If your goal is to produce learners who will not need to use the language creatively, then no model is needed. If your goal is to produce learners who will be able to come up with creative solutions to novel (for them) problems, then a concrete model early in learning is quite useful. More research is needed in order to determine the specific effects of concrete models on what is learned, and to determine the characteristics of a useful model.

TABLE 9. PROPORTION OF CORRECT ANSWERS ON TRANSFER TEST FOR MODEL AND CONTROL GROUPS—FILE MANAGEMENT LANGUAGE[a]

Group	*Type of Test Problem*				
	Sort-1	*Sort-2*	*Count*	*Compute-1*	*Compute-2*
Model	.66	.66	.63	.58	.45
Control	.63	.44	.43	.33	.22

[a] Adapated from MAYE80a.
Note. 20 subjects per group; group × problem-type interaction, $p < .07$.

3. DOES STUDENT ELABORATION ACTIVITY AID MEANINGFUL LEARNING?

3.1 Statement of the Problem

The previous section presented evidence that concrete models may influence learn-

ing of computer programming because they provide a familiar context for assimilating the new material. The second major technique for increasing the meaningfulness of technical information is elaboration—encouraging the learner to explain the information in his or her own words and to relate the material to other ideas or concepts. Elaboration techniques may influence meaningful learning because they encourage the activation of existing knowledge that is relevant for comprehending the newly presented material; that is, elaboration may affect the activation process (see Figure 1).

3.2 Putting It in Your Own Words

There is some evidence that asking subjects to put ideas into their own words during learning can enhance the breadth of learning. For example, Gagne and Smith [GAGN62] asked subjects to give a verbal rationalization for each step as they learned to solve a three-disk version of the Tower of Hanoi problem [EWER32]. These subjects took longer to learn than did those who did not verbalize; however they were able to transfer what they had learned to different problems, such as a six-disk version, much more efficiently (e.g., 3.8 minutes to solution) than the nonverbalizers (e.g., 10.0 minutes to solution).

More recently, Wittrock (WITT74) proposed the idea that "learning is a generative process"—that is, learning occurs when the learner actively generates associations between what is presented and what he already has in memory. As an example, Wittrock presented a study in which elementary school children read a passage and either generated a one-sentence summary for each paragraph or did not. Recall by the students who generated summary sentences was nearly double that of the control group. Apparently, when students are actively encouraged to put information in their own words, they are able to connect the new information to existing knowledge.

Elaboration techniques have long been used by experimental psychologists to enhance the learning of paired associates (such as HOUSE-CASA). For example, when students are asked to actively form mental images or a sentence involving word pairs, paired associate recall is greatly enhanced [BOWE72, PAIV69]. More recently, elaboration techniques have been used in school curricula [DANS78, WEIN78]. For example, in studying human physiology students are asked "How do arteries differ from veins?" Several researchers have argued that students should be given explicit training in "learning strategies" for actively processing new material [ONEI78].

TABLE 10. EXAMPLE OF THE MODEL ELABORATION EXERCISE IN THE PROGRAMMING TEXT

Model Elaboration
Consider the following situation. An office clerk has an in-basket, a save basket, a discard basket, and a sorting area on the desk. The in-basket is full of records. Each one can be examined individually in the sorting area of the desk and then placed in either the save or discard basket. Describe the FOR statement in terms of what operations the clerk would perform using the in-basket, discard basket, save basket, and sorting area.

The following is a series of studies that explore the role of elaboration techniques in learning computer programming. The main theme of this research is to determine how "putting it in your own words" influences the learning of a new computer language.

3.3 Effects of Model Elaboration on Transfer Performance

The first set of studies [MAYE80a] addresses the question of whether elaboration activity influences students' ability to engage in problem solving. In these studies subjects learned a new computer programming language and either were or were not encouraged to describe what they learned in their own words by relating it to a concrete familiar situation.

Method. Subjects read an instructional manual covering an information management language similar to that described in the previous section (see Tables 7 and 8). For subjects in the model elaboration group, there was an elaboration page after each page in the manual, while for subjects in the control group there was no elaboration exercise. The elaboration exercises asked the subject to describe the newly learned statement in terms of operations

TABLE 11. PROPORTION OF CORRECT ANSWERS ON TRANSFER TEST BY TYPE OF PROBLEM FOR MODEL ELABORATION AND CONTROL GROUPS[a]

Group	*Type of Test Problem*				
	Sort-1	*Sort-2*	*Count*	*Compute-1*	*Compute-2*
Model elaboration	.65	.58	.64	.64	.45
Control	.66	.64	.41	.38	.27

[a] Adapted from MAYE80a.
Note. 20 subjects per group; group × problem-type interaction, $p < .05$.

within a concrete model of the computer. Table 10 provides a typical exercise. Then all subjects took the same 20-item problem-solving test as described in the previous section.

Results. Table 11 shows the proportion of correct responses by type of problem for the two groups.[2] As can be seen, the control group performed well on simple retention-like problems, but the model elaboration group performed considerably better on problems requiring creative transfer. Thus there is evidence that requiring the learners to put technical information in their own words through relating the material to a familiar situation results in broader learning outcomes. The results are similar to those given in Table 9 and suggest that model advance organizers and model elaboration have similar effects.

3.4 Effects of Comparative Elaboration on Transfer Performance

In the previous study a concrete situation was presented and the learner asked to relate the new information to it. However, the results are ambiguous in the sense that they may be attributed either to elaboration activity per se or to the fact that additional information (about the concrete model) was presented to the model elaboration group. The purpose of the present studies was to use a kind of elaboration activity that does not add new information [MAYE80a]. Thus a set of studies was conducted in which some subjects were asked to compare newly learned statements in their own words.

[2] These tables are broken down by problem complexity, with more complex problems requiring transfer. The same general pattern is found for both generation and interpretation problems. Table 13 shows data for interpretation problems only, in order to avoid unnecessary complexity. However this table cannot be directly compared with Table 11.

TABLE 12. EXAMPLE OF THE COMPARATIVE ELABORATION EXERCISE IN THE PROGRAMMING TEXT

Comparative Elaboration
How is the FOR command like the FROM command? How is the FOR command different from the FROM command?

TABLE 13. PROPORTION CORRECTION TRANSFER TEST FOR COMPARATIVE ELABORATION AND CONTROL GROUPS[a]

Group	*Type of Problem*				
	Sort-1	*Sort-2*	*Count*	*Compute-1*	*Compute-2*
Comparative elaboration	.90	.90	1.00	.75	.55
Control	.90	.90	.65	.65	.25

[a] Adapted from MAYE80a.
Note. Data are for interpretation problems only; 13 subjects per group; group × problem-type interaction, $p < .05$.

Method. Subjects read the same manual about an information management language as in the previous study. However some subjects were given an elaboration page after each page in the booklet (comparative elaboration group), while for other subjects there was no elaboration (control group). The elaboration activity asked subjects to tell in their own words how two statements were similar and different. Table 12 provides a typical exercise. Then all subjects took the same test as in the previous study.

Results. Table 13 shows the proportion of correct answers by type of problem for the two groups. As can be seen, the control group excelled on retention-like problems, but the comparative elaboration groups excelled on the more complex transfer problems. Thus there is evidence corresponding to that found in the model elaboration studies that asking learners to put technical information in their own words (through making comparisons) results in broader learning that supports transfer.

TABLE 14. AVERAGE NUMBER OF RECALLED IDEA UNITS FOR MODEL ELABORATION, COMPARATIVE ELABORATION, AND CONTROL GROUPS[a]

Group	Type of Idea Units	
	Technical	Conceptual
	(19)	(52)
Model elaboration	5.3	13.9
Comparative elaboration	9.4	14.1
Control	7.5	7.5

[a] Adapted from MAYE80a.
Note. 20 subjects per group; group × type interaction, $p < .05$, for low-ability subjects. Numbers in parentheses indicate total possible.

3.5 Effects of Model and Comparative Elaboration on Recall

The previous studies suggest that elaboration activity can influence transfer performance. As a further test [MAYE80a] subjects were given manuals either with no elaboration questions, model elaboration questions, or comparative elaboration questions. It can be predicted that the elaboration subjects should recall more information that supports transfer—such as conceptual information—while the control group should recall more information about specific statements—such as technical information.

Method. As in the previous study subjects read a manual explaining the information management language that contained either no questions (control group), model questions (model elaboration group), or comparative questions (comparative elaboration group). Then subjects were asked to recall portions of the text.

Results. For purposes of scoring the recall protocols the text was divided into idea units. Some of the idea units presented information about how the computer operated (conceptual idea units), and others emphasized the grammar and technical aspects of each statement (technical idea units). Table 14 shows the average number of idea units recalled by type for the three groups. As can be seen, the control group recalled equal amounts of both types of information, but the elaboration groups each tended to emphasize recall of conceptual as compared with technical information. These results are consistent with the results of the transfer studies in that conceptual information is likely to be needed to support transfer.

TABLE 15. PROPORTION OF CORRECT ANSWERS ON TRANSFER TEST FOR NOTES AND NO-NOTES GROUPS[a]

Subjects	Problem Type	
	Generative	Interpretive
Low ability		
Notes group	.39	.56
No-notes group	.49	.33
High ability		
Notes group	.67	.62
No-notes group	.60	.60

[a] From PEPE78.
Note. 15 subjects per group; effect of ability, $p < .01$; interaction between group ability and problem type, $p < .025$.

3.6 Effects of Note Taking on Transfer and Recall Performance

The foregoing series of studies provides some evidence that elaboration techniques influence the breadth of learning. However, the generality of the results is limited by the fact that just one type of manual and two types of elaboration activity were used. In addition, previous studies did not control for amount of reading time. Thus an additional series of studies [PEPE78] was conducted using a different language (a BASIC-like language) and a different elaboration activity (note taking).

Method. Subjects watched a 20-minute videotape lecture, similar to the manual described earlier, describing seven BASIC-like statements. Some subjects were asked to take notes by putting the basic information in their own words. Other subjects simply viewed the lecture without taking notes. As a test subjects were given problems to solve or asked to recall portions of the lesson. Videotape presentations were controlled for presentation time in the two groups.

Results. Table 15 gives the proportion of correct answers on generative problems (similar to those in the lesson) and on interpretation problems (which were not in the lesson). As can be seen, for low-ability subjects (based on SAT mathematics scores) there is a pattern in which note

TABLE 16. AVERAGE NUMBER OF RECALLED IDEA UNITS FOR NOTES AND NO-NOTES GROUPS[a]

Group	Type of Idea Units		
	Technical	*Conceptual*	*Intrusions*
	(28)	(36)	
Notes	10.4	7.2	3.9
No-notes	9.4	4.7	2.4

[a] Adapted from PEPE78.
Note. 20 subjects per group; interaction between group and type of recall, $p < .025$. Numbers in parentheses indicate total possible.

taking helps performance on transfer but hurts performance on the retention-like problems. For high-ability subjects note taking has no effect, presumably because high-ability learners already possess strategies for actively assimilating the new information.

Table 16 shows recall of the lecture by type of idea unit for the two groups. As can be seen, the note takers recalled more conceptual information, but there is no difference between the groups in recall of technical information. Thus the results are consistent with the model elaboration and comparative elaboration studies concerning the effects of asking subjects to put new technical information in their words during learning.

3.7 Conclusion

The goal of elaboration is to help the learner describe the key concepts in his own words, using his existing knowledge. Unfortunately there is no foolproof way to design useful elaboration activities. Emphasis on format or grammatical details and emphasis on errorless verbatim recall of statements will not produce the desired effects. The learner should be able to describe the effects of each program statement in his own words.

4. UNDERSTANDING COMPUTER PROGRAMMING

The previous sections have focused on the issue of *how* to teach novices. This section briefly examines the issue of *what* to teach. Greeno [GREE76] has argued that instruction for problem-solving tasks should be based on *cognitive objectives*—statements of what the learner should have in his or her head at the end of instruction. Two major objectives that are relevant to enhancing a novice's understanding of computer programming are knowledge for understanding a statement and knowledge for understanding a program.

4.1 Understanding a Statement

What does it mean to say that someone "understands" a certain statement? In a recent analysis of BASIC each statement is described as a "transaction" [MAYE79c]. A transaction consists of an action, an object, and a location in the computer. For example, the statement LET $X = 5$, consists of the following six transactions:

(1) Find the number indicated on the right of the equal sign (ACTION: Find; OBJECT: Number; LOCATION: Program).
(2) Find the number in the memory space indicated on the left of the equal sign (ACTION: Find; OBJECT: Number; LOCATION: Memory).
(3) Erase the number in that memory space (ACTION: Destroy; OBJECT: Number; LOCATION: Memory).
(4) Write the new number in that memory space (ACTION: Create; OBJECT: Number; LOCATION: Memory).
(5) Go on to the next statement (ACTION: Move; OBJECT: Pointer; LOCATION: Program).
(6) Do what it says (ACTION: Allow; OBJECT: Command; LOCATION: Program).

Thus there is a general structure for each transaction; some action can be expected to be carried out on some object in some location in the computer. The two techniques cited in previous sections can be applied to teaching a transaction-type analysis of statements. It may be noted that statements with the same name may actually consist of different actions. For example, a "Counter Set LET" as given above is different from an "Arithmetic LET" such as LET $X = 5/2$. Explicit naming and describing of different types of statements with the same keyword may become a part of computer instruction.

4.2 Understanding a Program

What do experts know about computer programming that beginners do not know? One answer is that experts possess much more information and that the information is organized more efficiently. For example, a review of research on teaching people how to become better problem solvers concludes that good problem solving requires that the user have domain-specific knowledge: "All problem solving is based on knowledge" [GREE80]. Similarly, Simon [SIMO80] estimates that a person needs 50,000 chunks of domain-specific information to become an expert in some domain.

In a classic study subjects were asked to view briefly presented chessboard configurations and then to reconstruct them [CHAS73]. Chess masters performed much better than less experienced players on reconstructing board configurations if the board positions came from actual games; however the advantage was lost when random board patterns were presented. This finding suggests that experts in chess do not necessarily have better memories, but rather a repertoire of many meaningful patterns of board positions. They can chunk several pieces together into one meaningful pattern, while a less experienced player must try to remember each piece separately. In an analogous study reported by Shneiderman [SHNE80], experienced and inexperienced programmers were given programs to study. Experienced programmers were able to recall many more lines of code than inexperienced programmers when the program was a meaningful running program; however, when the program consisted of random lines of code, the two groups performed at similar levels. Apparently, the experts were able to chunk lines of code together into chunks, while less experienced users were less able to form such chunks.

For example, Atwood and Ramsey [ATWO78] suggest that experienced programmers encode a segment such as

```
SUM = 0
DO 1 I = 1, N
    SUM = SUM + (I)
1 CONTINUE
```

as "CALCULATE THE SUM OF ARRAY X." An experienced programmer has a "schema" for this task and is able to generate a variety of lines of code to accomplish it. In order to provide a more precise description of the "schemata" that are involved in understanding programs, Atwood and Ramsey [ATWO80] used a modified version of Kintsch's [KINT74] propositional analysis. Each statement in the program can be written as a predicate with arguments, and a macrostructure can be constructed. Although a detailed description of Atwood and Ramsey's system is beyond the scope of this paper, their work is promising in that it suggests that knowledge can be represented precisely.

One implication of this work is that it might be possible to teach the major "chunks" or "schemata" involved in computer programming explicitly using the techniques cited in previous sections. Explicit naming and teaching of basic schemata such as these may become part of computer programming curricula.

5. SUMMARY

This paper is concerned with how to make computers and computer programming more understandable for novices. Two instructional techniques from educational and cognitive psychology are described—using concrete models to represent the computer system and encouraging the learner to describe technical information in his own words. A review of the effectiveness of these techniques reveals that, under certain conditions, both techniques may enhance the learner's understanding as measured by ability to solve problems that were not explicitly taught. Finally, two major objectives of computing instruction are suggested—enhancing the novice's ability to understand (1) the meaning of individual program statements and (2) the program schemata that give the statements a higher level meaning.

ACKNOWLEDGMENTS

I wish to thank Tom Moran for his editorial comments. A shorter version of this paper was presented at the National Science Foundation Conference on National Computer Literacy Goals for 1985, held in Reston, Va., on December 18–20, 1980. Preparation of this paper was supported by Grant NIE-G-0118 from the National Institute of Education and Grant SED-80-14950 from the National Science Foundation.

REFERENCES

ANDE77 ANDERSON, R.C. "The notion of schemata and the educational enterprise," in *Schooling and the acquisition of knowledge*, R.C. Anderson, R.J. Spiro, and W.E. Montague, Eds., Erlbaum, Hillsdale, N.J., 1977.

ATWO78 ATWOOD, M.E., AND RAMSEY, H.R. "Cognitive structure in the comprehension and memory of computer programs: An investigation of computer programming debugging," ARI Tech. Rep. TR-78-A210, Science Applications, Englewood, Colo., Aug. 1978.

AUSU60 AUSUBEL, D.P. "The use of advance organizers in the learning and retention of meaningful verbal material," *J. Educ. Psychol.* **51** (1960), 267–272.

AUSU61 AUSUBEL, D.P., AND FITZGERALD, D. "The role of discriminability in meaningful verbal learning and retention," *J. Educ. Psychol.* **52** (1961), 266–274.

AUSU62 AUSUBEL, D.P., AND FITZGERALD, D. "Organizer, general background, and antecedent learning variables in sequential verbal learning," *J. Educ. Psychol.* **53** (1962), 243–249.

AUSU63 AUSUBEL, D.P. *The Psychology of meaningful verbal learning*, Grune and Stratton, New York, 1963.

AUSU68 AUSUBEL, D.P. *Educational psychology: A cognitive view*, Holt, Rinehart and Winston, New York, 1968.

AUSU77 AUSUBEL, D.P., NOVAK, J.D., AND HANESIAN, R. *Educational psychology: A cognitive view*, 2nd ed., Harper and Row, New York, 1977.

BARN75 BARNES, B.R., AND CLAWSON, E.U. "Do advance organizers facilitate learning? Recommendations for further research based on an analysis of 32 studies," *Rev. Educ. Res.* **45** (1975), 637–659.

BART32 BARTLETT, F.C. *Remembering*, Cambridge University Press, Cambridge, England, 1932.

BOWE72 BOWER, G.H. "Mental imagery and associating learning," in *Cognition in learning and memory*, L. Gregg, Ed., Wiley, New York, 1972.

BRAN72 BRANSFORD, J.D., AND JOHNSON, M.K. "Contextual prerequisites for understanding: Some investigations of comprehension and recall," *J. Verbal Learn. Verbal Behav.* **11** (1972), 717–726.

BRAN79 BRANSFORD, J.D. *Human cognition*, Wadsworth, Monterey, Calif., 1979.

BROW49 BROWNELL, W.A., AND MOSER, H.E. "Meaningful vs. mechanical learning: A study in grade III subtraction," in *Duke University Research Studies in Education, no. 8*, Duke University Press, Durham, N.C., 1949, pp. 1–207.

CHAS73 CHASE, W.G., AND SIMON, H.A. "Perception in chess," *Cognitive Psychol.* **4** (1973) 55–81.

DANS78 DANSEREAU, D. "The development of a learning strategies curriculum," in *Learning strategies*, H.F. O'Neal, Ed., Academic Press, New York, 1978.

DOOL71 DOOLING, D.J., AND LACHMAN, R. "Effects of comprehension on the retention of prose," *J. Exp. Psychol.* **88** (1971), 216–222.

DOOL72 DOOLING, D.J., AND MULLET, R.L. "Locus of thematic effects on retention of prose," *J. Exp. Psychol.* **97** (1973), 404–406.

DUBO76 DU BOULAY, B., AND O'SHEA, T. "How to work the LOGO machine," Paper No. 4, Dep. Artificial Intelligence, Univ. Edinburgh, Scotland, 1976.

DUBO80 DU BOULAY, B., O'SHEA, T., AND MONK, J. "The black box inside the glass box: Presenting computering concepts to novices," Paper no. 133, Dep. Artificial Intelligence, Univ. Edinburgh, Scotland, 1980.

EWER32 EWERT, P.H., AND LAMBERT, J.F. "The effect of verbal instructions upon the formation of a concept," *J. Gen. Psychol.* **6** (1932), 400–411.

FITZ63 FITZGERALD, D., AND AUSUBEL, D.P. "Cognitive versus affective factors in the

learning and retention of controversial material," *J. Educ. Psychol.* **54** (1963), 73–84.

GAGN62 GAGNE, R.M., AND SMITH, E.C. "A study of the effects of verbalization on problem solving," *J. Exp. Psychol.* **63** (1962), 12–18.

GOUL74 GOULD, J.D., AND ASCHER, R.M. "Query by non-programmers," IBM Research Rep., Yorktown Heights, N.Y., 1974.

GREE76 GREENO, J.G. "Cognitive objectives of instruction," in *Cognition and instruction*, D. K. Dahr, Ed., Erlbaum, Hillsdale, N.J., 1976.

GREE80 GREENO, J.G. "Trends in the theory of knowledge for problem solving," in *Problem solving and education: Issues in teaching and research*, D.T. Tuma and F. Reif, Eds., Erlbaum, Hillsdale, N.J., 1980.

GROT68 GROTELUESCHEN, A., AND SJOGREN, D.D. "Effects of differentially structured introductory materials and learning tasks on learning and transfer," *Am. Educ. Res. J.* **5** (1968), 277–202.

KATO42 KATONA, G. *Organizing and memorizing*, Columbia University Press, New York, 1942.

KINT74 KINTSCH, W. *The representation of meaning in memory*, Erlbaum, Hillsdale, N.J., 1974.

KOHL25 KOHLER, W. *The mentality of apes*, Liveright, New York, 1925.

LAWT77 LAWTON, J.T., AND WANSKA, S.K. "Advance organizers as a teaching strategy: A reply to Barnes and Clawson," *Rev. Educ. Res.* **47** (1977), 233–244.

LESH76 LESH, R.A. "The influence of an advance organizer on two types of instructional units about finite geometrics," *J. Res. Math. Educ.* **7** (1976), 82–86.

MAYE75a MAYER, R.E. "Information processing variables in learning to solve problems," *Rev. Educ. Res.* **45** (1975), 525–541.

MAYE75b MAYER, R.E. "Different problem-solving competencies established in learning computer programming with and without meaningful models," *J. Educ. Psychol.* **67** (1975), 725–734.

MAYE76 MAYER, R.E. "Some conditions of meaningful learning for computer programming: Advance organizers and subject control of frame order," *J. Educ. Psychol.* **68** (1976), 143–150.

MAYE77 MAYER, R.E. "Different rule system for counting behavior acquired in meaningful and rote contexts of learning," *J. Educ. Psychol.* **69** (1977), 537–546.

MAYE78 Mayer, R.E. "Advance organizers that compensate for the organization of text," *J. Educ. Psychol.* **70** (1978), 880–886.

MAYE79a MAYER, R.E. "Can advance organizers influence meaningful learning?" *Rev. Educ. Res.* **49** (1979), 371–383.

MAYE79b MAYER, R.E. "Twenty years of research on advance organizers: Assimilation theory is still the best predictor of results," *Instr. Sci.* **8** (1979), 133–167.

MAYE79c MAYER, R.E. "A psychology of learning BASIC," *Commun. ACM* **22** (1979), 589–594.

MAYE80a MAYER, R.E. "Elaboration techniques for technical text: An experimental test of the learning strategy hypothesis," *J. Educ. Psychol.* **72** (1980), 770–784.

MAYE80b MAYER, R.E., AND BROMAGE, B. "Different recall protocols for technical text due to advance organizers," *J. Educ. Psychol.* **72** (1980), 209–225.

MAYE81 MAYER, R.E., AND COOK, L. "Effects of shadowing on prose comprehension and problem solving," *Memory and Cognition* **8** (1981).

MERR66 MERRILL, M.D., AND STOLUROW, L.M. "Hierarchical preview vs. problem oriented review in learning in imaginary science," *Am. Educ. Res. J.* **3** (1966), 251–261.

MINS75 MINSKY, M. "A framework for representing knowledge," in *The psychology of computer vision*, P. Winston, Ed., McGraw-Hill, New York, 1975.

ONEI78 O'NEIL, H.F. *Learning strategies*, Academic Press, New York, 1978.

PAIV69 PAIVIO, A. "Mental imagery in associative learning and memory," *Psychol. Rev.* **76** (1969), 241–263.

PEPE78 PEPER, R.J., AND MAYER, R.E. "Note taking as a generative activity," *J. Educ. Psychol.* **70** (1978), 514–522.

RAYE73 RAYE, C. "Considerations of some problems of comprehension," in *Visual information processing*, W.G. Chase, Ed., Academic Press, New York, 1973.

REIS77 REISNER, P. "Use of psychological experimentation as an aid to development of a query language," *IEEE Trans. Softw. Eng.* **SE-3** (1977), 218–229.

RESN80 RESNICK, L.B., AND FORD, S. *The psychology of mathematics learning*, Erlbaum, Hillsdale, N.J., 1980.

RING71 RING, D.G., AND NOVAK, J.D. "Effects of cognitive structure variables on achievement in college chemistry," *J. Res. Sci. Educ.* **8** (1971), 325–338.

ROYE75 ROYER, J.M., AND CABLE, G.W. "Facilitated learning in connected discourse," *J. Educ. Psychol.* **67** (1975), 116–123.

ROYE76 ROYER, J.M., AND CABLE, G.W. "Illustrations, analogies, and facilitative transfer in prose learning," *J. Educ. Psychol.* **68** (1976), 205–209.

RUME75 RUMELHART, D.E. "Notes on a schema for stress," in *Representation and understanding*, D.B. Bobrow and A. Collins, Eds., Academic Press, New York, 1975.

SCAN67 SCANDURA, J.M., AND WELLS, J.N. "Advance organizers in learning ab-

stract mathematics," *Am. Educ. Res. J.* **4** (1967), 295–301.

SCHA77 SCHANK, R.C., AND ABELSON, R.P. *Scripts, plans, goals and understanding,* Wiley, New York, 1977.

SCHU75 SCHUMACHER, G.M., LIEBERT, D., AND FASS, W. "Textural organization, advance organizers, and the retention of prose material," *J. Reading Behav.* **7** (1975), 173–180.

SHNE80 SHNEIDERMAN, B. *Software psychology: Human factors in computer and information systems,* Winthrop, New York, 1980.

SIMO80 SIMON, H.A. "Problem solving and education," in *Problem solving and education: Issues in teaching and research,* D.T. Tuma and F. Reif, Eds., Erlbaum, Hillsdale, N.J., 1980.

SMIT69 SMITH, R.J., AND HESSE, K.D. "The effects of prereading assistance on the comprehension and attitudes of good and poor readers," *Res. Teach. Engl.* **3** (1969), 166–167.

THOR77 THORNDYKE, P.W. "Cognitive structures in comprehension and memory of narrative discourse," *Cognitive Psychol.* **9** (1977), 77–110.

WEAV72 WEAVER, F., AND SUYDAM, M. "Meaningful instruction in mathematics education," in *Mathematics Education Reports,* 1972.

WEIN78 WEINSTEIN, C. "Elaboration skills as a learning strategy," in *Learning strategies,* H.F. O'Neil, Ed., Harper and Row, New York, 1978.

WERT59 WERTHEIMER, M. *Productive thinking,* Harper and Row, New York, 1959.

WEST76 WEST, L.H.T., AND FENSHAM, D.J. "Prior knowledge or advance organizers as effective variables in chemistry learning," *J. Res. Sci. Teach.* **13** (1976), 297–306.

WHIT80 WHITE, R.T., AND MAYER, R.E. "Understanding intellectual skills," *Instr. Sci.* **9** (1980), 101–127.

WITT74 WITTROCK, M.C. "Learning as a generative process," *Educ. Psychol.* **11** (1974), 87–95.

RECEIVED APRIL 1980; FINAL REVISION ACCEPTED JANUARY 1981