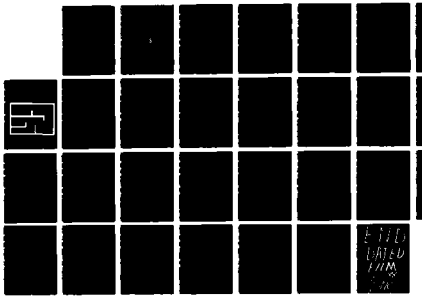
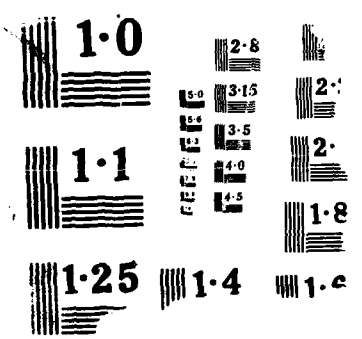
AD-A193 519 KNOWLEDGE ORGANIZATION AND THE ACQUISITION OF PROCEDURAL EXPERTISE(U) BBN LABS INC CAMBRIDGE MA C ZEITZ ET AL. OCT 87 BBN-6633 MDA903-87-C-0545

UNCLASSIFIED F/G 12/5 NL




1·0
1·1
1·25
1·4
2·8
3·15
3·5
4·0
4·5
2·
1·8

BBN Laboratories Incorporated

A Subsidiary of Bolt Beranek and Newman Inc.

Report No. 6633

Knowledge Organization and the Acquisition of Procedural Expertise

Colleen M. Zeitz
Brown University

Kathryn T. Spoehr
BBN Laboratories and Brown University

October 1987

Prepared for:

The U. S. Army Research Institute

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) Report No. 6633	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION BBN Laboratories 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION
6c ADDRESS (City, State, and ZIP Code) 10 MOULTON STREET CAMBRIDGE, MA 02238	7b ADDRESS (City, State, and ZIP Code)
8a NAME OF FUNDING/SPONSORING ORGANIZATION ARMY RESEARCH INSTITUTE 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA-903087-C-0545
8c ADDRESS (City, State, and ZIP Code) 5001 EISENHOWER AVENUE ALEXANDRIA, VA 22233	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
KNOWLEDGE ORGANIZATION AND THE ACQUISITION OF PROCEDURAL EXPERTISE

12 PERSONAL AUTHOR(S)
COLLEEN ZEITZ AND KATHRYN T. SPOEHR

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
TECHNICAL REPORT	FROM 11/86 TO 11/87	OCTOBER 1987	42

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	PROBELM SOLVING, ELECTRONICS, TROUBLESHOOTING, SKILL ACQUISITION, EXPERTISE

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The influence of the organization of a declarative knowledge base on the development and application of proceduralized knowledge was investigated in a complex troubleshooting domain. Although the two explanatory structures led to similar training performance, the two groups differed significantly in their overall level of performance in subsequent troubleshooting problems. Examination of objective measures of troubleshooting performance and think-aloud prototols indicated the breadth-first declarative knowledge representation fosters the use of mental models during problem solving in training. It also facilitates proceduralization of that knowledge into fast and accurate methods for localizing faults.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Dr. Judith Orasanu	22b TELEPHONE (Include Area Code) 202-274-5590 / 22c OFFICE SYMBOL

DD Form 1473, JUN 86 Previous editions are obsolete SECURITY CLASSIFICATION OF THIS PAGE

Abstract

The influence of the organization of a declarative knowledge base on the development and application of proceduralized knowledge was investigated in a complex troubleshooting domain. Domain explanations were generated in either depth-first or breadth-first manner for different groups of subjects who were also given experience learning to troubleshoot in the domain. Although the two explanatory structures led to similar training performance, the two groups differed significantly in their overall level of performance in subsequent troubleshooting problems. Examination of objective measures of troubleshooting performance and think-aloud protocols indicated that breadth-first declarative knowledge representation fosters the use of mental models during problem solving in training. It also facilitates proceduralization of that knowledge into fast and accurate methods for localizing faults.

Table of Contents

1. Introduction

One of the most important issues facing educators and designers of computer-based instructional systems is how to organize a body of knowledge in such a way as to optimize the acquisition of that knowledge by novices. The premise of this paper is that the well-structured presentation of information about a complex device can affect the knowledge organization of the user, which can, in turn, result in superior performance involving application of this knowledge in problem solving. The structure of the declarative knowledge improves performance by facilitating the creation of proceduralized representations.

In order to understand how to best convey the information required for effective performance, it is necessary to know the characteristics of relevant information possessed by expert problem solvers. Chi and Glaser (1985) contend that the cornerstone of expert problem solving is a large body of well-structured domain knowledge. They sugest that experts in a variety of domains, for example (Chase & Simon, 1973) and architecture (Akin, 1980), organize multiple stimuli into a smaller number of meaningful units called chunks. Experts are thus able to view a complicated situation in their domain of expertise in terms of a small number of chunks, and they encode especially complex situations as a hierarchy of chunks.

More evidence linking hierarchically organized knowledge and expert performance in the domain of troubleshooting comes from Rasmusen and Jensen (1974). Their electronics technicians used a general top-down hierarchical search through the subunits of the system rather than attacking a problem by consulting the documentation of the broken device. One reason for the prominence of general search procedures in this study may be that because the technicians repaired a wide array of equipment, they lacked frequent experience with each individual type of device, and thus more specific procedures may not have been optimal for their jobs. Despite this, the technicians were still able to discern the hierarchical structures of the devices they repaired and to use these to guide their search for faults. Hence their conception of electronic devices must have been based on some general, flexible, hierarchical representation.

When expert programmers design a complex program, they appear to construct their solution by a top-down, breadth-first, iterative method (Jeffries, Turner, Polson and Atwood, 1981). They generate an algorithm by successively decomposing abstract, higher-level problems into more manageable subproblems. In contrast, Jeffries et al. describe a novice's approach as mostly linear. Some novices worked from a middle level of abstraction of a central function outward, while another intermixed lines of quite specific code with general plans. Thus, an important difference between the expert and novice programmers in this study was that the experts exhibited a breadth-first hierarchical organization both in their conception of the problem and in their designs for solution.

Related evidence linking hierarchical representations and expertise can be found in the way advanced programmers attempt to memorize randomly ordered lines of computer code. By analyzing the consistency of the ordering of the lines across repeated attempts at recall by each subject, Adelson (1981) was able to demonstrate that the experts in her study had hierarchically reorganized the computer code on the basis of the procedural similarity of the lines. In contrast, novice programmers grouped the lines on a more superficial basis, according to their syntactic similarity. The novices were less consistent in their application of this mode of organization and recalled fewer of the lines of code than the experts. Thus, characteristics of expertise demonstrated in this study include the ability to impose a hierarchical organization on material to be memorized, and sensitivity to the procedural aspects of the stimuli.

A breadth-first approach also seems to be effective in the domain of medical diagnosis. Johnson, Duran, Hassebrock, Moller, Prietula, Feltovich and Swanson (1981) characterized subjects as using a breadth-first approach if they generated all the possible diseases which could be indicated by a set of symptoms before evaluating the relative merits of each. The technique of subjects who proposed and maintained one diagnosis until it was disproved was described as depth-first. Both of the subjects in their study who were characterized as breadth-first diagnosed

the symptoms correctly while only two of the seven subjects classified as depth-first produced the correct solution. Thus, the breadth-first approach appeared to lead to a successful diagnosis more frequently than the depth-first approach.

One of the most comprehensive analyses of skilled troubleshooting is Gitomer's (1984) assessment of the knowledge bases and cognitive abilities of groups of repairmen rated as highly proficient and less proficient by their supervisors. He found that the two groups were equally competent at describing components and identifying, describing, and recalling circuits, and demonstrated equivalent knowledge of basic electronic principles. However, the more proficient subjects were capable of describing a device at a more abstract level and displayed a greater degree of automaticity with common operators than the unskilled subjects. The most important difference was in the two groups' ability to conceptualize the problem space: the skilled troubleshooters seemed to view an electronic device as an integrated system of individual components while the less skilled subjects' model consisted of "bunches of cards to be swapped" (p. 47). Accordingly, proficient subjects spent relatively more time analyzing symptoms and tracing circuits, thereby attempting to gain a mental model of the problem, and less time running tests and swapping parts than the less proficient subjects. Gitomer describes the proficient subjects' method for troubleshooting as the systematic narrowing of the problem space, which is equivalent to the top-down hierarchical search reported by the technicians in Rasmussen and Jensen's (1974) study. As suggested earlier, a hierarchical model of the faulty device is required to account for this type of behavior. In contrast, Gitomer's less proficient subjects produced malfunction hypotheses such as, "There must be a short somewhere" (p.37), demonstrated either the lack of a hierarchical model or the inability to apply such a model.

Differences in knowledge organization may explain the discrepancy between Gitomer's two groups' performance, despite their apparently comparable declarative knowledge of electronics. The highly skilled group's search method and system model hierarchical and perhaps their

declarative knowledge base was organized in this fashion as well. Glaser (1985) suggests "The nature of this organization [of domain-related knowledge] determines the quality, completeness and coherence of the internal representation, which in turn determines the quality of further thinking" (p. 917). This suggests that a hierarchical organization may be necessary in order for the relevant knowledge a troubleshooter actually possesses to be evidenced in his or her problem solving attempts. Thus, Gitomer's unskilled repairmen know the same facts about the electronic devices as the other group, but the lack of structure in their understanding of the system prevents them from using this knowledge to competently troubleshoot the system.

The work of Anderson (1982) suggests that well-organized declarative knowledge is not sufficient by itself to produce expert problem-solving. He claims that initial attempts at solving a new kind of problem are performed by slow interpretation of one's relevant declarative knowledge, and that eventually, often-repeated sequences become compiled into procedural knowledge. Such proceduralization permits a skilled troubleshooter to mentally simulate how a device or subsection of a device should run, an ability that has come to be known as having a mental model of the system (Gentner and Stevens, 1983). To troubleshoot a complex device, a hierarchy of mental models, some detailed, encompassing a small portion of the system and some abstract, encompassing the whole device, is necessary. Such models could be used to test hypotheses about a fault location. One could introduce that fault into a mental model of the device, and then match the resulting predictions against the observed behavior. From experience with a specific malfunction, subjects could also form a mental model of how a system behaves when a particular fault occurs. Thus knowledge gained from individual troubleshooting episodes can also be proceduralized.

Taken together, the literature presents a picture of an expert who organizes domain knowledge hierarchically, has a large amount of proceduralized domain knowledge, and easily forms representations or mental models of problems within the domain. This suggests that the

hierarchical structure of declarative domain knowledge affects the likelihood of procedure formation. Certainly the repeated application of knowledge enables the formation of procedures as Anderson (1982) suggests, but there is an added constraint that practice in application only results in the proceduralization of knowledge that already has been appropriately structured. Before efficient procedures can be formed, declarative knowledge must be organized according to level of abstraction.

There are two methods by which the hierarchical organization of a declarative knowledge base might be imparted. The first is what might be called a depth-first approach and is commonly used as an explanatory mechanism in intelligent, computer-based instructional systems (e.g., Weld, 1983). Understanding is built up from explanations of each of the lowest level elements in the system (i.e., the deepest components in the hierarchical tree), with the elements being combined at successively higher levels until the structure of the entire domain or system has been described. For example, the description of a home heating system could begin with an explanation of how each part in the furnace works, followed by a description of how these parts are organized into larger functional units such as the burner and the water circulation reservoir. These larger units would then be combined in the explanation of the furnace as a whole. Similar descriptions of the thermostat, the baseboards, and so forth could also be constructed. Only when each of the major subsystems had been fully explained would their interworkings in the whole system be explained. The alternative explanatory method is to first generate an explanation at the highest level of domain knowledge, and then to recursively decompose the representation in a breadth-first manner, one level at a time, until the lowest level of information has been reached. Thus one could explain the same heating system first in terms of the overall functions of the major subsystems, such as the furnace and thermostat, followed by more detailed explanations of the major parts of each subsystem, until the level of individual parts was reached.

There is some evidence that explaining hierarchical organization by giving top-level, breadth-first information initially facilitates the acquisition of declarative knowledge. Smith and Goodman (1984) demonstrated that explanatory, hierarchical instructions allowed subjects to construct circuits more efficiently than bare-bones linear instructions. The hierarchical instructions facilitated the reading of, execution of, and memory for the instructional material. They argued that the well-organized structural and functional information they used gave rise to schemas, which, in turn, helped performance in three ways: 1) by filling gaps and forming connections between steps the subjects had to perform, 2) by forming a framework or "scaffold" on which to attach following information, and 3) by directing memory search. Smith and Goodman also suggested that breadth-first organization facilitated chunking of information; thus, they were suggesting that the structure of the input could have a positive effect on the subject's conceptualization of the device. An unresolved question raised by their study is whether it was the mere hierarchical organization of the structural and functional explanations, the conjunction of the hierarchical organization with the extra explanatory context, or just the explanatory material alone that actually was responsible for the observed facilitation. Further, Smith and Goodman provide only weak evidence on the issue of how hierarchical organization facilitates the construction of troubleshooting or problem solving procedures.

If experts organize their knowledge representations hierarchically and have proceduralized representations, and if hierarchically organized instructional materials facilitate learning, then it should be possible to accelerate some subjects' acquisition of procedural expertise relative to others by varying the organization with which domain knowledge is presented. The present study monitors the course of the development of troubleshooting expertise in a moderately complex domain by observing changes in the organization and proceduralization of the knowledge base. The study also seeks to determine if the course of this development can be altered by the way in which the initial information about the domain is presented to novices. In particular, it compares the knowledge and troubleshooting expertise acquired by two groups of subjects: those who learn

about the domain through a breadth-first exaplanatory sequence, and those who learn via a depth-first explanation.

Three specific questions are addressed by the present study. First, is it the case that subjects who are given hierarchical, breadth-first instructions will induce a more hierarchical representation of a complex physical system than subjects given the same information in a different order? Second, will the knowledge subjects in the breadth-first condition use to troubleshoot the system will of a more procedural nature? And finally, will subjects who received breadth-first instructions be able to troubleshoot the system more efficiently than the depth-first subjects (plausibly because of their more hierarchical, proceduralized knowledge base)?

1.1 Methodological Considerations

The present study examines the acquisition of these aspects of expertise through a longitudinal study of a few subjects' transitions from novices to experts. A longitudinal approach was taken not only because it permits the desired variation in instructional methodology, but also because it obviates an important problem in most studies of expert/novice differences. That is that the between-groups nature of such studies allows the possibility that the experts, being a self-selected group, may be quite different from a general set of novices in both cognitive and personal characteristics. This makes it impossible to attribute differences in performance entirely to differences in domain-specific experience.

In order to investigate the effects of the organization of declarative information in a well-controlled environment, a novel domain was constructed that was both hierarchical and complex enough to require sophisticated troubleshooting skills, while still being masterable in a reasonable amount of time. Subjects were taught about this domain using one of two structured explanatory frameworks and were given experience in troubleshooting each part of it. After training, performance was tested using two qualitatively different types of data. Some of the data were

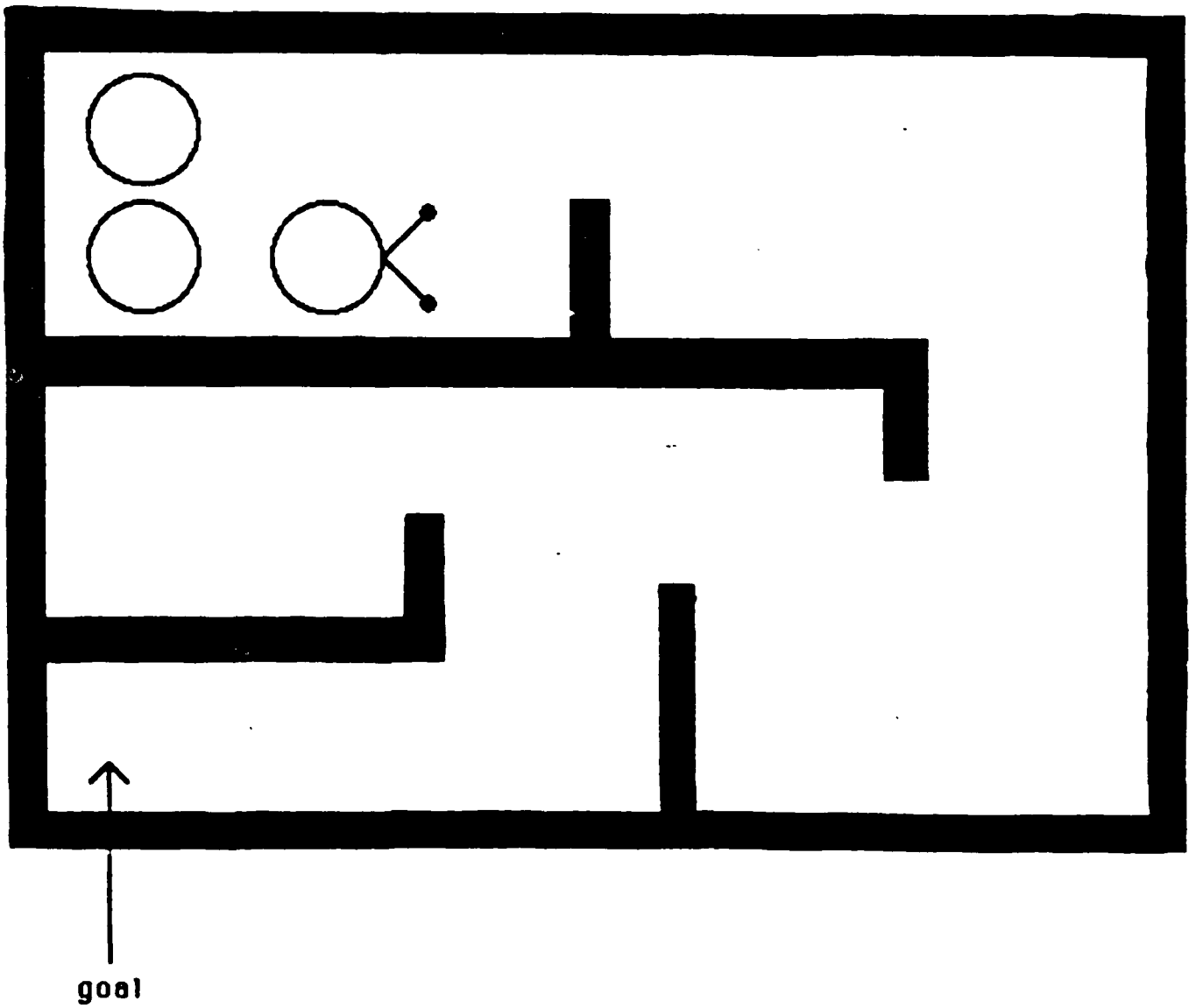
what might be termed "objective" measures of performance, including time or trials to reach some criterial level of performance, post-training ability to carry out targeted tasks, and memory for different kinds of information. Information on the knowledge representations and problem solving methods of subjects receiving the two types of explanatory materials were more directly assessed through the analysis of think-aloud protocols generated by subjects as they progressed through training and testing. Although the reliability of protocol data from small numbers of subjects does generate some concern, these concerns will be addressed below in light of the convergence between the protocol data and other performance measures.

2. Method

2.1 The Troubleshooting Domain

Subjects in this study learned to troubleshoot the MAZER system, which simulates the operation of a complex, goal-oriented robot learning to navigate through an arbitrary maze-like environment. The system runs in LISP on a VAX-780 computer, and subjects view and interact with it via a DEC GIGI display terminal. The output graphics are written in REGIS.

The primary MAZER display shows the automaton as a bug-like icon which moves in real time through a maze. Because knowledge of which direction the icon is facing is necessary for troubleshooting various of its internal components, the bug is shown with antennae on its forward face. MAZER always begins its maze-learning process at the top left-most position of the maze. The goal location is not visually indicated on the display, but subjects are told where it is. In addition, when working properly (i.e., in its unfaulted state) the MAZER icon appears to jump up and down when it reaches the goal. The MAZER icon leaves a trail of circles on the display as it finds its way through the maze, and sometimes goes back and retraces its route when it must determine which way to turn next. The visual path information is necessary for successful troubleshooting since MAZER is a self-modifying learning system. An illustration of the MAZER icon in a sample maze is shown in Figure 1.

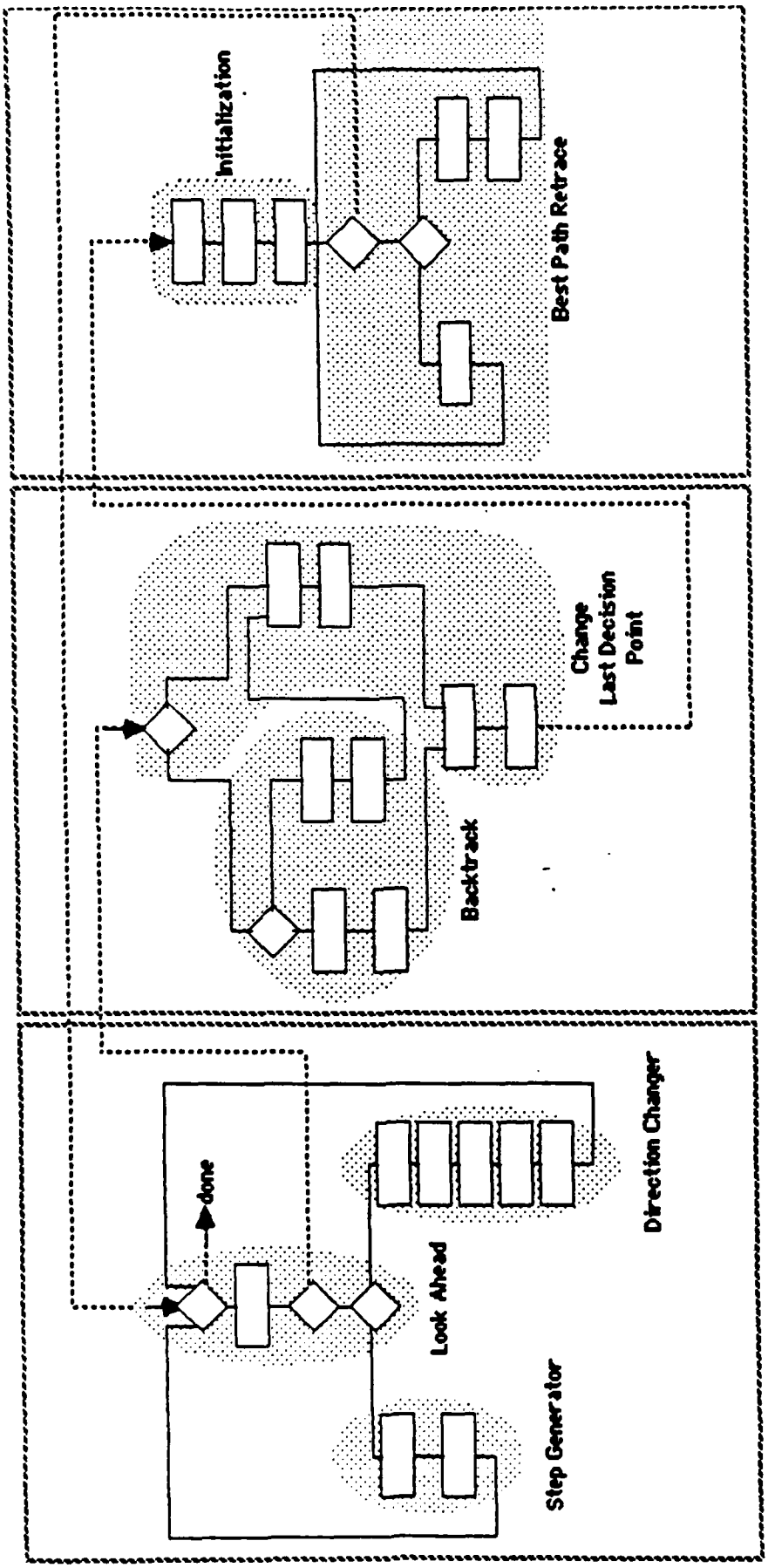

goal

The internal workings of the MAZER robot consist of four major functional components: 1) a navigator, 2) a path improver, 3) a path retracer, and 4) a sensory/memory system. The sensory/memory system is used by each of the other three systems and thus does not appear as an independent component on any functional diagram of MAZER. The navigator contains three functional subsystems, while the path improver and the retracer each have two. Each of these subsystems consists of a logical arrangement of multiple, low-level functional primitive operations. Depending on the functional component, the subsystems may contain straight-line information flow, feedback loops, or both. Figure 2 shows the major functional systems and subsystems in MAZER.

The user has access to a variety of information about the internal workings of the MAZER system via the display terminal keyboard. These include the ability to start and stop the robot's operation, the ability to examine both the contents of memory units and a record of the system's past performance, and the ability to test the robot's sensory apparatus. The system makes a record of the subject's use of these tools, and of more general indices of performance such as solution times and errors. Faults can be introduced into the MAZER system in any of the lowest level functional units for purposes of studying the user's troubleshooting ability. To create each fault, a single component was made to function in a way unlike how it was described to function. An example would be if a component that was supposed to change the icon's position by incrementing a variable by one, instead incremented it by two. This would cause the icon to jump over spaces on the screen and collide with walls. Thus, a a single fault would often bring about an array of symptoms and side effects.

2.2 Subjects

Four paid Brown University undergraduates served as subjects. Two pairs of subjects were matched for gender and performance on a pre-screening test of puzzles used to assess logical ability and motivation. One subject from each of these pairs was assigned to each condition.


Initialization
Best Path Retrace
Retracer
Change
Last Decision
Point
Backtrack
Path Improver
done
Look Ahead
Direction Changer
Step Generator
Navigator

2.3 Procedure

Since one of the goals of this study is to compare the effectiveness of breadth-first (BF) vs. depth-first (DF) structural explanations of domain information, the system was explained to two subjects in a depth-first fashion. For these subjects, each of the lowest-level components of a subsystem was explained in a sequential order initially. Then, the overall function of the subsystem these components composed was explained and named. After the other two subsystems were explained in a similar component-first manner, an overview of the entire system was presented as a summary. Thus the entire system was "built" up starting with the lowest level parts.

In the breadth-first condition, exactly the same information was presented in a different order to the other two subjects. The same overview that served as a concluding summary for the depth-first subjects, served as an introduction for the breadth-first subjects. The function and name of each subsystem was also presented just before its first component was discussed

Subjects in both conditions practiced localizing faults specific to a particular subsystem after that subsystem had been described. Altogether, there were twenty-four training faults: ten in the navigator, nine in the path improver and five in the retracer. The subjects diagnosed each fault at their own pace, watching as many displays and using as many system checks as necessary to localize the fault. A flowchart of the relevant part of the system was available to the subjects during troubleshooting. In the training sessions, whenever an incorrect solution was given, the subject was told that the answer was wrong, and then continued debugging. Subjects were asked to think aloud during half of the troubleshooting problems. Training took place during one- to two-hour sessions across approximately six consecutive days.

Following the completion of the training sessions, each subject's overall knowledge and skill was assessed during the test phase. Here subjects localized twelve faults, four from each

subsystem. Half of the faults had been encountered during the training phase and half were novel. Upon completing a test problem, subjects were asked to say whether they had solved that problem previously. Subjects then attempted to diagnose two faults that occurred simultaneously in the system. There were six of these double fault problems. received six problems in which the system contained two faults (double-fault problems). An important aspect of both parts of the test phase was that, unlike the training phase, subjects didn't know beforehand in which subsystem the faults would be located. Testing required approximately six hours spread across five 1- to 1-1/2-hour sessions on five consecutive days. Subjects generated think-aloud protocols on half of the test problems.

After the final training session and the final test session each subject's knowledge organization was assessed using a card sorting task. Subjects were asked to sort fault symptom descriptions into piles such that the faults in each pile "belonged together." Subjects were also permitted to subdivide each of the initial piles if they desired.

3. Results and Discussion

3.1 Performance Measures

Three measures of overall performance were computed. The first was *solution time*, the elapsed time from when the bugged program was initiated to when the subject declared that he or she had decided upon a final solution. For most of the data analyses the times from problems during which the subjects generated protocols are *not* included in this measure, since the amount that a subject chooses to say has a strong effect on solution times, and there are substantial individual differences in quantity of verbalization that are unrelated to problem-solving ability (Ericsson & Simon, 1984).

A second measure, *solution accuracy*, was scored on a scale from one to five. A score of one indicated that the subject had no guess at an answer, two meant the subject guessed the wrong

subsystem, three meant the subject guessed the wrong component in the right subsystem, four meant the right component was selected, but the subject could not explain how the impaired component was bringing about the current state of affairs, and finally, a score of five indicated that the subject had successfully located the malfunctioning component and explained how the component failed and thereby brought about the observed symptoms. The *percent correct* measure indicates the proportion of bugs the reasoner was able to locate successfully (accuracy scores of 4 or 5). The accuracy and percent correct measures are based on all troubleshooting episodes, regardless of whether think-aloud protocols were required.

3.2 Troubleshooting Performance

The effects of knowledge organization on level of expertise, as demonstrated by quantitative measures of troubleshooting performance were explored first. In the training phase each subject worked on each problem until a correct solution (accuracy score of four or five) was achieved. Thus the percent correct scores for both BF and DF subjects were 100% and the training accuracy scores for the two groups were virtually identical (4.73 for BF and 4.85 for DF, $X^2(1) = 1.69$, $p > .10$). BF subjects were slightly faster than DF subjects (318 secs. vs. 330 secs.).

Since the data were suggestive of a speed-accuracy tradeoff, solution times for the training faults were separated according to accuracy score as shown in Table 1. Training trials on which solution times were more than 2.5 standard deviation units away from the cell mean were not included in the analysis. A two-way analysis of variance for unequal *n*s showed that although BF subjects solved problems more quickly than DF subjects in both the four- and five-accuracy categories, the main effect of group was not significant (F (1,73) < 1). The data do, however, reveal a significant interaction between instructional group and accuracy category (F (1,73) = 7.87, $p < .01$). The times are comparable for complete solutions (accuracy = 5), but the groups diverge on problems on which they received an accuracy level of four. In addition, those problems that were less accurately solved (accuracy score = 4) also took significantly longer to solve

TABLE 1

Mean Solution Times (secs.) During Training Trials

	Breadth-First	Depth-First
Completely correct trials (accuracy score = 5)	277 (n = 27)	297 (n = 34)
Correct component located (accuracy score = 4)	313 (n = 10)	522 (n = 6)

(F (1,73) = 6.62, $p < .025$), indicating that they were a generally more difficult class of faults. Taken together these results suggest that the main benefit of the BF instructions during training was to permit the BF subjects to diagnose the more difficult problems more quickly.

In the test phase, BF subjects successfully solved 71% of the single-fault problems while DF subjects were successful on only 42% ($z = 2.46$, $p < .01$). Given that the BF subjects got more problems correct, it is not surprising that they also had higher average accuracy scores (4.10) than the DF subjects (3.50), a difference that was statistically significant ($X^2(1) = 3.81$, $p = .05$). Since the BF subjects also had shorter mean solution times (381 secs.) than did DF subjects (439 secs.), a speed/accuracy tradeoff does not account for the difference in accuracy scores.

The mean solution times on single-fault problems for the two instructional groups as a function of accuracy level and protocol condition are shown in Table 2. A three-way analysis of variance of the solution times for unequal *n*s on this data indicated that subjects reached solution significantly faster on trials on which they did not give protocols (F (1,37) = 4.615, $p < .05$), but that instructional group and accuracy level did not significantly influence solution time (F (1,37) < 1 for both factors). None of the two- or three way interactions reached significance.*

In the last test phase, in which subjects attempted to identify two faults occuring simultaneously, the percent correct (63% for BF and 67% for DF) and accuracy scores (3.98 for BF and 3.96 for DF) were similar across instructional groups. However, BF subjects solved these problems more quickly than their DF counterparts (854 sec. vs. 1083 sec.). The mean solution times, classified by combined accuracy score on the two faults for the two groups of subjects on the

---

* Two cells in Table 2 stand out as having abnormally large solution times, the no-protocol, inaccurate trials for the the BF subjects, and the protocol, accurate trials for the DF subjects. In the first case, there were only two observations in the cell, and one was abnormally large (over 1300 seconds). Although this artifically inflated the cell mean, the observation could not appropriately be discarded from the analysis without eliminating the within-cell variability entirely. The seemingly abberant DF cell resulted from a majority of observations that were 600 seconds or longer. The cell mean thus accurately summarizes subjects' performance on these trials.

double-fault problems are shown in Table 3. A two-way analysis of variance, with unequal *n*s for these data revealed that the BF subjects solved the problems significantly faster than did DF subjects (F (1,18) = 4.07, p = .05), and that, unlike in the single fault problems, the problems that were solved with the highest degree of accuracy also took longest to solve (F (2,18) = 17.37, $p < .001$). There was no significant interaction between the instructional group and the accuracy level factors (F (2,18) < 1). This pattern of results shows that it took a great deal of time to carefully sort out the mixed effects of two faults occurring simultaneously in the system. Premature guesses and less careful testing of fault hypotheses led to faster solution times but lower accuracy scores. To find both simultaneous faults with a mean accuracy between 4 and 5 took on average 1295 seconds, more than three times as long as subjects took to get the same accuracy score for a single fault (mean = 395 sec). The BF subjects were simply more efficient at performing this very challenging task.

To summarize, in all three phases of the experiment, we see evidence of superior performance by the BF subjects. In the training phase they solved the difficult problems more quickly. In the test phase, they tended to solve the single-fault problems more often and more accurately, and solved the double-fault problems more quickly than did the DF subjects. Thus, learning the system via a breadth-first approach allowed subjects to become better troubleshooters of this system. One reason for this may be that their knowledge of the system was more likely to be hierarchically structured, given the way in which the material was presented. If this is the case, this hierarchical organization should be observable in other aspects of performance.

TABLE 2

Mean Solution Time (secs.) During Single Fault Trials

	Breadth-First		Depth-First	
	No Protocol Trials	Protocol Trials	No Protocol Trials	Protocol Trials
Correctly solved (accuracy = 4 or 5)	219 (n = 8)	443 (n = 6)	289 (n = 5)	722 (n = 5)
Incorrectly solved (accuracy = 2 or 3)	733 (n = 2)	425 (n = 5)	354 (n = 7)	429 (n = 7)

TABLE 3

Mean Solution Times (secs.) During Double-Fault Test Trials

Combined Accuracy Score	Breadth-First	Depth-First
8.0 - 10.0	1169 (n = 6)	1450 (n = 5)
6.5 - 7.5	630 (n = 4)	914 (n = 6)
$\leq$ 6.0	359 (n = 2)	266 (n = 1)

3.3 Memory Organization Differences

Differences in domain memory organization are evident in the effects of repeated experience with a fault. Comparisons were made between the two kinds of faults that made up the test phase: malfunctions which the subject had diagnosed previously (familiar faults) and those which had not been encountered during the training phase (unfamiliar faults), and these results are summarized in Table 4. Average accuracy was higher on familiar (4.63 for BF, 3.67 for DF) than on unfamiliar faults (3.67 for BF, 3.34 for DF). The effects of familiarity can best be viewed in terms of savings from experience with the first occurrence of the bug during the training phase, with the BF subjects showing greater savings on the familiar faults than DF subjects ($X^2(3) = 9.71$ $p < .025$). An analysis of variance of the solution time data summarized in Table 4 revealed that although there was not an overall solution time advantage for the BF subjects ($F(1,35) < 1$), this was largely due to a significant instructional group by accuracy interaction ($F(1,35) = 4.50$, $p < .05$) in which the depth-first subjects finished very quickly on those problems on which they did not do well (low accuracy solutions). The analysis showed that familiar faults were solved more quickly overall than unfamiliar faults ($F(1,35) = 10.60$, $p < .005$), and that DF subjects had significantly more difficulty with unfamiliar relative to familiar faults then did BF subjects (group x familiarity interaction ($F(1,35) = 13.40$, $p < .001$). No other main effects or interactions were significant ($F < 1$ in all cases).

TABLE 4

Mean Test Solution Times (secs.) for Familiar and Unfamiliar Faults

	Breadth-First		Depth-First	
	Familiar Faults	Unfamiliar Faults	Familiar Faults	Unfamiliar Faults
Correctly solved (accuracy = 4 or 5)	301 (n = 10)	350 (n = 4)	328 (n = 6)	687 (n = 4)
Incorrectly solved (accuracy = 2 or 3)	475 (n = 2)	490 (n = 4)	104 (n = 5)	495 (n = 8)

One plausible account for this pattern of interactions in accuracy and solution time is that certain symptoms of the malfunctioning device can activate a subject's mental model of the faulty device. If the mental models of the BF subjects are more complete, by virtue of the training they underwent, then it is not surprising that they successfully solve a greater number of the repeated, familiar faults, and do so more quickly. This memory-based explanation is supported by the fact that BF subjects had better recognition memory for the familiar faults. If we consider the subclass of familiar single faults in which the training instance occurred relatively close to the test version of the fault (between 10 and 14 faults earlier), both groups recognized 84% of these as having been previously presented. Of the longer-lagged repetitions, where first presentations were 21 to 29 problems away, BF subjects recognized a substantial percentage (67%), while DF subjects recognized none (0%). Accordingly, the correlation between number of faults since first presentation and whether a fault was recognized was small for BF subjects ($r = -.19$, $p > .10$) and large for DF subjects ($r = -.86$, $p < .001$).

Table 5 summarizes performance on the single-fault familiar problems as a function of whether or not the problem was recognized by the subject as being repeated. BF subjects achieved higher accuracy and faster solution times on those problems they failed to recognize, while the reverse was true for the DF subjects. Statistical analyses showed significant interactions between instructional group, accuracy level, and recognition category for both accuracy ($X^2(3) = 8.97$, $p < .05$) and solution times ($F(1,12) = 7.87$ $p < .025$). Accuracy for the BF subjects was overall marginally higher than for DF subjects ($X^2(1) = 3.05$, $.05 < p < .10$). The DF subjects apparently had to rely heavily on memory for specific faults during the test. Although they recognized a substantially smaller percentage than BF subjects, their solutions were fast and accurate when they did recognize a fault as familiar. When no item-specific information was available (unrecognized faults) correct solutions were rare and were much more time-consuming. BF subjects, on the other hand did as well as DF subjects on recognized problems, though it took them extra time to recall the information. More importantly, BF subjects achieved even better

accuracy and faster times on the familiar problems they did not recognize from before. The fact that times for these unrecognized repeated problems were also faster than for test problems which were not presented earlier, indicates that BF subjects had built up appropriate solution procedures on the basis of past experience which permitted fast, accurate solutions even when direct problem solutions (symptom-diagnosis links), which would allow them to recognize the bug as familiar, were not available.

Differences in the structure of subjects' conceptualizations of the system were also demonstrated when subjects sorted cards which contained one-line symptomatic descriptions of each of the faults. The first sort by all subjects was into the obvious three piles for each of the main subsystems. When they further subdivided these piles, DF subjects made many piles, most of which corresponded to a solitary component, while BF subjects were able to divide each system into a small number of subsystems, forming a smaller number of more coherent categories. One BF subject noted that he knew he could sort the cards into one category per component, "But that would be boring." In sorting the 48 faults, DF subjects had a mean number of categories of 22.5 (which means about two faults per "pile"), while BF subjects had only 11.5.

TABLE 5

Performance on Recognized and Unrecognized Single-Fault Problems

	Breadth-First		Depth-First	
	Unrec.	Recog.	Unrec.	Recog.
Percent Correct (all problems)	100%	78%	33%	80%
Accuracy Score (all problems)	5.00	4.50	3.33	4.40
Solution Time (secs.) (correct problems)	233	329	453	266

When subjects were asked to sort the cards another way and to describe other possible bases on which faults could be classified, further conceptual differences between the BF and DF subjects emerged. Alternative BF organizational schemes were: 1) according to how the fault could be isolated, 2) according to entities that were common across the systems, 3) according to whether the bug interfered with transitions between the three systems, and 4) according to causes common to several types of component failures (for example, one category of this sort was array faults, involving problems with the system's two arrays, whether in reading from, writing to or resetting them). DF subjects reported bases for sorting that reflected a more surface-oriented view: 1) what the symptoms looked like, 2) the order in which they appeared, 3) whether the fault was immediately visible in test mode, and 4) how long it took to notice or diagnose each fault. These categories describe all of the labels subjects gave for categories with the following exceptions: one DF subject suggested one classification based on entities common to more than one subsystem, and one subject from each group suggested categorizing according to "fatal" and "non-fatal" faults. These differences in categorization reflect the more flexible, hierarchical, and procedural conception of the system by the BF subjects. The conceptualization displayed by the DF subjects' sorts is more superficial, and isomorphic to the organization of the instructional materials. The DF subjects appear to have failed to reorganize the information into a conceptual and more efficient representation.

Interesting parallels to the two groups' bases for categorization of faults have been demonstrated in the ways experts and novices categorize physics problems. Chi, Feltovich and Glaser (1981) found that physics experts classified problems according to the underlying physics principle and abstract solution procedures that were applicable to the problems. In contrast, novices in their study sorted according to the problems' literal surface features. Thus it seems that the present BF subjects categorized the faults by methods that are similar to the methods Chi et al. found are employed by experts, while the DF bases for sorting were more novice-like. This supports the contention that the breadth-first, hierarchical instructions accelerated the acquisition of system expertise by the BF subjects relative to the DF subjects.

3.4 Differences in Procedural Knowledge

Proceduralized representations are often found in studies of expertise. One characteristic of procedural knowledge is that it is automatic and thus not under conscious control. The implications of this automaticity are little or no drain of attentional resources, faster processing of appropriate information and inaccessibility of the procedures (Anderson, 1982).

Evidence of automatized procedures can be found in subjects' verbal protocols. The protocols were analyzed for statements that seemed to indicate a subject was executing a mental model of the device. The statement groups that were classified as mental models fell into one of four categories: 1) prescription - the sequence of steps the program should have gone through, 2) process of elimination - a step-by-step analysis to determine which components must be working and which could be at fault, 3) prediction - given a certain fault hypothesis, the sequence of actions the system will perform when next restarted, and 4) explanation - simulating the system in order to form a hypothesis about how a certain behavior could have been caused. To be classified as a mental model indicator, the portion of the protocol had to include a sequence of at least three system functions, without any intervening statements. The sequence had to describe events that would happen consecutively or that were causally related. An example is included in Appendix A.

BF subjects made statements showing evidence of mental model use 36 times, while the matched DF subjects evidenced only 12. However, almost all of the BF mental model protocols occurred in the training sessions while DF subjects used them equally throughout the training and test phases. Thus, it appears that this method of simulation was not only more likely to occur for BF subjects during training, but also became more automatic, and "went underground" as far as verbalizing is able to detect their presence, for the BF subjects. After the BF subjects received hierarchically organized information in the training sessions, the results suggest that they derived a hierarchically organized data base, which gradually evolved into a procedural representation. DF subjects, on the other hand, received the same information organized in a less servicable

fashion, and were forced to gradually transform this into an organized data base themselves. Such a difference accounts for why BF subjects showed superior performance throughout the study.

These findings suggest that there are at least three stages in the process of forming a proceduralized, expert-like representation of a system. In the first stage, the novice has a body of poorly organized knowledge that is difficult to access in an orderly fashion. In the second stage, this knowledge has become organized in an orderly, hierarchical fashion, which is easily verbalized. To reach the last stage, the learner must again reorganize the knowledge in a way that is efficient for the tasks to which it is frequently applied. The knowledge is still hierarchical, in the sense that the information is available at various levels of abstraction, but important associations across and within levels are also added. Overall, the learner's knowledge organization has to become more personalized and complex, and it is this more efficient organization of the knowledge which allows the formation of procedures. As a side effect of proceduralizing the knowledge, it becomes more difficult to verbalize.

This view can be used to explain the changes in quality of the system recalls across the two conditions. Between the pre-test and post-test system summaries, each group's understanding of the system has advanced one stage. Examples of pre- and post test recalls for both types of subjects are included in Appendix B. In summarizing each subsystem after learning about it and localizing several faults in that section (pre-test), the BF subjects gave clear, confident, succinct explanations of the functioning of that part of the device. The organization of the instructional material they received provided a coherent framework into which they were able to integrate each new piece of information they learned. Thus, at the end of each training session, the BF subjects' knowledge bases had reached the second stage level of organization.

In contrast, DF subjects' pre-test summaries were acausal, poorly ordered strings of unconnected actions and component names. The recalls give an indication that the DF subjects'

relevant knowledge bases consist of an unorganized jumble of facts about the system, which could be categorized at a stage one level.

The post-test system recalls were produced after subjects had solved many more problems in the process of completing both test phases. The BF subjects' post-test recalls began with a very brief summary of the goals of the entire device, and then proceeded into a more detailed summary that would seem at first glance to be poorly ordered and rambling. Component descriptions were more strongly linked to causes and effects they had throughout the three subsystems than to those components that proceeded and followed them within the same subsystem. Rather than describing each of the three subsystems once, that BF subject cycled through them several times. Rather than a rote recall of the system parts, they reasoned about what *must* happen next, even giving evidence of running a mental model of the device in order to produce the description of the system. The BF subjects' understanding of the system at that point is primarily procedural and difficult to verbalize, and so their recalls seem less organized. The BF subjects gave evidence of a well organized declarative knowledge base (stage 2) in their pre-test recalls and it is evident from their post-test recalls that this has further developed into a more procedural knowledge base (stage 3).

These descriptions are quite different from the post-test descriptions provided by the DF subjects. After the test phases, the DF subjects gave succinct descriptions of the parts that made up the system, subsystem by subsystem, component by component, in the order they were introduced in the instructions. For example, one DF post-test system summary was nearly a verbatim reproduction of the words from the diagram of the system. The parts were recited with little attention to how they function and interact. The loop-like structure of the subsystems was barely mentioned. The DF subjects' post-test recalls were well-ordered, perhaps in more of a linear than a hierarchical fashion, and so it seems that they were approaching stage 2. Their post-test recalls are more complete than the pre-test versions and were easily produced. However, a personalized understanding, and an appreciation of the procedural nature of the system are not in evidence.

Some of the qualities of the recalls described above can be quantified. Table 6 shows the number of certain key types of statements contained in the recalls. On such measure was the number of deviations from the order in which the lowest-level components were presented in the instructions and omissions of such components. BF subjects had only five such deviations in their pre-test recalls, but 21 deviations in their post-test recalls. It is unlikely that they abandoned an accurate, efficient representation of the system for a worse model. Instead, their post-test recalls seem to have more order deviations because their knowledge bases have been reorganized to be efficient for troubleshooting the system, not for producing rote recalls. In the DF subjects' recalls, the opposite effect occurs. The number of deviations decreases from nine to three, because over the course of the experiment they have managed to achieve a representation of the system which is an approximation of the organization in which it was presented to them. The interaction between the time when recall was performed and type of training was significant ($z = 2.21$, $p < .025$).

TABLE 6

Frequency of Key Statement Types in System Recall Protocols

	Breadth-First		Depth-First	
Statement Type	Pre-test	Post-test	Pre-test	Post-test
Ordering & omission errors	5	21	9	3
Reasoning about system operation	3	20	0	3
Links between components	13	29	5	17
Individual component explanation	17	11	7	3
"If-then" statements	24	22	19	19
Descriptions of actions	47	46	50	45

Table 6 also shows the number of statements in which subjects reasoned about how the system worked and "linking" statements, in which subjects described the connections between the subsystems and their branches. The number of these statements increases across pre- and post-test recalls for both groups, and it is reasonable to suppose that their increase reflects an increase in understanding of the system. The number of such statements in the DF post-test recalls is about the same as the number of such statements in the BF *pre-test* recalls. This is another indication that the sophistication of the DF representation of the system at the end of the experiment was approaching that of the BF subjects at the time of the pre-test recalls. It is also interesting to note that the number of explanation statements, in which subjects clarified how a component functioned, and "If-then" statements in which subjects discussed the conditional branching the system performed, stayed about the same over the course of the experiment, but overall, the BF subjects had many more such statements. This again indicates the greater depth and procedural nature of the BF subjects' representation. Also note that the number of statements which indicate actions that the system must perform remains the same over the course of the experiment and is equivalent for both groups of subjects.

The system summaries therefore indicate that over the course of the experiment, DF subjects wer attempting to get their poorly organized declarative knowledge into a more carefully structured form, while the BF subjects were getting their accessible and hierarchically organized knowledge transformed into automatized procedures which were effective for reasoning about the device but not for rote recall.

4. General Discussion

The present study has demonstrated that the degree of hierarchical structure present in explanatory material strongly influences the knowledge representation of the learner which, in turn, exerts a powerful influence over the degree to which that information can be proceduralized with practice into effective problem solving methods. Since the conclusions are based on the

performance data and protocols of a relatively small number of subjects, it is important to consider the extent to which the results may be considered reliable.

Although it is always possible to question the validity of information derived from verbal protocols, Ericsson and Simon (1984) have presented convincing arguments that verbal reports, especially when generated in the course of problem-solving, accurately reflect the principal information being used and the methods being employed. This being the case, to what extent should the information obtained from the protocols of only a very few subjects be trusted? The most common use for individual subject protocol data has been, as in the present study, to provide insight on the knowledge representation and procedures used by subjects in various problem solving situations. Single protocols from individual or very small numbers of subjects have been routinely used in the study of such diverse problem solving domains as computer programming (e.g., Anderson, Farrell, & Sauers, 1984), medical diagnosis (e.g., Kuipers & Kassirer, 1984), and cryptarithmetic (e.g., Bartlett, 1958; Newell & Simon, 1972). The present study goes beyond most other protocol studies, even those in which larger numbers of subjects were studied, by providing two lines of converging evidence on the conclusions. First, rather than obtaining only a single protocol, protocols from a large number of problem solving trials were obtained from each subject in the present study. This suggests that the phenomena we observed were not specific to a particular problem or situation, but were robust over time. Second, the protocol data from the present study is corroborated by objective measures of problem solving performance. Taken together, these data provide a good case for the conclusions drawn above.

The results of the present study add to a growing body of literature that suggests that hierarchical organization facilitates learning in a variety of settings. Direct demonstrations of the positive effects of hierarchical organization have been shown in text processing (Britton, Glynn, Meyer, & Penland, 1982; Kintsch & Keenan, 1974), list learning (Bower, Clark, Lesgold, & Winzenz, 1969), complex problem solving (Eylon & Reif, 1984; Larkin, 1980), and classroom

instruction (Mayer & Greeno, 1972; Reif & Heller, 1982). However, the present study does not merely add troubleshooting to the list of domains in which hierarchical organization facilitates learning. These results demonstrate that it is not only the existence of hierarchical structure in a domain that is important, but the way in which subjects come to learn this structure that is important. Both groups of subjects in the present study received the same information, and both groups eventually derived the hierarchical nature of the system, but those who learned the overall breadth-first organization first were able to use it as scaffolding to organize information during their acquisition of later declarative knowledge and problem solving procedures. It thus appears the the way in which information is presented has a far-reaching effect on subsequent performance.

There are two additional ways in which this study builds upon the results of earlier investigators. First, because the domain was novel and complex while remaining relatively circumscribed, it was possible to trace the influence of domain structure on the development of competence in complete novices as they acquired nearly complete domain knowledge. Prior information about the domain was therefore not a confounding factor, and the skills and knowledge acquired were in some sense "complete" in that by the end of the study there were no missing links to as yet unlearned aspects of the domain. Second, the use of protocol information in the present study made it possible to observe specific ways in which the domain structure had its effects. For example, large differences were found in the nature of the procedures used by the two groups of subjects in carrying about the troubleshooting tasks.

In conclusion, this study has demonstrated that the degree of hierarchical structure in instructional information will be reflected in the structure of the learner's knowledge representation. Further, we have found that a hierarchically organized knowledge base plus practice applying that knowledge will give rise to procedural representations. However, the same amount of practice applying knowledge that is not as well-structured will not allow the formation of procedural representations. Thus, this investigation has provided insight into both how instructional materials should be organized and how procedural representations are formed.

5. References

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

Akin, O. (1980). *Models of architectural knowledge.* London: Pion.

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review, 89*, 369-406.

Anderson, J. R., Farrell, R., & Sauers, R. (1984). Learning to program in LISP. *Cognitive Science, 8*, 87-130.

Barlett, F. C. (1958). *Thinking.* New York: Basic Books.

Bower, G. H., Clark, M. C., Lesgold, A. M. & Winzenz, D. (1969). Hierarchical retrieval schemes in recall of categorized word lists. *Journal of Verbal Learning and Verbal Behavior, 8*, 323-343.

Britton, B. K., Glynn, M., Meyer, B. J. F., & Penland, M. J. (1982). Effects of text structure on use of cognitive capacity during reading. *Journal of Educational Psychology, 74*, 51-61.

Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology, 4*, 55-81.

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

Chi, M. T. H., & Glaser, R. (1985). Problem solving ability. In Robert J. Sternberg (Ed.), *Human abilities: An information-processing approach.* New York: W. H. Freeman & Co.

Ericsson, K. A., & Simon, H. A. (1980). Verbal reports as data. *Psychological Review, 87*, 215-251.

Ericsson, K. A., & Simon, H. A. (1984). *Protocol analysis: Verbal Reports as Data.* Cambridge, MA: MIT Press.

Eylon, B., & Reif, F. (1984). Effects of knowledge organization on task performance. *Cognition and Instruction, 1,* 5-44.

Gentner, D., & Stevens, A. L. (1983). *Mental Models* Hillsdale, NJ: Lawrence Erlbaum Associates.

Gitomer, D. H. (1984). A cognitive analysis of a complex troubleshooting task. Unpublished doctoral dissertation, University of Pittsburgh, Pittsburgh, PA.

Glaser, R. (1985). Thoughts on expertise. Technical Report No. 8, Learning Research and Development Center, University of Pittsburgh, Pittsburgh, PA.

Johnson, P. E., Duran, A. S., Hassebrock, F., Moller, J., Prietula, M., Feltovich, P. J., & Swanson, D. B. (1981). Expertise and error in diagnostic reasoning. *Cognitive Science, 5,* 235-283.

Jeffries, R., Turner, A. A., Polson, P. G., & Atwood, M. E. (1981). The processes involved in designing software. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition,* Hillsdale, N. J.: Erlbaum.

Kintsch, W., & Keenan, J. M. (1974). Recall of propositions as a function of their position in the hierarchical structure. In W. Kintsch (Ed.), *The representation of meaning in memory.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Kuipers, B., & Kassirer, J. P. (1984). Causal reasoning in medicine: Analysis of a protocol. *Cognitive Science, 8,* 363-386.

Larkin, J. H. (1980). Skilled problem solving in physics: A hierarchical planning model. *Journal of Structural Learning, 6,* 271-297.

Mayer, R. E., & Greeno, J. G. (1972). Structural differences between learning outcomes produced by different instructional methods. *Journal of Educational Psychology, 63,* 165-173.

Newell, A., & Simon, H. A. (1972). *Human Problem Solving.* Englewood Cliffs, NJ: Prentice-Hall.

Rasmussen, J., & Jensen, A. (1974). Mental procedures in real-life tasks: A case study of electronic troubleshooting. *Ergonomics, 17,* 293-307.

Reif, R., & Heller, J. I (1982). Knowledge structure and problem solving in physics. *Educational Psychologist, 17,* 102-127.

Smith, E. E., & Goodman, L. (1984). Understanding instructions: The role of an explanatory schema. *Cognition and Instruction, 1,* 359-396.

Weld, D. S. (1983). Explaining complex engineered devices. Cambridge, MA: Bolt Beranek & Newman, Inc. Technical Report No. 5489.

Williams, M. D., Hollan, J. D., & Stevens, A. L. (1983). Human reasoning about a simple physical system. In D. Gentner and A. L. Stevens (Eds.), *Mental Models.* Hillsdale, NJ: Lawrence Erlbaum Associates.

6. Appendix A

Example Mental Model Sequence from a BF Subject Protocol

What it should do is, it should - should it just change direction? Well, it was there once, now it's there twice and it *should* stop. Well, then, actually, is it working correctly?

(beginning of mental model sequence)

Alright, well, let's see. It goes, it jumps on and it step, step, makes a turn. Realizes that there's a wall - it would update the array, change the coordinates, go back, choose left or right randomly. It chooses ... right. Updates its direction - it's direction should be north, so ... Oh, here it's not north - here it's south. Add turn to workpath. Hmmm... It realizes that it's been here before and it adds turn to the workpath. But then it stops and goes off? Let's see.

7. Appendix B

Sample System Summary Protocols

Pre-test Summary of the first half of the Navigator by a BF subject

First it tries, it checks...well, the top part is the goal recognizer and it,
That would just compare where it's at with the goal.
And if it's not, it continues.
It achieves its x and y coordinates,
Then it continues.
And it checks to see it it's the been-here-before realizer,
And then it goes on, whether it has or it hasn't
And there's this wall-finder, a wall--
Whatever, it's a box.
It sees -- it's kind of looking to see if the next thing in front of it, by incrementing, well whatever direction it is, whatever coordinate it would have to increment to go straight ahead of where it's pointing to see if there's a wall there.
And if the answer is "no", it would,
I mean, if there's no wall, it goes -- it updates...
Gosh, there's two boxes there.
It updates the been-here-before array.
There's one other thing it does. Huh! The goal, change x-y coordinates,
Changes x-y coordinates, wall and,
The array, then the array down here.
And there's one other box in there. What else would it have to do?
Nothing to do with direction.
I'm missing one thing, I know I'm missing one thing, but I don't know exactly what it is.
Alright, I'll try again.
I can't remember what it was.

Pre-test Summary of the first half of the Navigator by a DF subject.

Well the bug starts out -- the creature --what do you call it?
The hopper starts at the top of the maze and jumps into the first coordinate, 1,1.
And tests in advance the direction it's heading in to see if,
If there's a wall there -- or test if there's an empty space there, actually.
And if it's a direction in which he can head, he goes there.
He stays in that direction, which at this point is south.
And the steps, the x-y coordinates -- increment.
And, oh, before that,
The very first thing he checks is whether it's the goal.
And if the coordinates aren't (1,7), I think then, he goes on to the next steps.
So it increments the x-y coordinates, and the graphics move down.
And it also records into the array whether it's been there before or not before moving.
So when it reaches a new step it checks those arrays to see if it's been in that spot before.
If it has been in that spot, it turns red and starts from the beginning again.
If it hasn't, it just keeps going.
And it keeps moving and changing direction, if necessary, until it reaches its goal.

Post-test Summary of the Navigator by a BF subject.

And then it goes to the navigator,
And the navigator does, huh ...
The navigator seems like a long time ago.
Alright, well let's see, the first thing the navigator does is,
It tests the goal recognizer -- if it's a goal or not.
And if it is, then it's done.
And if it's not, it continues.
And it, let's see, oh... well, let's see, so
There's like three blocks or two blocks between the, or like one or two blocks, between recognizing for the goal and wall realizer.
I guess -- I can't really exactly remember the two blocks.
So I'm gonna skip over like one or two things.
And then, there's this wall realizer,
And if there's a wall there, if it realizes that there's a wall there, it will change it's direction.
Turn left or right randomly.
And it adds that to the workpath.
If it's not a wall, it, it um, if it's not a wall,
It changes it's x-y coordinates according to its direction.
And it updates the been-here-before array.
Gosh, and they both loop back to something.
I think they both loop back to update the been-here-before array.
Oh, ok, before the wall realizer, on the navigator still,
The block is been-here-before realizer.
And if it hasn't been here before, then it will go down to the wall realizer.
And if it has been there before, it will go to the path improver.
Now, wait a minute, does that make sense?
Ok, that doesn't make sense, but that's what happens.
It's not gonna stop there.
It's gonna go to the path improver.
It's gonna go to the path improver, at the end of the workpath.
I'm gonna go to the path improver and come back to this.

Ok, so now the navigator, again.
So there's this goal recognizer,
And this been-here-before realizer,
And a wall realizer.
And it can either say "yes, there is a wall."
Which, in case, which it will go one way and make a turn,
And go another way and take a step.
And it takes a step.
It has to update its been-here-before array,
It has to, it has to change its x and y coordinates in synch with with what its direction is.
And takes a step -- add the step to the workpath, that's what it has to do.
And then it swings back to the been-here-before realizer.
And then it's, if it doesn't have to --
If there is a wall, it will take a turn,
By, um, by changing its direction,
Adding turn to workpath,
And updating been-here-before array,

And then swing back to been-here-before realizer.
Ok, so, yeah, that *is* the navigator.
Is there one box I'm forgetting?
Goal recognizer, been-here-before realizer, wall realizer.
Umm ... No, cause it has three of workpath, the steps and the turns, turns, steps.
No, I think that's it because it only goes ... in the navigator.
And it goes to the path improver from the been-here-before realizer.
Ok, so that's the navigator, that's all set.

Post-test Summary of the Navigator by a DF Subject.

Ok, the system, the hopper first starts by checking to see if it's at its goal.
That's in the goal recognizer.
And then it changes the x and y coordinates -- increases them.
Checks the been-here-before array. Checks the realizer.
And then goes to the wall realizer.
Oh, in the been-here-before array,
If it has actually been,
If it's been there before, it goes to the path improver, but I'll get back to that.
Or ... and if it hasn't, it checks to see if it's a wall.
And if it *isn't* a wall,
It ... let's see, it's already increased the coordinates.
So it adds the step to the workpath and
I guess, records it in the been-here-before array
-- updates the been-here-before array first.
And then adds a step to the workpath, and goes back to the beginning again.
If it *is* a wall, it moves the x and y coordinates back...
Oh, no, first it ... it records it as a wall, updates the been-here-before array: hasn't tried it and is a wall.
Then it moves back the x and y coordinates.
Picks ... left or right randomly, trying to decide a turn and, (10).of 4;Ok, then, oh it makes a change in direction, updates the direction according to the turn.
And ... let's see, I'm trying to think whether it changes the coordinates again.
I think it just updates the direction and then ...
Oh, and adds the turn to the workpath, probably.
And then goes back to the beginning again and then the updating would be up at the beginning process again.
Ok, so, wait ... I'm just trying to think where the retracer comes in.
I guess ... oh, I'll go to the path improver first and then think about the retracer.
The first step in the path improver ... oh, we're starting from the navigator,
And it goes from the been-here-before array,
If...If the hopper has been in that spot before, it goes to the path improver.
And then it sends it to the navigator.
Ok, in the navigator...
Let me stop for a second so I can collect my thoughts.
Ok, in the navigator, it has the same thing,
It has a been here --
It starts in the navigator, it's moving along.
This is when it's on its own, it's not following whatever the best path might have been.
Or whatever. Whatever it would be following.
So it's moving all by itself.
And first it will change its x-y coordinates.
Um, it checks to see if it's -- No, wait.

Ok, it goes to its, it goes to a new space.
It will update --
If it can go to -- I'm not sure of the order.
Can you pause again? E:Sure.
Ok, this is about all I can say about the navigator.
What it will do is it moves around,
And if it comes to a spot that it's been to before, according to the been-here-before realizer,
It sends it to the path improver, which I'll talk about later.
But the other thing is it then moves on to a box that asks whether or not the next step is going to...
Oh! Here we go. This is what I did,
It changes x and y coordinates,
And asks if it's been here before.
If it has, it sends it to the path improver.
If not, it asks whether or not it's a wall.
Alright, if it's not a wall, it will change, it will...
Change the x and y -- No, it will, it's already changed them, but...
It will, um, it will let it move to the next set.
It will take a step.
It will add the word "step" to the workpath.
And then it will go back and ask if the next,
It will change x and y coordinates again,
And it will go back, it goes back to the beginning of the navigator.
It goes, once it's determined it can take a step.
It goes all the way back to the very beginning of the navigator.
If there *is* a wall there, it will change back x and y coordinates.
And it won't let it go, but it will, um, choose left or right randomly.
It will update the, the array.
And then it goes back to the top of the navigator again.
And runs through the navigator until it gets to the been-here-before realizer.
It says it has been here before.
It will send it to, to the um, path improver.

Post-test Navigator Summary of a DF Subject.

I'll start with the navigator.
Uh, first is the goal-recognizer.
Then is the, uh, change x-y coordinates,
The, then the been-here-before realizer...
If the been-here-before realizer says, yes, I've been here before, it goes to the path improver.
Uh, if not, it goes to the wall realizer.
It it's, if it *is* a wall, it updates the been-here-before array,
Changes back the x-y coordinates,
Chooses left or right randomly,
Updates the direction, an adds a step to the work path.
If it says no, it's not a wall --
What's that? if yes, if it says no it's not a wall,
It updates the been-here-before array,
And...and goes back into the loop, the navigator loop.

8. Author Notes

This research was supported by contract #N00014-83-C-0446 from the Personnel and Training Research Programs at the Office of Naval Research to BBN Laboratories with subcontract to Brown University, and by contract #MDA903-87-C-0545 from the U.S. Army Research Institute to BBN Laboratories with subcontract to Brown University. We wish to thank Gregory L. Murphy and Edward E. Smith for their helpful comments throughout the project. Requests for reprints should be addressed to Kathryn T. Spoehr, Department of Cognitive and Linguistic Sciences, Box 1978, Brown University, Providence, RI 02912.

END
DATED
FILM
8-88
Dtic