Int. J. Man–Machine Studies (1983) **19**, 391–398

Programming problem representation in novice and expert programmers

MARK WEISER AND JOAN SHERTZ†

Computer Science Department, University of Maryland, College Park, Maryland 20742, U.S.A.

(*Received 15 September 1982*)

The representation of computer programming problems in relation to the organization of programming knowledge is examined. An experiment previously done for physics knowledge is replicated to examine differences in the categories used for problem representation by novice and expert programmers. Results from sorting tasks show experts and novices begin their problem representations with specific different problem categories. Experts initially abstract an algorithm to solve a problem, whereas novices base their approach on a problem's literal features. A preliminary study of programming managers indicates an abstraction different from that used by programmers.

Introduction

Differences between novices and experts have been studied in chess (Chase & Simon, 1973; DeGroot, 1965), in physics (Larkin, McDermott, Simon & Simon, 1980), and in computer programming (McKeithen, 1979; Shneiderman, 1976). Most of these studies have focused on differences in solutions—the chess move, the physics law, the computer program. While demonstrating that novices and experts differ, these studies do not help account for the difference in terms of cognitive structure, prior knowledge, or problem-solving strategies. Shneiderman's "syntactic/semantic model" (Shneiderman & Mayer, 1979) proposes a theory of novice/expert differences in computer programming, but his supporting experiments are also consistent with alternative theories (for instance, that people who choose to become programmers have better memories and problem solving ability than people who do not so choose).

Chi, Feltovich & Glaser (1981) argue that solvers represent problems by category, and that these categories may direct problem solving. They investigate this hypothesis in a series of four studies on the representation of physics problems. Their first study had novice and expert subjects sort textbook physics problems based on similarities in how subjects would solve them. The data suggested that novices sorted on surface features of problems, such as ropes and inclined planes, while experts sorted on deep features, such as conservation of energy and Newton's second law. The second study confirmed the surface/deep difference by repeating the sorting task on a set of physics problems constructed to be crossed for surface and deep features. Third and fourth studies used thinking-aloud protocols to gain further insight into novice and expert cognitive maps.

We report here on a version of the second study of Chi *et al.*, as modified for novice and expert computer programmers. We were interested in two hypotheses: (1) that

† Currently at Intermetrics Inc., Bethesda, Maryland, U.S.A.

0020–7373/83/100381+08$03.00/0

expert computer programmers show a consistent categorization of programming problems, and (2) that this categorization is different from that used by novice programmers. Both were confirmed.

Materials

The first study of Chi *et al.* (1981) was exploratory and used standard physics textbook problems. Computer science has not yet developed a standard set of problems. Each textbook presents its own idiosyncratic style, and any choice of problems from textbooks would have invisibly biased the features used by our novice and expert subjects.

Instead, programming problems were constructed to cross at three levels of treatment each of application area, algorithm and data structures. Algorithm and data structure are widely considered to be at the foundation of programming knowledge (e.g. Wirth, 1976). Application area was chosen as a plausible source of surface features for novices. The application areas used were business, operating systems and string processing, the data structures were records, arrays and linked lists, and the three algorithm types were search, sort and numerical processing.

There were 27 problems altogether. Problems were three to five sentences long, and were on separate sheets of paper for easier sorting.

Our problems did not contain all the information necessary to write the program solution. They were thus unlike physics or computer science textbook problems, but more like problems actually encountered by practising programmers. The problems were typical of those presented at an early phase of program design, and had sufficient information to complete a detailed design. Two examples are shown in Fig. 1.

BUSINESS RECORD SORT

Company XYZ wants to review employee salaries, starting with those employees who have been with XYZ the longest. Employee data is stored on a disk file alphabetically by employee name. The problem is to create a listing of employees in order of longest service for XYZ to shortest.

OPERATING-SYSTEMS ARRAY SEARCH

An operating system keeps a file of current process-ids (integer) and the corresponding user-ids (integer) for each process. The file is ordered from most recently created process to oldest process. A user may have a maximum of n processes running at the same time, where n is set by the operator. When a user logs in, the system must check to see that the user does not already have the maximum number of processes running. The problem is to perform this check.

FIG. 1. Example problems.

Checking problem construction

Because our problems were constructed, we wished to verify that our desired categories were in fact evident in the problems. In fact, we had problems doing this, and the data structure dimension was never satisfactory.

Our check was to ask four expert programmers to match problems to categories. Each expert was told the problem categories by which the problems had been constructed, and attempted to match problems to categories. A large sheet of paper with a box printed for each category was available to the experts. Subjects were not restricted to one problem per box.

Results of the first pre-test show that 29% of the participants' categories differed from the construction categories. Three per cent of this difference was in application areas, 9% was in algorithms and 17% was in data structures.

As a result of the pre-test, the problems were changed to make them correspond more closely to their intended categories, and the same four participants took the pre-test again, with the same instructions.

The results of the second pre-test showed that data structures were still responsible for 17% of the total differences, application areas dropped to 1%, and algorithms dropped to 7%. The data structure dimension still showed considerable error, but the other dimensions were quite good. Since adjustment of the problems was producing no improvement in the data structures accuracy, the actual experiment was begun using the once-adjusted problems. As we will see in the results, there was no consistent organization of problems by data structure in any of the subject groups.

Method

There were three groups of participants: novice, expert and manager. The experts were nine computer science graduate students at the University of Maryland, all in their second year or later. The novices were six undergraduate computer science majors at the University of Maryland. Three of the novices had just finished a second semester programming course, and three had just finished a third semester programming course. The four manager participants were all from the same large programming organization and all were formerly programmers.

Each participant filled out an experience questionnaire, then received an envelope containing the 27 problems and a written instruction to "sort the problems into groups based on similarities in how you would solve them". Subjects were told to use as many categories as they liked, and to take as much time as they liked. When each participant finished putting the problems into piles, the time was recorded and the participant was asked to write down the problem numbers in each pile and the reason why those problems were placed together.

Results

Table 1 shows the basic statistics. Only the values for the four largest categories and the time to complete the problems show statistically significant differences among the groups by a Kruskal–Wallis analysis of variance (Siegel, 1955). For these values, pairwise differences were examined for significance at the 0·05 level using the Mann–Whitney U-test. Novices used significantly less time and had significantly fewer problems in their largest four categories than either experts or managers. There were no significant differences between experts and managers.

To evaluate patterns of problem grouping, a list was made of all the problem pairs which appeared in any subject's groups. Of particular interest is whether or not a pair

TABLE 1
Basic statistics

	Novices	Experts	Managers
N	6	9	4
Mean time	20·3	55·3	33·8
Mean no. of categories	8·2	6·9	5·3
Mean percentage of problems in largest category	26	32	42
Mean percentage of problems in four largest categories	73	85	91

has both members from the same category in an organization (such as sort in algorithms or business in application area), or has each member from a different category. The only pairs considered are those which were used by more than half the subjects. Tables 2, 3 and 4 show the pairings for novices, experts and managers. Within each table are three sub-tables, one for each of the organizations: algorithm, application area and data structure. A large percentage of points along the diagonal indicate that subjects paired problems according to that organization. The total number of points is a measure of the degree of internal agreement and consistency among subjects.

Table 2 shows that novices organized their problems by application area. Furthermore, there was not much agreement among the novices, but that when they did agree it was on the pairing of business problems with business problems.

Table 3 shows that experts agreed more often than novices (46 points versus 15), especially on problems requiring a sort. Experts organized by algorithm—overall only four of the 46 points do not pair two problems from the same algorithmic category.

TABLE 2
Novice clusters

Application area	Business	Op. sys.	Strings
Business	14	0	0
Op. sys.		1	0
Strings			0
Algorithm	Search	Sort	Number
Search	2	2	6
Sort		3	1
Number			1
Data structure	Records	Arrays	Pointers
Records	2	5	3
Arrays		1	3
Pointers			1

TABLE 3
Expert clusters

Application area	Business	Op. sys.	Strings
Business	5	12	11
Op. sys.		3	10
Strings			5
Algorithm			
	Search	Sort	Num crunch
Search	4	0	3
Sort		28	1
Num crunch			10
Data Structure			
	Records	Arrays	Pointers
Records	5	17	7
Arrays		5	8
Pointers			4

TABLE 4
Manager clusters

Application area	Business	Op. sys.	Strings
Business	22	21	3
Op. sys.		5	3
Strings			6
Algorithm			
	Search	Sort	Num crunch
Search	6	5	13
Sort		16	12
Num crunch			8
Data structure			
	Records	Arrays	Pointers
Records	4	20	10
Arrays		11	12
Pointers			3

The data for managers (Table 4) is more ambiguous. No clear favorite emerges from any of the three organizations. However, the managers had a high internal consistency.

Tables 2, 3 and 4 ignore the labels given by subjects to their problem groups. Tables 5 and 6 show the grouping of problems by similar labels. (Because of the small number

TABLE 5
Novice labels

Labels	Subjects using label (%)	Percentage of problems
Operating systems	83	20
String	83	10
Business	50	12
File	50	10
No reason	66	14
Other		33

TABLE 6
Expert labels

Labels	Subjects using label (%)	Percentage of problems
Sort	100	31
Numerical and statistical	100	17
Search	77	16
Pattern match	77	12
Reports	33	10
Data structure	33	5
Other		9

of subjects no such grouping is shown for the managers.) Each problem given to each subject is considered independently in Tables 5 and 6, giving 162 total novice problems and 243 total expert problems. Groups were assigned labels by the experimenters based on similarity of keywords used in describing the groups. Groups which could go under more than one label were placed under the label judged to be most important to the subject. Labels used no more than once are summarized in the label "Other".

Discussion

The interviews with the managers help explain their ambiguous clusters. Managers solve problems by delegating, and they sorted the problems into catergories by the *kinds of programmer* to whom they would give each problem. Since all four managers were from a single corporation, their high internal consistency is suggestive, but not definitive. The characterization of programmer type by managers is a worthy subject of future investigation.

Experts rarely used a label corresponding to data structure. This indicates that choice of data structure is not essential to expert modeling of problems to be solved, and perhaps explains our failure to obtain agreement about data types from our experts when the problems were checked.

As in the study of Chi *et al.* (1981), experts took more time than novices. This is presumably because of greater transformation was necessary from the surface features

of each problem to the experts' internal categories. Unlike the results of Chi *et al.*, novices accounted for significantly fewer problems in their first four categories than either experts or managers. This could be because our novices were more equivalent to the intermediate subjects of Chi *et al.*, who did show a tendency towards creating extra categories by dividing problems both by surface and deep features.†

The labels used by novices reflected problem surface features. In fact, the three category labels used most often by the novices were exactly our three application area—operating systems, strings, and business, in that order. The fourth novice category, file processing, could also be interpreted as a surface feature label, although not of an application area.

Novices were not very consistent in their labeling. More problems were grouped under "Other" in Table 5 than under any other category. Even the use of a label by several subjects did not indicate agreement on its meaning. Although the label "operating systems" was used the most, consistent pairing of problems with this label never occurred.

Interestingly, the large cluster of business problems shown in Table 2 nowhere appears in the labeling categories of Table 5. In fact, although each point in Table 2 indicates agreement by five or more subjects, only three subjects ever even used the label "business". This indicates novices put business problems together, but in groups without a business label. Novices were evidently unaware of their use of application area categories for their groups.

Labels of both novices and experts generally agreed with the categories for which the problems were constructed. Problems were rarely misclassified within a category type, such as placing a string problem in a category labeled operating system.

Conclusions and directions

Expert programmers use algorithm type as a category in representing problems, and novices do not. Expert programmers do not use application area to categorize problems, but novices do. Expert programmers are not unlike expert physicists in problem solving method—each represents a problem in terms of general principles not evident in the surface features of the problem.

Data structures showed no effect on problem categorization, but it was difficult to construct problems varying by data structure. Data structuring and data types are now of great interest in programming and more investigation with different problems is warranted.

A set of standard programming problems could lead to identifying all the categories used by expert programmers. This would provide a framework for organizing programming texts. Individual weaknesses in experienced programmers could be identified and remedied.

Programming managers show high internal agreement in categorizing problems, but do not categorize like novices or experts. There was some evidence that managers represent problems in terms of who will solve them. Understanding how successful managers do represent problems could lead to better program management.

† If it takes longer to become an expert physicist than an expert programmer, then our sophomore novices were closer to being experts than the sophomore novices of Chi *et al.*

We would like to thank John Gannon for his comments on presentation, Betsy Krusei for first pointing us to the literature on novice/expert differences in physics and suggesting that these paradigms would be effective for studying programming as well, and all of the experimental participants for their co-operation.

This work was supported by National Science Foundation grant no. MCS-80-18294, by Air Force Office of Scientific Research grant no. F49620-80-C-001, and by a grant from the General Research Board of the University of Maryland. Portions of this paper were presented at the National Conference of the Special Interest Group on Computer Personnel Research of the Association of Computing Machinery, June 1981.

References

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.

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

LARKIN, J., MCDERMOTT, J., SIMON, D. P. & SIMON, H. A. (1980). Expert and novice performance in solving physics problems. *Science*, **208**, 1335–1342 (20 June).

MCKEITHEN, K. B. (1979). Assessing knowledge structure in novice and expert programmers. *Ph.D. Thesis*, University of Michigan.

SHNEIDERMAN, B. (1976). Exploratory experiments in programmer behavior. *International Journal of Computer and Information Sciences*, **5**(2), 123–143.

SHNEIDERMAN, B & MAYER, R. (1979). Syntactic/semantic interactions in programmer behavior: a model and experimental results. *International Journal of Computer and Information Sciences*, **7**, 219–239.

SIEGEL, S. (1955). *Nonparametric Statistics for the Behavioral Sciences*. New York: McGraw-Hill.

WIRTH, N. (1976). *Algorithms + Data Structures = Programs*. Englewood Cliffs, New Jersey: Prentice-Hall.