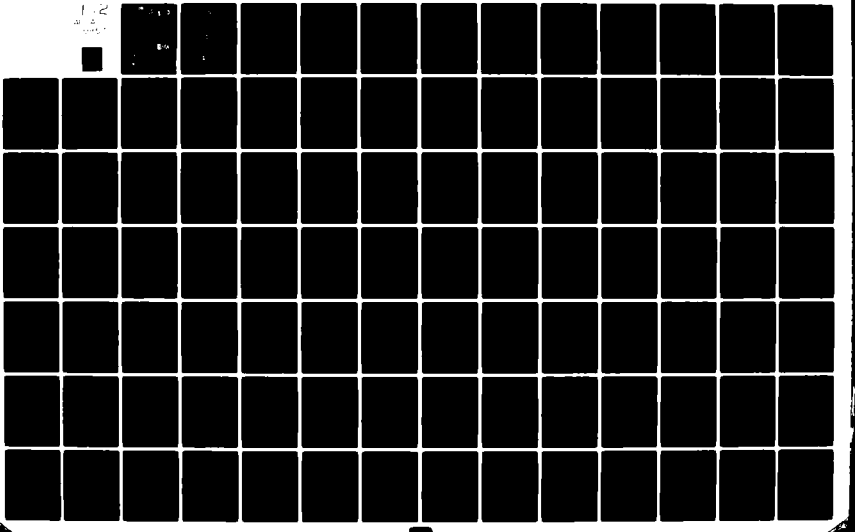
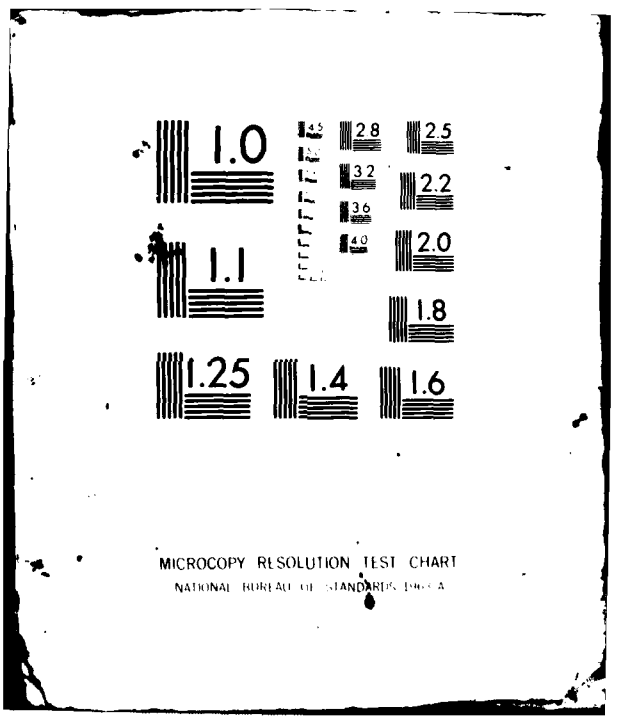
AD-A108 570 PERFORMANCE MEASUREMENT ASSOCIATES INC VIENNA VA F/G 9/2
EFFECT OF AUTOMATIC PROCESSING ON SPECIFICATION OF PROBLEM SOLU--ETC(U)
MAR 81 E M CONNELLY, R F COMEAU, P JOHNSON N00014-79-C-0739
UNCLASSIFIED PMA-81-361 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

LEVEL II

ONR CONTRACT NUMBER

N00014-79-C-0739

WORK UNIT NUMBER

NR 196-161

PAMA

Edward M. Connelly

Robert F. Comeau

Pamela Johnson

TECHNICAL REPORT

EFFECT OF AUTOMATIC PROCESSING ON SPECIFICATION OF PROBLEM SOLUTIONS FOR COMPUTER PROGRAMS

This research is supported by Engineering Psychology Programs, Office of Naval Research

MA-81-361

March 1981

Tech. Rep. #81-361

PMA
410 Pine Street, S.E.
Suite 300
Vienna, Virginia 22180
(703) 938-1600

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 81-361	2. GOVT ACCESSION NO. AD-A105 710
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): EFFECT OF AUTOMATIC PROCESSING ON SPECIFICATION OF PROBLEM SOLUTIONS FOR COMPUTER PROGRAMS	5. TYPE OF REPORT & PERIOD COVERED: Technical, 1 Sep. 1979 to 1 Mar. 1981
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Edward M. Connelly, Robert F. Comeau, Pamela Johnson	8. CONTRACT OR GRANT NUMBER(s): N00014-79-C-0739
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Performance Measurement Associates, Inc. 410 Pine Street, S.E., Suite 300 Vienna, Virginia 22180	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Work Unit #NR196-161 Task Area #RR0420901
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research Department of the Navy Arlington, Virginia 22217	12. REPORT DATE: March 1981
	13. NUMBER OF PAGES: 125
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Engineering Psychology Programs, Code 455 Office of Naval Research 800 N. Quincy Street Arlington, Virginia 22217	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software Development
Programmer Behavior
Automatic Programming
Automatic Processor

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The ability of computer users to specify problem solutions with the help of example solutions was investigated as a function of the user's background and experience, various levels of processing and various levels of problem complexity.

(over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Two participant groups (programmers and bookkeepers/ accountants) working with three levels of problem complexity and three levels of processor complexity were used. The experiment task employed in this study required specification of a logic for selection of a Navy Task force.

The results showed that specification of problem solutions by example-solutions led to low rates of errors-of-commission. Further, the rate of errors-of-omission was significantly affected by the degree of generalization of the example inputs by the automatic processor. The results also suggested that the strategy used in developing the example solutions may be a significant factor in the generation of accurate problem solution specifications.

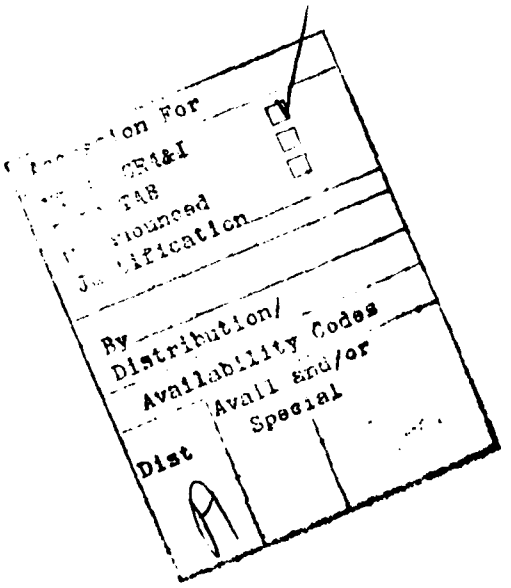


TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES (CONTINUED)

LIST OF TABLES (CONTINUED)

ABSTRACT

The ability of computer users to specify problem solutions with the help of example solutions was investigated as a function of the user's background and experience, various levels of processing, and various levels of problem complexity.

Two participant groups (programmers and bookkeepers/accountants) working with three levels of problem complexity and three levels of processor complexity were used. The experiment task employed in this study required specification of a logic for selection of a Navy task force.

The results showed that specification of problem solutions by example-solutions led to low rates of errors-of-commission. Further, the rate of errors-of-omission was significantly affected by the degree of generalization of the example inputs by the automatic processor. The results also suggested that the strategy used in developing the example solutions may be a significant factor in the generation of accurate problem solution specifications.

INTRODUCTION

The processes by which computer programs are developed and tested, as well as the process of revising established programs, are apparently so incompatible with the human memory and information processing capabilities that computer programs often cannot be synthesized in a precise manner. Instead, they are often developed in a "cut and try" manner, where the development proceeds on what is essentially an iterative refinement basis. For instance, Ramanoorthy (1978) proposed a design methodology, shown in Figure 1, that explicitly recommends repeated revisions of the four elements of the design: 1. requirement and specification; 2. design; 3. implementation; 4. evaluation and validation. A model that explains the success of a "cut and try" design method also points out its major weakness: If program errors are eliminated one-by-one as testing proceeds, the program can only be improved providing new errors are not introduced when old errors are corrected. Even moderately-sized computer programs often require repetitive debugging and patchup to make them acceptable; but errors in large programs may not be discovered until the program has actually been used in the field. Further, the program development process is uncertain since program development schedules and costs frequently exceed estimates thought to be more than adequate.

The research reported here involved an investigation of the ability of computer users to specify problem solutions. This ability is evaluated as a function of the user's background and experience, the complexity of the problem to be solved, the complexity of the available processor (computer), and the available feedback aids.

In order to provide background information, a review of current developments in software engineering is presented. This review indicates that developments in structured programming are, to some extent, based upon limitations in human information processing capabilities; but are mainly efforts to enhance computer program quality, an issue which is conceptually different from the specification of problem solutions. Further, developments in automatic programming technology, by which

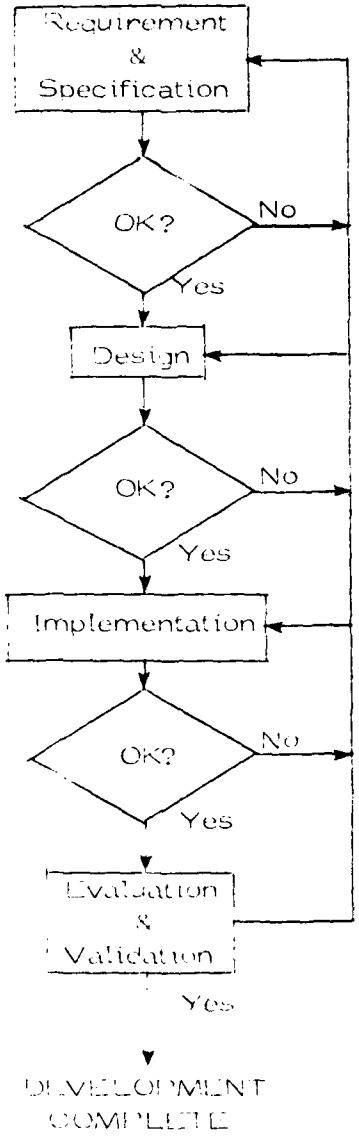


Figure 1. Proposed Design Methodology

(Extracted from Ramamoorthy, 1979)

computer programs can be automatically synthesized, have indicated that ultimately the user may be required only to specify accurately the problem solution, since the computer programs themselves will be automatically synthesized from solution specifications.

Developments in Software Engineering

Several possible reasons exist for the difficulties involved in the design and implementation of computer programs. For instance, it may be that a given task requires attention to so much detailed information and specialized knowledge of machine information (often unique to a particular computer installation) that the human analyst and/or programmer might have extreme difficulty in remembering and recalling the required information. Another possible reason may be that the step-by-step nature of the programming task prevents adequate consideration of the overall computer program. Whatever the true reason, a considerable number of ongoing efforts are designed to address such problems. These efforts fall into the following, somewhat overlapping, categories:

1. Design methodology,
2. Programming languages,
3. Automatic programming,
4. Program test and validation.

Design Methodology

Structured software design methodologies have been and continue to be developed based on the belief that the process of producing a computer program governs the quality of the program. As described by Bergland (1979), these structured techniques impact on all components of the software production process.

Various principles of structured design have been discussed by Yourdon & Constantine (1979), and by Jensen & Tonies (1979). Yourdon & Constantine have indicated that the limitations of human information processing should be used as

a basis for defining a computer program design process. Further, they suggest that Miller's "Magical Number 7 ± 2" item limitation on human short-term memory capability (Miller, 1956) should be a guiding principle in program design. Yourdon and Constantine have identified program complexity in "human terms" as being related to program size, and have suggested that there may be an effective limit to the number of pieces (modules) into which a problem can be decomposed.

In a somewhat similar approach, Jensen and Tonies (1979) recognized that characteristics of the human problem-solving process must be considered while formulating a software design process. These authors have suggested that software design should be broken into chunks which are more amenable to human comprehension. This is generally accomplished by attacking the problem at an abstract level and then proceeding to more detailed levels of design.

Although Yourdon and Constantine have provided what is considered to be an initial human processing capability basis for software development, short-term memory is by no means the only human information processing capability involved in software development. Additional factors of interest include the effect of problem and machine complexity, and the utilization of abstract and concrete levels of communication in each step of the design process.

Programming Languages

The continued development of new programming languages is based upon a rationale similar to that given above, i.e., that the process of generating a program, which is influenced by the programming language, ultimately governs program quality. Many new programming languages are a result of previous work by Dahl, Dijkstra, & Hoare (1972) and by Dijkstra (1976). For example, the PASCAL programming language (Welsh & McKeag, 1980) and its extensions, as well as the newer ADA language (Wegner, 1980) were both developed from this earlier work. These approaches, which involve the structured step-by-step refinement of information, have considerable appeal from a mathematical viewpoint and apparently enhance the development of "correct" machine code; however, as mentioned above, the present investigation is not designed to address the development of "correct" machine code, but rather to investigate

factors that affect the rapid and accurate specification of problem solutions from which computer programs are generated.

Automatic Programming

Another approach, which falls into both the automatic programming category and the program language category listed above, combines results from both programming language and artificial intelligence research. This research has already suggested a dramatic reduction in the effort required to provide machine code and has highlighted the increasing importance of adequate problem analysis and correct program specification. It is generally agreed by researchers in programming languages and in artificial intelligence that the process of generating a computer program can be logically divided into two steps: "program specificiation" and "program synthesis" (Biermann, 1976). Other researchers have referred to these steps by other names: "top part, bottom part" (Prywes, 1977), "program acquisition, program coding" (McCune, 1977), and "formulating the plan, formalizing the plan" (Wile, Balzer, & Goldmann, 1977). It is generally agreed that once a program has been specified, program coding can proceed without difficulty. Various program synthesis (coding) systems have been constructed and demonstrated by the researchers cited above, among others.

The first of the two steps, program specification, has been studied from various viewpoints, including artificial intelligence, as reported in a survey by Biermann (1976), where inputs from the user might include examples, or input-output specifications, or natural language commands; and the position of Wile, et al. (1977) who insist that the user is responsible for formulating the plan. At present, program specification is the major impediment to automatic programming; yet research to date has generally concentrated on the machine side of the man-machine interface, in which the human programmer (user) is considered as merely an inputter of program specifications. But since the user is the key element in the automatic programming process, machines should be adapted to human capabilities rather than humans to machine processes. Major issues here involve the unique capabilities of the human user in specifying

programs, as well as the influence of machine processing (generalizations) and feedback aids on the specification process.

Program Test and Validation

An enormous number of methods and tools for testing, inspecting, and checking computer programs have been devised (Meyers, 1979). Such tools, however, are frequently not geared to testing programs as problem solutions, i.e., does the program solve the problem at hand?

Two requirements govern program test and evaluation efforts. First, the testing must in part be problem independent (e.g., attempts to determine whether a program will reach completion); second, part of the testing must be problem dependent (e.g., to determine whether a given program solves the problem intended to be solved). The first of these requirements is a function of the program code only. The second is dependent in part on the availability of a correct specification of the problem solution. This is the subject of the research reported here.

Use of Computer Resources

Specification, testing, and use of computer programs require interactive communication between the user and the computer system. The level of abstraction necessary to specify a problem solution essentially defines the level of communication at the system interface. But whatever the aids are that prove to be superior in supporting the desired level of communication, they too require the use of computer resources, i.e., of processing time and memory. Thus, a need exists to determine the nature of the tradeoff between total system performance (speed and accuracy of communication) and computer resources in order to apply systems more efficiently to situations in which resources are limited. Determination of the exact nature of that tradeoff, in order to enable a system designer to consider such issues along with all other resource demands when designing the total system, is one objective of the present research effort.

Need for Flexible Communication

There is another way to view the user/computer process considered here. In the design of computer interface systems, a designer typically views the user with regard only to his/her level of knowledge about the system. For instance, Schneiderman (1980), in an extensive review of interface design methods, has identified various design check lists (Hansen (1971), Wasserman (1973), Pew & Rollins (1975), Gaines & Facey (1975), Cheriton (1976), Gebhardt & Stellmacher (1978), Turoff, Whitescarver & Hiltz (1978), and Kennedy (1974)) in which many worthwhile suggestions are offered, yet the cardinal rule agreed upon is "know the user."

Such limited views of user populations permit scaling of input only according to user knowledge of the system. Thus, the beginning users are typically presented with a "menu," listing a set of typical input sequences for potential use. The experienced user, however, is typically provided with a set of general commands, from which he can, with great freedom, design his own specific input commands.

This view, though important, neglects another interface dimension which is critical to efficient and effective communication. In every communication, whether it be between a user and a machine, or person to person, there is a great amount of information left unstated: information that is assumed either to be known or to be irrelevant. For instance, in user/computer communications, much information is stored in memory, system status registers, I/O buffers, etc., that is typically not communicated. This uncommunicated information is often the key to such difficult-to-answer questions as: Why didn't the system respond the way I anticipated? If I hit another RETURN, will I lose the data just entered? In what mode is the computer operating? What are alternative commands? How can I get this system to process my data without adding 1 to each entry?, etc. In general, the issue at hand is: how can a user establish communication or generate commands when unsure of the correct way to proceed?

The existence of such problems suggests that multi-level communication of both abstract and concrete information (i.e., both general statements and examples) may provide a method

for effective 2-way communication between users and computers. In such a situation, if the effect of a general command is not understood by a user, the computer can illustrate by presenting examples of specific cases. Thus a user can be provided with general program instructions and with detailed solution examples. If given general program instructions, the computer can return example solutions. If given detailed solution examples, the computer can generalize from them and return general instructions as well as additional example solutions. Such multi-level communications are used extensively and effectively in person-to-person communications. High levels of abstraction provide efficient compact communications; examples (concrete or low-level abstraction) are used to enhance the understanding of what is communicated in abstract terms.

With respect to the user/computer communication process described above, we note, first, that a user's capability to specify a program quickly and accurately (i.e., to specify the solution to his/her problem) is not only important, but is determined as a joint function of the type of problem, the available machine capability, and the availability of feedback aids. Second, the machine capability necessary to support the required problem solution specifications must be translated into computer resources, and the user's performance capabilities must be established as a function of these resources. Finally, when multiple levels of information abstraction are available, the capabilities of the users to specify actual programs must themselves be specified.

METHOD OF APPROACH

Recognizing that specification of a problem solution is the first step in implementing the solution and in time may be the only step required for specification of a computer program, the objectives of the work reported here were as follows:

1. To determine if different levels of abstract information enhance a user's speed and accuracy in specifying problem solutions;

2. To determine the effects of problem complexity, processor complexity, and user population experience and background on the speed and accuracy of specifying problem solutions;

3. To determine the computer resources required to support high levels of speed and accuracy in specifying problem solutions.

Since planning, designing, writing, and testing of a computer program are all required to implement and determine the adequacy of a problem solution, the process of actually developing a computer program was used as an initial model for identifying a suitable task for the experiment. In particular, literature describing various types of programming errors was examined in detail. Thayer (1976) and Motley (1977) for example, have developed a technique for coding programming errors, as well as statistics for measuring frequency of occurrence of different types of errors. Such efforts have shown that programming errors classified as "logical" constitute a larger percentage of errors than any other type. These authors have distinguished logical errors from such other categories of errors as computational errors, input/output errors, data handling errors, etc. Their conclusion, however, was that by far the greatest incidence of programming errors fell into the logical category.

The term "logical," as was used here, is defined in a somewhat broader context than that of either Motley or Thayer. For purposes of this effort, a "logical problem" is defined as a problem where the solution is a logical function of known independent factors. Using this definition, errors may arise if one neglects the fact that a change in a single variable may change the total result (i.e., each independent variable may be important to the solution). For background, a brief discussion of logical functions as a basis for the approach presented is provided.

Logical Functions

If N logical variables exist, it is possible to form 2^N logical combinations, and 2^{2^N} possible logical functions of those variables Thus, if three logical variables exist, there are eight possible combinations of those variables. A general logical function of three variables is given by:

$$D = K_1 ABC + K_2 AB\bar{C} + K_3 A\bar{B}C + K_4 A\bar{B}\bar{C} +$$

$$K_5 \bar{A}BC + K_6 \bar{A}B\bar{C} + K_7 \bar{A}\bar{B}C + K_8 \bar{A}\bar{B}\bar{C}$$

where the bar over a letter (e.g., $\bar{C}$) represents "not C," K_i is equal to 1 or 0 depending on whether the corresponding combination is or is not included in the logical function, and "+" represents the logical "OR" operation.

For example, if K_1 is equal to 1 and all remaining K's are set equal to 0, then the function specified is:

$$D = ABC$$

In another example, with K_1 and K_2 set equal to 1, and the remaining K's set equal to 0, the function is:

$$D = ABC + AB\bar{C}$$

which is read: "D equals A and B and C or A and B and not C" which simplifies to:

$$D = AB$$

A logical function can be specified in two ways. One is at a general level in which the logical function is simply written out; e.g.,

$$D = AB$$

In the second, the function is specified by giving example solutions, such as shown in Table 1.

Table 1

Example Logical Specification

D	A	B	C
T	T	T	T
T	T	T	F

Here we have two example solutions which state that when A and B and C are true, condition D is true; and also, when A and B and not C are true (i.e., C is false), condition D is also true. A complete specification of a three-variable equation requires eight independent solutions such as shown in Table 2. If, however, an associated computer processor were to generalize by the rule "The output is always false unless specified as being true," the two example solutions provided in Table 1 would be sufficient to completely specify the necessary logical function.

With the first method discussed above, the problem solution is completely specified by the formula. With the second method, the problem solution is implicitly specified by example solutions. Either method can be used to communicate the solution. The difficulty with specifying general solutions is that, as the function becomes increasingly more complicated, it becomes more difficult to understand and rectify. As a result, the user will often covertly generate for his/her own use, a number of example solutions which may then be employed to verify the general solution. This process often occurs in testing a computer program.

Table 2

Specification of Logical Function

D	A	B	C
F	F	F	F
F	F	F	T
F	F	T	F
F	F	T	T
F	T	F	F
F	T	F	T
T	T	T	F
T	T	T	T

The advantage provided by a general solution is that it is a compact way of representing and communicating the solution. The advantage of specifying solutions by means of examples is that they are easily understood and checked. Their disadvantage is that, as the number of variables (N) become large, the number of example solutions required to completely specify the logical function increases by 2^N.

Dealing with logical problems of this type is believed to be a fundamental difficulty not only in specifying problem solutions for computer programs, but also in many other problem areas where there are a large number of variables to be considered and where each can be of equal importance. These problem areas include specification of instructions for virtually any purpose (e.g., instructing individuals in military, industrial, and home situations), including instructions for developing solutions for computer implementation as well as for using computer systems.

Work Completed

The portion of the ongoing research completed and reported here consists of results obtained for two participant categories, each working with three levels of problem complexity, and three levels of processor complexity.

The experimental task required that the participant enter example solutions into a computer. The computer then examined those example solutions, and formed a general solution. This general solution was refined by the computer as the participant entered each new example solution. The participant was able to check the computer's general solution by:

1. requesting and examining example solutions generated by the computer in accordance with its general solution, and

2. by examining the computer's general solution function.

The Task

The experimental task employed in this study required specification of a logic for the selection of a hypothetical Navy task force. The task involved choosing ships from a ship list which identified the ship type, the transiting time (i.e., time required for the ship to get from its present position to the desired site), and stationing time (i.e., the number of days the ship could remain on station with available provisions). The participant was required to specify example ship combinations for each problem worked. This ranged from the simple task of specifying three different ship combinations in the first problem, to additional, more difficult combinations in later problems.

In addition to this specification of ship combinations, the participant also had to specify by example solutions the range of transiting and stationing times required. For instance, if the required transiting time was 10 days or less, the participant had to search the ship list to find one example ship of the proper type that would establish the desired upper limit on transiting time. Also, the participant had to search the list to find another ship of the proper type which had a transiting time that established the desired lower limit. Thus, the participant's task was not only to establish the proper combinations of ship types, but also, for each ship and combination of ships, to establish by example solutions the desired range of transiting and stationing times.

Each participant was also required to deal with various processor complexity levels. He/she did not have previous knowledge of how a particular processor (computer) would interpret the example solutions provided. For instance, at one complexity level, the processor would generalize transiting and stationing times over particular ship types only. At another processor complexity level, the processor would generalize transiting and stationing times over all ship types in a particular combination.

The participant evaluated the general solution (processor ship selection logic [SSL]) produced by the processor to make sure it was a correct solution. If it was incorrect, the participant modified it by providing additional example solutions. The participant could also evaluate the processor's general solution by asking the computer to generate its own example solutions and then by evaluating the newly generated solutions for accuracy and completeness.

Design of the Experiment

The experiment used a repeated measures Latin Square design (Plan #9 cited in Winer, 1971, pp. 727 - 736). The design is shown in Table 3. The factors investigated were:

a. 3 levels of problem complexity,* as measured by Halstead's E Metric, where each level required a different amount of effort to correctly specify the problem solution.

b. 3 levels of processor complexity* where each level required that a different amount of information be provided by the user to correctly specify the solution.

c. 2 participant categories: bookkeepers/accountants and expert computer programmers.

Each factor was fixed, but the group (G) and the participants within each group were random factors. (An additional factor, various levels of feedback aids, will be analyzed in later experiments.) The order of presentation for each group (G) was randomized, resulting in the presentation schedule shown in Table 4 for programmers and in Table 5 for bookkeepers/accountants.

*Measures of problem complexity and processor complexity are discussed in subsequent sections of this report.

Table 3

Schematic for Experiment Plan

Programmers

G_1	A_1B_2	A_2B_3	A_3B_1
G_2	A_1B_1	A_2B_2	A_3B_3
G_3	A_1B_3	A_2B_1	A_3B_2

Bookkeepers/Accountants

G_4	A_1B_2	A_2B_3	A_3B_1
G_5	A_1B_1	A_2B_2	A_3B_3
G_6	A_1B_3	A_2B_1	A_3B_2

A_i is a level of problem complexity.

B_j is a level of processor complexity.

Each combination of A_i and B_j is an experiment cell.

C_k is a group of 10 participants.

Table 4
Computer Programmers
Problem/Processor Order

Participant #	Session # 2	3	4	Group	Order
	Problem/Processor				
14	31/1	15/2	52/3	1	5
15	31/3	15/1	52/2	2	5
16	31/2	15/3	52/1	3	5
17	52/3	31/1	15/2	1	4
18	52/2	31/3	15/1	2	4
19	52/1	31/2	15/3	3	4
20	1[illegible]/2	52/3	31/1	1	[illegible]
21	15/1	52/2	31/3	2	1
22	15/3	52/1	31/2	3	1
23	31/1	52/3	15/2	1	6
24	31/3	52/2	15/1	2	6
25	31/2	52/1	15/3	3	6
26	52/3	15/2	31/1	1	3
27	52/2	15/1	31/3	2	3
28	52/1	15/3	31/2	3	3
29	15/2	31/1	52/3	1	2
30	15/1	31/[illegible]	52/2	2	2
31	15/3	31/2	52/[illegible]	3	2
32	15/2	52/3	31/1	1	1
33	15/1	52/2	31/3	2	1
34	15/3	52/1	31/2	3	1
35	15/2	31/1	52/3	1	2
36	15/1	31/3	52/1	2	2
37	1[illegible]/3	31/2	52/1	3	[illegible]
38	52/3	31/1	15/2	1	4
39	52/2	31/[illegible]	15/1	2	4
40	52/1	31/2	15/3	3	4
41	31/1	52/3	15/2	1	6
42	31/3	52/2	15/1	2	6
43	31/2	52/1	15/3	3	6

Notes: Problem numbers were assigned randomly to conceal the relative complexity of the problems. They were: #93, #15, #52, and #31 in order of increasing complexity.

Session 1 = 93/3 (Problem #93, Processor #3). This was the pre-test. Participants #1 - 13 were used in the Pilot Test, not in the experiment.

Table 5
Bookkeepers/Accountants
Problem/Processor Order

Participant #	Session # 2	3	4	Group	Order
	Problem/Processor				
44	31/1	15/2	52/3	4	5
45	31/3	15/1	52/2	5	5
46	31/2	15/3	52/1	6	5
47	52/3	31/1	15/2	4	4
48	52/2	31/3	15/1	5	4
49	52/1	31/2	15/3	6	4
50	15/2	52/3	31/1	4	1
51	15/1	52/2	31/3	5	1
52	15/3	52/1	31/2	6	1
53	31/1	52/3	15/2	4	6
54	31/3	52/2	15/1	5	6
55	31/2	52/1	15/3	6	6
56	52/3	15/2	31/1	4	3
57	52/2	15/1	31/3	5	3
58	52/1	15/3	31/2	6	3
59	15/2	31/1	52/3	4	
60	15/1	31/3	52/2	5	2
61	15/3	31/2	52/1	6	2
62	15/2	52/3	31/1	4	1
63	15/1	52/2	31/3	5	1
64	15/3	52/1	31/2	6	1
65	15/2	31/1	52/3	4	2
66	15/1	31/3	52/1	5	2
67	15/3	31/2	52/1	6	2
68	52/3	31/1	15/2	4	4
69	52/2	31/3	15/1	5	4
70	52/1	31/2	15/3	6	4
71	31/1	52/3	15/2	4	6
72	31/3	52/2	15/1	5	6
73	31/2	52/1	15/3	6	6

Notes: See Table 4.

Referring to Tables 4 and 5, the numbers in column one are the numbers assigned to the participants. The numbers in columns two, three, and four refer to the problem number and processor level number, respectively; e.g., 31/1 = Problem Number 31, Processor Number 1. The numbers in column five refer to the groups to which the participants were assigned and, in column six, to the order of problem presentation.

Each participant was given:

1. a description, via video tape, of the experiment task along with an illustrative solution to the task.

2. a description, via video tape, of the procedure for entering data into the computer.

3. a pre-test problem - a low-level complexity problem with Processor Level B_3 (least generalization).

4. Test Problem #X, Processor level l.

5. Test Problem #Y, Processor Level m.

6. Test Problem #Z, Processor Level n.

Note that values for X, Y, Z, l, m, n were taken from Tables 4 or 5 in accordance with the participant's group number

When Step 3 above was completed, the performance score on the pre-test was determined. If the participant's input example solution yielded more than one correct ship combination and the minimum/maximum times for transiting and stationing respectively were not equal (these two conditions were taken as evidence that the participant understood the instructions), the participant was permitted to enter the experiment. If the participant did not input example solutions yielding the result cited above, he/she was not permitted to enter the experiment.

Pilot Test

Eight programmers and five bookkeepers/accountants were used in the pilot test to determine suitable levels of problem complexity. The time required for presentation of instructions and for completion of each problem was also determined. Each pilot participant received the test instructions and was asked to solve four problems of different levels of complexity. All test problems in the pilot test used Processor Level B_3.

Results of the pilot test indicated that:

1. The ship list should be removed from the computer. Initially, the ship list was presented on the computer terminal, but it was found that participants continually referred back to it and took notes about suitable ships. This proved to be too time consuming, so, for reference, a typewritten copy of the ship list was given to each participant.

2. The initial problem design required excessive time and had to be revised so that a problem could be completed in one hour.

3. Time required to read the instructions was excessive in some cases. Therefore, it was decided to pace the instructions by presenting them simultaneously via video tape and typewritten copy - the participants could then follow along with their typewritten copy.

The Experiment Task

The experiment task was described to the participants as follows:

"In order to investigate the ability of individuals to develop accurate instructions and to test interpretations of

their instructions, we are conducting this experiment in which you will be presented with several problems of various levels of difficulty and asked to develop Instructions, in the form of Example Solutions to problems that we specify. You will also be asked to enter your example solutions into a computer terminal. The example solutions that you enter will be interpreted by the computer as Instructions to be used in solving the general problem for which your inputs are example solutions.

"Example solutions to problems are like specific instructions which can be interpreted (or generalized) for broad application. Thus, each example solution is part of a specification of a general solution. As several example solutions are developed and input to the computer. The computer will begin to form a general solution. In order to evaluate the computer's general solution for accuracy and completeness, you must test it by commanding that the computer develop its own solutions which you can then evaluate. If the computer's general solution is wrong or incomplete, you will have to develop additional example solutions to show the computer where it was wrong in its general solution. This iterative, interactive process will continue until you have provided enough example solutions, tested the computer output, and are satisfied that the general solution selected by the computer is the proper solution. During this process you will be provided with feedback aids from the computer to assist you in your task.

"Illustrative Problem (A problem and solution to show you how it is done)

The Navy has been ordered to be ready to assemble a task force for an extended mission in the Indian Ocean. In order to identify possible ships for this task force when the assembly order is given, Harry Smith, an analyst for the Navy, has been asked to specify the instructions necessary to scan a file of available ships, to apply the ship requirements necessary for the composition of the task force, and to select one or more "solutions" (i.e., suitable sets of ships) for the mission.

"Since it is not known when the assembly order will be given and the list of available ships changes from time to

time, Harry's job is not merely to select a suitable set of ships for the mission. Instead, Harry's job is to develop multiple example solutions that together define the logic (i.e., specify all combinations) of the ship selection process. The objective is to specify the logical rules for selecting ships; then later, when the ships are actually to be identified for the mission, the logic can be applied to the list of available ships and the appropriate ships selected automatically by the computer.

"Harry's first step is to obtain the criteria for this task force.

1. The ships* needed for the task force are:

   - One (1) nuclear attack aircraft carrier (CVAN)
   - Two (2) submarines (SS)
   - Two (2) destroyers (DD)
   - One (1) oiler (AO)

2. The present position of the ships and their speed must permit arrival at the assembly point within 5 days.

3. Fuel on board must permit stationing without resupply for a 10 day period.

"Harry's next step is to construct a list of ships which includes the necessary data on available ships. He constructs a ship list instead of using the actual ship file because the actual file of available ships may not contain all combinations of ship types, arrival times, and fuel loads necessary to permit selection of example solutions specifying the necessary combinations of ships (i.e., necessary to specify the ship selection logic). He developed the ship list shown in Table 7, and entered it into the computer."

(End of Experiment Task Description)

---

*Note: When a nuclear ship is specified, a non-nuclear ship is not acceptable; likewise, when a non-nuclear ship is specified, a nuclear ship is not acceptable.

Problem Variations

Four problems of different complexities were used in the experiment. Complexity is a function of the number of correct solution combinations and is defined in the next section. The problems were assigned random identification numbers so as not to reveal their relative complexity to the participant. One problem (#93) was used in the pre-test and the other three problems were used in the experiment. The least complex problem (Problem #93) required specification of three different ship combinations each involving four types of ships. The next level of problem complexity was Problem #15, which required six ship combinations again using four different ship types. In order of increasing complexity were Problem #52, which required 9 ship combinations each involving either 4 or 5 ship types, and, finally, Problem #31, which required 14 combinations each using 5 ship types. The four problem statements are provided in Appendix A.

Measurement of Problem Complexity

Halstead's E Metric (Halstead, 1977) is a measure of the mental effort (E) (i.e., the number of discriminations) required to accomplish a task. The metric permits decomposition of a task into a number of elementary discriminations required by the task. Halstead's E Metric is frequently used in software research. It permits comparing (and in some cases combining) results obtained by different research programs. Halstead's E Metric is given by:

$$E = V^2/V^* \quad (1)$$

where: V is the "program volume" and V^* is the "potential program volume." V is given by:

$$V = N \log_2 n \quad (2)$$

where:
$$N = N_1 + N_2 \quad (3)$$
$$n = n_1 + n_2 \quad (4)$$

N_1 = The total number of operators

N_2 = The total number of operands

n_1 = The unique operator count

n_2 = The unique operand count

V^* is given by:

$$V^* = n^* \log_2 n^* \quad (5)$$

where: $n^* = n_1^* + n_2^* = 2 + n_2^*$ (6)

n_1^* = The potential operator count

n_2^* = The potential operand count.

According to Halstead, the minimum possible number of operators n_1^* for any algorithm is 2, one operator for the function name and another for its assignment. Thus $n_1^* = 2$.

Furthermore, n_2^* is evaluated as the minimum number of different input/output parameters. The quantity N in equation (3) is called the "program length;" the quantity n in equation (4) is the "vocabulary size;" and the quantity n^* in equation (6) is the "potential vocabulary size." All of these quantities are abstract and may be difficult to visualize.

In order to compute the E metric for this experiment, the problem solutions were written in explicit logical form and the operators and operands were counted. Thus, for instance, the solution to Problem #93 was written:

(CVA and CVA and SS and SS)
or
(CVAN and CVAN and SS and SS)
or
(CVA and CVAN and SS and SS)

where: CVA is an aircraft carrier non nuclear
CVAN is an aircraft carrier nuclear
SS is a submarine,

According to this solution:

The number of unique operators (n_1) = 3

i.e., "AND," "OR," "()" are the three unique operators

The number of unique operands (n_2) = 3

i.e., CVA," "CVAN," "SS" are the three unique operands

The total number of operators (N_1) = 14

i.e., there are 9 AND's, 2 OR's and 3 ()'s in the solution

The total number of operands (N_2) = 12

i.e., there are 3 CVA's, 3 CVAN's and 6 SS's in the solution.

n_1^* = 2 as indicated previously,

n_2^* = 3, i.e., CVA, CVAN, SS are the three potential input/output parameters

The calculation of Halstead's E metric for each problem is given in Table 6.

The Experiment Processors

Based on the example solutions input into the computer by the participants, the processor (computer) is able to form a ship selection logic (SSL) by which the computer can select ships in a potential task force. The SSL consists of:

a. A set of ship combinations,

b. minimum/maximum (MIN/MAX) values for transit times for each ship type in each set of ship combinations,

c. MIN/MAX values for stationing times for each ship type in each set of ship combinations.

MIN/MAX values for transit and stationing times were a function of the processor generalization capability (i.e., processor complexity). The three complexity levels employed in this study were:

Level B_1: The MIN and MAX transit and stationing times apply to all ships regardless of type or combination.

Level B_2: The MIN and MAX transit and stationing times apply to all ships in each particular combination.

Level B_3: The MIN and MAX transit and stationing times apply to each ship type in each combination.

A solution to an illustrative problem showing the effect on the SSL of each level of processor complexity is presented in Tables 8A to 8D. Table 7 is the list of ships used in the illustrative problem.

Table 8A provides the problem statement and four example solutions that are assumed to have been entered into

Table 6

Halstead's E Metric

Problem	No. of Unique Operators n_1	No. of Unique Operands n_2	n_1+n_2	No. of Total Operators N_1	No. of Total Operands N	N_1+N_2	Program Volume(V) (BITS)	Potential Volume V* (BITS)	Halstead's E Metric
33	3	3	6	14	12	26	67.21	11.61	389.06
15	3	6	9	53	48	101	320.17	24	4,271.16
52	3	7	10	98	90	188	624.16	28.53	13,654.92
31	3	6	9	118	105	223	706.91	24	20,821.70

Note: $n_1^* = n_2$

Table 7

List of Ships

Ship #	Ship II	Transit Time required (Days)	Fuel Supply on Station (Days)
1	CVA	5	10
2	CVA	4	20
3	CVAN	5	200
4	CVAN	5	100
5	CA	4	3
6	CA	7	10
7	CC	4	10
8	CC	8	20
9	CGN	5	100
10	DD	5	10
11	DD	5	5
12	DD	6	10
13	DD	4	12
14	DD	7	10
15	DD	8	10
16	SS	5	7
17	SS	5	10
18	SS	4	10
19	SS	8	12
20	SS	3	12
21	SSN	5	190
22	SSN	6	200
23	SSN	4	250
24	AO	4	20
25	AO	5	30
26	AO	7	20

Legend:

CVA	Attack Aircraft Carrier	CGN	Guided Missile Cruiser (Nuclear Propulsion)
CVAN	Attack Aircraft Carrier (Nuclear Propulsion)	DD	Destroyer
CA	Heavy Cruiser	SS	Submarine
CC	Guided Missile Cruiser	SSN	Submarine (Nuclear Propulsion)
AO	Oiler		

Table 8A

An Illustration of the Effect of
The Three Processor Levels of Complexity

The criteria for the task force are:

1. The ships needed for the task force are:

   - One (1) nuclear attack aircraft carrier (CVAN) or attack aircraft carrier (CVA)
   - Two (2) submarines (SS)
   - Two (2) destroyer (DD)
   - One (1) oiler (AO)

2. The present position of the ships and their speed must permit arrival at the assembly point within 5 days.

3. Fuel on board must permit stationing without resupply for a 10 day period.

These problem criteria specify two distinct combinations of ships, as follows:

Combination 1: CVAN, SS, SS, DD, DD, AO

Combination 2: CVA, SS, SS, DD, DD, AO

Table 8A (Concluded)

An Illustration of the Effect of
The Three Processor Levels of Complexity

Suppose four example solutions are entered as follows:

Example Solution	Ship #	Ship Type	Transit Time Required(days)	Fuel Supply on Station (days)
1:	1	CVA	5	10
	17	SS	5	10
	20	SS	3	12
	10	DD	5	10
	13	DD	4	12
	24	AO	4	30
2:	2	CVA	4	20
	17	SS	5	10
	20	SS	3	12
	10	DD	5	10
	13	DD	4	12
	25	AO	4	30
3:	3	CVAN	5	200
	17	SS	5	10
	20	SS	3	12
	10	DD	5	10
	13	DD	4	12
	24	AO	4	20
	4	CVAN	5	100
	17	SS	5	10
	20	SS	3	12
	10	DD	5	10
	13	DD	4	12
	25	AO	4	30

Table 8B3

SSL Formed by Procedure Level B_1 for the Illustrative Problem

	# of Ships Req. per	Ship Type	Transit MIN	Transit MAX	Stationing MIN	Stationing MAX
Combination 1	1	CVA	3	5	10	200
	2	SS	3	5	10	200
	2	DD	3	5	10	200
	1	AO	3	5	10	200
Combination 2	1	CVAN	3	5	10	200
	2	SS	3	5	10	200
	2	DD	3	5	10	200
	1	AO	3	5	10	200

Table 8C

SSL Formed by Processor Level B_2 for the Illustrative Problem

	# of Ships Required	Ship Type	Transit MIN	Transit MAX	Stationing MIN	Stationing MAX
Combination 1						
	1	CVA	3	5	10	30
	2	SS	3	5	10	30
	2	DD	3	5	10	30
	1	AO	3	5	10	30
Combination 2						
	1	CVAN	3	5	10	200
	2	SS	3	5	10	200
	2	DD	3	5	10	200
	1	AO	3	5	10	200

Table 8D

SSL Formed by Processor Level B_3 for the Illustrative Problem

	# of Ships Required	Ship Type	Transit MIN	Transit MAX	Stationing MIN	Stationing MAX
Combination 1						
	1	CVA	4	5	10	20
	2	SS	3	5	10	12
	2	DD	4	5	10	12
	1	AO	4	5	20	30
Combination 2						
	1	CVAN	5	5	100	200
	2	SS	3	5	10	12
	2	DD	4	5	10	12
	1	AO	4	5	20	30

Note: The range of MIN/MAX values are limited by the available ships on the ship list.

* Numbers correspond to ships on list given in Table 7.

the computer. Table 8B is the SSL formed by Processor Level B_1 from the four example solutions. At Processor Level B_1 the processor scans the transiting and stationing times of all ships in all example solutions and determines the MIN and MAX values of the transiting time and of the stationing time. These MIN/MAX values are then assigned to each ship type in the SSL as shown in Table 8B. For instance, the MAX stationing time is the same (200 days) for all ship types in both ship combinations.

Processor Level B_2 generalizes over each ship combination (Table 8A) instead of over all ship combinations as does Processor Level B_1. This restricted generalization is illustrated in Table 8C where the MAX stationing time is different for the two ship combinations. The difference occurs because the ship with the greatest stationing time (AO) in Example Solutions 1 and 2, which define ship combination 1, has a stationing time of 30 days. In contrast, the corresponding MAX stationing time in Example Solutions 3 and 4 is 200 days for a CVAN.

Finally, Processor Level B_3 (Table 8D) generalizes only over each ship type, and, as a result, the MIN or MAX transiting and stationing times can easily be different for each ship type.

The impact of the various levels of processor generalization (complexity) on the user is that fewer example solutions may be required to establish a broad range of MIN and MAX transiting and stationing times in the SSL with Level B_1 than with Level B_2. In turn, Levels B_1 and B_2 each require less information from the user than does Processor Level B_3.

In contrast, while the amount of information required from the user increases with Processor Level B_1, B_2, and B_3, in that order, the amount of processing required decreases with that order. This is because the processor must make many comparisons of MIN and MAX transiting and stationing time values for Level B_1, fewer comparisons for Level B_2, and the least number of comparisons for Level B_3. Increasing the computer calculation load decreases the amount of information required from the user. As a result, processor complexity can be calculated from two viewpoints: the user's and the processor's. These processor complexity measures are discussed in the next two sections.

Measurement of Processor Complexity (User's Viewpoint)

Processor complexity, from the user's viewpoint, is a function of the amount of information required to specify the SSL for the computer. This amount is a direct function of the level of processor complexity and is computed in the following paragraphs for the problem requiring the greatest amount of information.

Variables to be specified by the user are:

a. The number and type of ships in each combination, i.e., each example solution.

b. The MIN/MAX stationing and transiting times required by the processor level.

Since the processors at each specified level must have sufficient storage to permit processing of all problems, the problems were examined to determine the maximum storage required for each of the two items listed above.

Consider first the amount of information required to specify the number and type of ships. Nine types of ships (see legend to Table 7) are available for selection. Adding "1" to account for a specification of "NO" ship results in a total of 10 required items, from which 1 is selected by the user. Ten items require 3.322 bits; however, since the computer must deal with distinct levels of storage, this value was rounded up to 4 bits.

The maximum number of ships per combination (which occurs in Problem #52) is 10. Thus, 4 X 10 = 40 bits are required to store data for each combination. (Note that some efficiency could be obtained by packing the data, since 4 bits have been allowed, but that because this number is constant for all processor levels, this efficiency was not used.)

The maximum number of ship combinations (from Problem #31) is 14. Thus the storage required for all combinations of ships was 4 X 10 X 14 = 560 bits.

Consider next the amount of information required to specify the MIN/MAX stationing and transiting times. This amount is a function of the processor level and is computed as

follows: since the allowable transiting times are integers within the limits 0 - 15 days, 4 bits each (4 bits permit discrimination of 16 items) are assigned to specification of MIN and MAX stationing times. Thus, specification of MIN/MAX transiting values requires 8 bits. Another 16 bits are required to specify the MIN/MAX values for stationing times, yielding a grand total of 24 bits per specification for a set of MIN/MAX values. Calculations for each processor level are as follows:

Level B_1 requires one specification of MIN/MAX values per problem; thus, the number of bits required = 24.

Level B_2 requires one specification of MIN/MAX values per combination, times the maximum number of combinations; thus, the number of bits required = 24 X 14 = 336.

Level B_3 requires one specification of MIN/MAX values per combination, times the maximum number of combinations, times the maximum number of ships per combination; thus, the number of bits required = 24 X 14 X 10 = 3360.

Thus, the amount of information required to form the SSL is 560 bits (which is the maximum storage required for all the ship combinations and is the same for all processor levels) plus the varying amounts of information (given above) required to specify the limits on transiting and stationing times.

Processor Complexity (Processor Viewpoint)

The processor must have the memory capacity to store all information required to specify the maximum amount of ship data as well as the greatest variety of MIN/MAX transiting and stationing times. Thus, the processor for each level of complexity uses 560 bits to store the ship data and 3360 bits to store the MIN/MAX transiting and stationing time data (based on the requirements for the problem requiring the largest amount of storage).

In addition to the memory storage requirements, computer processing time is required to perform the necessary comparison of ship transiting and stationing times. These comparisons are necessary to determine MIN/MAX transiting and stationing times and, as described previously, the number of comparisons is a function of processor complexity level.

The number of required comparisons may be determined by assuming that the user enters the minimum number of example solutions required to establish the rule selection logic (SSL). If the user enters additional example solutions, additional comparisons are required, but these are not considered in the following analysis.

Level 3A requires comparison of transiting times of all ships to identify the MAX transit time. If there are K_1 ship combinations and an average of K_2 ships per combination required for solution to a problem, then the total number of ships is $K_1 K_2$. Thus $K_1 K_2 - 1$ comparisons are required to find the largest transit time. A similar number of comparisons are required to determine MIN transit, and MAX and MIN stationing times. Thus, a total of $2(K_1 K_2 - 1)$ comparisons are required for a Level 3A processor.

Level 3B requires comparison of the transiting times of all ships in a combination. Thus $(K_2 - 1)$ comparisons are required for each combination. For K_1 combinations, $K_1 (K_2 - 1)$ comparisons are required. A similar number of comparisons are required to determine MIN transit, and MAX and MIN stationing. Thus, the total number of comparisons required is $2K_1 (K_2 - 1)$.

Level 3C requires no comparisons.

Participants

The two participant scenarios consisted of expert computer programmers, and experienced bookkeepers/accountants who were not expert programmers. Selection criteria for establishing whether

a programmer was considered an expert included: work with multiple languages, production of at least 10,000 lines of code, and experience with multiple computer systems. Selection criteria for the non-programming category (bookkeepers/accountants) included 4 years of schooling or experience in the bookkeeping field and no extensive experience as a computer programmer. Complete selection criteria for both participant categories are given in Appendix B.

Participants were obtained by commercial personnel organizations which specialize in placing programmers and bookkeepers/accountants in temporary and permanent positions. Initial screening of participants was done by the personnel agencies in accordance with the selection criteria provided them (see Appendix B). In addition, due to a low flow rate of participants, particularly in the bookkeeping/accounting area, an advertisement was placed in a local newspaper stating that an evaluation of computer equipment was being conducted. Twenty-six people answered this ad, of which 22 were found to be acceptable. A copy of the ad is given in Appendix C.

A total of 75 people participated in the experiment. Of the 75, 15 were not used because they did not achieve an entry level criterion score on the pre-test problem (#93). Also, 13 additional people participated only in the pilot test prior to the experiment. All participants were over 18 years of age.

Participants who were obtained through agencies (all computer programmers and eight of the bookkeepers/accountants) were paid in accordance with the policies of the agencies who obtained them. The bookkeepers/accountants who were obtained via the newspaper advertisement were paid $7.50 per hour.

Procedure

Participants were scheduled for either a morning session beginning at 8:00 or an afternoon session beginning at 1:00. When a participant arrived, he/she was greeted by name, offered a refreshment and asked to fill out a biographical questionnaire (the questionnaire is given in Appendix D) to verify that the participant's experience satisfied the experiment entrance criteria and to obtain additional information regarding the level of experience in his/her particular field. If the participant's experience did not satisfy the criteria, he/she

was not used in the experiment. If the participant was acceptable, the experiment was briefly explained, and the participant was provided with a consent form (Appendix E) having been assured that no personal risk was involved. The participant then signed this form to indicate that he/she understood these arrangements.

The participant was next seated in the experiment room. The room was approximately 12 X 16 feet in size, with a video tape recorder and video monitor located on one table, and the computer and terminal on a separate table. Participants were asked to make themselves comfortable and to adjust the light and ventilation to their satisfaction.

Instructions for the experiment were presented in two parts, both of which were on video tape. The first part described the experiment problem and gave a method for solving the problem including an example solution. (These instructions have been discussed earlier in the section entitled "Experimental Problem.") The second part gave instructions on how to enter data into the computer - and also included an illustrative problem solution. Since this portion of the instruction employed a dynamic display of the operation of the computer, it cannot be presented here.

After the instructions were presented, the participant was seated in front of the computer terminal. He/she was asked to use the manual key pad (consisting of keys labled 0 - 9 and ENTER). The participant was asked to enter example solutions using numbers that corresponded to a ship list (An example is shown in Appendix F) containing various types of ships and their transiting and stationing times. The participants could refer to this list at any time during the experiment. In addition, a pad of paper and pencils were provided for notes, calculations, etc. These sheets were kept in each participant's file for reference.

Participants were told that up to one hour was allotted for each problem, and that the computer would automatically stop the problem at that time. Participants were permitted to take a short break between test problems if they desired.

Data Entry. Participants used the key pad to enter data into the computer. A participant could select from the following functions:

Function #1: Change the Page of Example Solution. To change the example solution page on the terminal display, a participant would press Key #1 followed by the ENTER key. The computer would then request the page number that the participant wanted to view. The participant would enter the page number, resulting in a display of the desired page. A page is a term referring to the information that can be presented on a video monitor at a given time. Additional pages are used to present additional information. In this case, the pages contain the ship numbers for each example solution. This page could display eight example solutions and ships, as each page became full, another page would be generated.

Function #2: Enter Example Solution. When the participant wished to enter an example solution, he/she pressed Key #2 and then the ENTER key. The computer then requested the number of ships in the example. The participant would respond with the appropriate number and then proceed to enter the ship ID numbers.

Function #3: Change Status of Example Solution. To change the status of any solution (ACCEPT or CANCEL) the participant would press Key #3 followed by the ENTER key. The computer would then request the number of the example solution the participant wished to change. Pressing Key #3 changed the status of the example solution from ACCEPT to CANCEL or from CANCEL to ACCEPT. As changes were entered, the computer updated the display to conform to the new ACCEPT/CANCEL status. Only accepted example solutions were used to determine which the correct ship selection was (ESSL). A cancelled example solution was ignored by the computer. Any [illegible] display to be the proceed no [illegible] previously cancelled [illegible] reinstated at a later [illegible].

Function #4: Command the Computer to Generate Solutions. As a participant entered solutions, the computer generalized them to form a ship selection logic (SSL). To be sure that this generalization was correct, a participant could command the computer to generate and display its own example solutions. This was done by presssing Key #4 followed by the ENTER key. After viewing the computer's example solution, a participant could either accept or cancel it. To accept a solution, the participant pressed the ENTER key; to cancel a solution, he/she pressed Key 4 followed by the ENTER key.

Function #5: View the Computer's SSL. To view the SSL formed by the computer (based on the example solutions input by the participant) the participant pressed Key #5 followed by the ENTER key. Upon such a request, the computer would display the various ship combinations that together formed the computer's SSL.

Data Collection

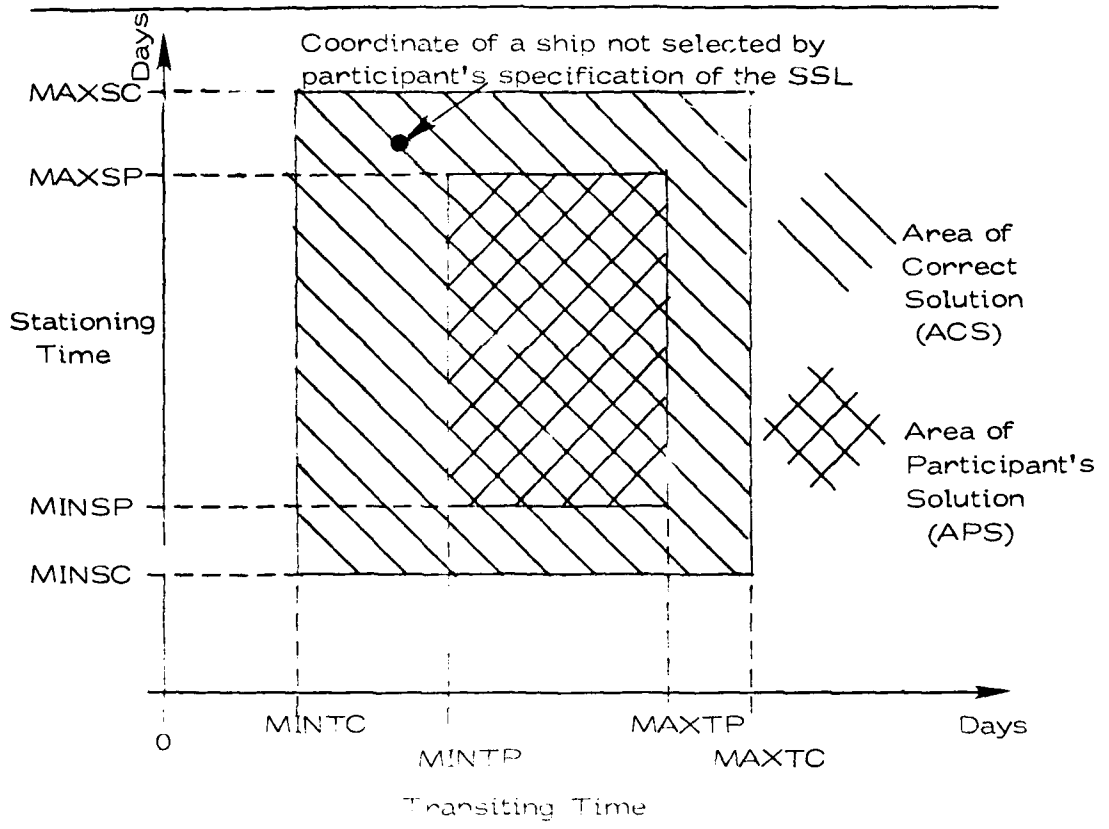
All example solutions were entered by participants into the computer. The computer then recorded all entries, as well as the time expended for each problem solution. This method of collecting data enabled generation of a printout listing detailed information on each participant's activities during the experiment.

Performance Measures

Probability of Correctly Selecting an Acceptable Ship.

Two summary performance measures were developed to reveal how well each participant solved the test problems. The two measures were identical with respect to scoring correct ship combinations, but differed in that Performance Measure 2 provided a penalty for an incorrect ship combination.

The measures were constructed by scoring the MIN/MAX transiting and stationing times for each ship type. As shown in Figure 2, the range of correct values for the MIN/MAX transiting and stationing times, which is a function of the ships on the available ship list, can be viewed as an area



(ACS) Area of correct solution = (MAXTC - MINTC) (MAXSC - MINSC)
(APS) Area of participant's solution = (MAXTP - MINTP) (MAXSP - MINSP)

MINSP	——	Minimum Stationing - Participant
MINSC	——	Minimum Stationing - Correct Solution
MINTC	——	Minimum Transiting - Correct Solution
MINTP	——	Minimum Transiting - Participant
MAXSC	——	Maximum Stationing - Correct Solution
MAXSP	——	Maximum Stationing - Participant
MAXTP	——	Maximum Transiting - Participant
MAXTC	——	Maximum Transiting - Correct Solution

Figure 2. Method of Computing Probability of Acceptable Ship Selection

in space. Every ship of a particular type whose coordinates fall within that area is acceptable. However, if a participant's ship selection logic provides a minimum for either transiting or stationing time greater than the correct minimum, or if the maximum for either transiting or stationing is less than the correct maximum, then the "area" specified by the participant will be contained entirely within the correct "area." As a result, it is possible that a ship with acceptable coordinates might not be selected according to the SSL specified by a given participant. The probability that an eligible ship of a given type (denoted for convenience by the subscript i) will be selected by the participant's SSL is the ratio of the areas just described. That is:

$$P_i = \frac{APS}{ACS} \tag{7}$$

where: APS = Area of participant's SSL for ship type i.

ACS = Area of correct SSL for ship type i.

The probability of selecting all eligible ships for a correctly specified combination of ships is the product of the P_i over all the ship types in that combination. That is:

$$PC_k = \Pi P_i \text{ over all } i \text{ in combination } k \tag{8}$$

where: K represents the k^{th} correctly specified ship combination.

If a problem requires N correct ship combinations, the <u>average probability</u> of selecting an eligible ship regardless of type, which we shall call Performance Measure 1, or PM1, is:

$$PM1 = \frac{1}{N} \sum_{k=1}^{N} PC \tag{9}$$

Performance Measure 1 reflects the probability that acceptable ships are in fact accepted by the SSL specified by the participant. If a particular ship combination is not specified by the participant, the corresponding PC_k value is zero. Note that PM1 does not carry any penalty for specifying incorrect

ship combinations. A measure that does penalize for incorrect combinations, PM2, is given as follows:

$$PM2 = \frac{1}{N}\left[\left(\sum_{k=1}^{N} PC_k\right) - I\right] \qquad (10)$$

where: I is the number of incorrect ship combinations.

PM1 and PM2 were the two summary measures used to analyze participants' data.

Procedure Measure

Two participant strategy measures were developed to test the relationship between the strategy used by participants in developing example solutions and their resulting performance scores. One strategy measure, termed a procedure measure, was designed to examine the pattern of choices among the options available to participants for working with the computer. Referring to Figure 3, it can be seen that, once a participant entered an example solution into the computer, he/she had three choices for the next step. One choice was to input another example solution. Another choice was to review the computer ship selection logic (SSL) to determine the effect of all example solutions entered up to the present time. The third choice was to request that the computer generate its own example solutions. The relative frequency of these three choices is represented by the Probabilities P_1, P_2, and P_3, respectively.

The value of P_1 reflects the propensity of the participant to input either a single example solution or multiple example solutions in sequence. If a participant's P_1 has a low value - near 0 - it may be concluded that the participant tends primarily to input single example solutions. The participant then either requests a display of the SSL or requests that the computer generate example solutions. If P_1 is very high - near 1.0 - then the value indicates that multiple examples were input before proceeding to one of the other input states. In fact, the value of P_1 provides an estimate of the mean number of times example solutions are input in sequence, as follows:

$$\text{Mean number of entries in sequence} = \frac{1}{1 - P_1}. \qquad (11)$$

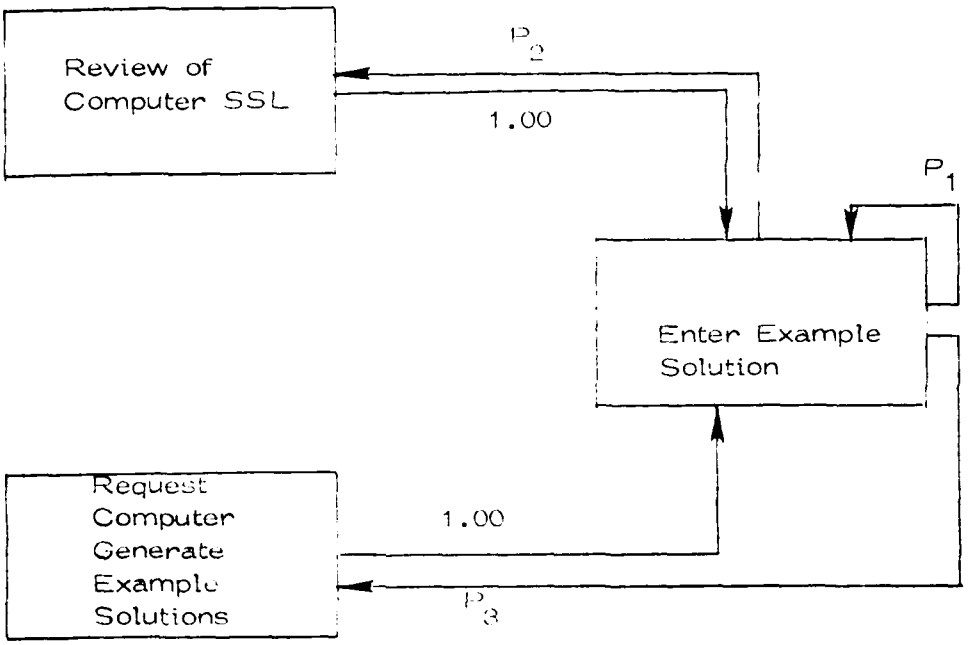


Figure 3. Participant Procedural Strategy

In a similar way the values of P_2 and P_3 reflect the propensity of a participant to view the SSL or to request the generation of sample solutions, respectively. A large value of P_2, for instance, indicates a strategy of inputting a single (or only a few) example solutions and then reviewing the effect on the SSL. A large value of P_3 indicates a propensity to input one (or only a few) input examples and then to request the computer to generate example solutions.

Since the three alternatives represent all of the available choices, the sum $P_1 + P_2 + P_3$ will always equal 1.0. Thus, as P_1 increases, the sum $P_2 + P_3$ will automatically decrease. As a result, the three variables are correlated, and any two may be used in a regression as independent variables to investigate the relationship between participant strategy and task performance.

Combinational Measure (CM)

Another measure of participant strategy involves the nature of the relationship between one example solution and the previous solutions input by a participant. In each problem logical OR conditions are specified which force a participant to develop several different ship combinations in order to completely specify an SSL. Many of these combinations require variations within one ship type - submarine, for example, either nuclear (SSN) or non-nuclear (SS) - while others may require variations over more than one type.

If two submarines are specified in conjunction with other ship types, then each specific combination of the other ships must be associated with three possible variations of SSN's and SS's, as follows:

2 SS and (other ships)

OR

1 SS and 1 SSN and (other ships)

OR

2 SSN and (other ships)

Similarly, if 3 submarines are specified, then each specific combination of other ships must be associated with four variations of SS's and SSN's as follows:

3 SS and (other ships)

OR

2 SS and 1 SSN and (other ships)

OR

1 SS and 2 SSN and (other ships)

OR

3 SSN and (other ships).

Often, variations (i.e., "OR" conditions) within one ship type (e.g., submarine) will be specified in combination with variations (OR conditions) within one or more of another type of ship (e.g., nuclear or non-nuclear aircraft carrier). Variations over multiple ship types require the generation of numerous combinations, and it is difficult to generate all such combinations in the absence of a systematic method.

The combinational strategy measure is designed to detect whether a participant tends to construct example solutions systematically by effecting changes only within one ship type (e.g., SS or SSN) in successive example solutions and by providing all such required variations within one ship type in sequence before proceeding to the next type.

The measure is computed as follows:

1. If on two successive solution examples more than one ship type is changed - score 0.

2. If on two successive solution examples changes occur only within one ship type - score 1.

3. If on three successive solution examples changes occur only within one ship type - score 1 for the first change (according to condition 2 above) and 2 for the second change.

4. In general, for multiple successive solution examples where changes are made only within one ship type - score 1 for the first change, score 2 for the second change, score 3 for the third change, etc.

5. The total score is the sum of the individual scores divided by the maximum score possible for the problem.

This measure is termed "combinational," since the extent to which ship combinations are generated in a systematic manner is the area of interest. A low value of the measure reflects the frequent use of input examples in which changes are made successively over more than one ship type.

DATA ANALYSIS

The term "treatment" in the paragraphs below refers to the effect on the performance data of variations in either problem or processor complexity. Two treatments may thus be applied, either individually or in combination, to the analysis of the performance of participant categories.

The following analyses were conducted:

Analysis 1

An ANOVA was used to determine the significance of the effect on performance of the two participant categories and the two treatments. Two ANOVAs were conducted, one each, for Performance Measures 1 and 2.

Analysis 2

An aposteriori multiple comparison of means test (Student-Newman-Keuls [SNK]) was used to determine whether the means for Performance Measure 1 for each level of the two treatments per participant category were significantly different. In addition, the SNK test was applied to the performance means for the nine treatment combination cells for each participant category.

Analysis 3

A performance count was conducted to determine the number of participants that achieved perfect scores in each treatment combination cell.

Analysis 4

Two-tailed t-tests were used to determine whether the performance scores achieved by participants assigned to various participant categories differed significantly.

Analysis 5

A univariate linear regression analysis for five independent demographic and two independent treatment variables was conducted. The variables were, in the order mentioned:

1. Participant group
2. Participant pre-test score
3. Participant age
4. Participant years of professional experience
5. Participant years of higher education
6. Problem complexity level
7. Processor complexity level

The dependent variable was the score the participants achieved on each test problem using Performance Measure 1. Also, a Step-Wise Linear Regression was conducted using the seven variables.

Analysis 6

To the variables in the analyses above were added the three independent strategy measures, P_1, P_2, and Combinational Measure (CM). (Note that P_3 is <u>not</u> an independent variable since $P_1 + P_2 + P_3 = 1$). The complete list of all ten independent variables therefore concluded with:

8. P_1 } Procedural Strategies
9. P_2 }

10. Combinational Measure (CM)

Univariate linear regression results were obtained for these additional variables, and a second Step-Wise Linear Regression, over all ten of the variables, was conducted.

RESULTS

Summarized Performance Data

Mean performance scores using Performance Measure 1 (probability of selection of an acceptable ship) for each experiment cell (i.e., each combination of the treatments) and each participant category are given in Table 9. Similar data were also obtained using Performance Measure 2 (probability of selection of acceptable ship minus a penalty for selecting an unacceptable ship). These data are given in Table 10.

Performance Measure 1 results achieved by programmers show a decrease in performance with increasing levels of problem and processor complexity. Performance Measure 1 scores achieved by bookkeepers/accountants are lower in every cell than those of the programmers, but do not reveal a monotonic decrease with increasing problem and processor complexity. Also, the participants in group 5 (bookkeepers/accountants) (reference Table 3) "appear" to have higher scores in cells A_1B_2, A_2B_3, A_3B_1 than would be expected if the three bookkeeper/accountant groups (4, 5, 6) had included individuals of equal capability. This result is further evaluated in Analysis 4.

Comparison of the performance scores obtained with Performance Measure 1 and Performance Measure 2 (Tables 9 and 10) reveals that performance evaluated with and without a penalty for errors of commission differs only slightly and shows the same trends in effects for both participant categories, and for both problem and processor complexity levels. The reason for this is that only a few errors of commission went undetected (and therefore not corrected) by the participants. An error of commission results when the SSL specifies one or more ships

Table 9

Mean Scores for Each Experiment Cell

(Performance Measure 1) by Participant Category

Programmers

	A_1	A_2	A_3
B_1	.833	.643	.658
B_2	.675	.619	.521
B_3	.519	.298	.332

Bookkeeper/Accountants

	A_1	A_2	A_3
B_1	.388	.225	.478
B_2	.435	.250	.151
B_3	.075	.139	.030

A_1, A_2, A_3 Problem complexity levels (in increasing order of complexity)

B_1, B_2, B_3 Processor complexity levels (B_1 having the most generalization and B_3 having the least generalization)

Table 1

Mean Scores for Each Experiment Cell

(Performance Measure 2) by Participant Category

Programmers

	A_1	A_2	A_3
B_1	.833	.599	.658
B_2	.660	.600	.515
B_3	.519	.270	.332

Bookkeeper/Accountants

	A_1	A_2	A_3
B_1	.275	.221	.459
B_2	.430	.183	.139
B_3	.071	.128	.030

A_1, A_2, A_3 Problem complexity levels (in increasing order of complexity)

B_1, B_2, B_3 Processor complexity levels (B_1 having the most generalization and B_3 having the least generalization)

that are unacceptable according to the problem instructions. The performance scores that result from Performance Measure 1 are a more direct measure of the effects of errors of omission - a more serious problem.

Plots of mean scores using PM1 for the three experiment problems versus the pre-test score for each participant category are shown in Figure 4, and plots of mean performance scores (PM1) for each participant category versus problem complexity are shown in Figure 5. A similar plot for processor complexity from the user's viewpoint is given in Figure 6. These data suggest that there is a positive correlation between scores on the experiment problem and pre-test score, and that increasing problem complexity results in an initial reduction of performance followed by a leveling off at higher complexity levels. In contrast, the effect of increasing processor complexity is an apparent monotonic decrease in performance.

Analysis 1. ANOVA

ANOVA results for the probability of selecting an acceptable ship (Performance Measure 1) are shown in Table 11. In this analysis, the main effects of participant category, problem complexity, and processor complexity were all found to be significant: $F(1, 54) = 21.38$, $p < .001$; $F(2, 108) = 7.53$, $p \leq .005$; and $F(2, 108) = 35.02$, $p \leq .001$, respectively.

ANOVA results for the probability of selecting an acceptable ship minus the penalty for selecting an unacceptable ship (Performance Measure 2) are shown in Table 12. Again, significant main effects for participant category, problem complexity, and processor complexity were found: $F(1, 54) = 21.96$, $p < .001$; $F(2, 108) = 8.34$, $p \leq .005$; and $F(2, 108) = 30.32$; $p \leq .001$, respectively. No significant interaction effects were found in either analysis.

The probable reason that similar results were obtained in the two analyses is that very few errors of commission were made. Thus, the measure that penalizes for such errors (PM2) provides results that are essentially the same as those that are provided by the measure that ignores such errors (PM1).

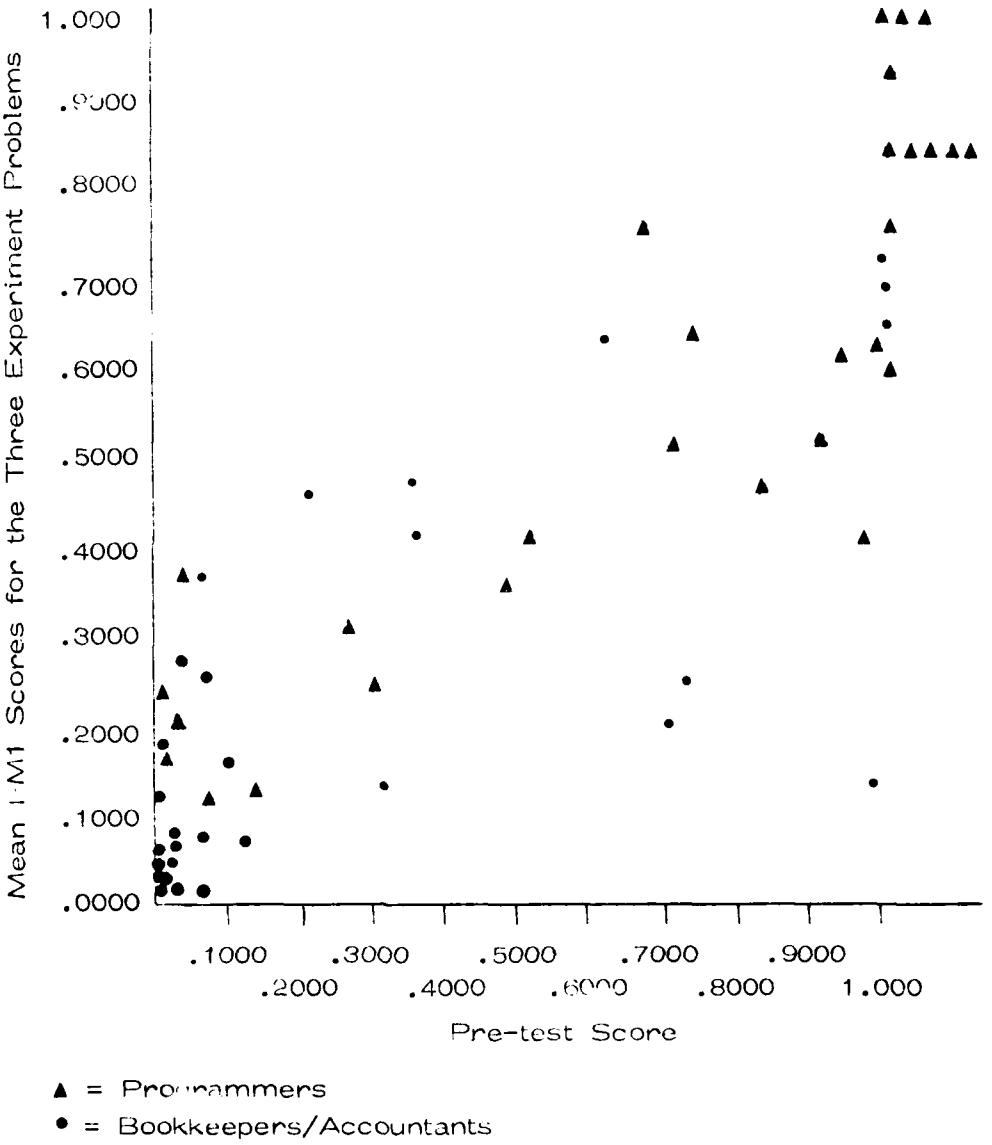


Figure 4. Mean PM1 Scores on Three Experiment Problems Versus Pre-Test Scores

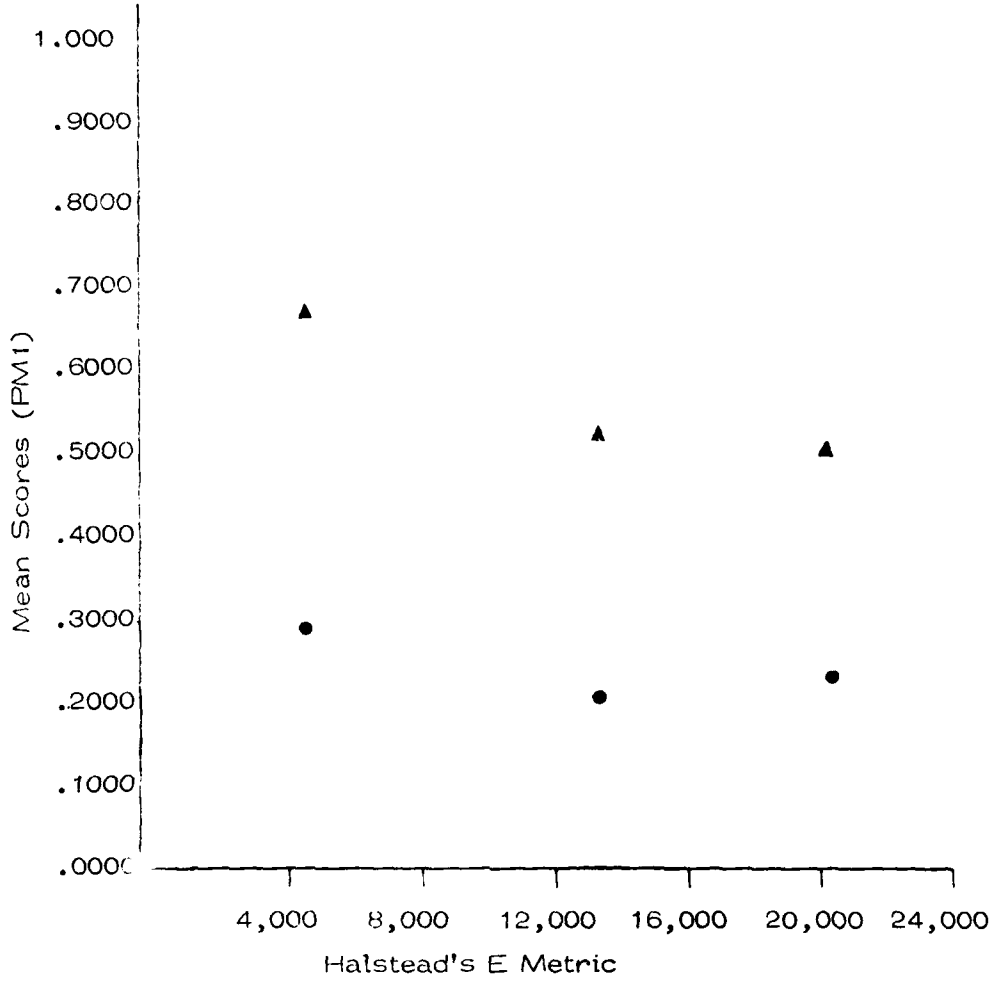


Figure 5. Mean Scores (PM1) for Each Participant Category Versus Problem Complexity

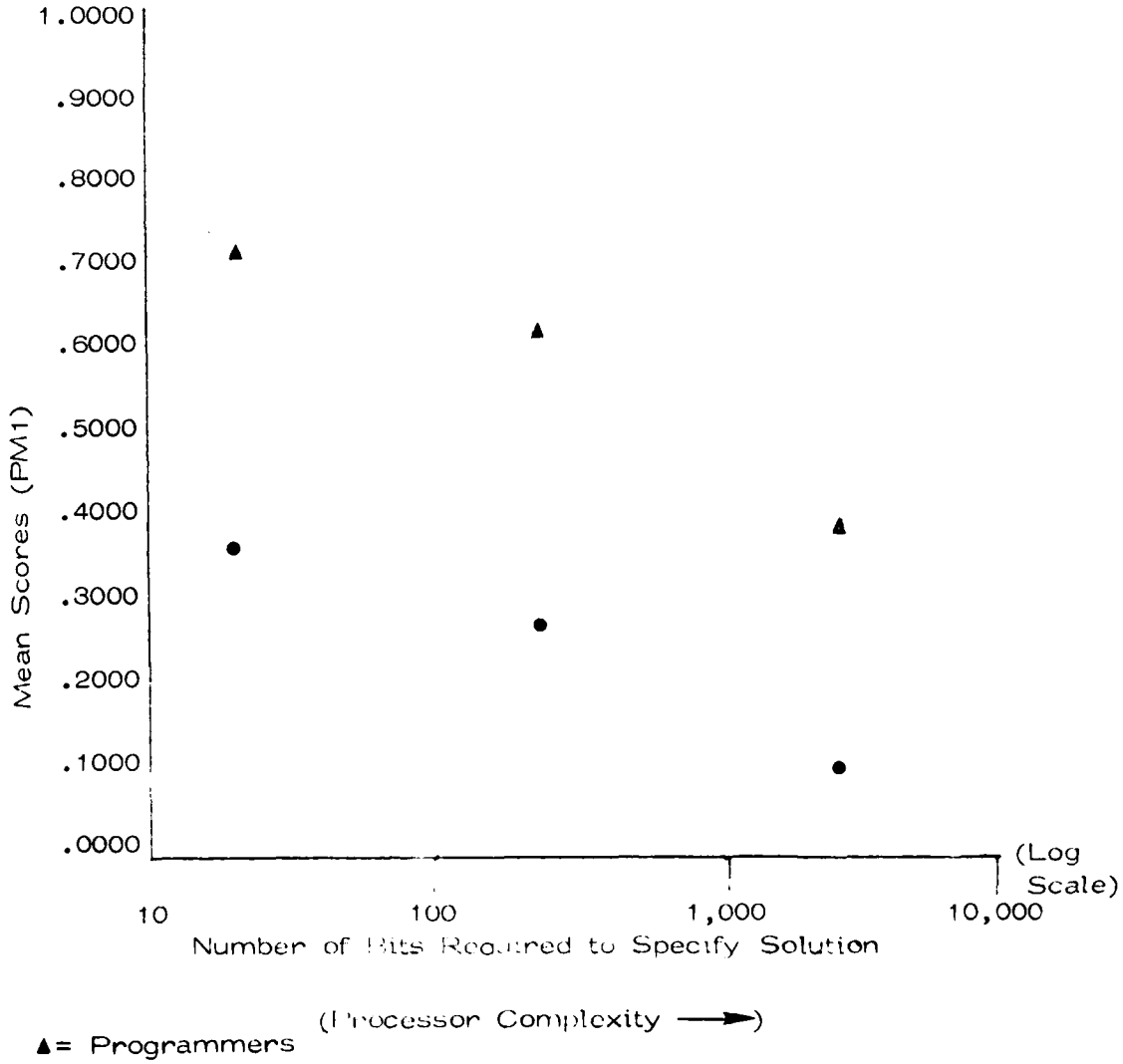


Figure 6. Mean Scores (PM1) for Each Participant Category Versus Processor Complexity (User's Viewpoint)

Table 11

ANOVA Results For Performance Measure 1

Source of Variation	SS	DF	MS	F
Between Subjects	17.422	59		
C (Participants)	4.755	1	4.755	21.382**
Rows	.252	2	.126	.567
C X Rows	.404	2	.202	.910
Sub Within Group	12.010	54	.222	
Within Subjects	8.241	120		
A - Problem Complexity	.630	2	.315	7.537*
B - Processor Complexity	2.930	2	1.465	35.024**
AC	.065	2	.032	.787
BC	.016	2	.008	.194
AB	.078	2	.039	.940
ABC	.000	2	.000	.009
Error (Within)	4.518	108	.041	

* $p < .005$
** $p < .001$

Ref: Plan #9, pgs. 727 - 736, Statistical Principles in Experimental Design, Winer (1971).

Table 12

ANOVA Results For Performance Measure 2

Source of Variation	SS	DF	MS	F
Between Subjects	17.389	59		
C (Participants)	4.834	1	4.834	21.965**
Rows	.243	2	.121	.553
C X Rows	.425	2	.212	.966
Sub Within Group	11.885	54	.220	
Within Subjects	8.642	120		
A - Problem Complexity	.763	2	.381	8.344*
B - Processor Complexity	2.776	2	1.388	30.328**
AC	.061	2	.030	.668
BC	.021	2	.010	.230
AB	.056	2	.028	.614
ABC	.020	2	.010	.219
Error (Within)	4.943	108	.045	

* $p < .005$
** $p < .001$

Ref: Plan #9, pgs. 727 - 730, Statistical Principles in Experimental Design, Winer (1971).

Analysis 2: Aposteriori Test: Comparison of Treatment Means

Tables 13 and 14 show the results of Student-Newman-Keuls (SNK) tests (Sokal & Rohlf, 1969) on the mean scores across all levels of problem complexity for programmers and bookkeepers/accountants, respectively. Tables 15 and 16 show results of the SNK tests on the mean scores across all levels of processor complexity for programmers and bookkeepers/accountants, respectively. Tables 17 and 18 show SNK results for two-way comparisons of the mean scores for the programmer and bookkeeper/accountant experiment cells, respectively.

Results shown in Tables 13 and 15 indicate that problem complexity affected the performance of programmers significantly, but that processor complexity did not have a significant effect. In contrast, the results shown in Tables 14 and 16 suggest that problem complexity did not effect the performance of bookkeepers/accountants significantly, but that processor complexity did significantly affect their performance. Table 17, which is a more detailed analysis dealing with individual treatment cells (i.e., combinations of processor and problem complexity levels), indicates that the performance of programmers was significantly better with Processor Level B_1 at all problem levels than with Processor Level B_3 at Problem Levels A_2 and A_3. With Processor Level B_2, significantly better performance is achieved only at Problem Levels A_1 and A_2. These results suggest two principles of problem/processor usage:

1. A high level of processor generalization (i.e., low complexity from the user's viewpoint) aids a computer user in accurately specifying problem solutions.

2. As problems become more complex, there must be a corresponding increase in processor generalization capability to support superior performance.

It is apparently important for the experienced computer user to be able to select processor generalization capability as a function of the complexity of the problem solution to be specified.

Table 13

Student - Newman - Keuls Test

Results For

Problem Complexity-Programmers

Problem Means*		A_3 .504	A_2 .520	A_1 .676
A_3	.504	.000		
A_2	.520	.016	.000	
A_1	.676	.172**	.156**	.000

*Mean of all programmers' performances on each problem, i.e., A_1, A_2, A_3, using PM1.

A_1, A_2, A_3 are the problem complexity levels (in increasing order of complexity)

Each matrix entry = row value - column value (e.g., entry in the third row, first column is .172 = .676 - .504)

**Indicates row value is significantly different from column value $p < .05$.

Table 14

Student - Newman - Keuls Test

Results For

Problem Complexity-Bookkeepers/Accountants

Problem		A_2	A_3	A_1
	Means*	.205	.220	.300
A_2	.205	.000		
A_3	.220	n/s .015	.000	
A_1	.300	n/s .095	n/s .080	.000

*Mean of all bookkeepers'/accountants' performances on each problem, i.e., A_1, A_2, A_3, using PM1.

A_1, A_2, A_3 are the problem complexity levels (in increasing order of complexity)

Each matrix entry = row value - column value (e.g., entry in the third row, first column is .095 = .300 - .205

n/s = Not significant

Table 15

Student - Newman-Keuls Test

Results For

Processor Complexity-Programmers

Processor Level		B_1	B_3	B_2
Means*		.544	.561	.595
B_1	.544	.000		
B_3	.561	n/s .017	.000	
B_2	.595	n/s .051	n/s .034	.000

*Mean of all programmers' performances on each processor level B_1, B_2, B_3, using PM1.

B_1, B_2, B_3 are the processor complexity levels (B_1 having the most generalization and B_3 having the least generalization)

Each matrix entry = row value - column value (e.g., entry in the third row, first column is .051 = .595 - .544)

n/s = Not significant

Table 16

Student - Newman - Keuls Test

Results For

Processor Complexity-Bookkeepers/Accountants

Processor Level		B_3	B_2	B_1
	Means*	.151	.256	.351
B_3	.151	.000		
B_2	.256	.105**	.000	
B_1	.351	.200**	.095**	.000

*Mean of all bookkeepers'/accountants' performances on each processor level, i.e., B_1, B_2, B_3, using PMT

B_1, B_2, B_3 are the processor complexity levels (B_1 having the most generalization and B_3 having the least generalization)

Each matrix entry = row value - column value (e.g., entry in the third row, first column is .200 = .351 - .151

**Indicates row value is significantly different from column value $p \leq .05$.

Table 17

Student - Newman - Keuls Test

Results for Programmers

Problem/ Processor Level		A_2B_3	A_3B_3	A_1B_3	A_3B_2	A_2B_2	A_2B_1	A_3B_1	A_1B_2	A_1B_1
Means*		.298	.332	.519	.521	.619	.643	.658	.675	.833
A_2B_3	.298	.000								
A_3B_3	.332	.034	.000							
A_1B_3	.519	.221	.187	.000						
A_3B_2	.521	.223	.189	.002	.000					
A_2B_2	.619	.321**	.287	.100	.098	.000				
A_2B_1	.643	.346**	.312**	.125	.123	.025	.000			
A_3B_1	.658	.360**	.326**	.139	.137	.039	.014	.000		
A_1B_2	.675	.377**	.343**	.156	.154	.056	.031	.017	.000	
A_1B_1	.833	.535**	.501**	.314	.312	.214	.189	.175	.158	.000

*Means of all programmers' performances in each cell, i.e., i.e., A_1B_1, A_1B_2, etc., using PM1.

A_1, A_2, A_3 are the problem complexity levels

B_1, B_2, B_3 are the processor complexity levels

Each matrix entry = row value - column value (e.g., entry in second row, first column is .034 = .332 - .298).

**Indicates row is significantly different from column value $p \leq .05$.

Table 18

Student - Newman - Keuls Test

Results For

Bookkeepers/Accountants

Problem/ Processor Level		A_3B_3	A_1B_3	A_2B_3	A_3B_2	A_2B_1	A_2B_2	A_1B_1	A_1B_2	A_3B_1
Means*		.031	.075	.139	.151	.225	.250	.338	.435	.478
A_3B_3	.030	.000								
A_1B_3	.075	.045	.000							
A_2B_3	.139	.109	.064	.000						
A_3B_2	.151	.121	.076	.012	.000					
A_2B_1	.225	.195	.150	.086	.074	.000				
A_2B_2	.250	.220	.175	.111	.099	.025	.000			
A_1B_1	.338	** .308	** .263	.199	.187	.133	.880	.000		
A_1B_2	.435	** .405	** .360	** .296	** .284	.210	.185	.047	.000	
A_3B_1	.478	** .448	** .403	** .339	** .327	.253	.228	.090	.043	.000

*Means of all bookkeepers'/accountants' performance in each cell, i.e., A_1B_1, A_1B_2, etc., using PM1

A_1, A_2, A_3 are the problem complexity levels
B_1, B_2, B_3 are the processor complexity levels

Each matrix entry = row value - column value
(e.g., entry in second row, first column is
.045 = .075 - .031

**Indicates row is significantly different from column value $p \leq .05$

Data presented in Table 18 indicates that performance with Processor Level B_3 for all problem levels for bookkeepers/accountants is significantly lower than for problem/processor complexity combinations A_1B_2 and A_3B_1. Table 18 also contains other treatment combinations that provide statistically significant differences in performance, such as cell (treatment combination) A_1B_1 which leads to significantly improved performance over that of cells A_3B_3 and A_1B_3. These results are more difficult to summarize than those for programmers; however, they support the suggested principles of problem/processor usage given above.

Analysis 3: Percentage of Participants Achieving a Perfect Score

Table 19 shows the percentage of participants (both programmers and bookkeepers/accountants) who achieved a perfect score for each experimental treatment combination. This table indicates that on Problem/Processor Combination A_1B_1 60% of the programmers achieved a perfect score, with a lower percentage achieving perfect scores in other cells. It was possible for participants to achieve a perfect score on any of the problem/processor combinations presented.

Analysis 4: t-Test of Group Mean Scores

Table 20 shows the results for t-tests performed to determine whether the mean scores for the three groups in each participant population were different. No evidence was found to reject the null hypothesis that the participant groups did not differ.

Analysis 5: Step-Wise Linear Regression Analysis #1

Each of the seven variables was tested in a univariate regression with results shown in Table 21. Among the seven variables, pre-test score was the best univariate predictor of performance, suggesting that initial participant capability had a strong influence on performance scores. Next in importance in score prediction, as determined by the amount variance explained in a univariate regression, were: participant cateogry, participant age, processor complexity, and participant years of experience, in that order, followed by years of higher education and problem complexity.

Table 19

Percentage of Bookkeepers/Accountants and Programmers with Perfect Scores

Bookkeepers/Accountants

	A_1	A_2	A_3
B_1	10%	0%	10%
B_2	30%	0%	0%
B_3	0%	0%	0%

Programmers

	A_1	A_2	A_3
B_1	60%	30%	50%
B_2	30%	30%	10%
B_3	40%	0%	20%

Table 20

t-Test for Differences in Participant Groups

Programmer Groups	DF	t-Score
Group 1 compared with Group 2	18	.1980 n/s
Group 1 compared with Group 3	18	.2020 n/s
Group 2 compared with Group 3	18	.0227 n/s

Bookkeeper/Accountant Groups	DF	t-Score
Group 4 compared with Group 5	18	1.4209 n/s
Group 4 compared with Group 6	18	.6152 n/s
Group 5 compared with Group 6	18	1.1960 n/s

Note: n/s = Not significant
Two-tailed t-score required for significance at the .05 level = 2.101
Group 1 and Group 4 used problems 15/2, 52/3, 31/1 assigned in random order.
Group 2 and Group 5 used problems 15/1, 52/2, 31/3 assigned in random order.
Group 3 and Group 6 used problems 15/3, 52/1, 31/2 assigned in random order.

Table 21

Univariate Analysis of the Independent Variables

	Pearson's-R Correlation with the Dependent Variable	Variance Explained	F-Ratio**	
Pre-test Score	.674	46%	143.382	Analysis 5
Participant Category	.437	19%	40.515	Analysis 5
Participant Age	-.359	13%	25.423	Analysis 5
Processor Complexity	-.335	11%	21.681	Analysis 5
Participant Years Experience	-.206	4%	7.657	Analysis 5
Participant Years Higher Education	.196	4%	6.889	Analysis 5
Problem Complexity	-.156	2%	4.289	Analysis 5
Combinational Measures (CM)	.796	63%	296.542	Analysis 6
P_2*	-.527	28%	66.125	Analysis 6
P_1*	.384	15%	29.752	Analysis 6

*Procedural Measures; see p. 42.

**All scores are significant at the $\leq .001$ level
$F_{.001[5,168]} = 4.10$.

Table 22 provides the results of a Step-Wise Linear Regression analysis using performance scores on PM1 as the dependent variable and seven candidate independent variables. The seven independent variables tested in the Step-Wise Linear Regression were: pre-test score, processor complexity, participant category, problem complexity, participant years of experience, participant years of higher education, and participant age. The first five of these accounted for 65% of the variance; the addition of the remaining two increased the amount of variance explained by less than 1%.

Table 23 shows a regression table for the first five of the independent variables listed above. The addition of the remaining variables, participant age and higher education were found not to increase the variance explained. In this table it can be seen that pre-test scores correlate very highly with the dependent variable experiment scores (.674). Also, participant category correlates highly with the dependent variable experiment scores (.437). In the univariate regression, the coefficients for pre-test score were found to be highly significant (t (173) = 9.994, $p < .001$), as were those for participant group (t (173) = 4.663, $p < .001$) and processor complexity (t (173) = -7.367, $p \leq .001$). Also found to be significant were the coefficients for problem complexity (t (173) = -3.175, $p \leq .01$) and participant experience (t (173) = -3.604, $p \leq .001$). The regression equation was also significant (F (5,168) = 61.511, $p \leq .001$).

Analysis 6: Step-Wise Linear Regression Analysis #2

Table 24 provides the results of a second Step-Wise Linear Regression analysis using scores on PM1 as the dependent variable and the following independent variables: procedure measures P_1 and P_2, combinational measure (CM), pre-test score, age, experience, education, problem complexity and processor complexity. Of these independent variables, combinational measure was the best univariate predictor of performance, explaining 63% of the performance variance. This suggests that the method used to systematically develop ship combinations has a strong influence on performance scores. Next in importance in score prediction were processor complexity, and pre-test score. Use of additional variables increased the percentage of variance explained by less than 1%.

Table 25 shows a regression table for the independent variables. Combinational scores correlate very highly with the

Table 22

Step-Wise Multiple Linear Regressions

for

Seven Independent Variables

Number of Independent Variables	Independent Variables Explaining Largest Variance	Variance Explained
1	Pre-test Score	46%
2	Pre-test Score, Processor Complexity	57%
3	Pre-test Score, Processor Complexity, Participant Category	60%
4	Pre-test Score, Processor Complexity, Participant Category, Problem Complexity	62%
5	Pre-test Score, Processor Complexity, Participant Category, Problem Complexity, Years Experience	65%
6 - 7	Less than 1% increase in the amount of variance explained	

Table 23

Correlation and Step-Wise Regression with Five Independent Variables

Independent Variable	Correlation* Coefficient	Regression Coefficient	Standard Deviation	t-Value
Participant Categ.	.437	.200	.043	4.663**
Pre-test Score	.674	.475	.049	9.694**
Experience	-.206	-.022	.006	-3.604**
Problem Complexity	-.156	-.082	.026	-3.175***
Processor Complexity	-.335	-.009	.001	-7.367**

Intercept = .216

Multiple Correlation Coefficient = .804

Explained Variance = .647

Standard Error of Estimate = .230

Analysis of Variance Explained

Source	Sum Sq	DF	Mean Sq	F-Ratio
REG	16.220	5	3.24403	61.511 **
RES	8.860	168	.05274	

*Correlation with Dependent Variable: Score on Each Experiment Problem

** t (173) = 4.663, $p < .001$
t (173) = 9.694, $p < .001$
t (173) = -7.367, $p < .001$
t (173) = -3.604, $p < .001$
F (5,168) = 61.511, $p < .001$

*** t (173) = -3.175, $p < .01$

Table 24

Step-Wise Multiple Linear Regressions

for

Ten Independent Variables

Number of Independent Variables	Independent Variables Explaining Largest Variance	Variance Explained
1	Combinational Measure (CM)	63%
2	CM, Processor Complexity	72%
3	CM, Processor Complexity, Pre-test Score	80%
4 - 10	Less than 1% increase in amount of variance explained	

Table 25

Correlation and Step-Wise Linear Regression with Three Independent Variables

	Correlation	Regression		
Independent Variable	Coefficient	Coefficient	Standard Deviation	t-Value
Combinational Measure	.796	.758	.056	13.584*
Processor Complexity	-.335	-.008	-.001	- 8.821*
Pre-test Score	.674	.304	.038	7.921*

Intercept = .142

Multiple Correlation Coefficient = .891

Explained Variance = .795

Standard Error of Estimate = .174

Analysis of Variance Table

Source	Sum Sq	DF	Mean Sq	F-Ratio
REG	19.927	3	6.64246	219.140*
RES	5.153	170	0.03031	

*t (173) = 13.584, $p < .001$
t (173) = - 8.821, $p < .001$
t (173) = 7.921, $p < .001$

F (3,170) = 219.140, $p < .001$

dependent variable test score (.796). In this regression the coefficients for combinational score were found to be highly significant (t (173) = 13.584, $p \leq .001$), as were those for pre-test scores (t (173) = 7.921, $p \leq .001$) and processor complexity (t (173) = [illegible], $p < .001$). The regression equation was also found to be significant (F (3,170) = 219.140, $p \leq .001$).

DISCUSSION AND CONCLUSIONS

Based on the data analyses reported in the previous section, the following conclusions are offered:

1. Specification of problem solutions via example solutions results in a low rate of errors of commission - as evidenced by the similar performance results obtained from Performance Measurement 1 and Performance Measurement 2. These results do not suggest that input errors were not made in the experiments. Indeed, input errors were made, but most were detected and corrected as work on the problem continued.

   Additional work is planned to compare subject error rates using example solutions, as in this study, to other ways of specifying problem solutions.

2. Programmers as a group performed significantly better than did bookkeepers/accountants on problems studied in this effort. This is evident both from the ANOVA results and from a simple inspection of the raw data. The experiment problem format was designed to be neutral; i.e., favoring neither programmers nor bookkeepers/accountants. Both participant categories are familiar with accurate and detailed work; however, bookkeepers/accountants may be used to more fixed or structured problems whereas programmers may be more at ease with a variety of problems. Thus, the novelty of the problem may have favored programmers over bookkeepers/accountants. Further, programmers may be more comfortable with computer displays; however, bookkeepers/accountants may be more familiar (and may therefore enter

data faster) with a key pad. (A key pad has only numerals arranged in the calculator "matrix" format.) Programmers appear to be more familiar with typewriter-like key arrangements, often using two fingers to enter data. The extent to which these aspects of the situation have affected the results is not known. However, since the combinational measure in a univariate regression explained more variance than did participant category (i.e., programmers, bookkeepers/accountants), the use of systematic methods of generating ship combinations is more important for performance than participant category; and a high percentage of programmers used that systematic method. Further, it is likely that programmers would be more familiar with efficient combination-generating analogs such as the binary number system. Additional work is planned using feedback aids which present combinations in usable forms to participants to determine if performance in both participant categories can be improved.

3. It is clear that problem complexity and processor complexity significantly affect performance. In addition, the two-way multiple comparison test results (SNK) reveal that the level of processor complexity was the dominant factor affecting performance of both participant categories. Programmers working with Processor Level B_3, which provides very little automatic processing and thus little generalization of the input data, experienced a significant reduction of performance. This reduction of performance occurred for all levels of problem complexity. Programmers performed significantly better with Processor Level B_1, which provides considerable generalization of the input data. Also, programmers performed significantly better with Processor Level B_2, provided that the problems were not difficult, i.e., provided that Problem Levels A_1 and A_2 were used. In contrast, the bookkeepers/accountants performed significantly better with Processor Level B_1 on Problem Levels A_1 and A_3 only.

Bookkeepers/accountants failed to show a significant improvement in performance with Processor Level B_2 over Processor Level B_3. A low rate of errors of commission was observed from both participant categories using example solutions. This may be a property of specifying solutions with examples instead of a more general specification. However, the performance measures used (PM 1 and PM 2) combine the effects both of the selection of ship combinations and the development of a range of transiting and stationing times in combination to form the SSL. It would be expected that the use of increased processor generalization might improve the continuous or parametric portions of the specification (i.e., the correct range of transiting and stationing times), but would not affect the frequency with which factors are incorrectly omitted (errors of omission) or incorrectly included (errors of commission). It is further expected that, as stated previously, the use of feedback aids structured to encourage systematic generation of factor combinations (i.e., ship combinations) might aid in reducing errors of omission. Additional analysis and experiments are planned to further investigate those areas.

4. The first regression analysis (Analysis 5) showed that pre-test score was the single most important factor of the seven factors analyzed in predicting performance scores. Pre-test score, obtained on a common problem and problem complexity cell, may measure inherent participant ability and is a strong factor in predicting performance.

5. It is interesting to note that participant experience, measured in years of professional training (and/or work), was not a strong factor in predicting the performance score variance. This result suggests that the skills required to perform well on the experiment problems are not skills naturally developed by years of training and work experience. But, it may also indicate that skills taught some time ago, plus skills learned over years of experience, are of equal value to skills learned recently.

6. Participant procedural strategies measured by P_1 and P_2 explained a substantial portion of the score variance in a univariate regression. However, since P_1 was positively and P_2 negatively correlated with performance, it appears that most participants tended to generate and enter multiple input examples before checking the effect of these inputs, i.e., successful participants developed a plan and correctly interpreted the effects of entering multiple example solutions.

7. Since the combinational measure explained 63% of the score variance and together with processor complexity and pre-test score explained 80% of the score variance, the use of a systematic method for identifying and generating combinations of ships must be a key skill in solving the experiment problems. Indeed, the combinational strategy measure has a higher correlation with the performance score than does participant category. Apparently, therefore, a systematic strategy was used both by successful programmers and bookkeepers/accountants to achieve high scores.

REFERENCES

REFERENCES

Bergland, G.O., & Gordon, R.D. Tutorial: software design strategies. New York: The Institute of Electrical and Electronics Engineers, Inc., 1979.

Biermann, A.W. Approaches to automatic programming. Advances in computers, vol. 15. New York: Academic Press, 1976.

Dahl, O.J., Dijkstra, E.W., & Hoare, C.A.R. Structured programming. New York: Academic Press, 1972.

Dijkstra, E.W. A discipline of programming. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1976.

Halstead, M.H. Elements of software science. New York: Elsevier Holland, Inc., 1977.

Jensen, R.W., & Tonies, C.C. Software engineering. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1979.

McCune, B.P. The psi program model builder synthesis of very high-level programs. Proceedings of the symposium on artificial intelligence and programming languages, ACM, 1977.

Miller, E. & Howden, W.E. Tutorial: software testing and validation techniques. New York: The Institute of Electrical and Electronics Engineers, Inc., 1978.

Miller, G.A. The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychological Review, 1956, 63, pp. 81 - 97.

Motley, R.W., et al. Statistical prediction of programming errors (AD/A - 041 106). Griffiss Air Force Base, New York: Rome Air Development Center Air Force Systems Command, 1977.

Myers, G.J. The art of software testing. New York: John Wiley & Sons, 1979.

Prywes, N.S. Automatic generation of computer programs. Advances in computers, vol. 16. Academic Press, 1977.

Ramanoorthy, C.U., & Yeh, R.T. Tutorial: software methodology. New York: The Institute of Electrical and Electronics Engineers, Inc. 1978.

Schneiderman, B. Software psychology. Cambridge, Massachusetts: Winthrop Publishers, Inc. 1980.

Sokal, R.R., & Rohlf, F.J. Biometry. San Francisco: W.H. Freeman and Company, 1969.

Thayer, T.A. Software reliability study (AD-A030 798). Griffiss Air Force Base, New York: Rome Air Development Center Air Force Systems Command, 1976.

Wegner, P. Programming with ADA: an introduction by means of a graduated sample. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1980.

Welsh, J., & McKeag, M. Structured system programming. London: Prentice-Hall International, Inc., 1980.

Wile, D., Balzer, R., & Goldmann, N. Automated derivation of program control structure from natural language description. Proceedings from the Symposium on artificial intelligence and programming languages, ACM, August 1977.

Winer, B.J. Statistical principles in experimental design (2nd ed. pp. 727 - 736). New York: McGraw-Hill, 1971.

Yourdan, E., & Constantine, L.L. Structured design. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1979.

APPENDIX A

PROBLEM STATEMENTS

TEST PROBLEM # 93

Now suppose the mission has been modified. You are asked to develop the example solutions required to specify the ship selection logic for the modified mission. The selection criteria for the task force has been modified to:

1. The ships needed for the task force are:
   - 2 Attack Aircraft Carriers (CVA or CVAN), and
   - 2 Submarines (SS)
2. The present position of the ships and their speed must permit the ships to arrive at the assembly point within 5 days.
3. Supplies on board each ship, including fuel, must permit stationing without resupply for a 10 day period.

This task force criteria specifies 3 combinations of ship types as follows:

- 2 CVA and 2 SS

  or

- 2 CVAN and 2 SS

  or

- 1 CVA and 1 CVAN and 2 SS

You must develop example solutions for each ship combination using ships with suitable transit and stationing times.

Harry's example input list has been expanded and input to the computer. Please use the expanded example input list (to identify ships by their item number) to develop example solutions, enter the solutions into the computer, and test the computer's generalization of your examples by requesting additional solutions for the computer. If a computer solution is in error, develop and enter additional solutions using the aids the computer automatically provides. Continue this process (inputting example solutions and testing the computer) until you are satisfied that the computer has properly generalized your solution examples, i.e., it uses your examples to provide its own correct solutions.

PLEASE START NOW.

WHEN YOU HAVE FINISHED:

When you believe the computer SSL is correct, give it this test:

Would the SSL select the following ships for the task force?

	Transit Time	Stationing Time
CVA	3 days	17 days
CVAN	2 days	75 days
SS	3 days	11 days
SS	5 days	25 days

Yes ☐

No ☐

TEST PROBLEM #31

Now suppose the mission has been modified. You are asked to develop the example solutions required to specify the ship selection logic for the modified mission. The selection criteria for the task force has been modified to:

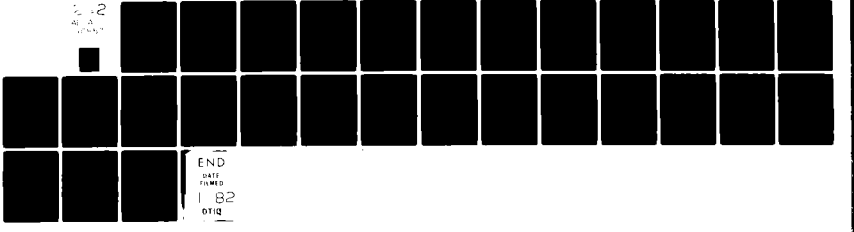
1. The ships needed for the task force are:
   - 2 Attack Aircraft Carriers (CVA) or 1 Nuclear Attack Aircraft Carrier (CVAN), and
   - (2 Submarines (SS or SSN) and 3 Destroyers (DD)) or (3 Submarines (SS or SSN) and 2 Destroyers (DD)) and
   - 1 Oiler (AO).
2. The present position of the ships and their speed must permit the ships to arrive at the assembly point within 5 days.
3. Supplies on board each ship, including fuel, must permit stationing without resupply for a 10 day period.

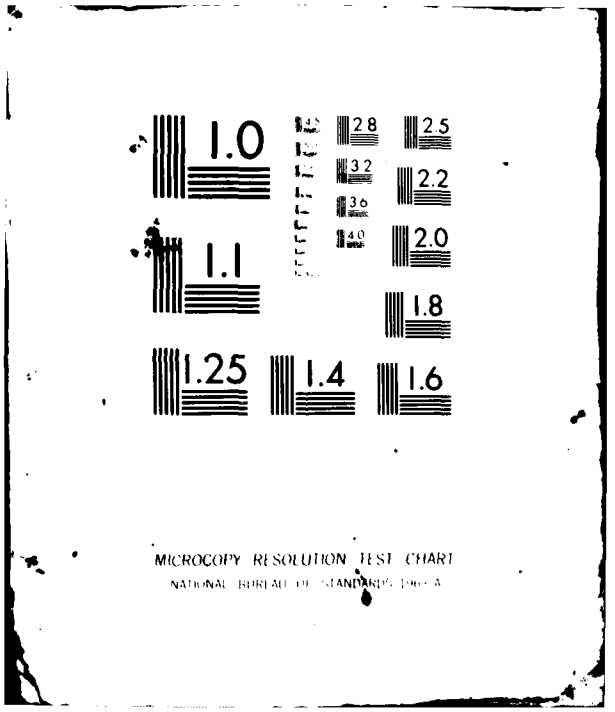
Harry's example input list has been expanded and input to the computer. Please use the expanded example input list (to

identify ships by their item number) to develop example solutions, enter the solutions into the computer, and test the computer's generalization of your examples by requesting additional solutions for the computer. If a computer solution is in error, develop and enter additional solutions using the aids the computer automatically provides. Continue this process (inputting example solutions and testing the computer) until you are satisfied that the computer has properly generalized your solution examples, i.e., it uses your examples to provide its own correct solutions.

PLEASE START NOW.

AD-A108 570 PERFORMANCE MEASUREMENT ASSOCIATES INC VIENNA VA F/G 9/2
EFFECT OF AUTOMATIC PROCESSING ON SPECIFICATION OF PROBLEM SOLU--ETC(U)
MAR 81 E M CONNELLY, R F COMEAU, P JOHNSON N00014-79-C-0739
UNCLASSIFIED PMA-81-361 NL




1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

TEST PROBLEM #15

Now suppose the mission has been modified. You are asked to develop example solutions for the modified mission. The selection criteria for the task force has been modified to:

1. The ships needed for the task force are:
   - 1 Attack Aircraft Carrier (CVA or CVAN), and
   - 2 Submarines (SS or SSN), and
   - 4 Destroyers (DD), and
   - 1 Oiler (AO)
2. The present position of the ships and their speed must permit the ships to arrive at the assembly point within 5 days.
3. Supplies on board each ship, including fuel, must permit stationing without resupply for a 10 day period.

Harry's example input list has been expanded and input to the computer. Please use the expanded example input list (to identify ships by their item number) to develop example solutions, enter the solutions into the computer, and test the computer's generalization of your examples by requesting additional solutions for the computer. If a computer solution is in error, develop and enter additional solutions using the aids the computer automatically provides.

Continue this process (inputting example solutions and testing the computer) until you are satisfied that the computer has properly generalized your solution examples, i.e., it uses your examples to provide its own correct solutions.

PLEASE START NOW.

TEST PROBLEM #52

Now suppose the mission has been modified. You are asked to develop example solutions for the modified mission. The selection criteria for the task force has been modified to:

1. The ships needed for the task force are:
   - 1 Attack Aircraft Carrier with Nuclear Propulsion (CVAN), and
   - 2 Guided Missile Cruisers (CG or CGN), and
   - 2 Submarines (SS or SSN), and
   - 3 Destroyers (DD), and
   - 2 Oilers (AO)
2. The present position of the ships and their speed must permit the ships to arrive at the assembly point within 5 days.
3. Supplies on board each ship, including fuel, must permit stationing without resupply for a 10 day period.

Harry's example input list has been expanded and input to the computer. Please use the expanded example input list (to identify ships by their item number) to develop example solutions, enter the solutions into the computer, and test the computer's generalization of your examples by requesting additional solutions for

the computer. If a computer solution is in error, develop and enter additional solutions using the aids the computer automatically provides. Continue this process (inputting example solutions and testing the computer) until you are satisfied that the computer has properly generalized your solution examples, i.e., it uses your examples to provide its own correct solutions.

PLEASE START NOW.

APPENDIX B

PARTICIPANT CRITERIA

Programmer Selection Criteria

Programmers selected to participate in the experiment should have the following experience:

1. Coding and Testing Experience:

   Participants shall have experience in at least two higher level languages from among the following list:

   a. BASIC

   b. FORTRAN

   c. COBOL

   d. PL-1

   e. ALGOL

   f. PASCAL

   Participants shall have been responsible for the entire process of coding and testing the programs and shall have produced at least 10,000 lines of program code (100 programs of at least 100 lines each).

2. Computer Systems:

   Participants shall have programmed with two different computer systems, i.e., OS and DOS, or equipment from different manufacturers, such as Honeywell, IBM, DEC, etc.

3. Timesharing

   Participants shall have at least 160 hours of on-line terminal usage on a timesharing system.

4. Application Systems:

   Participants shall have coded at least two systems of programs of at least ten programs each.

5. JCL:

   Participants shall have been responsible for at least twenty programs.

6. Participants shall have used "modular" or "structured" programming techniques.

7. Each participant must be over 18 years of age.

8. English must be the participants' native language.

Bookkeeper/Accountant Selection Criteria

Bookkeepers and accountants selected to participate in the experiment shall have the following experience and qualifications:

1. Participants must be over 18 years of age.

2. English must be the participants' native language.

3. Participants should never have been a professional computer programmer.

4. Participants should have at least 2 years of bookkeeping/accounting schooling AND 2 years of bookkeeping/accounting work experience,

   OR

5. Participants should have at least 4 years of bookkeeping/accounting work experience,

   OR

6. Participants should have at least 4 years of bookkeeping/accounting schooling.

APPENDIX C

NEWSPAPER ADVERTISEMENT

NEWSPAPER ADVERTISEMENT

Accountants/Bookkeepers

A firm in Vienna needs the help of accountants and bookkeepers in evaluating a computer system for use by non programmers. 4 - 5 hour evaluation. 4 years bookkeeping/accounting schooling or experience needed. $7.50 per hour. 938-1600.

APPENDIX D

PARTICIPANT BIOGRAPHICAL FORM

C-36

FOR OFFICIAL USE ONLY

Participant No. _____

Date: _____

PARTICIPANT BIOGRAPHICAL FORM

Name: _____

Age: _____ Sex: _____

Education: High School Graduate: Yes _____ No _____

_____ years of higher education Degree: _____

Major: _____ Minor: _____

Bookkeeping and Accounting Experience

Please check all areas in which you have worked and enter the number of months of full-time experience in that area (2 years part time = 1 year full time).

_____Accounts Payable/Cash Disbursements _____

_____Accounts Receivable/Cash Receipts _____

_____Trial Balance Preparation _____

_____Preparation of Periodic Reports (Income Statement, Balance Sheets, etc.) _____

_____Accounting for Inventory _____

_____Payroll _____

_____Taxes _____

Circle the number of years in which you have worked as a full-time bookkeeper or accountant: 1 2 3 4 5 6 7 8 9 10 10+

Please make any additional comments you believe best describes your professional expertise.

Higher Level Language Programming Experience

Please circle all languages in which you have programmed and check your best estimate of the number of programs coded in that language.

TYPE	1-10	11-20	21-30	31-50	51-100	100^{+}
BASIC						
FORTRAN						
COBOL						
PL-1						
ALGOL						
PASCAL						
_____						
_____						

Circle your best estimate of the total number of programs you have coded (any language, any computer).

0 1-100 101-200 201-300 301-500 500^{+}

Data Processing Experience

Check all areas in which you have worked and enter the number of months of full-time experience in that area (2 years part time = 1 year full time).

_____Data Entry	_____	_____System Analysis	_____
_____Production Control	_____	_____Data Base Administration	_____
_____Operations	_____	_____Data Communications	_____
_____Applications Programming	_____	_____Other(s)	_____
_____System Programming	_____		

Circle the number of years in which you have worked as a full-time data processing professional: 1 2 3 4 5 6 7 8 9 10 10^{+}

Circle your best estimate of the number of programs for which you have coded the Job Control Language (JCL) necessary for testing or production runs.

1-10 11-20 21-30 31-50 51-100 100^{+}

Please list up to 5 computer/operating systems on which you have worked which you believe best represents your experience. Examples: IBM 370/155 OS, PDP-11 RT-11.

Please make any additional comments you believe best describes your professional expertise.

APPENDIX E

SAMPLE RELEASE FORM

CONTRACT TO ACT AS A RESEARCH SUBJECT
FOR THE DEPARTMENT OF THE NAVY

NAME OF PARTICIPANT_____ DATE:_____
(Please Print)

ADDRESS:_____ TELEPHONE:_____

1. I authorize Mr. Edward M. Connelly who is the Principal Investigator for this project or his representatives, Mr. Bob Comeau or Mrs. Joanne Connelly to collect and analyze my examples of solutions to test problems.

2. This project has been explained to me by _____.
(Print)

It has been pointed out to me that my solution examples will not be made known to any individual , other than appropriate members of the research team, or for any purpose other than the data analysis to be performed by the research team. No information concerning my performance on the experimental task will be disclosed to anyone other than the research team without my written permission.

3. I understand that there are no special risks of any kind associated with my participation in this study.

4. I understand that the project may further the understanding of how various programming aids can assist a programmer or other person to specify a computer program.

5. I understand that Mr. Edward M. Connelly, Mr. Bob Comeau, Mrs. Joanne Connelly will answer any questions I may have about the project.

6. I understand that by signing this form, I have waived none of my legal rights that may be associated with liability for negligence on the part of Performance Measurement Associates, Inc.

7. I state that I am 18 years of age or older.

Subject:_____ Witness:_____
(Signature) (Signature)

_____
Edward M. Connelly
Principal Investigator

APPENDIX F

EXAMPLE SHIP LIST

SHIP LIST

Ship ID Number	Ship Type	Transit Time	Station Time
1	CVAN	2	11
2	CVAN	5	11
3	CVAN	2	19
4	CVAN	2	112
5	CVAN	4	10
6	CVAN	4	176
7	CVAN	1	43
8	CVAN	3	72
9	CVAN	3	200
10	CVAN	2	148
11	CVAN	3	168
12	CVAN	3	10
13	CVAN	3	69
14	CVAN	1	158
15	CVAN	2	193
16	CVAN	4	69
17	CVAN	3	145
18	CVAN	3	149
19	CVAN	5	179
20	CVAN	3	200
21	CVA	2	20
22	CVA	3	15
23	CVA	2	16
24	CVA	4	17
25	CVA	3	17
26	CVA	3	13
27	CVA	5	17
28	CVA	2	16
29	CVA	4	18
30	CVA	2	16
31	CVA	1	14
32	CVA	2	10
33	CVA	4	18
34	CVA	5	12
35	CVA	2	18
36	CVA	4	19
37	CVA	1	19

Ship ID Number	Ship Type	Transit Time	Station Time
38	CVA	2	11
39	CVA	2	10
40	CVA	4	20
41	CA	4	29
42	CA	4	30
43	CA	4	29
44	CA	4	23
45	CA	3	24
46	CA	5	14
47	CA	4	22
48	CA	4	10
49	CA	2	30
50	CA	4	26
51	CA	3	29
52	CA	2	28
53	CA	2	29
54	CA	1	19
55	CA	1	19
56	CA	4	10
57	CA	3	23
58	CA	3	28
59	CA	4	20
60	CA	5	26
61	CGN	4	289
62	CGN	1	155
63	CGN	4	46
64	CGN	4	228
65	CGN	2	147
66	CGN	5	215
67	CGN	2	300
68	CGN	3	10
69	CGN	2	17
70	CGN	4	148
71	CGN	3	243
72	CGN	4	290
73	CGN	3	150
74	CGN	2	191
75	CGN	2	10
76	CGN	2	300
77	CGN	3	237

Ship ID Number	Ship Type	Transit Time	Station Time
78	CGN	3	154
79	CGN	1	194
80	CGN	5	102
81	CG	3	20
82	CG	3	14
83	CG	2	14
84	CG	2	16
85	CG	2	28
86	CG	5	14
87	CG	2	18
88	CG	3	10
89	CG	2	30
90	CG	4	30
91	CG	3	25
92	CG	2	21
93	CG	5	18
94	CG	4	10
95	CG	2	27
96	CG	2	14
97	CG	2	16
98	CG	1	27
99	CG	1	14
100	CG	4	29
101	DD	3	10
102	DD	5	26
103	DD	2	19
104	DD	2	24
105	DD	3	26
106	DD	4	27
107	DD	1	21
108	DD	2	24
109	DD	2	11
110	DD	2	29
111	DD	4	23
112	DD	2	30
113	DD	3	21
114	DD	4	17
115	DD	3	30
116	DD	4	10
117	DD	1	12

Ship ID Number	Ship Type	Transit Time	Station Time
118	DD	5	19
119	DD	4	19
120	DD	3	13
121	DD	4	19
122	DD	3	25
123	DD	5	23
124	DD	4	29
125	DD	4	16
126	DD	4	29
127	DD	4	30
128	DD	4	18
129	DD	2	29
130	DD	5	23
131	DD	1	15
132	DD	2	30
133	DD	2	10
134	DD	2	22
135	DD	3	10
136	DD	3	11
137	DD	1	28
138	DD	4	19
139	DD	4	19
140	DD	4	16
141	SSN	2	226
142	SSN	3	192
143	SSN	2	17
144	SSN	3	10
145	SSN	4	152
146	SSN	2	210
147	SSN	2	300
148	SSN	3	111
149	SSN	4	136
150	SSN	4	229
151	SSN	3	181
152	SSN	1	156
153	SSN	4	148
154	SSN	3	300
155	SSN	2	10
156	SSN	4	295
157	SSN	5	47

Ship ID Number	Ship Type	Transit Time	Station Time
158	SSN	1	206
159	SSN	5	12
160	SSN	4	198
161	SS	2	10
162	SS	4	10
163	SS	1	20
164	SS	3	29
165	SS	3	19
166	SS	4	22
167	SS	2	15
168	SS	4	27
169	SS	2	28
170	SS	4	16
171	SS	5	11
172	SS	3	17
173	SS	1	16
174	SS	2	26
175	SS	2	17
176	SS	4	17
177	SS	4	30
178	SS	4	23
179	SS	5	29
180	SS	4	30
181	AO	4	50
182	AO	2	46
183	AO	4	10
184	AO	4	28
185	AO	3	50
186	AO	4	11
187	AO	5	17
188	AO	2	40
189	AO	1	19
190	AO	3	39
191	AO	4	27
192	AO	2	20
193	AO	3	23
194	AO	3	15
195	AO	4	10
196	AO	3	27
197	AO	4	12

Ship ID Number	Ship Type	Transit Time	Station Time
198	AO	1	30
199	AO	5	25
200	AO	2	45

DISTRIBUTION LIST

OSD

CDR Paul R. Chatelier
Office of the Deputy Under
Secretary of Defense
OUSDR (E & LS)
Pentagon, Room 3D129
Washington, D.C. 20301

Department of the Navy

Director
Engineering Psychology Programs
Code 442
Office of Naval Research
800 North Quincy Street
Arlington, Va. 22217

Director
Communication & Computer Technology
Code 240
Office of Naval Research
800 North Quincy Street
Arlington, Va. 22217

Director
Information Systems Program
Code 411-IS
Office of Naval Research
800 North Quincy Street
Arlington, Va. 22217

Director
Manpower, Personnel & Training
Code 270
Office of Naval Research
800 North Quincy Street
Arlington, Va. 22217

Department of the Navy

Director, Physiology & Neuro
Biology Programs
Code 441B
Office of Naval Research
800 North Quincy Street
Arlington, Va. 22217

Special Assistant for Marine
Corps Matters
Code 100M
Office of Naval Research
800 North Quincy Street
Arlington, Va. 22217

Commanding Officer
ONR Eastern/Central
Regional Office
ATTN: Dr. J. Lester
Building 114, Section D
666 Summer Street
Boston, Mass. 02210

Commanding Officer
ONR Branch Office
ATTN: Dr. C. Davis
536 South Clark Street
Chicago, Ill. 60605

Commanding Officer
ONR Western Regional Office
ATTN: Dr. E. Cloye
1030 East Green Street
Pasadena, CA. 91106

Office of Naval Research
Scientific Liaison Group
American Embassy, Room A-407
APO San Francisco, CA. 96503

Department of the Navy

Director
Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375

Dr. Michael Melich
Communications Sciences Division
Code 7500
Naval Research Laboratory
Washington, D.C. 20375

Dr. Robert G. Smith
Office of the Chief of Naval Operations, OP987H
Personnel Logistics Plans
Washington, D.C. 20350

Dr. Jerry C. Lamb
Combat Control Systems
Naval Underwater Systems Center
Newport, RI 02840

Naval Training Equipment Center
ATTN: Technical Library
Orlando, Florida 32813

Human Factors Department
Code N215
Naval Training Equipment Center
Orlando, Florida 32813

Dr. Alfred F. Smode
Training Analysis and Evaluation Group
Naval Training Equipment Center
Code N-00T
Orlando, Florida 32813

Department of the Navy

CDR R. Gibson
Bureau of Medicine & Surgery
Aerospace Psychology Branch
Code 513
Washington, D.C. 20372

CDR Robert Biersner
Naval Medical R & D Command
Code 44
Naval Medical Center
Bethesda, MD. 20014

Dr. Arthur Bachrach
Behavioral Sciences Department
Naval Medical Research Institute
Bethesda, MD. 20014

Dr. George Moeller
Human Factors Engineering Branch
Submarine Medical Research Lab.
Naval Submarine Base
Groton, CT. 06340

Head
Aerospace Psychology Department
Code L5
Naval Aerospace Medical Research Lab.
Pensacola, Florida 32508

Dr. James McGrath
CINCLANT FLT HQS
Code 04E1
Norfolk, VA 23511

Department of the Navy

Navy Personnel Research and Development Center
Planning & Appraisal Division
San Diego, CA. 92152

Navy Personnel Research and Development Center
Management Systems
Code 303
San Diego, CA. 92152

Navy Personnel Research and Development Center
Performance Measurement & Enhancement
Code 309
San Diego, CA. 92152

Human Factors Engineering Branch
Code 1226
Pacific Missile Test Center
Point Mugu, CA. 93042

Dean of the Academic Departments
U.S. Naval Academy
Annapolis, MD. 21402

Dr. Gary Poock
Operations Research Department
Naval Postgraduate School
Monterey, CA. 93940

Department of the Navy

Dean of Research Administration
Naval Postgraduate School
Monterey, CA. 93940

Mr. Warren Lewis
Human Engineering Branch
Code 8231
Naval Ocean Systems Center
San Diego, CA. 92152

Dr. A.L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380

Commanding Officer
MCTSSA
Marine Corps Base
Camp Pendleton, CA. 92055

Mr. Arnold Rubinstein
Naval Material Command
NAVMAT 0722 - Rm. 508
800 North Quincy Street
Arlington, VA. 22217

Commander
Naval Air Systems Command
Human Factors Programs
NAVAIR 340F
Washington, D.C. 20361

Commander
Naval Air Systems Command
Crew Station Design,
NAVAIR 5313
Washington, D.C. 20361

Department of the Navy

Mr. Phillip Andrews
Naval Sea Systems Command
NAVSEA 0341
Washington, D.C. 20362

Commander
Naval Electronics Systems Command
Human Factors Engineering Branch
Code 4701
Washington, D.C. 20360

Human Factors Section
Systems Engineering Test Directorate
U.S. Naval Air Test Center
Patuxent River, MD. 20670

CDR W. Moroney
Code 55MP
Naval Postgraduate School
Monterey, CA. 93940

Mr. Merlin Malehorn
Office of the Chief of Naval Operations (OP-115)
Washington, D.C. 20350

Department of the Army

Mr. J. Barber
HQS, Department of the Army
DAPE-MBR
Washington, D.C. 20310

Dr. Joseph Zeidner
Technical Director
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, Va. 22333

Department of the Army

Director, Organizations and Systems Research Laboratory
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, Va. 22333

Technical Director
U.S. Army Human Engineering Labs.
Aberdeen Proving Ground, MD. [illegible]

Maj. Gerald P. Krueger
USA Medical R & D Command HQ
SGRD - PLC
Fort Detrick, MD. 21801

ARI Field Unit - USAREUR
ATTN: Library
C/O ODCSPER
HQ USAREUR & 7th Army
APO New York 09403

Department of the Air Force

U.S. Air Force Office of Scientific Research
Life Sciences Directorate, NL
Bolling Air Force Base
Washington, D.C. 20332

Chief, Systems Engineering Branch
Human Engineering Division
USAF AMRL/HES
Wright-Patterson AFB, OH [illegible]

Air University Library
Maxwell Air Force Base, AL [illegible]

Dr. Earl Alluisi
Chief Scientist
AFHRL/CCN
Brooks AFB, TX 78235

Foreign Addressees

North East London Polytechnic
The Charles Myers Library
Livingstone Road
Stratford
London E15 2LJ
ENGLAND

Dr. Kenneth Gardner
Applied Psychology Unit
Admiralty Marine Technology Establishment
Teddington, Middlesex TW11 OLN
ENGLAND

Director, Human Factors Wing
Defense & Civil Institute of Environmental Medicine
Post Office Box 2000
Downsview, Ontario M3M 3B9
CANADA

Dr. A.D. Baddeley
Director, Applied Psychology Unit
Medical Research Council
15 Chaucer Road
Cambridge, CB2 2EF
ENGLAND

Other Government Agencies

Defense Technical Information Center
Cameron Station, Bldg. 5
Alexandria, Va. 22314

Dr. Craig Fields
Director,
System Sciences Office
Defense Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, Va. 22209

Dr. M. Montemerlo
Human Factors & Simulation Technology, RTE-6
NASA HQS
Washington, D.C. 20546

Other Organizations

Dr. H. McI. Parsons
Human Resources Research Office
300 N. Washington Street
Alexandria, Va. 22314

Dr. Jesse Orlansky
Institute for Defense Analyses
400 Army-Navy Drive
Arlington, Va. 22202

Dr. Arthur I. Siegel
Applied Psychological Services, Inc.
404 East Lancaster Street
Wayne, PA. 19087

Dr. Robert T. Hennessy
NAS - National Research Council
2101 Constitution Ave., N.W.
Washington, D.C. 20418

Other Organizations

Dr. M.G. Samet
Perceptronics, Inc.
6271 Variel Avenue
Woodland Hills, CA. 91364

Dr. Robert C. Williges
Human Factors Laboratory
Virginia Polytechnic
Institute and State
University
130 Wittemore Hall
Blacksburg, Va. 24061

Dr. Alphonse Chapanis
Department of Psychology
The Johns Hopkins University
Charles and 34th Streets
Baltimore, MD. 21218

Dr. Elizabeth Kruest
General Electric Company
Information Systems Programs
1755 Jefferson Davis Highway
Arlington, Va. 22202

Dr. James H. Howard, Jr.
Department of Psychology
Catholic University
Washington, D.C. 20064

Journal Supplement Abstract
Service
American Psychological Association
1200 17th Street, N.W.
Washington, D.C. 20036

Other Organizations

Dr. Edward R. Jones
Chief, Human Factors Engineering
McDonnell-Douglas Astronautics
Company
St. Louis Division
Box 516
St. Louis, MO. 63166

Dr. Babur M. Pulat
Department of Industrial
Engineering
North Carolina A & T State
University
Greensboro, N.C. 27411

Dr. Richard W. Pew
Information Sciences Division
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA. 02138

Dr. Douglas Towne
University of Southern
California
Behavioral Technology Laboratory
3716 S. Hope Street
Los Angeles, CA. 90007