AD-A043 028 NAVAL HOSPITAL JACKSONVILLE FLA* F/G 9/2
USE OF AN IQF-LIKE QUERY LANGUAGE BY NON-PROGRAMMERS.(U)
FEB 75 J D GOULD, R N ASCHER N00014-72-C-0419
UNCLASSIFIED RC-5279 NL

| OF |
AD
A043028

END
DATE
FILMED
9-77
DDC

AD A043028

IBM Research

USE OF AN IQF-LIKE QUERY LANGUAGE BY NON-PROGRAMMERS[1]

John D. Gould/Robert N. Ascher

February 20, 1975

RC 5279

6

B S.

DDC
RECEIVED
AUG 10 1977
D

This research was supported in part by the Engineering Psychology Programs, Office of Naval Research, Contract Number N00014-72-C-0419, Work Unit Number NR-197-020.

Reproduction in whole or in part is permitted for any purpose of the United States Government.

Approved for public release; distribution unlimited.

AD No. ____
DDC FILE COPY

Yorktown Heights, New York

San Jose, California

Zurich, Switzerland

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) RC5279	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) Use of an IQF-Like Query Language by Non-programmers,		5. TYPE OF REPORT & PERIOD COVERED (9) Interim Technical Report,
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) John D. Gould, Robert N. Ascher		8. CONTRACT OR GRANT NUMBER(s) (15) N00014-72-C-0419
9. PERFORMING ORGANIZATION NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (12) 35 p.
11. CONTROLLING OFFICE NAME AND ADDRESS (11) (20) International Business Machines, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598		12. REPORT DATE FEB [illegible] 1975
		13. NUMBER OF PAGES 30
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research, Code 455, Arlington, Virginia		15. SECURITY CLASS. (of this report)
		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

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

query language man-computer question-asking IQF programming human factors

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

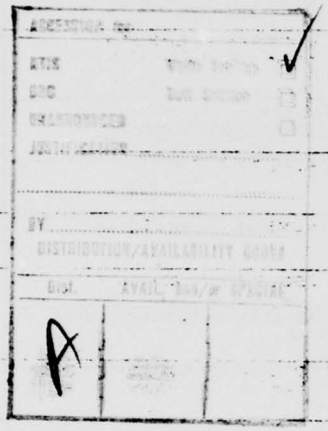
ABSTRACT: This exploratory experiment attempts to examine separately the formulation, planning, and coding of queries. College students and file clerks required about ten hours to learn a query language which was somewhat similar to IBM's IQF query language, but contained more function. They were then given 15 test problems that varied in complexity and how well they were expressed. Subjects were required to formulate, then to plan (writing each in their own words), and finally to code each problem. Results provide some suggestions

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

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

349 250

about which problem variables affected which "stages" in writing queries. For example, whether or not a problem was well expressed seemed to affect problem formulation time, but had no affect upon problem planning or problem coding times. Specific language constructions (additionso IQF), such as contextual referencing and a new method to handle limited disjunctive problems, were shown to be useful. The types of coding errors that subjects made were identified and discussed.



RC 5279
(#23163)
2/20/75
Psychology

USE OF AN IQF-LIKE QUERY LANGUAGE BY NON-PROGRAMMERS[1]

John D. Gould and Robert N. Ascher

Behavioral Sciences Group
Computer Sciences Department
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

Typed by Linda Callahan on MTST

ABSTRACT: This exploratory experiment attempts to examine separately the formulation, planning, and coding of queries. College students and file clerks required about ten hours to learn a query language which was somewhat similar to IBM's IQF query language, but contained more function. They were then given 15 test problems that varied in complexity and how well they were expressed. Subjects were required to formulate, then to plan (writing each in their own words), and finally to code each problem. Results provide some suggestions about which problem variables affected which "stages" in writing queries. For example, whether or not a problem was well expressed seemed to affect problem formulation time, but had no affect upon problem planning or problem coding times. Specific language constructions (additions to IQF), such as contextual referencing and a new method to handle limited disjunctive problems, were shown to be useful. The types of coding errors that subjects made were identified and discussed.

DDC
RECEIVED
AUG 30 1977
D

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication elsewhere and has been issued as a Research Report for early dissemination of its contents. As a courtesy to the intended publisher, it should not be widely distributed until after the date of outside publication.

Copies may be requested from:
IBM Thomas J. Watson Research Center
Post Office Box 218
Yorktown Heights, New York 10598

This is an exploratory experiment on how people with no experience using computers write queries. It was undertaken for two reasons: to collect some data on how people formulate, plan, and code queries; and to understand better how people learn and use a query language.

Query languages let people ask questions about data stored in a computer. For example, a person might want to inquire about census data, performance of particular stocks, personnel files, sales forecasting, geological findings, or course enrollments. The importance of query languages is expected to increase during the next decade because of the huge predicted expansion of data base applications (cf. Datamation, 1973).

Formulating appropriate queries involves problem-solving skills. Most experimental work on so-called problem-solving tasks has been limited to studying games, such as chess (deGroot, 1965), or mental puzzles such as cryptarithmetic (Newell and Simon, 1972), Tower of Hanoi (Newell and Simon, 1972), and *Hobbits and Orcs* (Thomas, 1974). These tasks generally require a person to formulate the problem, to plan a strategy or algorithm to solve the problem, and finally to try out, or implement, this plan. In the case of instructing a computer, this last component or "stage" requires a person to code the problem. The present experiment attempts to study these stages in a gross way by requiring subjects first to write a formulation of the problem, then to write a plan to solve the problem, and finally to code the problem in a formal query language (described below). Previous work on how people write computer programs (Boies and Gould, 1974; Miller 1974; Youngs, 1969; cf. also Weinberg, 1971) and debug computer programs (Gould and Drongowski, 1974; Gould, 1975) studied only this last stage. What we

operationally define as "formulation", "planning", and "coding" are not, of course, independent, mutually exclusive stages in problem-solving. Our attempt is to isolate some significant portion of each process as it relates to the use of computer languages. Of specific interest were how these stages are affected by the complexity of the problem and by how well the problem is initially defined.

IBM's Interactive Query Facility language (1972), called IQF, was used as a prototype language for the query language actually used here. IQF was designed to be relatively easy for non-programmers to learn and to be appropriate for simple queries. IQF differs in many important respects from the language used in this experiment, however. First, the experiment was not run interactively on a computer, but instead subjects wrote their queries on pieces of paper. Second, there were several specific language features not in IQF that we wanted to examine. Subjects were allowed to select and operate upon more than one subset of records, whereas in IQF they can only query one subset at a time. An assignment statement capability was introduced so that this aspect of procedure specification could be explored in a language that did not require a user to specify transfer of control. The use of contextual references, through a demonstrative pronoun, was allowed because people often use this construction in specifying procedures in their own words (Miller and Becker, 1975). A new method designed to aid people in coding limited disjunctive problems was invented and used. People have difficulty in procedurally specifying disjunctive ("or") concepts, even when they have just demonstrated they can follow the same disjunctive rule (Miller, 1975).

METHOD

Subjects. Seven female liberal arts college students and ten female middle-age file clerks (high school graduates of about 25 years ago) were paid to be subjects. Four file clerks did not complete the experiment. All subjects had no experience using computers.

Query Language. Besides the major differences mentioned above, several minor syntactic features of IQF were modified for ease of use, based upon informal pre-experimental testing. In using the language, a subject first

See page 24 for Table 1

selected a file, then selected one or more subsets of records within the file, and finally operated upon these records. In Table 1, line (1) selects the file. Lines (2) to (4) and lines (10) to (12) select the subsets of records of interest. The remainder of the lines operate upon the records.

The language consists of commands, attributes, values, operators, and variables. For illustration, the command portion of each instruction in Table 1 is capitalized. The FROM_____FILE command selects one file. The FOR commands and the AND FOR commands select records of interest. The LET_____BE CALLED_____ command is the assignment statement. A variable could be used in the first blank of the assignment statement, as in line (6); or a contextual reference referring to the result of the previous line could be used, as in line (14). Variables are intermediate data structures that a subject has created. More than one variable, separated by operators, could also be used in the first blank, as in line (17). The ALSO command,

as in line (9), separated different sets of records. The COUNT and TOTAL commands do what their names imply.

Besides the commands shown in Table 1, there was a COUNT BY_____ command, a TOTAL BY_____ command, and a SORT BY_____ command. These involve the concept of operating on an entire set of records at once, e.g., Sort By Department.

The operators used in this study were "or less", "or more", "+", "-", "*", and "/".

Training. Based upon pre-experimental work and experiences of colleagues teaching laboratory programming languages to non-programmers, a 10-page training program was developed. Each subject wrote about ten practice queri[illegible] An experimenter was always in the room, provided feedback on the acc[illegible]y of her queries, helped her, and answered all questions. (Consequently, the exact training that each subject received varied somewhat.) Following this, a subject coded six test problems. If she was correct on four of them, she proceded to the formal experiment. Otherwise, some re-training occurred and then she did the six problems over until she was correct on four of them.

Problems. The 15 test problems were worded in various ways, but they all required subjects to write queries that would select from a personnel file sets of records, or employees, having specified characteristics. For each set, subjects sometimes had simply to list the records, sometimes count the records (e.g., as in line (5) of Table 1), sometimes total, for example, the number of dependents of the selected employees, and sometimes compute an average, e.g., as in lines (5), (6), (7), (8), and (17) in

Table 1. The problems varied systematically in two ways. Each required the selection of one, two, or three sets of records. The problem in Table 1 requires two sets. Secondly, each set had 1, 3, or 5 modifiers or qualifiers. The problem in Table 1 has 3 modifiers for each set. For example, lines (10) to (12) select engineers over 50 years of age working in St. Louis.

There were nine so-called "standard" problems, one for each combination of sets and modifiers. These standard problems served as control problems for the other six problems. Of these six, one was similar to the 2-set, 5-modifiers-per-set standard problem and was used to check the reliability of results for problems of different wording. Three others were so-called "temporary data" problems. Two temporary data problems required subjects to create and save temporary data and to use these results later in their query to compute averages (i.e., average salaries or average numbers of dependents). The third required, in addition, that subjects compare two averages that they had calculated earlier in their query. The remaining two problems were so-called "poorly expressed" because one contained irrelevancies and the other was vague.

Procedure. Each subject was given the fifteen problems in a random order, usually in 2 or 3 sessions. After reading a problem, a subject was required to write a sentence beginning with "I want to know...", and ending with the information she needed to solve the problem. As explained to her, this was to be a measure of problem formulation. Second, she was required to write a plan, which was to consist of a series of steps to solve that problem. She again used her own words to write this plan, and she used her own judgment about how detailed to make the plan, an outcome of some

interest in itself. Finally, she was required to code the problem, numbering each line of code in the order she wrote it. No erasing was allowed. This provided a complete record of all insertions and line changes.

Subjects had a chart in front of them that contained thirteen columns, each with a different heading or attribute, e.g., name, age, sex, year hired. This chart represented the personnel file used for all problems. Under each attribute was a list of all permissible words, or values. Subjects were required to use these exact attributes and values, including their exact spellings, in selecting records; otherwise, their code was considered erroneous. Table 1 illustrates the use of attributes (underlined words) and values (words beginning with lower case letters). (In practice, however, subjects were not required to underline, and the case of letters was irrelevant.)

Subjects wrote on paper placed on a large brass plate, which was attached to an IBM System/7 computer[2]. The writing pencil was also connected to the computer, and when a subject touched the brass plate with it, she completed a circuit that controlled a clock in the computer. A subject touched the plate at the start of problem formulation and again when she completed writing her formulation. This gave a measure of formulation time. In problem planning, a subject touched the brass plate after writing each planning step, which provided a time for each step. Similarly, in problem coding, a subject touched the plate after writing each line of code. The results reported are based upon the first attempt at writing each query.

Error Analysis. Following data collection, accuracy of code was analyzed manually in several ways by one experimenter going through the data

several times. Each successive manual pass led to slightly different results and sometimes required arbitrary decisions. Consequently, these error analyses are approximate only.

One error classification characterized errors in relation to the commands in which they occurred. If a necessary command was omitted, and not replaced by another (wrongly chosen) command, an "omission" error for that missing command was recorded. If a wrongly chosen command was present, it was classified as "inappropriate" (and no omission error was recorded). If a command was appropriate, but contained an error, this error was classified in terms of the erroneous element in that command, e.g., heading (attribute), value, operator, or the command itself.

A second, independent error analysis attempted to characterize each error as clerical, syntactic, or conceptual. This proved impossible to do.

A third error analysis, related to the second, attempted to classify the general accuracy characteristics of each entire query, rather than individual errors within it. If the query was "correct", this was recorded. If it was incorrect, the errors were examined to see if the query could best be characterized as containing "clerical-syntactic" errors. A clerical error included misspellings, mis-transcription of a heading or value from the data structure chart, or obvious typographical mistakes. A syntactic error was a violation of the rules of the language relating to an <u>individual</u> statement. If the errors could not be best described as clerical-syntactic, they were then examined to see if the query could be described as belonging to a "conceptual/language" error category. An incorrect query was thus classified if it was judged that the errors, which would otherwise be

judged as "conceptual", were the result of a systematic misunderstanding of the rules of the language. Finally, an incorrect query was classified as due to a "conceptual" error. It was thus classified if the errors primarily reflected misunderstandings of the problem rather than of the language, e.g., if the query was syntactically correct but caused an incorrect result. Subjects' formulations and plans were sometimes valuable in making judgments about these last two categories.

A fourth error analysis attempted to determine which stage (formulation, plan, code) was initially responsible for a coding error. Although we were hopeful of learning where coding errors first arise, this proved unsuccessful, primarily because of ambiguities in subjects' wordings of their formulations and plans, and because of inter-subject variations in the detail of their formulations and plans. For example, if a formulation or plan was detailed it was easier to attribute a coding error to it than if it were ambiguous.

Fifth, some general characteristics of the formulations and plans were noted.

RESULTS

Training. Mean training time was longer than we had initially expected, being 8.4 hours for college students (range = 4 to 13.5 hours) and 14.7 hours for the file clerks (range = 7.5 to 24.5 hours). Among the concepts that subjects required much time to learn were (a) how to parse an instruction, e.g., how to parse line (4) of Table 1 into command, attribute, and value; (b) how to use data structure, i.e., attributes and values; (c) the necessity to pay attention to details, such as exact spellings; (d) mapping the

problem statement into the query language by determining, for example, which attributes were necessary and which ones were not necessary to mention. Subjects' training times correlated positively with their subsequent coding times ($r = .66$; $p < .02$), but were not related to their formulation times, their planning times, or their accuracy on test problems.

General Formulation Results. Subjects' formulations were typically re-statements of the problem, and contained about the same number of words (about 10-60) as did the problem statement. Mean formulation time per problem was about 1.5 minutes (which is twice as long as these subjects required to write the same amount of memorized material).

General Planning Results. In general, subjects' plans contained details relevant to using the data structure, and they resembled, both syntactically and semantically, the query language itself. Mean planning time per problem was 3.3 minutes, again roughly twice the subjects' writing rates for meaningful material. Subjects had nearly (78%) as many statements or steps in their plans as in their code. Subjects varied considerably in the style of their plans. Some subjects were relatively general and specified several characteristics in a step, e.g., all the modifiers in a set of records. This would often be followed in the next step by the action to be applied to that set, e.g., count, total, or average. Other subjects had simpler steps, specifying, for example, only one modifier in a step. It appeared to us that much of the thinking and problem-solving occurred in the planning stage. For example, while coding a problem some subjects put one finger on one step in their plan, wrote the code for that step, moved their finger to the next step in their plan, coded it, and continued this for each successive step in the

plan. Some subjects used transitive verbs like "obtain", "make", "go to", "get", "check", "find" in their plans, whereas other subjects used personal pronoun constructions like "I need", "we have". Plans were rarely modified, and insertions were also rare.

General Coding Results. Once a subject began coding, she usually wrote 10 to 20 successive lines of code, rarely inserting or deleting a line. Presumably, the fact that she had first written a plan contributed to this surprisingly organized effort. Mean coding time was 4 minutes per problem, and subjects wrote 37% of their queries correctly.

Group Differences. No difference could be found in the accuracy or kinds of errors made by each group. College subjects were faster than file clerks in formulating, in planning, and in coding at least 12 of the 15 problems. All three of these differences were statistically significant by a one-tailed binomial test (all $p < .01$), but were not significant when tested with analyses of variance.

Standard Problems. Figure 1 shows that the total time to formulate, to plan, and to code problems increased about linearly with the number of required sets in a problem ($F(2,24)=77.11$; $p < .001$) and with the number of modifiers per set in a problem ($F(2,24)=67.79$; $p < .001$). Figure 2 shows

See page 27 for Figure 1
See page 28 for Figure 2

that, when considered separately, formulation times, planning times, and coding times each increased about linearly with the number of required sets in a problem ($F(2,24)=90.36$, 25.37, and 83.41, respectively; all $p < .001$)

and with the number of modifiers per set in a problem ($F(2,24)=83.96$, 22.94, 71.48, respectively, all $p < .001$). Both planning time per step and coding time per statement were generally about 20 seconds, regardless of problem complexity. The form of the Set x Modifier interaction is consistent with this (for formulation time, $F(4,48)=17.73$; $p < .001$, for plan time, $F(4,48)=3.40$; $p < .05$, and for code time $F(4,48)=14.80$; $p < .001$).

There were very high correlations between time to formulate a problem and time to plan a problem ($r = .97$; $p < .001$), between time to formulate a problem and time to code a problem ($r = .98$; $p < .001$), and between time to plan a problem and time to code a problem ($r = .98$; $p < .001$).

More complex problems required more writing in each stage. This was corrected by subtracting, from each of the 195 time scores at each stage, the product of the rate with which subjects wrote memorized material (.5 words per second) times an estimate of the number of words written at each stage (based upon sampling about 20% of the 195 queries). Figure 3 shows that formulation times, planning times, and coding times still all increased with

See page 29 for Figure 3

the number of sets in a problem ($F(2,24)=39.01$, 4.55, and 32.64, respectively; all $p < .05$) and with the number of modifiers per set in a problem ($F(2,24)=33.66$, 5.08, and 29.63, respectively; all $p < .05$). These remaining times can be thought of as "thinking" times, and this result indicates that additional thinking was required as the problems became more complex.

See page 25 for Table 2

Subjects wrote fewer correct queries for problems with more sets and also for problems with more modifiers per set (line a, Table 2). The simplest problem was coded correctly by 10 of 13 subjects, whereas the most complex problem was coded correctly by only one subject. With the exception of the center cell of Table 2, the probability of an error in a line of code was .1 to .2, regardless of problem complexity.

Performance on the non-standard 2-set 5-modifiers-per-set problem was about the same as performance on the comparable one that differed in wording and the exact sets and modifiers that were required.

Temporary Data Problems. Requiring subjects to create, save, and later use temporary variables in calculating averages led to longer times to complete these three problems than to complete three standard problems with the same number of sets and modifiers, (means = 432 seconds for standard problems versus 761 seconds for temporary data problems, $F(1,12)=28.19$; $p < .001$). This difference of 329 seconds was due to a difference of 139 seconds in plan times and 186 seconds in code times. Some of this additional plan time and code time probably reflected additional time for thinking, whereas some of it was due to writing more statements. The left panel in Figure 4 shows that, when corrected for the amount of writing involved, the times actually spent thinking in both planning and coding were significantly longer on the temporary data problems than on the comparable standard problems ($F(1,12)=19.05$ and 4.82, respectively, $p < .01$ and $p < .05$, respectively). Formulation times were the same for the two types of problems, however.

See page 30 for Figure 4

An average of 4.3 queries was correctly written on the temporary data problems, compared with an average of 5.3 on the comparable standard problems. Conceptual errors accounted for seven of the incorrectly written temporary data queries but for only one of the incorrectly written "control" queries.

Poorly Expressed Problems. Total time to formulate, plan, and code the two poorly expressed problems exceeded that for the comparable standard problems (means = 533 and 430 seconds, respectively; $F(1,12) = 4.46$; $p < .10$). Of the difference of 103 seconds, 52 seconds was due to a difference in formulation times, 20.5 seconds to a difference in plan times, and 30.5 seconds to a difference in code times. The right panel in Figure 4 shows that, after correction for amount of writing, formulation times were still greater on poorly expressed problems than on the comparable standard problems ($F(1,12) = 27.86$; $p < .001$), but planning times and coding times were the same on the two types of problems (both $F < 1.0$). Thus, a problem that was poorly expressed affected formulation time only, whereas the requirement to create temporary data variables affected planning time and coding time.

Seven queries were written correctly for the poorly expressed problem that contained irrelevant information. Subjects' written formulations showed that these irrelevancies were usually stripped away during the formulation stage. No queries were written correctly on the other poorly expressed problem, which was a tricky one requiring subjects to seek information that would settle an argument between two men. All 13 subjects selected information that evaluated only one of the men's views.

Coding Error Analyses. Of the 195 queries written, 127 were incorrect. Ten of these were syntactically correct but would not get all the data

required to solve the problem. These 10 and 18 others, including a few that indicated confusion in calculating an average, were judged to be due to conceptual errors. Sixty-four erroneous queries were judged to be due to conceptual/language errors and 35 to syntactic-clerical errors.

Table 3 indicates the commands in which coding errors occurred in the 115 queries that contained such errors (excluding two that were simply rife with errors). Subjects made about the same number of errors in using record-selection commands (182 errors; probability of an error = .12) as in using record-operation commands (156 errors; probability of an error = .17).

See page 26 for Table 3

The main reason for errors in record selection was that subjects used wrong values (102 of the 182 errors). Thirty-five of these 102 errors were due to difficulties subjects had in (a) converting problem statements such as "over 50 years old" or "over $8000" into the appropriate values and operators (i.e., into "51 or more" or "8001 or more"); or (b) in transforming a problem statement such as "have worked for us for more than 5 years" into the appropriate attribute and value (i.e., "year hired 1969"). Another 30 of these 102 errors were due to the way subjects were required to use the "Highest Degree" attribute. This was the only attribute with which they were not allowed to use the "or more" or "or less" operators. Instead of merely writing "HS or more" if they wanted people with a high school degree, they had to write out all higher values as well, e.g., HS, JC, BS, MS, PhD. This approach caused many errors, and should not be used in query systems. Another 20 of the 102 errors were clerical errors such as misspellings, use

of synonyms, and inappropriate insertion of dollar signs.

Besides these 102 value errors, another main type of data selection error was the apparently inadvertent omission of a modifier, or FOR or AND FOR command (24 errors). This almost always occurred in problems involving many modifiers, and would of course be a dangerous error in real-life. Another 19 errors were due to subjects either using the wrong operator (e.g., 'or more' instead of 'or less') or in omitting a required operator.

The 156 errors in record operation commands were spread across all of these commands, as shown in Table 3. One-third (52 of 156) of the errors were due to inappropriate uses of the commands, a type of error that did not occur much during record selection. Most of these 52 errors were due to confusing the COUNT command (which counts items, as the number of bills in one's wallet) and the TOTAL command (which, for example, produces the total amount of money in one's wallet), misplacing a statement in a procedural sequence, or using superfluous statements. Most (40 to 52) occurred in programs classified as containing conceptual/language errors.

Another forty-eight of the record operation errors were due to erroneous use of attributes, or headings. Mainly subjects used incorrect headings, omitted headings, or misspelled headings. Another 37 record operation errors were due to subjects omitting required commands (one subject committed 15 of these).

Additional Language Features. Several specific language features, additions to the real IQF, were examined. First, the ALSO command for handling the limited disjunctions required for selecting multiple sets of records was successful. It was used 165 times and only one error occurred.

Second, subjects rarely used the COUNT BY and TOTAL BY commands, which indicates that they preferred to think about and manipulate data across only one dimension at a time. Of course, this "preference" result is dependent upon the stress in training given to these commands, which was not much and probably variable. Third, the assignment statement (LET command) was not especially difficult for subjects to use (cf. Table 3). Fourth, when a variable in the LET command referred to the result of a calculation in the immediately preceding statement, subjects were allowed a choice of using an "absolute" reference (e.g., as in line (6) of Table 1) or a "contextual" reference (i.e. through the demonstrative pronoun, "this", as in line (8) of Table 1). Nine subjects always used contextual referencing when possible, three subjects never did and one subject sometimes did. This was essentially a matter of preference, as no difference in accuracy occurred between these two constructions.

Individual Differences. Subjects' mean times varied, from the briefest to the longest, by a factor of 6 in training (4 to 24.5 hours), a factor of 2 in the formulation stage (60 to 133 sec. per problem), a factor of 5 in the planning stage (69 to 334 sec. per problem), and a factor of 3 in the coding stage (138 to 337 sec. per problem). The most accurate subject wrote 9 of 13 queries correctly, whereas the least accurate subject wrote only one query correctly. With the exception of value errors in the AND FOR command, individual subjects differed considerably in the errors they made. Subjects who were fast at problem formulation were also fast at problem planning (r=.87; p < .01) and at problem coding (r=.62; p < .05). Thus, formulation time is a moderately good predictor of coding time. The

correlation between subjects' plan times and code times was .56 ($p < .05$). There was no relation between subjects' coding times and accuracies ($r=-.32$; $p > .10$). Incidently, although subjects were generally encouraged to write a single query about a problem, they could have broken up a complex problem and written a series of simple queries about it. They never did this though, which may merely reflect a powerful experimental demand to write a single query.

Practice. The number of correctly written queries increased from an average of 2.7 (out of 13) on the first three problems to an average of 6 on the last three problems; the number of errors decreased from about 3 per query on the first three problems to about 1 per query on the last 3 problems.

DISCUSSION AND CONCLUSIONS

It is perhaps again appropriate to point out that this is an exploratory study. Three methodological aspects should be mentioned in this regard. The first has to do with lack of control over subjects during training and testing. During testing, an experimenter corrected each query immediately after each subject wrote it and discussed any errors and answered any questions before she went on to the next test problem. While this was valuable in understanding subjects' difficulties and was a sensible pedagogical tool, it did result in "non-standard" test conditions and probably led to the practice effect just mentioned.

Second, the wording of each standard problem differed in ways other than those defined by the two independent variables studied. This lack of control was intentional so that a greater domain of potential difficulties

in using query languages might be identified. Some problems required subjects to transform the exact wording of a problem statement into different words required by the data base and some problems required subjects to select records based upon people over a certain age or with certain types of educational degrees. These requirements, which led to more errors than use of other modifiers, were more often associated with the more complex problems than with simpler problems. Thus, the apparent systematic effects of the number of sets and the number of modifiers per set on formulating, planning, and coding queries cannot be strictly accounted for by these two variables.

Third, this was a crude, knowingly over-simplified, beginning attempt to study separately the processes of formulating, planning, and coding a problem. On the positive side, a number of interesting results were found about the locus of query and programming variables on people's performance. The number of sets and the number of modifiers per set in a problem led to approximately linear increases in formulation times, in planning times, and in coding times. (Increases in problem complexity required subjects to do more of the same thing rather than do different things, however.) Poorly formulated problems affected only the formulation stage, whereas the requirement to create temporary data variables affected the planning and coding stages. This agrees with intuition. About twice as much thinking time was needed for planning and for coding problems that required creating intermediate data structures as for planning and coding problems that did not. Subjects stripped away the irrelevancies in a poorly stated problem when they formulated it, although their formulations were not always correct. Formulation times were a moderately good predictor of coding times.

On the negative side, we could not determine which stage was initially responsible for a coding error. This problem of identifying the reasons for conceptual coding errors is important, and may be studied more successfully by assessing independently a person's understanding of the problem, of the language, and the mapping of the two. Most significant is the unknown relation of the actual cognitive processes involved in formulating, planning, and coding to the three stages so-named here. Even though subjects rarely modified an earlier stage while working in a later stage, formulating, planning, and coding are not ordinarily independent processes when one works on more complex problems.

Requiring subjects to formulate and plan a problem prior to coding it may have created an artificial situation, but arguments can be made that this either increased overall time and errors or that it decreased them. Determining the value of a formal planning exercise prior to coding a problem is an important future research question. Incidently, opinions of professional programmers differ on the value of their formally planning (e.g., with a flowchart) a program prior to coding it. Because of its simplicity our methodology can be useful in the future. For example, one might compare how different query or programming languages affect different stages. The ratios of formulation time to coding time for various languages would be an interesting and useful statistic, and the contents of the formulations would indicate the degree and manner which individual languages affect how people think about a problem.

Compared with college students, file clerks were about 25 years older, had less education, had no contact with formal education for about 25 years, probably had lower IQ's, and seemed to find it difficult to consider

imaginary situations. These several differences and the fact that the groups were so small make it impossible to identify the bases of why the file clerks required more time in training and on the test problems. The fact that four file clerks failed to finish training suggests additional difficulties if this population should be required to use a language such as this.

This study was not intended to be an evaluation of the IQF language itself. Rather it was an exploratory attempt to identify the difficulties non-programmers have in writing queries, and IQF served as a prototype query language. Training times would have been considerably shorter if only the IQF concepts were taught. Subjects, both college students and file clerks, coded about half their queries correctly on the problems that IQF itself is designed for (problems requiring only one set of records), which is somewhat less than the accuracies recently found for newer and more powerful query languages (Reisner, 1974; Thomas and Gould, 1974).

The general clerical errors of misspellings, the apparent inadvertent omission of required attributes or headings (which will produce an incorrect set of records), confusion of operators like TOTAL and COUNT, and difficulties with theta operators like "or more" and "or less" were all found by Thomas and Gould (1974) to occur with Zloof's (1974) Query By Example language also. Some of these errors could be reduced or eliminated by an interactive system providing various types of feedback. For example, the dangerous inadvertent omission of an intended modifer (24 errors) might be caught if the system re-stated the user's query, perhaps in English, prior to executing it. The accuracy of queries involving logical constructions that are difficult for casual users could perhaps be improved through feedback that provides

examples to illustrate the subtle (to the user) differences between the answer to his query and answers to related ones that he might potentially have asked.

The idea for examining contextual referencing came from data that showed people frequently use this when generating procedures in their own words (Miller and Becker, 1974). The present experiment suggests that subjects might prefer to use contextual referencing with formal languages also, since most subjects used, when possible, a contextual reference to a previous calculation rather than an absolute reference. This is merely suggestive, as the experimental demand for using either one was not carefully controlled.

Some problems required subjects to select more than one set of records or specify more than one value for a particular attribute. Two techniques were used to help subjects do this, and both of them involved the elimination of the word "or" from the query language. First, when more than one value was required for a particular attribute, subjects wrote these values, separated by commas, and did not use the word "or". Second, the ALSO command was invented for separating the code for separate sets of records[3]. This worked well, as the ALSO command was used 165 times, and only one error resulted. (The English problem statements of standard and temporary data problems were clear, usually because of punctuation, in demarcating the separate sets of records required.) Thus, because of the ambiguous way "or" and "and" are used in English, with the semantics of the situation often over-riding conventional rules of logic, elimination of the word "or" in formal query languages may be a good idea.

REFERENCES

Boies, S. J. and Gould, J. D., Syntactic Errors in Computer Programming. Human Factors, 1974, 16, 253-257.

Datamation, November, 1973, 126.

deGroot, A. D. Thought and Choice in Chess. The Hauge: Mouton, 1965.

Gould, J. D. Some Psychological Evidence on How People Debug Computer Programs, International Journal of Man-Machine Studies, 1975 (in press).

Gould, J. D. and Drongowski, P. An Exploratory Study of Computer Program Debugging. Human Factors, 1974, 16, 258-276.

IBM Interactive Query Facility, Terminal User's Reference Guide, GH20-1223, 1972.

Miller, L. A. Programming By Non-Programmers. International Journal of Man-Machine Studies, 1974, 6, 237-260.

Miller, L. A., 1975, in preparation.

Miller, L. A. and Becker, C. A. Programming in Natural English. IBM Research Report RC-5137, 1974.

Newell, A. and Simon, H. A. Human Problem Solving. Englewood Cliffs, New Jersey: Prentice-Hall, 1972.

Reisner, P. Human Factors Evaluation of Two Data Base Query Languages: Square and Sequel. IBM Research Report, RJ-1478, 1974.

Thomas, J. C. An Analysis of Behavior in the Hobbits-Orcs Problem. Cognitive Psychology, 1974, 6, 257-269.

Thomas, J. C. and Gould, J. D. A Psychological Study of Query By Example. IBM Research Report, RC-5124, 1974.

Weinberg, G. M. The Psychology of Computer Programming, New York: Van Nostrand Reinhold, 1971.

Youngs, E. A. Error-proneness in Programming. Unpublished Ph.D. thesis, University of North Carolina, 1969.

Zloof, M. Query by Example. IBM Research Report, RC-4917, 1974.

FOOTNOTES

1. We thank John Thomas for many helpful suggestions, and him and K. William Sholz for helpful comments on an earlier version of this manuscript.
2. We thank Stephen Boies for helping us program this.
3. This was done jointly with Clayton Lewis.

TABLE 1

Program for the problem "List the average salary of employees who were in Chicago, have at least a Bachelor's degree, and are married; and the average salary of engineers over 50 years of age working in St. Louis". For illustrative purposes, commands are in all upper case letters, attributes are underlined, values are in all lower case letters, and variables begin with an upper-case character.

```
(1)  FROM Personnel FILE
(2)  FOR chicago Location
(3)  AND FOR Highest Degree bs, ms. ph.d
(4)  AND FOR married Marital Status
(5)  COUNT
(6)  LET Count BE CALLED Count 1
(7)  TOTAL Salary
(8)  LET This BE CALLED Total 1
(9)  ALSO
(10) FOR Age 51 or more
(11) AND FOR engineer Jobtype
(12) AND FOR st. louis Location
(13) COUNT
(14) LET This BE CALLED Count 2
(15) TOTAL Salary
(16) LET This BE CALLED Total 2
(17) LET Total 1/ Count 1 BE CALLED Avg 1
(18) LET Total 2/ Count 2 BE CALLED Avg 2
(19) LIST Avg 1, Avg 2
```

TABLE 2

Performance of the 13 Subjects on the Nine Standard Problems

Number of Modifiers Per Set		Number of Sets 1	2	3	Means
1	(a)	10	9	6	8.7
	(b)	1	0	0	.3
	(c)	2	4	2	2.7
	(d)	0	0	5	1.7
	(e)	6	12	12	10.0
3	(a)	6	1	5	4.0
	(b)	1	3	1	1.7
	(c)	5	7	4	5.3
	(d)	1	2	3	2.0
	(e)	14	48	22	28.0
5	(a)	3	4	1	2.7
	(b)	0	0	0	0.0
	(c)	6	7	8	7.0
	(d)	4	2	4	3.3
	(e)	13	22	47	27.3
	(a)	6.3	4.7	4.0	
	(b)	0.7	1.0	.3	
	(c)	4.3	6.0	4.7	
	(d)	1.7	1.3	4.0	
	(e)	11.0	27.3	27.0	

(a) = Number of correct programs
(b) = Number of "conceptual error" programs
(c) = Number of "conceptual/language error" programs
(d) = Number of "clerical/syntactic error" programs
(e) = Total number of errors

TABLE 3

Programming/Query Language Statements in Which Errors Occurred

	Number of Errors	Total Errors	Actual Occurrences
Non-Existent Command		3	3
File Selection		1	195
Data Selection		182	
FOR Command	(27)		365
Inappropriately used	5		
Omitted	1		
Heading (Attribute)	9		
Value	12		
AND FOR Command	(154)		754
Inappropriately used	12		
Omitted	23		
Command	2		
Heading (Attribute)	8		
Value	90		
Operator	79		
ALSO Command	(1)		165
Omitted	1		
Data Operation		156	264
Count Command	(26)		
Inappropriately used	16		
Omitted	6		
Command	4		
Count By Command	(4)		17
Inappropriately used	2		
Command	1		
Heading (Attribute)	1		
Total Command	(24)		94
Inappropriately used	10		
Heading (Attribute)	14		
Total By Command	(0)		0
List Command	(44)		207
Inappropriately used	12		
Omitted	4		
Command	5		
Heading (Attribute)	23		
Let Command	(48)		363
Inappropriately used	9		
Omitted	20		
Command	3		
Heading (Attribute)	10		
Operator	6		
Sort By Command	(10)		10
Inappropriately used	3		
Omitted	7		
		342	2437

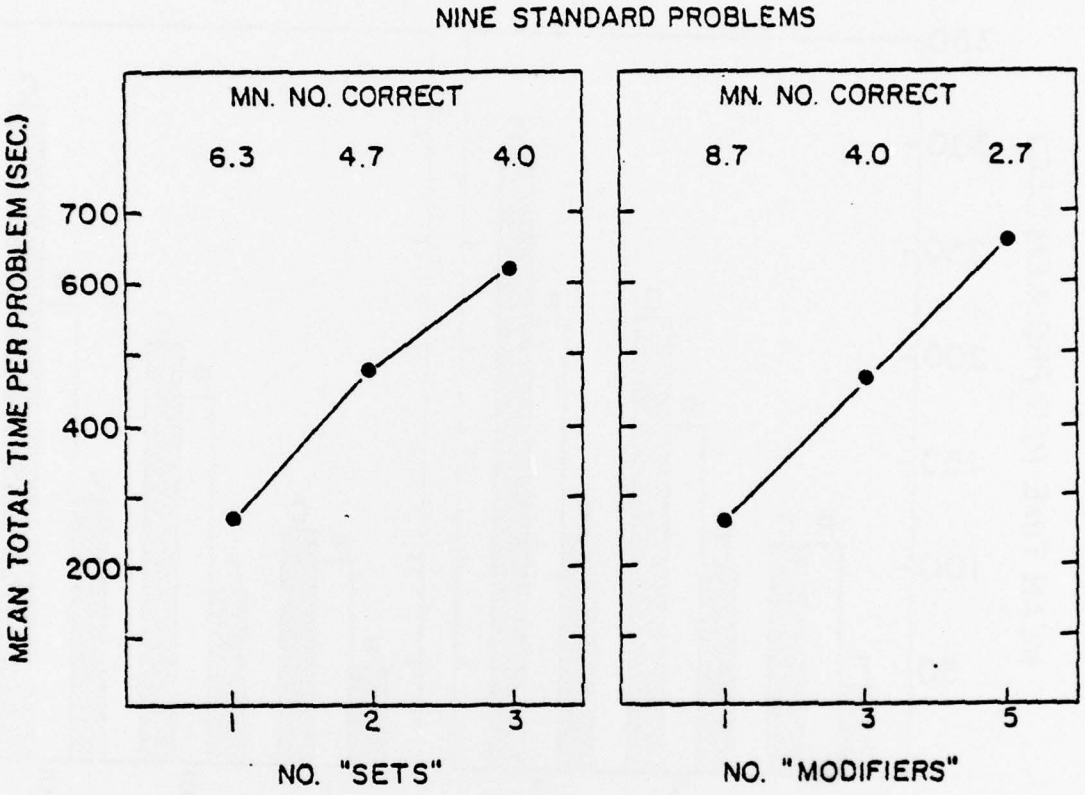


Figure 1. Average time for subjects to complete each of the nine standard problems. Accuracy is based upon the mean number of the 13 subjects that coded each problem correctly.

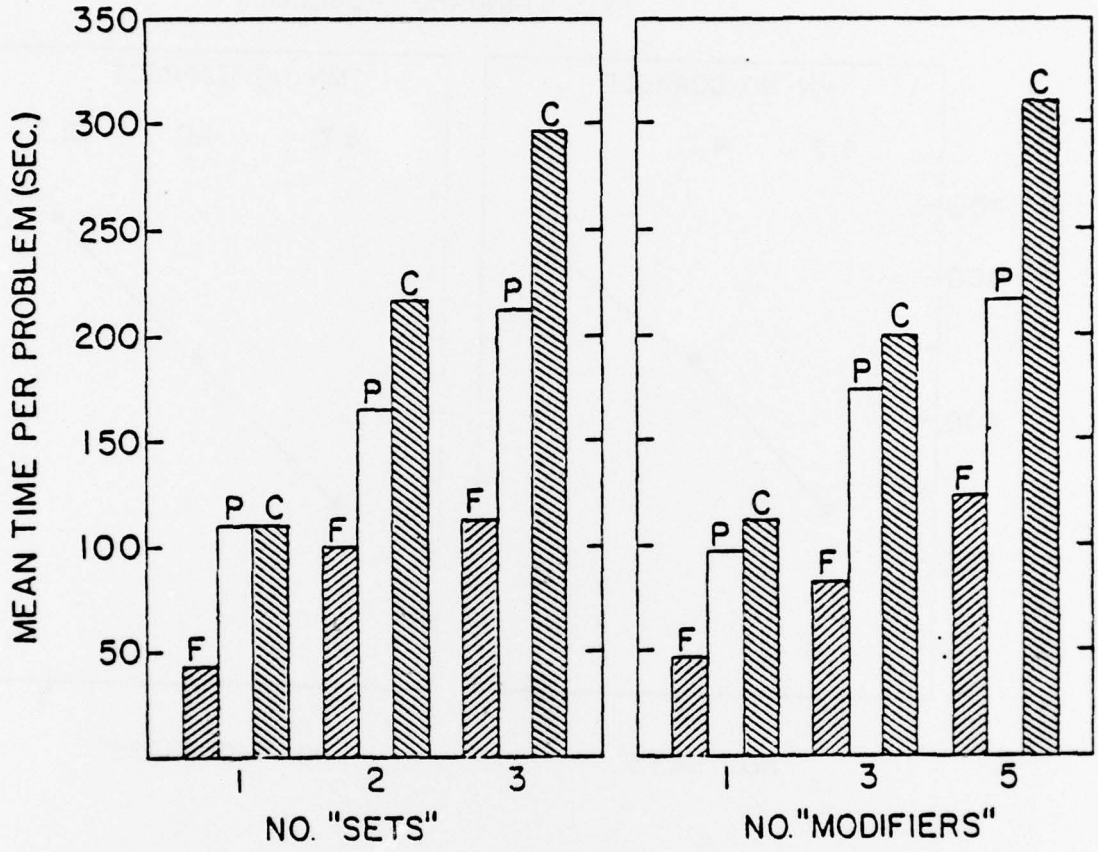


Figure 2. Average time for subjects to formulate, to plan, and to code the nine standard problems.

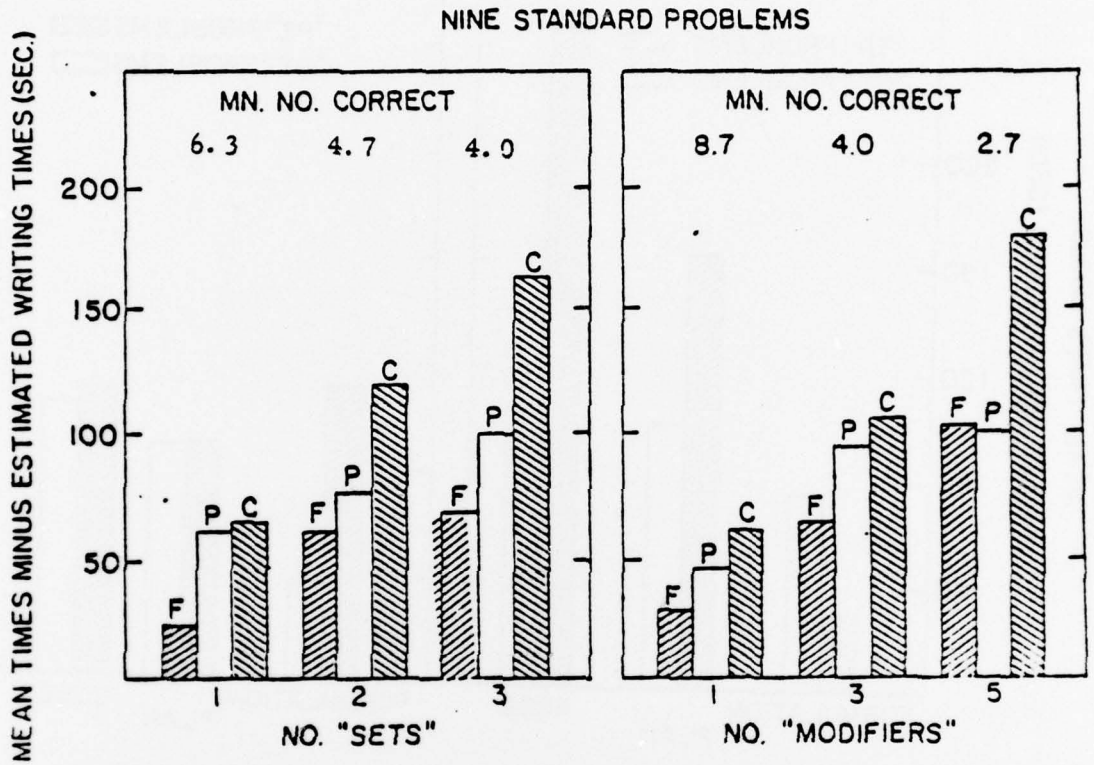


Figure 3. Average time, when corrected for the amount of writing involved, for subjects to formulate, to plan, and to code the nine standard problems.

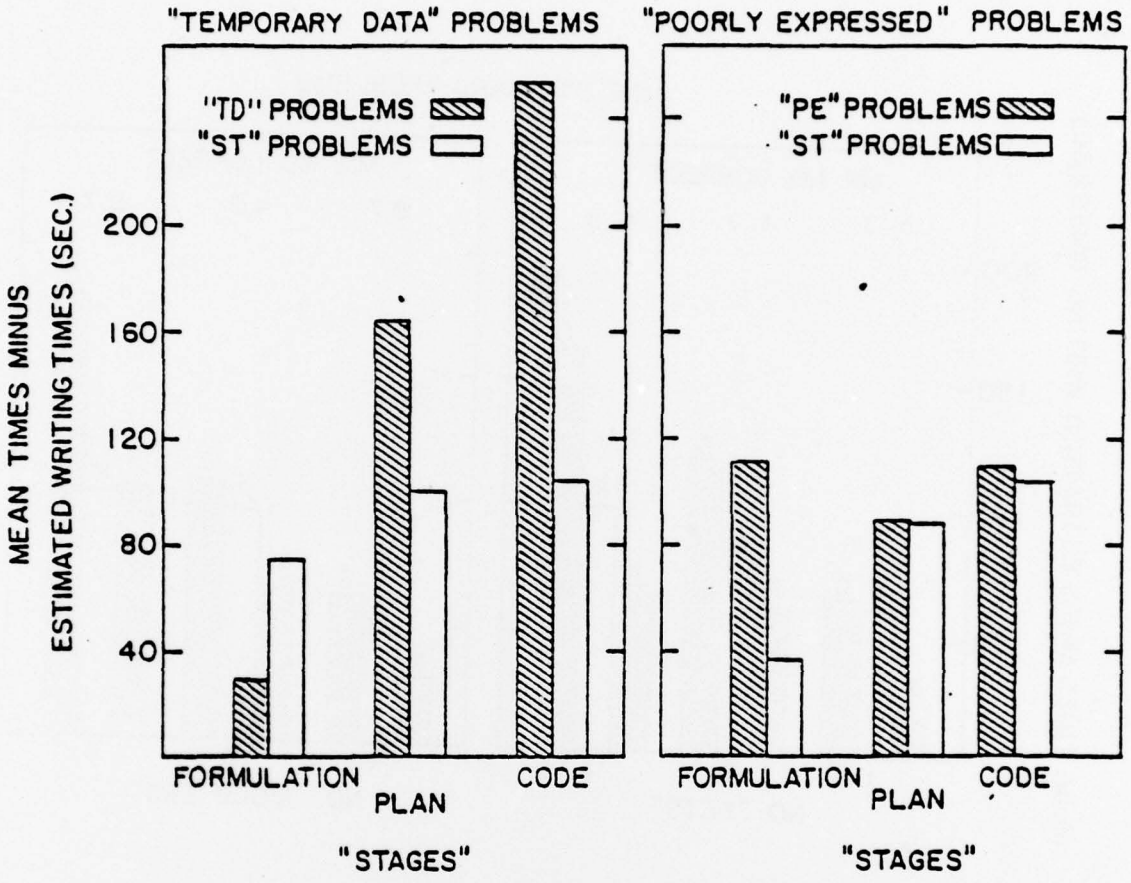


Figure 4. Average time, when corrected for the amount of writing, for subjects to formulate, to plan, and to code three temporary data problems and comparable standard problems (left panel) and the two poorly expressed problems and comparable standard problems (right panel).

TECHNICAL REPORTS DISTRIBUTION LIST

CODE 455

Director, Engineering Psychology (5 cys)
Programs, Code 455
Office of Naval Research
800 North Quincy Street
Arlington, Virginia 22217

Defense Documentation Center (12 cys)
Cameron Station
Alexandria, Virginia 22314

Director, ONR Branch Office
ATTN: Dr. J. Lester
495 Summer Street
Boston, Massachusetts 02210

Director, ONR Branch Office
ATTN: Dr. M. Bertin
536 S. Clark Street
Chicago, Illinois 60605

Director, ONR Branch Office
ATTN: Dr. E. Gloye
1030 East Green Street
Pasadena, California 91106

Director, ONR Branch Office
ATTN: Mr. R. Lawson
1030 East Green Street
Pasadena, California 91106

Dir., Naval Research Laboratory (6 cys)
Technical Information Division
Code 2027
Washington, D. C. 20375

Dir., Naval Research Laboratory (6 cys)
ATTN: Library, Code 2029 (ONRL)
Washington, D. C. 20375

Mr. John Hill
Naval Research Laboratory
Code 5634
Washington, D. C. 20375

Office of Naval Research
Information Systems Program, Code 437
800 North Quincy Street
Arlington, Virginia 22217

Office of Naval Research
Operations Research Program, Code 434
800 North Quincy Street
Arlington, Virginia 22217

Office of Naval Research
Naval Analysis Programs, Code 431
800 North Quincy Street
Arlington, Virginia 22217

Office of Naval Research
ATTN: Dr. K. T. Wallenius, Code 436
800 North Quincy Street
Arlington, Virginia 22217

Office of the Chief of Naval
Operations, Op-987E
Department of the Navy
Washington, D. C. 20350

CDR H. J. Connery
Office of the Chief of Naval Operations,
Op-987M4
Department of the Navy
Washington, D. C. 20350

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

Dr. Heber G. Moore
Hqs. Naval Material Command
Code 0331
Department of the Navy
Washington, D. C 20360

Mr. Arnold Rubinstein
Naval Material Command, NAVMAT 03424
Department of the Navy
Washington, D. C. 20360

Commander, Naval Electronics
Systems Command
Command and Control Div., Code 530
Washington, D. C. 20360

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

Bureau of Medicine and Surgery
Human Effectiveness Branch, Code 713
Department of the Navy
Washington, D. C. 20372

CDR Robert Wherry
Human Factors Engineering Branch
Crew Systems Department
Naval Air Development Center
Johnsville
Warminster, Pennsylvania 18974

LCDR Robert Kennedy
Human Factors Engineering Br., Code 5342
U.S. Naval Missile Center
Point Mugu, California 93042

Lt. Col., Henry L. Taylor, USAF
OAD*E&LS) ODDR&E
Pentagon, Rm. 3D129
Washington, D. C. 20301

Mr. Richard Coburn
Head, Human Factors Division
Naval Electronics Laboratory Center
San Diego, California 92152

Dean of Research Administration
Naval Postgraduate School
Monterey, California 93940

Navy Personnel Research and
Development Center (Code 10) (5 cys)
San Diego, California 92152

Mr. James L. Long
Weapons Systems Research (N-332)
Naval Education and Training Command
Naval Air Station
Pensacola, Florida 32408

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

Dr. George Moeller
Head, Human Factors Engineering Branch
Submarine Medical Research Laboratory
Naval Submarine Base
Groton, Connecticut 06340

Lt. Col. Austin W. Kibler
Director, Human Resources Office
Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, Virginia 22209

U.S. Air Force Office of Scientific
Research
Life Sciences Directorate, NL
1400 Wilson Blvd.
Arlington, Virginia 22209

Dr. J. M. Cristensen
Chief, Human Engineering Division
Aerospace Medical Research Laboratory
Wright-Patterson AFB, OH 45433

Dr. J. E. Uhlaner
Dir., U.S. Army Research Institute
for the Social & Behavioral Sciences
1300 Wilson Blvd.
Arlington, Virginia 22209

Chief of Research and Development
Human Factors Branch
Behavioral Science Division
Department of the Army
ATTN: Mr. J. Barber
Washington, D. C. 20310

Dr. Joseph Zeidner
Dir., Organization and Systems
Research Laboratory
U.S. Army Research Institute for
the Behavioral & Social Sciences
1300 Wilson Blvd.
Arlington, Virginia 22209

Dr. Stanley Deutsch
Chief, Man-Systems Integration
OART, Hqs., NASA
600 Independence Avenue
Washington, D. C. 20546

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

Dr. Edgar M. Johnson
Organizations & Systems Research Lab.
U.S. Army Research Institute for the
Behavioral and Social Sciences
1300 Wilson Blvd.
Arlington, Virginia 22209

Dr. James Parker
BioTechnology, Inc.
3027 Rosemary Lane
Falls Church, Virginia 22042

Dr. Edwin A. Fleishman
Foxhall Square
3301 New Mexico Avenue, N.W.
Washington, D. C. 20016

American Institutes for Research Library
135 N. Bellefield Avenue
Pittsburgh, Pennsylvania 15213

Psychological Abstracts
American Psychological Association
1200 17th Street, N.W.
Washington, D. C. 20036

Dr. A. I. Siegal
Applied Psychological Services
404 E. Lancaster Street
Wayne, Pennsylvania 19087

Dr. Joseph Wulfeck
Dunlap and Associates, Inc.
115 South Oak Street
Inglewood, California 90301

Dr. Robert R. Mackie
Human Factors Research, Inc.
Santa Barbara Research Park
6780 Cortona Drive
Goleta, California 93017

Mr. Wes Woodson
Man Factors, Inc.
4433 Convoy Street, Suite D
San Diego, California 92111

Dr. C. H. Baker
Director, Human Factors Wing
Defense & Civil Institute of
Environmental Medicine
P. O. Box 2000
Downsville, Toronto, Ontario
Canada

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

Dr. Bruce M. Ross
Department of Psychology
Catholic University
Washington, D. C. 20017

Mr. Harry Chipman
WR Systems, Inc.
2531 S. Jefferson Davis Highway
Arlington, Virginia 22202

Dr. David Meister
U.S. Army Research Institute
1300 Wilson Blvd.
Arlington, Virginia 22209

Mr. George Graine
Naval Ship Systems Command
(SHIPS 047C12)
Department of the Navy
Washington, D. C. 20362

Lt. Col. John F. Ahlborn
Headquarters, AFSC-DLSE
Andrews AFB
Washington, D. C. 20334

Dr. H. H. Wolff
Technical Director (Code N-2)
Naval Training Equipment Center
Orlando, Florida 32813

Dr. Donald A. Topmiller
Chief, Systems Effect. Branch
Human Engineering Division, USAF
Wright Patterson AFB, Ohio 45433

Col. Robert O. Viterna
DA/OCRD
Headquarters, Dept. of the Army
DARD-ARS-B
Washington, C. C. 20310

Dr. Anthony Debons
IDIS
University of Pittsburgh
135 N. Bellefield Avenue
Pittsburgh, Pennsylvania 15260

Dr. Alfred F. Smode
Training Analysis & Evaluation Group
Code N-OOT
Orlando, Florida 32813