Naive programmer problems with specification of transfer-of-control

by LANCE A. MILLER
IBM Thomas J. Watson Research Center
Yorktown Heights, New York

INTRODUCTION

We have conducted a series of experiments concerning the programming performance of persons with no prior contact with computers other than the training received in the experimental sessions. Our objective in these experiments is to identify design principles for facilitating communication between the naive user, as a problem solver, and a computer system. We view programming as a problem-solving activity, an instance of what generally may be called “procedure specification.” We believe it possible to design computers as optimal problem-solving tools only if the operating characteristics of the problem solvers are known and taken into account. Consequently, we are seeking to discover the problems and processes involved in human specification of procedures, using experimental laboratory methods.

A typical research method is to create a laboratory programming language designed to investigate questions of interest, teach this language to subjects, and study performance using this language for solving a variety of problems, either using computer terminals or pencil and paper tests. In addition to this approach we have applied content analysis methods to various kinds of procedure specifications expressed in natural English, these being collected either in our laboratory or in written publications. Our objective in this latter approach is to identify normative communication modes and mechanisms which may be usable for man-machine interactions.

The presentation is organized around information and graphics charts and covers the following topics: (1) initial studies demonstrating feasibility of investigating programming in the laboratory and suggesting expression of transfer-of-control as a locus of difficulty; (2) studies comparing various means for expressing transfer-of-control; (3) results of experiments using our “procedure table”; (4) results of analysis of specifications in natural language.

Introductory and explanatory text accompanying each chart is presented below.
PROCEDURE SPECIFICATION RESEARCH

1. IDENTIFY GEN'L. PROBS.

TWO EXPS. WITH

- VERY SIMPLE LAB. PROG. LANG.
- COLLEGE SUBJECTS
- SIMPLE PROBLEMS
- POWERFUL LAB. PROG. LANG.
- MORE COMPLEX PROBLEMS

RESULTS

*** SPECIFYING TRANSFER OF CONTROL A KEY DIFFICULTY (TIME ERRORS)

*** VARIETY OF CONTROL STRUCTURES PRODUCED

- PROGRAMS W/ REDUNDANCY BETTER THAN THOSE WITHOUT
- PROGRAMS W/ COMPLICATED FLOW OF CONTROL OFTEN IN ERROR

CONCLUSIONS

*** KEY PROBLEM TO FOCUS ON IS HOW BEST TO SPECIFY CONTROL.

Chart I—Our first two experiments demonstrated the feasibility of studying programming in the laboratory using naive (to computing) subjects. We found that a majority of the difficulty was concerned with specifying transfer-of-control.

ALTERNATIVES FOR SPECIFYING TRANSFER OF CONTROL

1. GRAPHICAL --
   E.G. FLOW DIAGRAMS

2. IF - THEN - ELSE
   (LEADS TO "TREE" OR HIERARCHICAL STRUCTURES)

Chart II—There are three primary ways of expressing the set of contingencies governing appropriateness of executing a particular action—where expression is in a procedural manner.

3. BRANCH - TO - LABEL (IF A YES: AI NO: B2)
   FLOW DIAGRAM

Other

DECISION TABLES
IPO & HIPO
CAUSE EFFECT GRAPHS

Chart III—Flow diagrams utilize two-dimensional representation with the yes/no actions to be taken depending on the truth-value of the predicate expressed as connective lines to other areas.

Chart IV—The "if-then-else" method involves hierarchical or nesting structures... (cf. also Chart VII).
3. BRANCH - TO - LABEL
(SYMBOLIC, LEADS TO COMPLEX BUT NOT EMBEDDED STRUCTURES)

GREEK: YES: NEXT NO: B2

B2: DO ACTION ...

Chart V—While the "branch-to-label" means permits a much wider variety of path connections (cf. also Chart VII).

EXPERIMENTAL EVALUATION IF/BRANCH/FLOW

HYPOTHESES UNDER TEST

IF VS. OTHERS
HIERARCHIC (EMBEDDED) VS. ONE-LEVEL NON-LINEAR STRUCTURES
BRANCH VS. FLOW
(1.) SYMBOLIC VS. GRAPHICAL TRANSFER
(2.) ONE VS. TWO DIMENSIONAL REPRESENTATION

PROCEDURE
GROUP TRAINING VIA BOOKLET
2 PRACTICE PROBLEMS WITH CORRECTNESS FEEDBACK
6 EXP'L. PROBS. WITH SYNTACTIC FEEDBACK ONLY

MEASURES
TRAINING -- TIME/ERRORS/COMPLETION
EXPERIM'L. -- TIME/ERRORS

Chart VI—We, like Sime, Green and Guest (1973), were interested in comparing these means of expressing transfer-of-control. A comparison of the IF method vs. the others is, among other things, a comparison of planning and procedure specification using hierarchical vs. non-hierarchical structures. A comparison of FLOW to BRANCH permits evaluation of graphical vs. symbolic representation of transfer-of-control and one vs. the highly commended two-dimensional representation.

IF (A) THEN
   IF (B) THEN
      IF (C) THEN X
      ELSE W
   ELSE V

EXP. EVAL. IF/BRANCH/FLOW (CONT.)

LANGUAGE FEATURES ALSO MANIPULATED
1. ABILITY TO USE LOGICAL OPERATOR "AND" "OR"
2. ABILITY TO USE NEGATION -- "NOT"

THESE WERE MANIPULATED NOT TO SEE IF DESIRABLE (THEY ARE) BUT TO FORCE DIFFERENT -- MORE COMPLICATED -- CONTROL STRUCTURES.

SUMMARY

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>NUMBER OF CONDITIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROG. MODE</td>
<td>3</td>
</tr>
<tr>
<td>USE OF &quot;AND, OR&quot;</td>
<td>2</td>
</tr>
<tr>
<td>USE OF &quot;NOT&quot;</td>
<td>2</td>
</tr>
<tr>
<td>PROBLEMS</td>
<td>6</td>
</tr>
</tbody>
</table>

Chart VII—Use of the IF procedure restricts the variety of incorrect transfers which may occur—e.g., given the hierarchical specification of tests A, B, erroneous transfer from C to action Z instead of W cannot occur. With a BRANCH method, given appropriate means of identifying commands (command numbers or labels) transfer may occur to any other command without restriction.

Chart VIII—Each of the three means of expression conditionals was studied under four language-feature conditions resulting from combinations of two variables—capability to employ the negation operator "not", and capability to employ logical connectives "and" and "or". The intent of these manipulations was not to study the desirability of providing such features (they are highly desirable) but to force subjects to use various features of the conditional modes. For example, when use of the operator "not" is not permitted in the IF mode, subjects cannot simply test for "not-x" by specifying "IF not-x..." but they must employ the "else" aspect of the conditional—e.g., "IF X... ELSE K" where K represents the actions appropriate for the condition of "not-x."
EXP. EVAL. IF/BRANCH/FLOW (CONT.)

PROBLEMS: SIX PROBLEMS VARYING ON TWO DIMENSIONS
LOGICAL OPERATOR -- AND vs. OR (INCL.)
POS. VS. NEG. STATEMENT -- POS. NEW NEG. MIXED

EXAMPLE:

PUT THINGS IN BOXES INDICATED WHEN THE CONDITIONS ARE MET

BOX 4: ALL THINGS THAT ARE OLD AND NOT BROKEN
BOX 2: ALL OTHER THINGS THAT ARE OLD OR NOT WOODEN
BOX 1: EVERYTHING ELSE

Chart IX—Hierarchical multiple-action problems of the type used by Sime et al., were employed, using a fixed level of three actions and three relevant attributes. Within this framework the logical characteristics of the problems were varied on two dimensions—the logical connective, and vs. or, and the presence or absence of the operator not. The second action-line of the problem was always the place of variation (e.g., an affirmative conjunctive problem would read “… BOX 2: ALL OTHER THINGS THAT ARE OLD AND WOODEN…”).

EXP. EVAL. IF/BRANCH/FLOW (CONT.)

RESULTS WITH COLLEGE STUDENTS

1. TRAINING -- IF SLIGHTLY MORE DIFFICULT
2. EXP. PROBS. -- NO DIFFERENCES AMONG THREE MODES
   SIG. EFFECTS OF OTHER VARIABLES

RESULTS WITH NAVY RATINGS

1. TRAINING -- IF CONSIDERABLY MORE DIFFICULT
2. EXP. PROBS. -- IF SIG. MORE DIFFICULT, NO OTHER DIFF.

CONCLUSIONS

1. SYMBOLIC ADDRESSES AND 1-DIM. REPRESENTATION ARE O.K.
2. POSSIBLE POPULATION DIFFERENCES
3. CLOUDS OF SUSPICION FOR IF

Chart X—The finding that IF was either no different from or poorer than the other modes is in contrast to Sime et al.’s finding of IF superiority to BRANCH. We believe that our experimental testing situation was more neutral than theirs, with automatic indentation, for example, facilitating IF performance, and forcing a jump on positive outcome for BRANCH interfering with BRANCH performance. The finding of no differences, over language conditions or populations, between FLOW and BRANCH brings into question the often expressed, but untested, claim of flow-diagram superiority over other methods.

Chart XI—At this point we decided to test out our understanding of the processes and problems involved in programming specification of transfer-of-control by attempting to generate a method for specification which would not be subject to the problems characterizing the previously tried modes of specification, at least for the same problems. We considered a variety of possibilities, keeping in mind the insignificant BRANCH-FLOW results which suggested that one-dimensional, symbolic transfer structures did not lead to greater difficulty than FLOW representation.
Naive Programmer Problems with Specification of Transfer-of-Control

**First Procedure Table Experiment**

Same problems as in If/Branch/Flow Experiment

(Permitted use of "And, Or, Not")

Training same method -- booklet -- about same time (20 min.)

**Results**

No subject had difficulty with training

Significantly better than for other modes

in both college and navy populations

**Conclusions**

Maybe we are on to something

Best alternative so far

Chart XIII—In the same manner as the experiments comparing the three conditional modes we developed training and testing procedures and materials for the Procedure Table, using the same problems. Performance was significantly better in training and in testing than with the other modes of specification suggesting that there was merit in our present tabular structuring and syntax.

**Complex Problems**

How we are interested in taking real world problems and seeing how our subjects fare.

**Took Examples From a Number of Sources**

- Recipes
- Instruction Manuals
- Repair Manuals
- Math. Problems

Chart XIV—The problems in most of the preceding experiments were fairly simple. We now wished to see how subjects would fare in translating various complex examples of specified procedures into procedure tables.
SECOND PROCEDURE TABLE EXPERIMENT

USED 18 REAL WORLD PROBLEMS
BOTH COLLEGE AND NAVY SUBJECTS
SOLUTIONS GIVEN USING PROCEDURE TABLES

RESULTS DATA PRESENTLY BEING ANALYSED

APPARENT TRENDS

DIFFICULTY DOES NOT APPEAR TO BE IN USE OF
PROCEDURE TABLE, THIS IS O.K. FOR SIMPLER PROBLEMS.

PRIMARY DIFFICULTY APPEARS TO BE IN
(1) FORMULATING PROBLEM
(2) FORMULATING SOLUTION, APPROACH, TO PROBLEM

Chart XV—While results continue to be analyzed, a key trend appears to be that incorrect specifications are due not so much to difficulty in using the procedure table but errors in understanding or resolving the ambiguity in the procedures expressed in natural language. This suggests the utility of adding pre-programming phases which facilitate planning and consideration of the solution algorithm eventually to be specified. We are currently evaluating various such planning aids.

PROCEDURE SPECIFICATION

<table>
<thead>
<tr>
<th>NATURAL LANGUAGE</th>
<th>PROGRAMMING LANGUAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTEXT:</td>
<td>HEAVY USE</td>
</tr>
<tr>
<td>TRANSFER OF</td>
<td>INCOMPLETE</td>
</tr>
<tr>
<td>CONTROL</td>
<td>SPECIFICATION</td>
</tr>
<tr>
<td>ITERATION:</td>
<td>VAGUE</td>
</tr>
<tr>
<td>INITIALIZATION/</td>
<td>NEVER STATED</td>
</tr>
<tr>
<td>EDF</td>
<td></td>
</tr>
<tr>
<td>EXCEPTION HANDLING</td>
<td>NEVER STATED</td>
</tr>
</tbody>
</table>

Chart XVII—The natural language characteristics contrast strongly with features of programming languages . . .

NATURAL LANGUAGE SPECIFICATIONS

RECENT REPORT DESCRIBES RESULTS OF EXPERIMENT ---
ASKED 14 COLLEGE STUDENTS TO SPECIFY THE
SEQUENCE OF STEPS (IN DETAIL) THEY WOULD
FOLLOW TO SOLVE CERTAIN FILE SEARCH AND
MAINTENANCE PROBLEMS.

EXPERIMENTAL RESULTS --
DATA REFERENCING -- 42% REQUIRED CONTEXT
TRANSFER OF CONTROL -- LITTLE (7-9%) COMPARED
WITH REAL PROGRAMS (E.G.,
KNUTH=20%)
(SOME "IF-THEN" B T NO
"ELSE")

OPERATORS -- HIGH LEVEL OPERATING ON WHOLE DATA
AGGREGATE (FIND THOSE WHO...), LIKE APL

OMISSIONS -- NO EOF, INITIALIZATIONS, DECLARATIONS,
DIMENSIONING DATA TYPE SPECIFICATION,
EXCEPTION HANDLING

Chart XVI—In parallel with our laboratory studies of programming we have also been interested in learning about how people naturally specify procedures. Our first study involved college students writing procedures to interrogate a data base of a mythical company, and we found some interesting features . . .

OTHER NATURAL LANGUAGE ANALYSES

EXAMINATION OF REAL-WORLD SPECIFICATIONS OF PROCEDURES --
** KITCHEN RECIPES
** KNITTING INSTRUCTIONS
** KIT TESTING/ASSEMBLY INSTRUCTIONS
** TROUBLE SHOOTING MANUALS

PRESENTLY ANALYZING "JOY OF COOKING"

* HOW ARE PROCEDURES CALLED?
* HOW ARE DATA ENTITIES REFERRED TO?
* HOW ARE PARAMETERS EXPRESSED AND PASSED
* HOW EXPRESS TRANSFER OF CONTROL, EXCEPTION
HANDLING, ITERATION?
* HOW DATA SELECTION SPECIFIED?
* WORD USAGE

Chart XVIII—In view of the interesting results obtained from this study, we felt it might be even more informative to investigate how highly stable and evolved procedures are specified—e.g., knitting instructions, kitchen recipes, kit assembly instructions. We chose kitchen recipes to begin this study (for a variety of reasons, including some important secondary considerations as evaluating output after actual execution).
Programming Language

vs.

Natural Language

seems to be the case of...

Conditional

vs.

Qualificational

IF THING

is red,

put in box 1

else...

Put red things in box 1

Charts XX to XXIII—We had as one of our verb categories an IF classification to provide for statements of the conditions that had to be satisfied before the subsequently specified actions should be taken. This category corresponds roughly to the conditional command in programming languages, and it is of interest to note how rarely incidents of this class occurred. Most often actions or recipe components were characterized not in terms of a sequence of testing for certain attribute values and then specifying the appropriate actions but in terms of specifying the action to be taken in certain qualified ways for certain qualified components. The qualificational method may well prove a more useful model for language design for naive users than the traditional conditional method of programming languages of the present.

REFERENCES

