

GOVERNMENT INFORMATION
University of Washington
Seattle, Washington 98195

PROCEEDINGS

March 15-17, 1982
Gaithersburg, Maryland



HUMAN FACTORS IN COMPUTER SYSTEMS

Sponsored By



INSTITUTE FOR COMPUTER
SCIENCES AND TECHNOLOGY
NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE



WASHINGTON, D.C., CHAPTER
ASSOCIATION FOR
COMPUTING MACHINERY

In Cooperation With



OFFICE OF NAVAL RESEARCH
ENGINEERING PSYCHOLOGY PROGRAMS

SOFTWARE PSYCHOLOGY SOCIETY

COMPUTER SYSTEMS GROUP
HUMAN FACTORS SOCIETY

IFIP WORKING GROUP 6.3

ENGINEERING AND SCIENCE LIBRARY
WEAN HALL
CARNEGIE-MELLON UNIVERSITY
PITTSBURGH, PA 15213

INDENTATION, DOCUMENTATION AND PROGRAMMER COMPREHENSION

A. F. Norcio

Department of Applied Science
U. S. Naval Academy
Annapolis, Maryland 21042

INTRODUCTION

Recent investigations into the psychological factors underlying computer programming have focused on the effects of internal documentation and statement indentation on programmer performance [1, 5, 7]. Using memory recall approaches several studies have concentrated on the relationship between logic segments of a program's algorithm and the memory organization of the programmer. Evidence suggests that programmer memory organization of program statements closely parallels the logic segments of the algorithm [2, 7]. While statement indentation and internal documentation are considered important elements in good programming style, their effects on programmer performance have not been observed decisively even under differing experimental methodologies [1, 2, 8].

Since memory organization appears to be a functional psychological process, it seems reasonable to assume that indentation and documentation function as aids to comprehension rather than as organizers for memory [4]. The purpose of this study was to examine experimentally the relationship between documentation, indentation and the comprehension of computer programs.

In examining the comprehension of computer programs, Norcio [3, 4] has proposed the Cloze Procedure [6] as a possible reliable and valid technique. With this approach blank lines are substituted for appropriate program statements. The subjects' tasks then become supplying correct statements for the blank lines. Using this procedure, this study examined the effects of documentation and indentation on program comprehension in two separate experiments.

METHOD

Overview

Two separate experiments were conducted. While the two experiments were structurally similar, they were conceptually distinct. In the first experiment, methodological manipulation occurred at the beginning of logic segments. In the second experiment, the manipulation occurred within logic segments.

©1981 ASSOCIATION FOR COMPUTING MACHINERY

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

Experiment 1

Subjects. Seventy computer programming students participated in groups of ten students each: 1) one panel, and 2) six experimental conditions. The six experimental conditions represented three documentation levels (ND-no documentation, BD-beginning documentation, ID-one line of interspersed documentation between logic segments) and two indentation levels (NIN-no indentation, IND-indentation).

Materials. Five Fortran programs which varied from twenty to thirty-three lines were used. These programs were appropriately complex for programming students.

Procedure. The panel was used to define the logic segments in each program. A consensus of opinion decide the logic segments in each program.

No documentation was included in the programs which the ND groups viewed. Internal documentation was incorporated at the beginning of each program given to the BD groups. One line of interspersed documentation between the logic segments of the programs which the ID groups viewed. The programs given to the IND groups had their statements indented according to the logic segment definitions. The program given to the NIN groups had no indentation at all.

In this experiment, blank lines were substituted for program statements at the beginning of each logic segment.

Subjects were randomly assigned to each of the six experimental conditions. Programs were presented to the subjects in random order so that order would be balanced. The subjects were instructed to study the first program and supply correct versions of the missing statements. They were further instructed to repeat this process until all five programs had been examined. Subjects were tested individually and had thirty minutes to complete the experiment.

The number of correctly supplied statements for each of the five programs was recorded for all subjects.

Results. A 2 x 3 multivariate analysis (MANOVA) was computed with documentation and indentation as the independent variables and the numbers of correctly supplied statements for each of the five programs as the dependent variables. The results generally indicate that groups with indented programs and interspersed documentation were able to supply significantly more correct statements.

Specifically, there was a significant multivariate interaction effect ($p < .01$) in favor of the ID/IND group. There was a significant multivariate indentation effect ($p < .05$) in favor of the ID group. There was a significant multivariate documentation effect ($p < .05$) in favor of the ID groups. There were also several significant univariate effects between the various programs.

Experiment 2

Subjects. Sixty computer programming students participated in groups of ten students each in six experimental conditions, which were identical to experiment 1. These subjects did not participate in experiment 1.

Materials. The same materials which were used in experiment 1 were also used in this experiment.

Procedure. The logic segments defined by the panel in experiment 1 were also used in this experiment.

The procedure used in this experiment was identical as in experiment 1 with one fundamental difference. In this experiment blank lines were substituted for program statements within logic segments.

Results. A 2 x 3 MANOVA was computed with documentation and indentation as the independent variables and the numbers of correctly supplied statements for each of the five programs as the dependent variables. The results generally indicate that groups with indentation were able to supply more correct statements. Specifically, there was a significant multivariate interaction effect ($p < .04$) in favor of the ID/IND group. There was a significant multivariate indentation effect ($p < .008$) in favor of the IND groups. There were also several significant univariate effects between the various programs.

DISCUSSION

While previous studies have not observed significant indentation effects on recall, the present investigation presents evidence that indentation aids comprehension. It is possible that indentation enhances comprehension by detailing and maintaining the logic segment organization in the context of the overall program. Since significant indentation effects were observed in both experiments, perhaps this detailing provides a programming logic template which coincides with the logic segments' mental representations.

It is not surprising that documentation effects were significant in the Beginning Segment experiment and not in the other. According to psychological theory, the first element of a chunk provides a key to its contents. In the second experiment, the initial statement of each logic segment was present. Apparently, the combination of the beginning statement and indentation provided sufficient assistance to comprehension. The task in the first experiment was presumably more difficult, since the first statement of each segment had to be supplied. The significant documentation effect suggests that one line interspersed documentation provided a conveniently

located clue to the entire segment and consequently to its first necessary statement. The significant interaction effect in favor of the indented/interspersed group in both experiments suggests that this combination might provide the optimal combination for enhancing comprehension.

These experiments suggest that highlighting a program's logic segments provides a useful style in comprehending computer programs. The results further indicate that statement indentation and interspersed documentation are appropriate techniques in accomplishing this goal.

REFERENCES

1. Love, L. T. Relating Individual Differences in Computer Programming Performance to Human Information Processing Abilities. Ph.D. Thesis, University of Washington, 1977.
2. Norcio, A. F. Memory Organization and Computer Program Logic. Paper presented at the Annual Meeting of the Human Factors Society, Detroit, 1978.
3. Norcio, A. F. The Cloze Procedure: A Methodology for Analyzing Computer Program Comprehension. Paper presented at the Annual ACM Computer Science Conference, Dayton, 1979.
4. Norcio, A. F. Factors Affecting the Comprehension of Computer Programs. Paper presented at the National Computer Conference, New York, 1979.
5. Norcio, A. F. and Kerst, S. M. Documentation and Indentation Effects on Memory Organization of Computer Programs. Paper presented at the Annual Meeting of the American Psychological Association, New York, 1979.
6. Rankin, E. F. An Evaluation of the Cloze Procedure as a Technique for Measuring Reading Comprehension. Ph.D. Thesis, University of Michigan, 1957.
7. Sheiderman, B. Exploratory Experiments into Programmer Behavior, International Journal of Computer and Information Sciences, 1976, 5, 123-143.
8. Sheiderman, B. and Mayer, R. Syntactic/Semantic Interactions in Programmer Behavior: Analysis and Experimental Results. International Journal of Computer and Information Sciences, 1979.

APPENDIX

Sample Program 1

```
C THIS PROGRAM CONVERTS ROMAN NUMERALS TO THEIR
C CORRESPONDING ARABIC NUMBERS. THE ARRAY R
C CONTAINS THE ROMAN NUMERALS AND THE ARRAY V
C CONTAINS THEIR CORRESPONDING ARABIC EQUIVALENTS.
C
C THE ROMAN NUMERAL IS INPUTTED TO THE VARIABLE
C IN. STARTING FROM RIGHT TO LEFT THE LOWEST ORDER
C NUMERAL IS DETERMINED AND THE APPROPRIATE
C CONVERSION IS MADE.
C
```

```
IMPLICIT INTEGER (A-Z)
DIMENSION R(7),V(7),IN(15)
DATA R/'M','D','C','L','X','V','I'/
DATA V/1000,500,100,50,10,5,1/
```

```

10     FORMAT (15A1)
      LAST=0
      IF(IN(I).EQ.'') GO TO 100
      DO 80 J=1,7
      IF(IN(I).NE.R(J)) GO TO 80
      IF(V(J).GE.LAST) OUT=OUT+V(J)
      IF(V(J).LT.LAST) OUT=OUT-V(J)
      LAST=V(J)
80     CONTINUE
100    CONTINUE

20     FORMAT(1X,I10)
      GO TO 1
      END

```

Sample Program 2

```

C THIS PROGRAM CONVERTS ROMAN NUMERALS TO ARABIC
C NUMBERS
      IMPLICIT INTEGER (A-Z)
      DIMENSION R(7),V(97),IN(15)
      DATA R/'M','D','C','L','X','V','I'/
      DATA V/1000,500,100,50,10,5,1/
C INPUTS THE ROMAN NUMERALS

10     FORMAT(15A1)

      LAST=0
C CHECKS THE ROMAN NUMERALS BACKWARDS
      IF(IN(I.EQ."") GO TO 100
      DO 80 J=1,7
C CONVERTS THE NUMERAL TO ITS CORRESPONDING NUMBER
      IF(IN(I).NE.R(J)) GO TO 80
      IF(V(J).GE.LAST) OUT=OUT+V(J)
      IF(V(J).LT.LAST) OUT=OUT-V(J)
      LAST=V(J)
80     CONTINUE
100    CONTINUE
C OUTPUTS THE ARABIC EQUIVALENT

20     FORMAT(1X, I10)
      GO TO 1
      END

```