Part I

COMPUTER SCIENCE

BIT 20 (1980), 394–405

A METHODOLOGY FOR MEASURING THE READABILITY AND MODIFIABILITY OF COMPUTER PROGRAMS

ANKER HELMS JØRGENSEN

Abstract.

This paper describes an experimentally based methodology for investigating how the readability and modifiability of computer programs can be expressed in terms of programming style.

As one part of the methodology three different measures of the readability and modifiability of a program are established. The measures are obtained by human gradings or scores. As another part these measures are related to computer-evaluable programming style characteristics. The result is a formula yielding the readability and modifiability as a function of measurable program characteristics.

In testing the methodology one experiment provided a formula of readability containing six computer-evaluable programming style characteristics. A further experiment indicated that the general validity of this formula of program readability needs further investigation.

Keywords: Experiment, software quality, readability, modifiability, programming style.

1. Introduction.

Since the software crisis was recognized in the late sixties much effort has been spent in developing concepts and designing tools facilitating the construction of high quality software. Nevertheless, program development and maintenance is a steadily increasing fraction of total cost of computer usage (Boehm [4]). Moreover, the cost of maintenance is about double that of development (Zelkowitz [14]). In maintenance of computer programs, readability and modifiability are important issues; among the factors that cause difficulty in reading and modifying programs, the task and algorithm will undoubtedly enter, but a large part will presumably be determined by the style of programming. The purpose of the study presented here is to obtain objective empirical evidence on how the readability and modifiability of computer programs can be expressed by programming style characteristics.

Along with the development of the discipline of software science, many theoretical and experimental studies have been reported, see Halstead [7] and Fitzsimmons and Love [6]. In these studies, measures of program complexity and clarity have been developed and tested. The measures, however, are highly correlated to the program length; little attention has been paid to the

Received September 27, 1979. Revised August 18, 1980.

programming style, which is an attribute of programs that is independent of the program length.

A few publications on programming style based on personal experience exist, such as Roberts [12] and McCracken and Weinberg [11]. Kernighan and Plauger [8] treated the elements of programming style in terms of examples from the literature. Boehm et al. [3] discussed the characteristics of software quality and proposed metrics for the characteristics. Elshoff [15] studied the readability of 120 commercial PL/I programs. He claimed: "Basically, the programs are unreadable. An experienced programmer will find them extremely difficult to read". He concluded that the readability of programs cannot be automatically measured. He admitted, however, that specific attributes which effect readability can be identified.

An experimental approach to the problem is described by Weissman [13]. He conducted a series of experiments on the psychological complexity of programs. He found that variable names, program structuring, comments, control flow, and paragraphing were significant factors.

Recently Curtis et al. [5] measured the psychological complexity of software maintenance tasks using the Halstead and McCabe program complexity metrics. These metrics were compared to programmer performance on two maintenance tasks. Programs with various levels of commenting, variable name mnemonicity, and control structures were studied by the programmers. No significant differences in performance across the various levels of these programming style factors were found.

In natural languages readability of texts has been studied for many years (Klare [9]). In Sweden for example, Björnsson [2] constructed the so-called LIX-measure. It measures the readability of a text by the average sentence length in words plus the percentage of words of more than six letters. The smaller the figure the more readable the text.

In the present study an experimentally based methodology is developed for investigating how the readability and modifiability of computer programs can be expressed in terms of programming style characteristics. The elements of the methodology are first described. Next, details and results of two experiments that make use of the methodology are described.

In the first experiment a formula expressing the readability of programs as graded by computer science experts was found. The formula consisted of six computer-evaluable programming style characteristics concerning comments, indentations, blank lines, length of variable names, arithmetic operators, and **goto**-statements.

The second experiment was designed with great care, especially regarding the methodological aspects. For example, variation caused by external factors was attempted reduced. Nevertheless, the second experiment yielded no results similar to the formula found in experiment 1. This indicates that the general validity of this formula needs further investigation.

In using the present methodology, programming style characteristics are investigated as they occur in real programs. This contrasts with the method used by Weissman [13] and Curtis et al. [5], in which a few characteristics are varied in levels defined by the experimentor. Moreover, the present methodology imposes no restrictions upon the number of characteristics that can be investigated in an experiment. This is due to application of a statistical procedure, which selects a small number of characteristics from a large number.

The most notable result of the present study is the formula of readability found in the first experiment, where ten computer science experts' gradings of the readability were expressed quite precisely by a formula of six computer-evaluable programming style characteristics.

2. Elements of the methodology.

The elements of the methodology are introduced in detail in this section. Two experiments making use of the methodology are described in sections 3 to 6.

As an introduction, consider the following experiment which aims at the establishment of a computer-evaluable measure of program readability. First, a set of programs having a reasonable range of style characteristics is selected. Second, a measure of the readability of each program is established by having a group of programmers work with the set in suitable manners and state their impressions of the readability. Third, suitable style characteristics are chosen and their use in each of the programs is determined by an analysis program that takes the text of each program as input. Fourth, using a statistical selection procedure, the readability obtained in the second step is expressed approximately in a formula that combines the style characteristics obtained in the third step.

The methodology is based on the concepts readability, modifiability, and programming style. These concepts are introduced in the following paragraphs.

The definition of the concept readability given by Björnsson [2] in natural languages is used here: "The *readability* of a text is the sum of stylistic factors that make the text more or less understandable to the reader". The readability of a program can be determined by several methods. A reader (subject) can evaluate his or her understanding of the program on some nominal scale or by ranking; or it can be determined by someone's performance in an examination that depends on his or her understanding of the program.

In order to reduce the influence of individual personal factors, the readability of a program is normally determined by several persons. The average of these determinations is denoted the *readability-index*. This is a measure of the readability of a program.

The definition of the concept modifiability given by Boehm et al. [3] is used here: "A software product possesses the characteristic *modifiability* to the extent that it facilitates updating to satisfy new requirements".

The modifiability of a program can be determined by having a subject carry out modifications to the program. The accuracy of the modifications can be evaluated, giving the subject's performance. Normally, several subjects carry out the modifications in order to reduce the influence of individual personal factors. The average of these performances is denoted the *modifiability-index*. This is a measure of the modifiability of the program.

The concept programming style is defined as follows: *Programming style* includes those characteristics of a program that are chosen by the programmer when the task, the algorithm, and the programming language is given. Typical style characteristics are comments and indentations.

A programming style characteristic must fulfil several requirements. It should occur in practically all programs in order not to restrict its range of application; it should be such that an average programmer will be able to write programs with an acceptable level of the characteristic; it should not imply a trend towards vulgarization of the language; and finally, it should not be difficult to evaluate.

There are two kinds of programming style characteristics. The first kind can be expressed algorithmically and thus be evaluated by a computer program. Such characteristics are here denoted computer-evaluable characteristics or *C-characteristics*. An example is the average length of variable names. Application of C-characteristics in an experiment requires an evaluation program. This program inputs the set of programs and outputs the so-called *C-figures*. These are measures of the quality of the program with respect to the C-characteristics.

The other kind of programming style characteristics requires evaluation by a human being. An example is the meaningfulness of the variable names. Such characteristics are here denoted human-evaluable characteristics or *H-characteristics*. In order to reduce the influence of individual personal factors the H-characteristics of a program are normally evaluated by several persons. The average of these evaluations is denoted the *H-figure*. This is a measure of the quality of the program with respect to the H-characteristics. The evaluation of the H-characteristics can be made by the subjects participating in the experiment or by independent experts.

In using the present methodology, the indices of the readability and the modifiability enter in a statistical selection procedure. This procedure aims at finding the "best" approximating formula expressing the readability- and modifiability-indices as functions of the C- and H-figures. Björnsson [2] applied a stepwise linear regression analysis technique. This is used here:

$$R \sim R' = a_1 c_1 + a_2 c_2 + \ldots + a_k c_k + b$$

where R is the readability-index, R' is an approximating formula (a linear regression equation), the c_i's are a subset of the C- and H-figures, and the a_i's and b are regression coefficients.

The technique requires carrying out of a large number of linear regression analyses using selected subsets of the C- and H-characteristics.

The selection of the "best" approximation R' is a compromise. It is based on the following criteria. The residual variance of the deviation $R-R'$ should be small.

When the number of characteristics in the subset is increased, the residual variance should decrease. The signs of the regression coefficients should be in accordance with the "nature" of the corresponding characteristic. For each characteristic, the sign and magnitude of the coefficient should be stable when other characteristics are included or excluded in the subset. The formula should be comprehensible; thus, the number of characteristics in the formula should be small.

The result of the statistical selection procedure is a formula including a subset of the characteristics. A formula consisting only of C-characteristics can be implemented as a computer program. Such a program is an automatic tool for assessing the readability or modifiability of programs. However, such a program has the weakness that it can be "cheated" by a programmer who adds for example meaningless comments. Cosmetics, however, do not aid readability of programs in practice.

3. Programs and subjects.

It was decided to use only programs written in high level programming languages. These are widely used and they are machine independent.

Two kinds of programs can be used, namely artificially constructed programs and real-world programs. Weissman [13] used the former kind. In investigating the mnemonicity of variable names he used three programs for solving the eight-queens problem with three levels of mnemonicity: fully mnemonic, shortened mnemonic, and meaningless, respectively. Real-world programs are not written with artificially varied levels of programming style characteristics, however. Therefore we decided to use real-world programs.

The subjects determining the readability and modifiability of a set of programs should evidently be familiar with the programming language. They should also be familiar with the task of the programs and the applied algorithms so as to make their performances as far as possible independent of these irrelevant matters.

4. Experiment 1.

The purpose of this experiment was to gain experience with the methodology. Only readability of programs and computer-evaluable programming style characteristics were considered. The experiment was conducted in 1972–74 as a student project at the Institute of Datalogy, University of Copenhagen. Participants were I. Andersen, I. Carstensen, H. Friis, L. Fischer, and the author. The experiment is described in Danish by Andersen et al. [1].

The experiment was based on a set of 48 Algol 60 application programs written at universities and software houses. The tasks performed by the programs ranged from numerical analysis applications to a simple operating system. The programming style varied widely. There were programs without indentations, comments, etc., and there were programs with thorough descriptions, paragraphing, etc. The minimum, average, and maximum program size was 2, 6, and 14 pages, respectively.

The readability graders were ten computer science experts. They were given an instruction along with the programs and they were allowed several weeks for doing the grading. Each expert stated his impression of the readability of each program on a 7-point scale, where 1 is excellent, 4 is fair, and 7 is very poor.

It turned out that the gradings produced by the various graders were in good agreement. For each of the 48 programs the variance between the 10 experts' gradings was calculated. The minimum, average, and maximum variance was 0.4, 1.2, and 1.7, respectively (on the 1–7 scale).

A total of 57 programming style C-characteristics were defined. Examples of these are the average number of procedure parameters, **if-then-else**-statement density, and maximum depth of parentheses nesting. The C-characteristics were evaluated by a program producing 57 C-figures for each of the 48 programs in the set.

The next step was application of the statistical selection procedure. Several thousand linear regression analyses were performed. One particular regression equation turned out to be the "best" according to the criteria mentioned in section 2. All the criteria were fulfilled. The characteristics of the equation were:

–	Extensiveness of comments	C1
–	Extensiveness of blanks in the left margin	C2
–	Extensiveness of blank lines	C3
–	Average length of the variable names	C4
+	Average number of arithmetic operators per 100 lines	C5
+	Average number of **goto**-statements per label	C6

The regression formula and the definition of the six characteristics is found in appendix I. Typical values of the C-figures are also given there.

The signs in the leftmost column above are interpreted as follows: A minus means that the characteristic aids the readability when the C-figure increases, while a plus means that the characteristic counteracts the readability when the C-figure increases. Notice the direction of the scale: 1 is excellent and 7 is very poor.

The validity of the statistical selection procedure is described in appendix II. This yields an estimate of the deviation between the graded readability R and the approximation R'. For 68% of the programs this deviation is less than 0.4, while it is less than 0.8 for 95% of the programs (on the 1–7 scale).

5. Discussion of experiment 1.

There are three serious drawbacks of experiment 1. The first is that the programming style characteristics are computer evaluable. This implies, for example, that a lengthy comment automatically is interpreted as a meaningful comment. The result, however, shows that comments in general have aided understanding of the programs.

The second drawback is the measure of readability applied. The experts stated their impression of the readability of the programs. These statements are a questionable estimate of the readability of the programs.

The third drawback is that other issues than the programming style varied among the programs, namely the tasks and the algorithms. The difficulty in understanding the programs caused by these two issues could not be distinguished from the difficulty caused by the programming style.

In spite of these drawbacks and the vagueness of the concept of readability, it proved possible to describe the experts' gradings of the readability by a formula. This formula appeared as the unique "best" in the statistical selection procedure. Further, it is intuitively clear that the characteristics do affect the readability of programs. The characteristics are also in accordance with the generally agreed opinion on programming style. An exception is the characteristic C5 (average number of arithmetic operators per 100 lines). This characteristic is due to a few programs performing tasks in numerical analysis.

6. Experiment 2.

In this experiment it was attempted to avoid the drawbacks of experiment 1 and also to include the best features of the methodology developed by Weissman [13]. Further, it was decided to study modifiability in addition to readability since reading a program is only an initial step in modifying the program. Moreover, the results found by Weissman and in experiment 1 were to be re-examined. Experiment 2 was conducted in 1975–77 at the Institute of Datalogy, University of Copenhagen, as a student project supervised by Larry Weissman. The methodology was fitted for this experiment by I. Carstensen, L. Fischer, and the author, while the author conducted the experiment.

In experiment 1 the extraneous influence from task and algorithm was not avoided. In order to reduce this influence, experiment 2 is based on programs that realize the same task and the same algorithm. The task is topological sorting and the algorithm is the one given by Knuth [10]. The programs, 10 Pascal programs, were written by computer science students at the University of Århus.

The variation in the programming style between the programs was not large. This is due to the similarity of the students' educational backgrounds. The minimum, average, and maximum program size was 2, 3, and 7 pages, respectively.

One hundred first year computer science students participated in the experiment as subjects. Before the experiment they had studied a Pascal program similar to the 10 experiment programs. Consequently they were familiar with the programming language, the task, and the algorithm.

The experiment was conducted as a 2-hour session. Each subject was given one program; thus each program was studied by 10 subjects. An instruction and a questionnaire was handed out along with the programs.

The readability of the programs was measured by two methods. The first was the subjects' evaluations of their own understanding of the program. The 7-point scale applied in experiment 1 was also applied here. The second measure was an examination of the subjects' understanding based on their answers to 9 questions about the programs. The answers were evaluated by the author on a 3-point scale.

As a measure of the modifiability, the subjects' performances in carrying out two modifications to the programs were used. Both modifications consisted of updating the program to satisfy new requirements. The author evaluated the subjects' performances on a 5-point scale.

The six computer-evaluable programming style characteristics found in experiment 1 were re-examined in this experiment. They were transformed to Pascal. The C-figures were evaluated by a program producing six C-figures for each of the 10 programs. The characteristic C5 (average number of arithmetic operators per 100 lines) and C6 (average number of **goto**-statements per label) were represented most sparsely in the programs. They were consequently omitted.

As mentioned in the introduction Weissman [13] had found five significant programming style characteristics. They are:

Choice of variable names	H1
Structure of the program	H2
Use of comments	H3
Choice of control structures	H4
Paragraphing of the listing	H5

Weissman varied these characteristics in levels in artificially constructed programs. In experiment 2 real-world programs were used. Consequently, the quality of the use of the characteristic in the programs had to be evaluated. On basis of a pilot-experiment it was decided to let the subjects do the evaluation of these H-characteristics; a 7-point scale was used. It turned out that many of the subjects had not understood the characteristic H4 (choice of control structures). This characteristic was therefore omitted.

The final step of experiment 2 was to carry out the statistical selection procedure. All possible regression analyses of the C- and H-figures on the two readability indices and the modifiability index were performed. Only for one selection of C- and H-figures were all the criteria mentioned in section 2 fulfilled. This case was approximation of the question-performance readability-index solely by the characteristic H3 (use of comments); for this approximation the residual variance was 0.025 (on the 1–3 scale).

This result confirms one of the factors found by Weissman. The result, however, is a bit disappointing, considering the methodological care lavished on the experiment and in view of the success of the preceding experiments, namely experiment 1 and Weissman's study. Some reasons for the meagre result are discussed below.

One reason for this is that the subjects' experience in programming was at the lowest acceptable level. Records and pointers were used extensively in the programs for building non-trivial data structure dynamically. Although the subjects had learned about records and pointers, they had not themselves written programs using these concepts. Thus, use of these language elements in the programs may have caused trouble to the subjects. Moreover, the idea of the algorithm given by Knuth [10] is hard to grasp. Although the subjects had worked with the algorithm beforehand, its difficulty may have influenced their performances.

Yet another reason for the meagre result is that the programming style varied little among the programs. This means that a possible systematic variation in the subjects' performances and evaluations may have been dominated by the individual variations among the subjects.

Experiment 2 provided other information, however, namely about pairs of C- and H-characteristics covering the same stylistic aspect and about the two measures of readability.

An example of a pair of C- and H-characteristics covering the same stylistic aspect is C4 (average length of variable names) and H1 (choice of variable names). Such pairs were compared by correlation analysis. The correlation coefficient between the characteristics C1 (extensiveness of comments) and H3 (use of comments) was 0.84. Thus, extensive comments are considered more meaningful than sparse comments. In all other pairs the correlation was less than 0.5.

Comparison of the two measures of readability (self-evaluation and question-performance) yielded a correlation coefficient of 0.85. This means that self-evaluation is as valid a measure as performance in answering questions. Notice that the readability-indices are averages; large individual differences occurred.

7. Conclusions.

A methodology for investigating empirically how the readability and modifiability of computer programs can be expressed in terms of programming style characteristics has been established.

Elements of the methodology are: Human subjects; high level language programs; evaluable programming style characteristics; human measures of program readability and modifiability; and a statistical selection procedure. The methodology yields a formula expressing the readability and modifiability as functions of evaluable programming style characteristics.

In an experiment making use of the methodology a formula for the readability was found. It consisted of the following style characteristics:

- Extensiveness of comments,
- Extensiveness of blanks in the left margin,
- Extensiveness of blank lines,
- Average length of the variable names,
- Average number of arithmetic operators per 100 lines,
- Average number of **goto**-statements per label.

These characteristics are intuitively acceptable and they are in accordance with the generally agreed opinion on programming style.

The result shows that such a vague concept as readability of computer programs as graded by computer science experts can be expressed quite precisely in terms of computer-evaluable programming style characteristics.

The outcome of a further experiment, which was methodologically well founded, was a bit disappointing; the readability was shown to be expressed solely by the programming style characteristic use of comments. The meagre result in this experiment is probably due to a very small variation in the programming style applied in the programs. Thus, the second experiment does not contradict the result found in the first experiment. Rather, it indicates that the general validity of the formula found in experiment 1 needs further investigation.

In the second experiment, two measures of readability were applied, namely self-evaluation and performance in answering questions about the program. These measures were found to be highly correlated.

APPENDIX I

Details of the formula found in experiment 1.

The formula was

$$R' = 5.7 - 0.061*\mathrm{C1} - 0.047*\mathrm{C2} - 0.023*\mathrm{C3} - 0.15*\mathrm{C4} + 0.0065*\mathrm{C5} + 0.21*\mathrm{C6}$$

The coefficients are of differing magnitude because the C-figures are not normalized. The values of the formula when actual C-figures are inserted are interpreted on the 1 to 7 scale, where 1 is excellent and 7 is very poor. Examples of C-figures and contributions to the formula value are given in table 1 below.

Next is given the definition of the six characteristics.

C1: Extensiveness of comments.
The total number of characters in comments divided by the total number of characters (blanks and non-blank) in the program, multiplied by 100.

C2: Extensiveness of blanks in the left margin.
The total number of blank characters in the left margin divided by the total number of characters (blank and non-blank) in the program, multiplied by 100.

C3: Extensiveness of blank lines.
The total number of blank lines divided by the total number of lines (blank and non-blank) in the program, multiplied by 100.

C4: Average length of the variable names.
The total number of characters in all occurrences of variable names divided by the number of occurrences of variable names, in characters.

C5: Average number of arithmetic operators per 100 lines.
The total number of arithmetic operators (+, −, *, /, and //) divided by the total number of lines in the program, multiplied by 100.

C6: Average number of **goto**-statements per label.
The total number of **goto**-statements divided by the total number of labels in the program.

Table 1.
C-figures and contributions to the formula value from the 48 *programs in experiment* 1.

Contribution	C-figure values			Formula contribution		
	min.	ave.	max.	min.	ave.	max.
C1 Comments	0	7.2	34	−0	−0.44	−2.1
C2 Left margin blanks	0	16	52	−0	−0.75	−2.4
C3 Blank lines	0	13	41	−0	−0.30	−0.94
C4 Variable names	1.4	3.6	7.5	−0.21	−0.54	−1.1
C5 Arith. operators	6	48	211	+0.039	+0.31	+1.4
C6 **goto**-statements	0	1.6	4.0	+0	+0.34	+0.84

APPENDIX II

Validity of the statistical approximation in experiment 1.

A presumption for the application of the statistical selection procedure is that the deviations $R - R'$, divided by the residual variance of the regression equation, are distributed normally with mean 0 and variance 1. A chi-square test of the residual variance and a t-test of the mean did not invalidate this presumption. Consequently the deviations between the graded readability and the formula values are distributed normally with mean 0 and variance 0.16.

The deviation between the regression values and the residual variance was tested with an F-test and were found to be significant at the 0.1% level.

REFERENCES

1. I. Andersen, I. Carstensen, H. Friis, L. Fischer, A. H. Jørgensen: *PLIX – programlæsbarhedsindex.* Rapport 75/1, (1975), Datalogisk Institut, Københavns Universitet.
2. C. H. Björnsson: *Læsbarhed.* Gad, København (1971).
3. B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. J. MacLeod, M. J. Merritt: *Characteristics of software quality.* TRW Systems Group, One Space Park, Redondo Beach, California (1973).
4. B. W. Boehm: *Software and its impacts: A quantitative assessment.* Datamation, vol. 19, no. 5, (1973) pp. 48–59.
5. B. Curtis, S. B. Sheppard, P. Milliman, M. A. Borst, T. Love: *Measuring the psychological complexity of software maintenance tasks with the Halstead and McCabe metrics.* IEEE Trans. on Software Engineering, vol. SE-5, no. 2, (1979) pp. 96–104.
6. A. Fitzsimmons and T. Love: *A review and evaluation of software science.* Computing Surveys, vol. 10, no. 1 (1978), pp. 3–18.
7. M. H. Halstead: *Advances in software science.* Advances in Computers, vol. 18, (1979) pp. 119–172. Academic Press, New York.
8. B. W. Kernighan and P. J. Plauger: *The Elements oj Programming Style.* McGraw-Hill, New York (1974).
9. Klare: *Measurement of Readability.* Iowa University Press (1963).
10. D. E. Knuth: *The Art of Computer Programming,* vol. 1. Addison-Wesley, New York (1973).
11. D. D. McCracken and G. M. Weinberg: *How to write a readable FORTRAN program.* Datamation, vol. 18, no. 10 (1972), pp. 73–77.
12. K. V. Roberts: *The readability of computer programs.* Computer Bulletin, vol. 10, no. 4 (1967), pp. 17–24.
13. L. Weissman: *A methodology for studying the psychological complexity of computer programs.* Technical Report CSRG-37, University of Toronto (1974).
14. M. V. Zelkowitz: *Perspectives on software engineering.* Computing Surveys, vol. 10, no. 2 (1978), pp. 197–216.
15. J. L. Elshoff: *An analysis of some commercial PL/I programs.* IEEE. Trans. on Software Engineering, vol. SE-2, no. 2, (1976) pp. 113–120.

INSTITUTE OF DATALOGY
UNIVERSITY OF COPENHAGEN
SIGURDSGADE 41
DK-2200 COPENHAGEN N