STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLV**, Number 2, 2000

ON PROGRAMMING STYLE – PROGRAM CORRECTNESS RELATION

M. FRENŢIU

Abstract. There is little empirical information about the relation between the quality of the programs and the style of the programmer. One experiment in this direction is presented in this paper.

It is considered that the style of the programmer affects his efficiency, and the correctness of his programs. To sustain this hyphotesis, the papers at a written examination were analysed and the conclusions are presented.

Key words: programming methodology, style, quality, software metrics, education

1. Introduction

The need to measure various attributes met in software engineering is underlined in [4]. Certainly, it is very important to asess the time needed to realize a software project, or to evaluate the quality, maintainability, reliability, or usability of a program, or the productivity of a programmer. Also, we think it is very useful to assess the effect of programming style on the above mentioned attributes.

Is there a relation between the style of a programmer and the quality of his work? We need some definitions of the concepts we use. What is a style? In [2] the word style is considered to be the general way in which something is done, "the general attitudes and usual ways of behaving", "the style of a product is its design", and the style of writing is "the choice of words and the way in which sentences and paragraphs are structured". It is somebody's manner of speaking, acting, writing, for expressing his thought.

In Software Engineering when we define the style we can think only to how the programs look [15, 17]. Therefore, in a narrow sens we have:

Definition 1. Programming Style consists of the ways in which the programer writes programs easy to read, and easy to understand, the ways in which these qualities are achieved.

2000 *Mathematics Subject Classification.* 68N30.

1998 *CR Categories and Descriptors.* D.2.3 [**Software**] : Software Engineering – *Coding Tools and Techniques* D.2.7 [**Software**] : Software Engineering – *Distirbution, Maintenance and Enhancements* .

Readability is considered to be the main attribute of style [11]. And readability depends on indentation, good names, and on the comments present in the texts of the programs.

We will briefly describe the elements of style. But we must say that they may differ from person to person, although each programmer must have and think to his own style. Citing Gries, "Whatever conventions you use, use them consistently" [10].

Therefore, the elements of style (in a narrow sens) are:

- Comments;
- Text Formatting (Indentation, White Spaces);
- Good Names for Entities of the Program.

Comments are very important for internal documentation. Every program ought to have documentation in it. Comments must be used:

- to state the specification of the problem solved by the program, to precise the author, date, and other useful information for the reader;
- to show the purpose of each variable;
- to explain what a procedure does: to show the specification of the procedure, and the meaning of the parameters;
- to write the loop invariants in those places of the program where they hold;
- to explain the conditions in which some parts of the programs are reached, and the role of these parts;
- and to transmit other useful information to the reader.

Indentation rules are used to enrich the clarity of the program. There are a variety of suggestions for such rules. Gries suggests the following indentation rules [10]:

- succesive short commands can be written on the same line provided that, logically, they belong together;
- commands of a sequence that appear on succesive lines should begin in the same column;
- subcommands of a comand must be indented 3 or 4 spaces from the column where the command begins;
- the pre- and postcondition of a command should begin in the same column as the command;
- a loop should be preceded by an invariant and a bound function; these should begin in the same column as the beginning of the loop.

Then, the names of variables, functions, types, contribute to the clarity of programs [13, 14]. Here are some rules in this direction:

- choose meaningful names for all entities;

- do not use a single name for two variables (i.e. the same name has one meaning in a part of the program, and a second meaning in another part);
- define your variables before you use them, and then be sure to adhere to the definitions;
- when a name is composed of two words, start the second one with a capital letter.

But everybody admits that a badly conceived program remains a bad program. It may be well documented through comments, it may be nice indented, and it may use meningful names, if it is not correct it is useless. And, also, if it cannot be maintained, it is not of a good quality.

Therefore, we consider a larger aspect of programming style.

Definition 2. Programming Style consists of all means taken by the programmer in his activity for producing reliabile products easy to read, and easy to understand, the ways in which these qualities are achieved [13, 10], [8, page 137].

This definition sets in the main frame the way in which the programmer respects the general accepted rules for good programming. It starts with the specification of the program, with the way in which the design is done, with the clarity of documentation for all stages of work. As Floyd [5] said, we must permanently fight to acquire better programming methods for producing correct and easily maintainable programs. The style depends on how the general accepted rules for good programming are respected. And there are many books and papers that contain such rules [13, 14, 10, 6, 7, 8, 9].

2. The experiment

The opportunity to do this study was offered by the Graduate Licence Examination held in June 2001. 94 students took part in this exam (for B.Sc. in Computer Science). The subject consisted of two parts: theoretics (the first two subjects), and programming (next two subjects). Here are the subjects:

(1) Sorting. Quicksort;
(2) Merging;
(3) Specify, design, and implement an Abstract Data Type ***SET***;
(4) Write a program which prints the longest sequence of consecutive primes from a given sequence of natural numbers.

The results are given in Table 2. Since the results for the theoretical subjects are not directly used in the analysis, only the total number of points (for all four subjects) are given in the fourth column (denoted by T). The P3, and P4 columns contain the points given for the subject (3), and (4), respectively. Then, the elements of style are measured by grades from 0 to 10. The grade 0 is given when the corresponding attribute is not present at all, and 10 if it is considered perfectly.

No	P3	P4	T	P3	P4	No	P3	P4	T	P3	P4
				I C N G	I C N G					I C N G	I C N G
1	23	22	73	8 0 6 7	8 0 8 5	48	24	21	80	6 5 5 6	5 4 4 5
2	19	18	66	7 2 7 5	6 3 4 3	49	18	23	75	4 0 4 4	5 0 3 5
3	25	8	73	8 1 3 6	7 0 8 2	50	25	25	90	9 6 6 9	8 5 5 8
4	15	18	57	7 3 5 7	8 2 6 2	51	24	19	79	7 0 4 6	8 8 5 7
5	22	15	70	7 0 7 4	9 0 5 4	52	20	18	72	7 2 6 7	6 2 4 6
6	15	19	69	8 1 6 5	6 6 8 7	53	22	23	85	6 1 5 6	7 2 3 7
7	8	19	57	7 0 5 2	7 1 8 4	54	11	18	64	4 1 4 4	5 3 2 4
8	15	20	71	5 1 5 2	8 3 7 6	55	23	20	78	5 4 5 6	5 1 4 5
9	22	25	81	7 5 5 5	8 5 8 6	56	20	21	75	3 1 2 3	4 3 4 4
10	25	22	79	7 6 5 6	7 8 6 8	57	20	0	52	4 0 3 4	0 0 0 0
11	25	20	84	8 4 6 7	7 0 4 6	58	10	20	52	5 4 4 5	6 1 3 6
12	22	19	78	8 0 4 8	8 2 5 5	59	25	22	87	6 7 6 8	6 0 4 6
13	20	20	77	6 6 5 8	4 8 7 7	60	20	5	65	7 5 5 7	6 4 3 5
14	9	5	42	0 0 0 0	2 0 2 0	61	6	10	42	2 2 1 2	5 0 5 5
15	18	17	75	6 0 3 5	5 4 3 4	62	18	15	64	5 0 3 4	4 1 3 3
16	22	20	82	5 0 4 5	5 1 4 5	63	23	20	82	4 2 5 5	5 3 4 5
17	0	2	28	0 0 0 0	2 0 0 0	64	20	20	80	6 0 5 6	6 7 5 7
18	23	22	80	6 0 4 6	5 0 4 5	65	24	21	82	7 8 7 8	7 7 6 7
19	24	20	82	9 7 7 8	8 6 6 8	66	20	18	74	6 0 4 5	6 9 6 8
20	13	17	63	6 1 3 3	7 0 4 7	67	21	20	73	6 0 4 5	7 3 4 6
21	15	15	57	6 2 3 5	6 3 5 5	68	9	21	65	2 0 3 2	3 4 5 4
22	17	18	71	7 7 6 7	7 6 5 6	69	14	20	67	3 0 2 2	5 0 2 4
23	5	13	42	3 0 1 1	2 4 2 2	70	25	15	78	6 3 5 6	5 2 3 4
24	12	20	66	5 4 5 4	5 6 6 5	71	12	10	55	2 0 3 3	4 0 2 2
25	25	23	88	6 8 6 9	7 7 7 8	72	20	19	76	4 1 3 3	4 2 3 3
26	10	19	49	2 0 1 3	6 1 3 6	73	23	15	76	7 3 5 6	4 3 4 4
27	0	0	15	0 0 0 0	0 0 0 0	74	9	15	57	4 3 3 4	5 3 4 4
28	13	15	61	2 7 4 4	7 0 4 4	75	20	20	72	5 3 4 5	5 2 4 5
29	18	21	78	6 4 5 5	7 5 5 7	76	22	13	75	6 7 5 6	5 5 3 4
30	24	18	81	8 6 6 7	5 5 3 5	77	22	25	76	7 3 6 7	8 9 5 6
31	24	21	85	7 1 6 6	5 4 4 5	78	17	22	79	4 0 2 3	6 6 4 6
32	25	23	85	7 6 6 8	7 6 5 8	79	13	18	64	4 0 3 3	7 5 5 7
33	24	15	74	6 3 6 7	5 3 6 5	80	23	19	82	6 0 4 5	5 4 5 5
34	23	22	65	5 7 7 7	6 0 5 6	81	25	23	85	5 0 4 5	6 1 4 5
35	18	20	77	6 2 6 6	7 5 4 5	82	22	21	82	2 4 5 5	5 3 5 5
36	24	20	79	9 9 8 9	5 6 6 6	83	18	22	73	6 4 6 6	6 4 5 6
37	19	18	76	6 5 4 5	3 4 3 5	84	11	18	68	3 2 3 3	4 0 2 3
38	20	4	53	5 1 5 5	2 0 1 1	85	8	18	46	4 2 3 3	5 2 3 4
39	20	18	66	8 5 4 7	5 7 5 5	86	12	19	59	5 0 4 4	5 0 3 5
40	22	19	79	7 0 5 6	6 0 4 4	87	25	20	83	6 0 5 6	6 4 3 5
41	20	22	65	6 0 4 6	7 1 4 7	88	9	18	54	4 0 3 3	5 1 4 4
42	11	3	46	2 1 1 1	1 0 1 1	89	25	20	82	7 0 6 6	6 6 5 6
43	9	11	30	2 0 1 2	2 2 1 2	90	23	22	82	8 7 6 8	8 7 6 8
44	21	20	61	3 0 4 4	3 5 3 3	91	24	20	79	7 5 6 7	7 6 6 7
45	22	13	72	5 2 3 5	4 1 3 4	92	17	21	72	6 5 5 6	6 6 6 6
46	24	21	80	6 5 5 6	6 4 4 5	93	25	23	88	7 4 5 7	6 7 5 7
47	21	20	76	5 5 4 5	4 5 3 4	94	24	15	77	5 3 4 4	5 4 3 4

TABLE 1. Primary data for attributes of style

Line no.	X1	X2	Y	C(X1,Y)	C(X2,Y)
1	P3	T	P3/I	0.70	0.67
2	P3	T	P3/C	0.37	0.37
3	P3	T	P3/N	0.70	0.70
4	P3	T	P3/G	0.79	0.73
5	P4	T	P4/I	0.64	0.57
6	P4	T	P4/C	0.42	0.45
7	P4	T	P4/N	0.56	0.50
8	P4	T	P4/G	0.76	0.66
9	P3	P4	P3/I + P4/I	0.63	0.61
10	P3	P4	P3/C + P4/C	0.43	0.40
11	P3	P4	P3/N + P4/N	0.60	0.63
12	P3	P4	P3/G + P4/G	0.72	0.69
13	P3/I + P4/I	P3/N + P4/N	T	0.68	0.67
14	P3/C + P4/C	P3/G + P4/G	T	0.48	0.77

TABLE 2. The correlation coefficients for various attributes

For example, in column P4/C the grades for comments in the program corresponding to the problem 4 are given. The minimum amount of comments required to obtain 10 is formed from the statement of the problem, the precondition and the postcondition for each procedure, the meaning of each variable, and, in some important places, the situation in which that part of the procedure is reached.

The papers were independently analysed by two teachers, and the points were given for the global correctness of programs. It was similar to an inspection of the program, such that the given number of points reflects the measure of program correctness (columns P3, and P4, respectively), and acquired knowledges (column T). Although everybody knew that the correctness of programs is important, and this was watched carefully, the students also knew that the teachers look at their style of programming.

The columns marked by (C), (I), and (N) contain the points for the measures in which the rules connected to comments, indentation, and good names are respected, as explained above. The column (G) contains the points (from 0 to 10) for the way in which all the general accepted programming rules are respected, starting with the specifications of the problem and of all used modules, analysing the design and the documentation of all activities. The points contained in the columns (C), (I), (N), and (G) were given by the author of this paper.

3. Conclusions

It is known that the measure of linear dependence between two characteristics is given by the correlation coefficient of these characteristics. Therefore, the correlation coefficients for various attributes were computed. They are given in Table 3, where C(X,Y) denotes the correlation coeficient of the attributes X and Y.

As we expected, these coefficients are positive, and show that there is a strong dependence between the corresponding attributes. This confirms the idea that programming style has an important impact on program correctness. Also, it was expected that the largest coefficients are between the correctness and the way in which the general rules are satisfied (column G).

Moreover, we must observe that these correlation coeficients are stable for both problems, i.e. $C(P3, A)$ is closed to $C(P4, A)$ for all attributes $A \in \{I, C, N, G\}$. This confirms that the students have been convinced of the necessity to respect the above mentioned rules, and have acquired an acceptable programming style.

Nevertheless, we must observe some anomalies, and, for educational purposes, take some measure to eliminate them. First, we can observe that the smallest coefficients correspond to the column C: $C(P3, P3/C) = 0.37$ is the smallest of all. Therefore, students do not like writing comments. In this direction we must observe that there are 56 programs (from $188 = 2 \times 94$) that have no comments at all!

This is in contrast with the case of the other attributes, where the presence of zeros is an exception, only the lines with $P3 = 0$, or $P4 = 0$, having the grades for these attributes equal to zero.

The indentation rules are much better respected. There is one more reason for this. At all lectures, when the teachers write algorithms or code, they respect these rules in all lines. But only sometimes they write comments.

We may conclude that a good programming style and a correct programming habit must be taught in parrallel. As can be seen [6, 7, 8, 9] there were many important programming style rules in my lectures, but they were not compulsory, as is the case of many universities [1, 3, 11, 12, 15, 17]. As a consequence of this analysis, I think such rules must become compulsory.

References

[1] Adams, David, and Dan Beckett, Programming Style, http://www.island-data.com/downloads/ papers/programmingstyle.html, 2001.

[2] BBC English Dictionary, HarperCollins Publishers, 1993.

[3] Craig E.Wills, Programming Assignments, http://www.cs.wpi.edu/∼cew/courses/2005/style/style.html

[4] Fenton, N.E., Software Metrics. A Rigorous Approach, Int. Thompson Computer Press, London, 1995.

[5] Floyd, R.W., The Paradigms of Programming, Comm.ACM, 22(1979),8, 455-460.

[6] Frenţiu M., B.Prv, Programming Proverbs Revisited, Studia Univ. Babeş- Bolyai, Mathematica, XXXVIII (1993), 3, 49-58.

[7] Frenţiu M., On Program Correctness and Teaching Programming, Computer Science Journal of Moldova, vol.5 (1997), no.3, pp.250-260.

[8] Frenţiu M., Lazar I.(Romanian), Programming Fundamentals. Algorithms Design, Ed.Univ."Petru-Maior", Târgu-Mureş, 2000.

[9] Frentiu M., Verifying Program Correctness (Romanian), Ed.Univ."Petru-Maior", Târgu-Mureş, 2001.

[10] Gries, D., The Science of Programming, Springer Verlag, Berlin, 1981.
[11] Haahr, P., A Programming Style for Java, http://www.webcom.com/~haahr/essays/java-style
[12] Keith Gabryelski, Wildfire C++ Programming Style, http://www.cs.umd.edu/users/cml/cstyle
[13] Kernigham, Brian W., and P.J.Plauger, The Elements of Programming Style, McGraw-Hill Book Company, New York, 1974.
[14] Ledgard H.F., Programming Proverbs for Fortran Programers, Hayden Book Company, Inc., New Jersey, 1975.
[15] McCann, Toward Developing Good Programming Style, http://www.comsc.ucok.edu/~mccann/style_p.html
[16] Meyer, B., Object Oriented Software Construction, Prentice Hall, Englewood Cliffs, 1988.
[17] David R.Tribble, Notes About Programming Style, http://www.flash.net/~dtribble/src/sys/style.htm, 1998-04-16

Department of Computer Science, "Babeş-Bolyai" University, 1, M. Kogălniceanu, RO-3400 Cluj-Napoca, Romania
E-mail address: mfrentiu@cs.ubbcluj.ro