Text and source readability – A step to cognition

Conference Paper *in* AIP Conference Proceedings · March 2021

DOI: 10.1063/5.0042068

CITATIONS
0

READS
93

4 authors:

Galina Momcheva
Free Varna University
23 PUBLICATIONS **10** CITATIONS
SEE PROFILE

Veselina Spasova
Free Varna University
9 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Antonina Ivanova
Free Varna University
7 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Milen Zhelyazkov
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Text and source readability - A step to cognition

Cite as: AIP Conference Proceedings **2333**, 070013 (2021); https://doi.org/10.1063/5.0042068
Published Online: 08 March 2021

Galina Momcheva, Veselina Spasova, Antonina Ivanova, and Milen Zhelyazkov

Text and Source Readability - A Step to Cognition

Galina Momcheva, [a] Veselina Spasova [b], Antonina Ivanova [c] and Milen Zhelyazkov [d]

Varna Free University "Chernorizets Hrabar, Yanko Slavchev 84, Chayka Resort, 9007 Varna, Bulgaria

[a] galina.momcheva@vfu.bg
[b] Corresponding author: vspasova@vfu.bg
[c] antonina.ivanova@vfu.bg
[d] 172831001@vfu.bg

Abstract. The article aims to explore the relationship between readability of texts from instructions (assignments texts, technical documentation, law/regulation texts) and the corresponding source code (task solution) from a particular education resource for programming on the topic with neural/positive sentiment 'identity documents'. The article examines two types of studies: one of applying software metrics and second one from empirical research of text understanding for a target group of 15 years old students (on texts and programming source code). The results are a strong foundation for continuing research in order to receive objective methodology for readability for the people of this age. This has both research and also publishing companies' business added value as objective recommendations for the authors of the textbooks and also for teachers in Computer Science and languages for general teaching practices.

INTRODUCTION

Readability and comprehension are of the focus for years [3]. The understanding of texts is also reflected in PISA international tests for functional literacy. It measures the students' abilities to understand the material studied, deal with applied problems based on learning, and to do problem solving.

Obviously, for most newcomers in learning programming, studying a programming language is similar to studying a foreign language. However, to study Computer Science is something more than to study the language. According to modern paradigms, it also has to do computational thinking. It is not a secret for educators in this area that if a person speaks/knows the English language then learning a programming language is much easier. The huge set of new research methods in Computer Science in general and in Data Science in particular as natural language processing and biometric-type of research (in Neuroscience) gives new power for readability research. [6] Soon, the results in this area will influence the teaching and learning natural languages, too. Now we have the chance to do this research to improve the skills for a new generation of learners.

The new methodology of learning Computer Science, as this of the book Computer Science 8th grade [7], is a risky mission because of colliding new methodological practices with old executives (people who have some stereotypes – teachers, students, parents).

Generally, the new methodology is embedded in the curricular (MoE) for 8th grade in specialized schools that is to apply a software-first approach in learning Computer Science. Authors of the book [7] suggest one more new approach in subject Computer Science for students in specialized STEM-like schools in Bulgaria that is the multi-language programming methodology. The reason why this is done is that the authors recommend to use one programming language during labs (practical sessions) but to visualize different approaches of modern programming languages and to compare/show analogy, and also to allow the development of these students who already have the experience, to use the tetrad (C#, Java, JavaScript, Python).

Applications of Mathematics in Engineering and Economics (AMEE'20)
AIP Conf. Proc. 2333, 070013-1–070013-6; https://doi.org/10.1063/5.0042068
Published by AIP Publishing. 978-0-7354-4077-7/$30.00

At the same time from a software engineering point of view, the process of code reading is an important skill for every programmer in particular and every member in a software team (e.g. code review practical sessions). The coding skills are as important as skills in reading and understanding code written before from somebody else.

Up to now, there is not any metric (readability) that can evaluate readability and comprehension in Bulgarian texts by age. As an author of textbooks for schools for 20 years, one of the authors of this article meets a lot of misunderstandings with editors from publishing companies. In many cases, they (the editors) prefer to force authors to simplify texts unnecessarily. There are not (up to now) corpora (from specific subject areas like Computer Science or for a specific age, etc.). This research is the first one from a chain of long-term research that also will be scaled up to other media (not only text) using UI/UX practices for evaluation of readability of software apps, videos, mobile apps, etc. We cannot miss mentioning that we are focused on a future software tool that combines different methods and approaches for the sake of truth in readability.

RELATED WORK

Many people have been working on the topic of readability, comprehension, understandability, and cognition since the 1960s for both written texts and for programming sources. Some of the first metrics concern word difficulty and number of sentences. Research on the readability of technical texts was taken in the early 1970s. One of the most popular formulas is that of Rudolf Flesh.

A formula that received acceptance is the Fog Index. Another technique for evaluating readability is the 'cloze' technique. The technique requires the deletion of words from a passage and then scoring the reader on the percentage of words, he can correctly replace [1].

Software metrics for evaluation of the complexity of the code typically use parameters such as number of objects, number of methods, number of comments, number of variables, number of lines code, number of operators, etc. As a major metric in most sources, considering the issue of code complexity is considered McCabe's metric [5]. Although some programmers in criticize the use of McCabe's metrics [2,9] because in some cases the assessment using the method based mainly on the number of branches in the program differs from the subjective evaluation of the code complexity given by experienced programmers, it has found its place in practice and is present, in both, in the developed multilingual libraries for estimating complexity (devMetrics 1.0: http:/www.free-software), Eclipse metric plugin [8], as well as in well-established software analysis tools such as Understand [9] and Sonarcube [10].

In order to overcome the drawbacks of using cyclic complexity in terms of subjectively assessing the complexity of understanding of otherwise complicated testing, due to a large number of possible paths, program constructs from Sonar Source offer the use of a modified version that reflects the complexity of the code comprehension process, called cognitive complexity [11]. However, it is virtually a slight modification of McCabe's metric, which reduces the complexity of case-switch constructs that in the original version lead to high values increment when structures that break the flow are nested and ignore shorthand structures that readably condense multiple lines of code into one.

Very interesting are methods and results published from Jbara and Feitelstone [4]. They describe that usually used metrics for software complexity are ignoring the sources' global structure on account of the structure of the modules and demonstrate that code regularity – where the same structures are repeated time after time – may significantly reduce complexity because once one figure is out the basic repeated element it is easier to understand additional instances using the volume of the zipping code.

Some new ideas (formulaic approaches), related to the readability of texts, considering some cognition aspects are presenting from Ziafar and Namaziandost [9].

This article gives the idea to combine well-known software complexity metrics with other methods to obtain results that are more related to the environment and the particular case (like educational and learning processes in our case).

METHODOLOGY OF RESEARCH

Software complexity metrics mainly reflect the effort required to write the code, believing that the complexity of understanding is almost equal to the complexity of writing. This, however, applies when the code development and reading are done by experienced programmers. When it comes to assessing the complexity of the code and its understanding of the beginner (trainee) programmers who are not yet familiar with the constructions of the language, we can assume that the parameters described above are insufficient for the evaluation and is good with the traditional software metrics to add metrics for text comprehension, because in practice for the learner any code is the text written in a little-known language. No less important in the training process than the sample codes are the conditions and explanations of the tasks, and how clear and understandable.

Determining how novices think about style and readability is crucial for teaching programming effectively and for designing tools to support novice programmers [8].

In this article, an interesting approach is provided to explore simultaneously both reading skills and programming readability skills of students in 8th grade in Bulgaria studying Computer Science in an obligatory subject Computer Science in schools with a specific profile. The empirical research is done in three of High schools in Mathematics in Bulgaria. A task assignment (on page 136 [6]) and its solution as a programming source from the textbook Computer Science (grade 8) are analyzed for readability with well-known metrics and an examination of two types of tasks is provided with students.

The aim is to figure out relations between the text of the assignment and the source code of a program.

The texts in Test A and the source in Test B are on the topic of Identity Document (a card, driving licenses, etc.) – in what age what type of documents are allowed to have in Bulgaria. In Test A (Text Readability) questions are for readability only. They explore if students can find or understand facts. Some of the texts are quotes from laws/national regulations. In Test B (Source Code Readability/Knowledge in C#) questions could be divided into two subgroups: for understanding (readability) and for knowledge of programming.

Both tests for readability consist of questions about finding facts in the text and raising decisions from text statements. In one of the schools (10 % of responders), the students were asked to work in pairs.

For software metrics tests we used two versions of the code for the same problem – the first is optimal from the professional programmer point of view (Source 1) and the second was written with the purpose to use as more as possible elementary constructions at the expense of efficiency (Source 2).

EXPERIMENTS AND RESULTS

The target group of responders consists of 127 students from 8th grades in three schools in the country who answer all of the raised questions individually. In addition, there are 10 students working on tests A and B in pairs. The empirical research with students was done in February - March 2018.

Some experiments for relationships between two tests are done in the following way.

The outliers from the second test are skipped from the analysis number of respondents in this part of experiment 110.

The assigned points to questions of both Test 1 and Test 2 are:

Test 1 (3 questions, total number of points 12)

Test 2 (10 questions, total number of points 48)

Test 2.(R) (2 questions for readability, applied in the programming

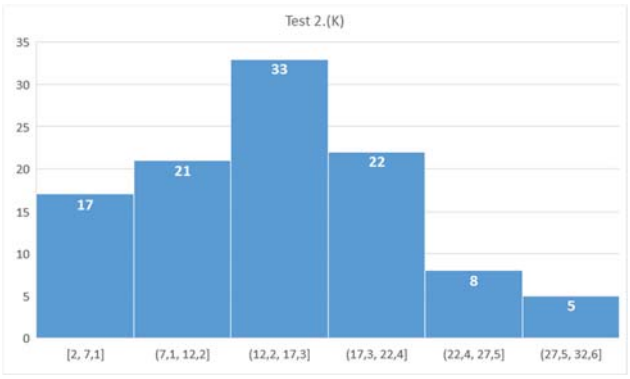

FIGURE 1. The distribution of data in the Test 2.(K)

The average results (in points) are 14.75. The median is 13.5. The mode is 13.
This experiment does not aim to evaluate the level of knowledge.

The control group for pair testing shows better results than individual respondents.

Related to Jbara and Feitelstone [4] methodology we received for source file (Source 1) the results in Table 1, applying the compression algorithms: LZMA, GZIP, and BICOM.

TABLE 1. Source file volumes

	Initial file size	**LZMA size**	**GZIP size**	**BICOM size**
In bytes	1894	723	605	526
Percent of zipping		38 %	32 %	28 %

As for the software metrics experiments, the Python package Textstat 0.4.1 is used to calculate statistics from text to determine readability, complexity, and grade level of a particular corpus. After calculating the values of metrics in this library only, the Coleman Liau index is taken into consideration in this article and the results are represented in Table 2.

TABLE 2. Coleman Liau index for Source 1 and Source 2

	Source 1 with Texts in Bulgarian in Printing Method	**Source 1 with Texts in English in Printing Method**	**Source 2**
Coleman_liau_index	33.17	36.96	33.29

Two versions of source code for the same program written on C# are compared, and in the second version, the complex combination conditions are converted to more simple, combined with more if-else statements. This is also evident from the length of the code, which is twice as much compared to the original version, both for the declarations and for the statements (without declaration ones – executive statements). At the same time, the calculated complexity indexes are essentially similar in value, practically the same. However, the values of the Fog Index (text complexity metric) that is applied to the source of the two versions of the program are lower, which means that especially for novice programmers, this code, although twice as long, is easier to understand, especially for young programmers (Table 3).

TABLE 3. Calculated metrics for Source 1 and Source 2

Metric	Source 1	Source 2
AvgCyclomatic	6	7
CountLine	50	102
CountLineCode	47	93
CountLineCodeDecl	5	8
CountLineCodeExe	25	59
MaxCyclomatic	11	13
MaxCyclomaticModified	11	13
MaxNesting	2	3
Fog Index	8.84	8.44

CONCLUSION

Where the evaluation is done for English speaking countries persons we may suggest that there is some connection between linguistics term ‘“dependent clause fragments” and conditionals in programming, but evaluating Bulgarian students in readability with English texts is testing the triad of understanding in a foreign language, too that is not the focus of this research. This will continue in the experiment continuation.

Future research is focused on programming practices in pair-programming practices, cross-language support, natural language dependences, brain research, team’s competence, and understanding.

ACKNOWLEDGMENTS

Our thanks to the teachers of the students from 8th grade from High Schools in Mathematics in Bulgaria: Julia Dimitrova (Varna), Milen Mavrodiev (Varna), Georgi Nikov (Varna), Zornitsa Dzhenkova (Gabrovo), Galya Nedelcheva (Gabrovo), Galina Bobeva (Rousse), who helped us with the organization of this study.

REFERENCES

1. Booher, H. R. Survey of research on readability ot technical publications. National Technical Information Service Springfield, 1971
2. Crossley, S. A., Greenfield, J., & McNamara, D. (2008). Assessing text readability using cognitively based indices. TESOL Quarterly, 42(3), 475-493, Wiley. 30 December 2011
3. Fenton N., J. Bieman, Software Metrics. A Rigorous and Practical Approach, 3-th edition, Taylor & Francis Group, 2015
4. Jbara, A., Feitelson Dror, G. On the Effect of Code Regularity on Comprehension.
5. McCabe, A Complexity Measure, IEEE Transactions on Software Engineering, 1976
6. Marter T., P. Babucke, Ph. Lembken, and S. Hanenberg, Lightweight programming experiments without programmers and programs: an example study on the effect of similarity and number of object identifiers on the readability of source code using natural texts. In Proceedings of the 2016 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! 2016). Association for Computing Machinery, New York, NY, USA, 2016

7. Momcheva, G., T. Glushkova, R. Marinova. Computer Science (8th grade), Anubis-Bulvest 2000, 2017 (in Bulgarian)
8. Wiese, Eliane S., A. N. Rafferty, A. Fox. Linking code readability, structure, and comprehension among novices: it's complicated. In Proceedings of the 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '19). IEEE Press, 84–94, 2019
9. Ziafar M., E. Namaziandost, "A Formulaic Approach to Propositional Density and Readability, International Journal of Innovation and Research in Educational Sciences, Vol. 6, Issue 6, 2020
10. https://www.cqse.eu/en/blog/mccabe-cyclomatic-complexity (april 2020)
11. http://eclipse-metrics.sourceforge.net (april 2020)
12. https://scitools.com (april 2020)
13. https://www.sonarqube.org (april 2020)
14. Campbell,G. Ann. Cognitive complexity. A new way of measuring understandability SonarSource SA, 2018