International Journal For Technological Research In Engineering
Volume 7, Issue 3, November-2019 *ISSN (Online): 2347 - 4718*

A FACTUAL STUDY OF CODE READABILITY AND SOFTWARE COMPLEXITY

Deepa Dhabhai
(Assistant Professor - CS)
Shekhawati Institute of Engineering and Technology, Rajasthan, India

Abstract: Important software quality metrics such that Code readability and software complexity that impact other software metrics such as maintainability, reusability, portability and reliability. This paper presents a Factual study of the relationships between code readability and program complexity. This analysis includes many readability and complexity metrics. Our study factually confirms the existing wisdom that readability and complexity are negatively correlated.

Keywords: readability, complexity, metrics, correlation, feature ranking, Factual study

I. INTRODUCTION

Code Readability is defined as a human judgment as to how much source code is understandable and easy to read. It has been traditionally considered as an important software quality metric as it has a great influence on software maintenance. Typically, maintainability phase consumes 40% to 80% of the total life-cycle cost of software [1]. Aggarwal et al. [2] claim that source code readability and documentation readability are crucial for maintainability of a software. Some researchers identify reading code as a key activity in software maintenance, and also recognize it as the most time-consuming activity among all the maintenance activities [3], [4], [5]. Software complexity is defined in IEEE glossary standards as: "the degree to which a system or component has a design or implementation that is difficult to understand and verify" [12]. The complexity of code can be affected by many factors, such as: lines of code, total number of operators and operands, coupling between objects, and number of control flows [13]. These factors are used in software metrics for measurement and approximate quantification of software complexity. Software complexity is considered as an "essential" property of the software since it reflects the complexity of the real-worlds problem a software deals with [6]. On the other hand, code readability is considered as an "accidental property", not an essential one, as it is not determined by the problem space, and can largely be controlled by the software engineers. While software complexity metrics measure the size of classes and methods, coupling, and interdependencies between modules, the code readability considers local and line-byline factors such as: names of identifies, indentations, spaces, and length of lines of code. Software quality is a critical topic in software engineering, and thus many researchers have performed studies in this area. Code readability and software complexity have a substantial impacts on software quality. For better quality, low complexity and high readability are desired. \ Complexity may impact code readability, while low readability also may result in higher perceived complexity. Thus, readability and complexity are related. Research Questions: A proper understanding of the relationship between these two attributes (i.e., readability and complexity) is necessary. In this paper, we present an Factualstudy of the relationships between code readability and software complexity.

II. BACKGROUND AND RELATED WORK

In the next sub-sections we highlight the importance of source code lexicon (i.e., terms extracted from identifiers and comments) for software quality; in addition, we describe state-of-the-art code readability models. To the best of our knowledge, three different models have beed defined in the literature for measuring the readability of source code [7], [8], [9]. Besides estimating the readability of source code, readability models have been also used for defect prediction [7], [9]. Recently, Daka et al. [17] offered a specialized readability model for test cases, which is used to improve the readability of automatically generated test cases. A. Software Quality and Source Code Lexicon Identifiers and comments play a crucial role in program comprehension and software quality since developers express domain knowledge through the names they assign to the elements of a program (e.g., variables and methods) [10], [11], [12], [13], [14], [15], [16]. For example, Lawrie et al. [14] showed that identifiers containing full words are more understandable than identifiers composed of abbreviations. From the analysis of source code identifiers and comments it is also possible to glean the "semantics" of the source code. Consequently, identifiers and comments can be used to measure the conceptual cohesion and coupling of classes [18], [19], and to recover traceability links between documentation artifacts (e.g., requirements) and source code (e.g., [20]). While the importance of meaningful identifiers for program comprehension is quite consolidated, there is no agreement on the importance of the presence of comments for increasing code readability and understandability. While the previous studies pointed out that comments make source code more readable [21], [22], [23], the more recent studies, for instance by Buse and Weimer [7], showed that the number of commented lines is not an important factor in their readability model. However, the consistency between comments and source code has been shown to be more important than the presence of comments, for code quality. Binkley et al. [24] proposed the QALP tool for computing the textual similarity between a component comment and its code. The QALP score has been shown to correlate with

human judgements of software quality and is useful for predicting faults in modules. Specifically, the lower the consistency between identifiers and comments in a software component (e.g., a class), the higher its fault-proneness [24]. Such a result has been recently confirmed by Ibrahim et al. [25]; the authors mined the history of three large open source systems observing that when a function and its comment are updated inconsistently (e.g., the function code is modified, the comment not), the defect proneness of the function increases. Unfortunately, such a bad practice is quite common since very often developers do not update comments when they maintain source code [26], [27]. B. Structural Features as a Proxy of Readability Buse and Weimer [7] proposed the first model of software readability and provided evidence that a subjective aspect like
Buse & Weimer have proposed a code readability metric and developed a readability tool that automatically measures proposed readability metric. They selected Java code snippets and made them available to the selected human annotators for the judgement of readability of those code snippets. The results obtained from the experts were compared with results from the propose readability tool. The overall accuracy of the tool was found to be 80%. The study also showed that the readability is strongly correlated with some software quality attributes such as code changes, defect log messages and automated defect reports. However, Daryl Posnett et al. [16] argued that code readability is a subjective property and it is not persuadable to generate readability score using automated readability tool. Further, they included that readability very much depends on the information contained in the source code and thus the readability score can be calculated based on size and code entropy. Similarly, Ankit performed a review of metrics for software quality. The authors reviewed various readability metrics in the literature such as Flesh-Kincaid metric, Gunning-Fog metric, SMOG index, Automated Readability Index and Coleman-Liau Index and concludes that the choice of readability metrics depend on different employments. They mention various elements of code that improves and degrades readability, for example appropriate comments and poorly defined variables respectively. Complexity Metrics: Many software complexity metrics have been proposed in the past. Almost all the proposed complexity metrics measure complexity based on three attributes: software size, data flow, and control flow. Halstead Complexity Model [18], McCabe [19], Line of Code (LOC) [20] and Chidamber & Kemerer [21] metrics suites are all examples of proposed complexity metrics.

2.2 Analyses of Relationship

There have been few investigations into the relationship between software readability and complexity.In the study of. [7], the investigation was based on Component Based Software Engineering (CBSE).. Readability and complexity results indicate a negative correlation between both of these. Buse and Weimer proposed an approach for constructing a readability tool. Researchers investigated correlation between readability score from their proposed readability tool and cyclomatic complexity with help of Pearson product moment correlation. They found that readability is weakly correlated with complexity in the absolute sense, and it is effectively uncorrelated to complexity in relative sense. 3. Study Setup In this section, we discuss readability and complexity metrics used in this work, as well as the tools and methodology adopted in this research. 3.1 Readability Metrics Different metrics are developed to estimate the readability of code. The readability metrics used in this work are described below. A sentence difficulty is determined as words per sentence and word difficulty is that by calculating letters per words. The equation for calculating ARI is: ARI = 4.71(characters) + 0.5 (words) – 21.43 The numeric value of the ARI metric it approximates the grade level needed to comprehend the text. For example, ARI = 3 means, students in 3rd grade (ages 8-9 yrs. old) should be able to comprehend the text [28]. B. The Simple Measure of Gobbledygook (SMOG): SMOG is suggested by G Harry McLaughlin [22] in 1969. This metric evaluated the time (in years) required by any person to read the text. The equation for calculating SMOG is: SMOG = 3 + Square Root of Polysyllable Count The SMOG metric value signifies a U.S. school grade level indicating that an average student in that grade level can read the text [28].

III. CONCLUSION:

This paper define factual study of Code Readability and software complexity .Using facts of previous research work that complexity metric measures interface complexity for software components and shows how it is correlated to readability matrix

REFERENCES

[1] Dubey, S. K., & Rana, A. (2011). Assessment of maintainability metrics for object-oriented software system. ACM SIGSOFT Software Engineering Notes, 36(5):1-7.
[2] Aggarwal, K. K., Singh, Y., & Chhabra, J. K. (2002). An integrated measure of software maintainability. In Annual Proc. of Reliability and maintainability symposium, pp. 235-241.
[3] Raymond, D. R. (1991, October). Reading source code. In Proc. of the 1991 conference of the Centre for Advanced Studies on Collaborative research, pp. 3-16.
[4] Deimel Jr, L. E. (1985). The uses of program reading. ACM SIGCSE Bulletin, 17(2): 5-14.
[5] Rugaber, S. (2000). The use of domain knowledge in program understanding. Annals of Software Engineering, 9(1-2): 143-192.
[6] Brooks, F. P. (1987). No silver bullet essence and accidents of software engineering. IEEE Computer, 20(4): 10-19
[7] Tashtoush, Y., Odat, Z., Alsmadi, I., and Yatim, M. (2013). Impact of programming features on code readability. International J. of Software Engineering and Its Applications, 7(6): 441-458.
[8] Batool, A., ur Rehman, M. H., Khan, A., and Azeem, A. (2015). Impact and Comparison of Programming Constructs on JAVA and C# Source

Code Readability. International J. of Software Engineering and Its Applications, 9(11): 79-90.
[9] Buse, R. P., and Weimer, W. R. (2008). A metric for software readability. In Proc. of ACM international symposium on Software testing and analysis, pp. 121- 130.
[10] https://github.com/ipeirotis/ReadabilityMetrics
[11] http://www.virtualmachinery.com/jhdownload.htm
[12] Radatz, J., Geraci, A., and Katki, F. (1990). IEEE standard glossary of software engineering terminology. IEEE Std, 610121990(121990), 3.
[13] Misra, S., and Akman, I. (2008). A model for measuring cognitive complexity of software. In Proc. of International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, pp. 879-886.
[14] Buse, R. P., and Weimer, W. R. (2010). Learning a metric for code readability. IEEE Transactions on Software Engineering, 36(4): 546-558.
[15] Pahal, A., and Chillar, R. S. (2017). Code Readability: A Review of Metrics for Softw. Quality. Intl. J. of Comp. Trends and Tech, 46 (1): 1-4.
[16] Posnett, D., Hindle, A., & Devanbu, P. (2011). A simpler model of software readability. In Proc. of the 8th working conference on mining software repositories, pp. 73-82.
[17] Wang, X., Pollock, L., and Vijay-Shanker, K. (2011). Automatic segmentation of method code into meaningful blocks to improve readability. In 18th Working Conference on Reverse Engineering ,pp. 35- 44.
[18] Halstead, M. H. (1977). Elements of software science. Elsevier.
[19] McCabe, T. J. (1976). A complexity measure. IEEE Trans. on software Engineering, SE-2 (4): 308-320.
[20] Parareda, B., & Pizka, M. (2007). Measuring productivity using the infamous lines of code metric. In Proc. of SPACE 2007 Workshop, Japan.
[21] Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. IEEE Transactions on software engineering, 20(6): 476-493.
[22] http://www.readabilityformulas.com/smogreadabilit y-formula.php
[23] http://www.readabilityformulas.com/gunning-fogreadability-formula.php
[24] Flesch-Kincaid Readability Index. http://www.mang.canterbury.ac.nz/writing_guide/wr iti ng/fles ch.shtml
[25] Coleman-LiauIndex. http://en.wikipedia.org/w/index.php?title=Meri_Col eman&ac tion=edit&redlink=1
[26] https://www.cs.waikato.ac.nz/~ml/weka/
[27] Goswami, P., Kumar, P., & Nand, K. (2012). Evaluation of complexity for components in component based software engineering. IJREAS, 2(2).
[28] http://www.readabilityformulas.com/free-readabilityformula-tests.php
[29] https://ipeirotis-hrd.appspot.com
[30] Montgomery, D. C., Jennings, C. L., and Kulahci, M. (2008). Introduction to Time Series Analysis and ForeCasting. Wiley Series in Probability and Statistics. John Wiley and Sons, Inc.
[31] Software Code Readability Dataset. https://github.com/zibranm/dataset.git