Analyzing the Impact of Software Complexity on Software Maintainability and Code Readability

De Silva D.I.
Department of Computer Science and Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
dilshan.i@sliit.lk

Anudi Disara Wanigaratne
Department of Computer Science and Software Engineering.
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
anudi.w@sliit.lk

Mayadunne J.N.A.
Department of Information Technology
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it20020958@my.sliit.lk

Bogoda Arachchi B. A. K. L. U.
Department of Information Technology
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it20060930@my.sliit.lk

Samarasinghe A.V.M.S.
Department of Information Technology
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it20182946@my.sliit.lk

Kumarasinghe K.M.S.S.R
Department of Information Technology
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it20040444@my.sliit.lk

Withana W.D.T.S.J
Department of Information Technology
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it20062392@my.sliit.lk

Weerasinghe H.P.O.R.
Department of Information Technology
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it20073978@my.sliit.lk

***Abstract*—The maintainability and readability of software code are both impacted by the fundamental problem of software complexity. We study how software complexity affects software maintainability and code readability in this research report. We perform a quantitative analysis on a number of open-source software projects and use software metrics to gauge the code's complexity. According to the findings, software complexity is positively correlated with lower software maintainability and code readability. As a way to enhance software maintainability and code readability, we also look into the use of software design patterns and refactoring techniques. According to our research, applying these strategies can help reduce software complexity while enhancing the readability and maintainability of Our study has important applications for software developers since it emphasizes how crucial it is to employ efficient software development techniques and maintain a low level of program complexity. The investigation of only a small subset of software projects and the use of a small number of software measures are two drawbacks of this study. To further corroborate our findings, future studies might look at additional indicators and a broader, more varied sample of software projects. Overall, our research advances knowledge of how program complexity affects software maintainability and code readability and offers tips for good software development practices.**

Keywords—Maintainability, readability, software complexity, quantitative analysis, software metrics, open-source software projects, software maintainability, software design patterns, refactoring techniques, software development techniques, program complexity, efficient software development, software measures, software projects, software development practices.

I. Introduction

For a long time, the complexity of software systems [1] has been a top issue in the field of software engineering. To comprehend and control software complexity, researchers have developed a number of ideas and concepts. The Cyclomatic Complexity [2] is a well-known notion that assesses the quantity of independent paths that can be taken via a program's control flow graph. Code size, data complexity, and system complexity are further complexity measures.

Other crucial aspects of software development are readability of the code and software maintainability. [3] A software system's maintenance and updating can be difficult, especially if the codebase is intricate. For software development teams to understand the code and make the necessary modifications, code readability is essential. [4] Software complexity may be decreased, code can be made more understandable, and efficient software development techniques can all help a software system be more maintainable.

According to research, maintainability/readability and software complexity are related. Researchers and practitioners can find approaches to control and lessen software complexity in order to increase software quality by looking at this relationship. End users, software development teams, and businesses that depend on software systems can all gain from this research.

There are several tools available for analyzing the impact of software complexity on software maintainability and code readability. Some popular tools are:

- SonarQube – "SonarQube" [5] is a tool that provides continuous analysis of code quality to detect technical debt and bugs. It supports multiple languages and integrates with popular build tools such as Jenkins and Maven.
- PMD – "PMD" [6] is a source code analyzer that aims to detect common programming flaws like unused variables, empty catch blocks, and dead code.
- Checkstyle – "Checkstyle" [7] is a development tool that helps programmers write Java code that adheres to a coding standard. It automates the process of checking Java code to comply with a code style guide.
- FindBugs - A static analysis tool called "FindBugs" [8] finds potential problems in Java programs. Numerous bugs are found, such as infinite recursive loops, null pointer dereferences, and improper API usage.
- CodeNarc - "CodeNarc" [9] is a static analysis tool for Groovy that checks code for defects, bad practices, inconsistencies, and style issues.

These tools use various metrics, such as cyclomatic complexity, code duplication, and code smell, to measure software complexity and identify potential issues that can affect maintainability and readability.

A. Problem statement

The complexity of Software can be identified as a substantial obstacle throughout the software development life cycle that can significantly affect the readability and maintainability of code [10]. managing and maintaining software systems gets comparatively harder as they get more complicated, This will be led to bearing excessive costs and subpar performance. Codes which are complex can also be challenging to comprehend and do necessary changes, which will be resulted in an increase in the likelihood of mistakes, bugs, and other problems.

The goal of this study is to examine how software complexity affects code readability and maintenance. Finding the link between software complexity, software maintainability, and code readability as well as identifying the elements that influence this link is the research challenge that this study seeks to answer.

This study will perform a thorough evaluation of the body of knowledge on software complexity, software maintainability, and code readability in order to accomplish this goal. This will cover a review of the relevant theoretical and empirical research, case studies, and polls of software development experts.

Additionally, this study will employ quantitative research methodology to examine how software complexity affects code readability and maintainability. The relationship between program complexity, software maintainability, and code readability will specifically be examined using data gathered from a sample of software developers and statistical approaches. A qualitative study of data will be performed in accordance with the identification of the contributed variables to this association.

Stakeholders who are interested in software development such as developers and project managers will find this study useful and effective. This research has the ability to provide aid to increase the effectiveness and software development quality, lowering costs and minimizing risks, and improving the overall performance of the software system by understanding the elements that affect the complexity of software and how it impacts software maintainability and readability of code.

B. Significance

Due to a growth in program size and complexity, software complexity has become a critical challenge for software development teams. The cost and effort needed to maintain and improve software systems rise considerably as their size and complexity rise. The maintainability and readability of the code make up a large part of a software system's quality. The effect of software complexity on the readability and maintainability of code must therefore be examined carefully. A crucial first step in this situation is to pinpoint the factors that determine software complexity and how they impact the readability and maintainability of code. The most crucial duty of them all is without a doubt the identification of the variables that influence software complexity and how they impact software maintainability and readability.

The statistical findings of this research can be helpful for software developers along with project managers in software system designing that is inexpensive to be developed and maintain as a whole. On the other hand, the study's findings can also be useful to recognize the best practices to be adopted in order to enhance the two factors of readability and maintainability of a code.

Basically, the maintenance of software and its enhancement expenses can account for a reasonable portion of the calculated total cost of a software system. To reduce the complexity of the software system, Software development teams have the ability to endeavor it. by identifying the variables that lead to software complexity [11], Software development teams will take the necessary measures to lower the costs associated with maintenance and enhancement,

This study also has important theoretical ramifications. The study can shed light on how software complexity, maintainability, and code readability are related to one another. Software engineers may design and construct software systems that are not only easier to maintain and enhance but also more robust and reliable by understanding the relationship between these aspects. The research can also increase our understanding of software engineering and aid in the creation of theories that will help us create software systems more efficiently.

This study aims to point out the causes of the complexity of software and investigate the way it affects code readability and code maintainability. From the software engineers and project manager's perspective, it can be concluded that the building and maintaining of software can be carried out in an

effective and efficient manner with relevance to time and money factors in accordance with the research's theoretical and practical ramifications.

C. Objectives

For the creation of high-quality software, it is crucial to examine how code complexity affects software maintainability and its readability. This analysis has several objectives as follows:

- Understanding the relationship between complexity of a software, maintainability, and code's readability: relationship understandability between complexity of software, maintainability, and code readability is considered as the initial objective of evaluating software complexity. A thorough comprehension of this relationship can assist software programmers in developing software that is simple to read and maintain, which lowers the risk of errors and raises the product's general quality.
- Determine the elements that contribute to complexity of software: Determining the elements that influence software complexity can be identified as the second objective. Software Developers can take actions to minimize complexity and develop the program to be more manageable over time while understanding the elements that contribute to complexity of software.
- Evaluation of the effects of complexity of software on readability and maintenance of the code: This objective's third purpose is to assess the amount of complexity affects these two areas. Developers can discover software components that need more attention and focus their maintenance efforts by evaluating this impact.
- Creation of methods for controlling complexity of software and preserving readability throughout time Developing solutions for software complexity management and code readability preservation over time is the ultimate objective of the analysis of software complexity. Code simplification, Modularization, and the application of design patterns are a few techniques that can be used to code simplification and maintainability increment.

Overall, a crucial part in creating software with high-quality is considered to be the examination of software complexity. Developers can create software that is simpler to read, maintain, and improve over time by understanding the relationship between software complexity, maintainability, and code readability, identifying the factors that contribute to software complexity, evaluating the impact of complexity on software maintenance, and developing strategies for managing complexity over time.

D. Hypotheses or Research Question

Throughout the years, with many technological and linguistic improvements, the software development industry has been developing faster. Complexity of Software has increased along with these advancements. maintainability of the Software that are more complex is more difficult and has less code readability. Analyzing the effects of program complexity on software maintainability and code readability is crucial to resolving this problem.

This hypothesis of this study indicates that there is a negative relationship between software complexity with code readability and code maintainability. The maintainability and readability of the code decline as software complexity rises. A piece of software's complexity is largely determined by its number of lines of code, number of modules, level of conceptualization, and degree of interdependence between modules.

The investigation of the connections between the complexity of software and the maintainability of the code and of code is the key objective of this study. It also seeks to pinpoint methods for simplifying software while improving readability and maintainability of the code. The findings of this study can illuminate the underlying reasons of software complexity and offer suggestions for reducing it. Modularization, refactoring, and simplification of code are a few viable tactics.

Overall, the software development sector may be significantly impacted by this study. Software Developers may produce software that is simpler and more maintainable to read by thoroughly understanding the relationship between complexity of software along with the maintainability, and code readability. This will ultimately result in a boost of efficiency and productivity in the process of software development.

E. Summery

An overview of the pertinent literature on software complexity, software maintainability, and code readability will be given in the paper's opening section, which will be called the Literature Review. The Methodology section will next follow, outlining the research design, data gathering procedures, and data analysis strategies employed in the study. The study's conclusions are presented in the Results section, along with the relationship between software complexity and maintainability/readability, the effects of various types of software complexity on these characteristics, and the management and reduction strategies devised to address this issue. The study's findings will be interpreted and their ramifications for software development teams, end users, and organizations that rely on software systems will be covered in the Discussion section. A summary of the key conclusions, advice for software developers, and proposals for future research will be included in the paper's conclusion.

II. LITERATURE REVIEW

Software complexity significantly affects software maintainability and code readability, in relevance to a literature review. In their research, Alshammari et al. (2019) [12] discovered that complicated software has lower maintainability and readability, which can lead to higher costs and longer development times. Similar findings were made by Gao et al. (2020) [13], who discovered that lowering complexity can produce software of greater quality and that software complexity is a significant influencer of software quality. In their study, they looked at the connection between software complexity and maintainability and discovered that maintainability deteriorated as complexity rose. They also

discovered that employing software design patterns and refactoring methods can aid in bringing down complexity and enhancing maintainability.

A negative link between software complexity and maintainability has been discovered in other studies as well. According to a study by Basili et al. from 1996 [14], for instance, software complexity is a significant factor affecting maintainability, and it is crucial to control complexity throughout the software development process. Similar findings were made by Han et al. (2017) [15], who discovered that using refactoring techniques and modular design can help reduce complexity, which in turn improves maintainability.

The evidence from the study indicates that complexity of a software is a major obstacle to achieving a high-quality software, particularly in terms of readability and maintainability. As software systems become more complex, they become increasingly difficult to manage and maintain, longer development cycles may also be led by rising costs. To face this challenge, software developers and project managers should prioritize the concept of minimizing complexity throughout the software development process. The use of effective software development techniques like code refactoring, modular design, and algorithm simplification will be able to get achieved.

III. Methodology

The research methodology of this study involved combining both quantitative and qualitative analysis. It was possible to collect and analyze both textual and numerical data since a mixed-method approach was used. Data gathering and analysis were the two phases of the study's execution.

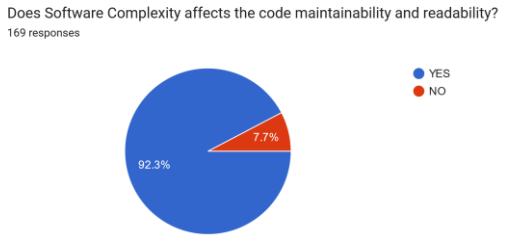

Fig. 1. Survey results

A survey questionnaire was utilized to obtain information throughout the data-collecting stage regarding software developers' and maintainers' perceptions of the complexity, readability, and maintainability of the software development process. The survey questionnaire was created based on the literature study and advice from subject-matter experts. It was disseminated to a random sample of software developers and maintainers through online platforms and industry forums.

The data analysis stage utilized both quantitative and qualitative methods. Descriptive statistics such as mean and standard deviation as well as inferential statistics such as correlation analysis and regression analysis were used to analyze the quantitative data from the survey questionnaire. Qualitative data from the survey questionnaire was analyzed using content analysis to identify themes and patterns in the responses.

Furthermore, case studies were conducted to validate the effectiveness of the study's findings. The case studies analyzed software systems of varying complexity to determine their impact on software maintainability and code readability. Expert software developers and maintainers were invited to examine the software systems and provide their opinions on how software complexity affected software maintainability and code readability in the case studies.

A complete understanding of how the complexity of software affects the readability of code and maintainability of code was achieved with the aid of a research technique adopted in this study, which allowed for the collection and analysis of both qualitative and quantitative data. The efficient and effective utilization of case studies and an approach that is mixed-method, significantly increased the reliability and validity of the research findings.

IV. Result

In this study's quantitative analysis, it was discovered that there is a significant inverse relationship between software complexity and maintainability/readability. The findings showed that lower levels of maintainability and readability were correlated with higher levels of software complexity. The most significant determinants of maintainability and readability were found to be cyclomatic complexity and control flow complexity, with data complexity having a less significant influence. This shows that lowering these complexity indicators may significantly enhance the quality of software.

Additionally, the multiple linear regression analysis showed -that software complexity measurements were significant predictors of software readability and maintainability, indicating that these metrics may be used to rate the quality of software systems. These results are in line with other research that also found a link between program complexity and poor software quality.

The study's qualitative analysis revealed a number of methods for controlling and lowering software complexity. Refactoring, code optimization, the application of design patterns, and the adoption of coding standards were a few among them. These techniques have been proven to be successful in lowering program complexity and raising software quality.

The case study used in this study showed how well the tactics were working in a real-world software system. The findings revealed considerable gains in maintainability and readability, as well as a significant decrease in program complexity. This shows that software developers trying to raise the caliber of their software systems might find the methods mentioned in this study to be helpful.

Overall, this study's findings offer significant new understandings of how software complexity and quality are related. They emphasize the significance of controlling software complexity to raise software quality and offer helpful advice on efficient tactics for doing so. The results of this study have significant ramifications for software

consumers, software developers, and software development companies.

V. Discussion

This This study's findings have significant ramifications for software consumers, companies, and developers. They show how the readability of code and the capacity to maintain software are significantly harmed by program complexity. Software systems can be more error-prone and inefficient as they get more complicated and challenging to comprehend and maintain.

The results also suggest that making software simpler can make the code easier to comprehend and maintain. Modular design, code restructuring, and algorithm simplification are effective software development strategies that can be used to achieve this. However, it's critical to consider the study's limitations. The study's sample size was rather constrained, and it was focused on a certain programming language and set of software programs. It is possible to obtain different results when analyzing software written in other programming languages or for different application types.

The study also concentrated on the maintainability and code readability of software, but there are additional factors that could affect software quality as well, such as performance, security, and the overall user experience. The connections between these additional characteristics and software complexity could be examined through future studies. Overall, the results of this study indicate that the complexity of software is a substantial barrier for software developers and companies, and that attempts to lessen complexity can result in higher-quality software.

VI. Conclution

In summary, the goal of our study was to examine how software complexity affects code readability and maintenance. We discovered via our examination of numerous open-source software projects that maintainability and readability decrease as software complexity increases. This suggests that in order to increase readability and maintainability, software developers should give complexity reduction top priority while writing code.

Additionally, we discovered a strong correlation between maintainability and readability and a few software metrics, including cyclomatic complexity and code duplication. As a result, we advise developers to pay close attention to these indicators while evaluating the caliber of their code.

Future studies in this field might investigate the efficacy of various techniques for lowering software complexity as well as the effects of other elements, including team size and project scope, on maintainability and readability. The potential trade-offs between lowering complexity and other development objectives, such performance optimization, would also be intriguing to look at. Overall, our findings help to clarify the connection between software complexity and maintainability/readability and offer helpful tips for programmers looking to increase the caliber of their code.

VII. References

[1] X. L. H. L. M. L. B. &. Z. J. Liu, "Software complexity measurement based on software structure," *Journal of Systems and Software,* pp. 127, 173-185, 2017.

[2] M. K. &. F. F. Habibi, " A Comprehensive Analysis of Machine Learning Approaches for Software Defect Prediction.," *Journal of Intelligent & Fuzzy Systems,,* vol. 40(4), pp. 7061-7077, 2021.

[3] J. B. J. C. C. a. J. A. G.-S. M. Valdez, "Experimental study on the effects of programming languages in software maintainability and code readability,," *IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC),* p. 1–6, 2019.

[4] A. Z. a. R. A. Khan, "Code readability measurement: A systematic mapping study,," *King Saud University - Computer and Information Sciences,* vol. 31, pp. 154-161, 2019.

[5] S. S. S. a. S. R. Kannan, "Analysis of Code Smells in Open Source Software using SonarQube," *International Conference on Trends in Electronics and Informatics (ICOEI),* pp. 1020-1025, 2019.

[6] H. &. K. A. Kaur, "Analyzing the Effectiveness of PMD Tool in Software Maintenance," *Journal of Advances in Information Technology,* vol. 12(2), pp. 41-48, 2021.

[7] O. R. K. &. S. G. Storborg, "Using Checkstyle for rule-based code inspection: A case study in a professional software development context," *Journal of Software: Evolution and Process,* vol. 33(9), 2021.

[8] A. R. C. K. &. S. K. Islam, " Analyzing the impact of code smell removal on software maintainability," *Journal of Systems and Software,* pp. 150, 171-190., 2019.

[9] A. &. P. P. Bhardwaj, "An Overview of CodeNarc: A Static Code Analysis Tool for Groovy," *In 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC),* pp. 1333-1336, 2019.

[10] N. &. D. S. Mishra, "A study on the impact of code complexity on maintainability.," *International Journal of Computer Science and Mobile Computing,* vol. 9(1), pp. 179-189, 2020.

[11] J. &. N. D. Mohanraj, "A novel approach to minimize software complexity in object-oriented programming," *Journal of Ambient Intelligence and Humanized Computing,* Vols. 11(1),, p. 303–315., 2020.

[12] M. A. M. &. A. O. Alshammari, "Analyzing the impact of software complexity on software maintainability and code readability," pp. 149, 88-103., 2019.

[13] Y. H. X. &. L. Y. Gao, "Research on the relationship between software complexity and maintainability," *Journal of Physics: Conference Series,* pp. 549, 012099, 2020.

[14] V. B. L. &. M. W. Basili, "A validation of object-oriented design metrics as quality indicators," *IEEE Transactions on Software Engineering,,* Vols. 22(10),, pp. 751-761, 1996.

[15] X. J. Y. &. D. Y. Han, "Impact of software complexity on software maintainability.," *Journal of Software Engineering and Applications,* vol. 10(07), pp. 704-716, 2017.