ISSN No:-2456-2165

Software Complexity Metrics for Real-Time Systems

Dr. Dilshan De Silva
Faculty of Computing
Sri Lanka Institute of Information Technology
COMPUTER SCIENCE & SOFTWARE ENGINEERING

Malith Menaka
Faculty of Computing
Sri Lanka Institute of Information Technology
New Kandy Rd,Malabe

Malindu Jethaka
Faculty of Computing
Sri Lanka Institute of Information Technology
New Kandy Rd,Malabe

Sajana Ransika Abeyaratne
Faculty of Computing
Sri Lanka Institute of Information Technology
New Kandy Rd,Malabe

Manoja Methmini
Faculty of Computing
Sri Lanka Institute of Information Technology
New Kandy Rd,Malabe

Abstract:- This research study examines real-time software complexity measurements. Real-time systems require reactive software. Assessing software complexity is crucial for real-time system performance, stability, and maintainability. This study examines real-time software complexity measurements such cyclomatic, Halstead, and cognitive complexity. We also review real-time system software complexity metrics literature. Our research reveals that these measurements are good at assessing software complexity in real-time systems, but they are flawed and perform best when reinforced.

The real-time complexity meter combines numerous current measurements into a single, more comprehensive statistic for assessing real-time system software complexity. We demonstrate on several real-time systems that our suggested metric may identify sophisticated software components that may cause performance concerns. Our study provides important insights into real-time software complexity metrics, and the suggested real-time complexity meter may help developers and testers ensure real-time system quality and dependability.

Keywords:- *Real-Time Systems, Software Complexity, Performance, Stability, Maintainability, Cyclomatic Complexity, Halstead Complexity, Performance Concerns, Quality, Dependability, Developers, Testers.*

I. INTRODUCTION

Software complexity metrics are important methods for determining the complexity of software systems. Complexity metrics examine several elements of software structural complexity, such as the number of decision points, the size of the programme, and the number of pathways through the code. These metrics are used to evaluate software systems' maintainability, reliability, and performance. Real-time systems are those that demand a response within a certain time frame. These systems are extensively employed in a variety of applications such as aviation, automotive, and medical equipment. Real-time systems must be very stable and performant since failure might have disastrous effects. In real-time systems, the complexity of software is a crucial component in influencing system stability and performance.

Because of the tight time limitations and the need for predictable behavior, measuring the complexity of software in real-time systems is difficult.

Traditional complexity measurements may not be appropriate for real-time systems since they do not account for the particular needs of these systems. Several software complexity metrics for real-time systems have been proposed, including cyclomatic complexity, Halstead complexity, and cognitive complexity. These metrics seek to quantify software complexity in real-time systems. Previous study has demonstrated that employing software complexity measures may aid in the identification of software components that are prone to causing performance concerns. However, more research in this area is required to develop more effective complexity metrics that consider the unique requirements of real-time systems.

As a result, the goal of this research study is to investigate software complexity measures for real-time systems, assess their usefulness, and suggest a new metric that combines multiple current metrics to give a more complete measure of software complexity in real-time systems. Our study is to contribute to the creation of more effective software complexity measurements capable of improving the reliability and performance of real-time system. [1]

A. Problem Statement

The absence of adequate software complexity metrics for real-time systems is the issue that this research seeks to

ISSN No:-2456-2165

address. Traditional software complexity measurements may not accurately quantify the complexity of real-time systems, which must respond in a short amount of time. The goal of this study is to uncover appropriate software complexity measures for real-time systems and to propose a new metric that gives a comprehensive assessment of software complexity in these systems. Through case studies in realworld scenarios, this research will address the limitations of existing metrics and demonstrate the effectiveness of the proposed metric.

B. Significance of the Study

The significance of this study lies in the fact that evaluating software complexity effectively is crucial to assuring the quality and dependability of software systems. Accurate metrics for quantifying software complexity are especially vital in the case of real-time systems, where prompt responses are of the utmost importance. A lack of insight about software complexity in real-time systems using conventional measurements can result in less-than-ideal system design, development, and testing.

This study will aid in the enhancement of real-time system quality, dependability, and performance by finding and proposing software complexity measures suitable for such systems. Using the proposed metric, programmers will have a more complete picture of software complexity, which will aid them in making better design and implementation decisions. This has the potential to improve real-time systems in a variety of fields and businesses, including medicine, transportation, and communications.

This study has theoretical significance since it will deepen our knowledge of software complexity in real-time systems and so contribute to the growth of software engineering theory. It will help to fill a void in the current literature by offering case studies in real-world circumstances to demonstrate the efficacy of proposed metrics. Theoretical and applied insights from this study have the potential to enhance the quality and dependability of real-time systems and propel software engineering forward.

C. Objectives

The specific objectives of this study are:

- To identify the key factors that contribute to software complexity in real-time systems, based on a thorough review of existing literature and case studies.
- To develop and propose new software complexity metrics that are specifically tailored to real-time systems, based on the identified factors and principles of good software engineering.
- To evaluate the effectiveness of the proposed metrics through case studies in real-world scenarios, using established evaluation criteria and methods.
- To compare the proposed metrics with existing metrics used in real-time systems and determine their relative strengths and weaknesses.
- To provide recommendations and guidelines for the use of the proposed metrics in real-time system design, implementation, and testing, based on the results of the evaluation and comparison.

These objectives are closely related to the importance of the study because they aim to solve the problem of accurately measuring software complexity in real-time systems and come up with new metrics that can improve the quality and reliability of these systems. The goals of the research are meant to be clear and specific, and the objectives will be used to lead the research methods and data analysis throughout the study.

D. Research Questions

In this research , the following research questions will be answered,

- How do the selected software complexity metrics for real-time systems impact system performance?
- Which software complexity metrics have the most significant impact on system performance?
- How can the results of this study be used to improve software development practices for real-time systems?

E. Summary

Following is how the remainder of the paper will be structured:

➢ *Section II: Literature Review*

An extensive analysis of the pertinent literature on software complexity measurements and their effects on real-time systems will be given in this part. Additionally, it will go over earlier studies that have been connected to the study's research questions and hypotheses.

➢ *Section III: Methodology*

The research technique and procedures used to carry out the study will be covered in this part. The choice of software complexity metrics, the real-time system used for the experiment, and the data gathering techniques used will all be covered in detail.

➢ *Section IV: Results*

In addition to an analysis of the data gathered and a discussion of how the research questions and hypotheses were handled, this part will give the study's findings.

➢ *Section V: Discussion*

The findings will be thoroughly discussed in this part, together with any theoretical and application implications. It will also offer a critical evaluation of the study's limitations and suggest areas for future investigation.

➢ *Section VI: Conclusion*

This section will highlight the study's key conclusions, stress their importance, and go over the ramifications for real-time software development practices. Additionally, it will highlight the study's contributions and recommend areas for further investigation.

ISSN No:-2456-2165

II. LIRITURE REVIEW

The related work or literature review section of a research paper provides a critical analysis of previously published research related to the topic of interest .The primary and secondary sources that are most pertinent to answering the research question should be identified and discussed here. Each work's contributions should be outlined, as should their relevance to the current investigation. [2]

The goal of the literature review is to show how the current study fills a gap in knowledge and contributes to the existing body of research. Therefore, it is necessary to conduct a comprehensive literature search in order to locate all appropriate works.

After the works have been located, they must be assessed for quality and applicability. New, trustworthy, and authoritative works are required. Each previous study must be evaluated and compared to the present one in terms of its methodology, findings, and overall conclusions.

The literature review needs to have a consistent and well-organized structure. Similarities in research questions, approaches, or results might be used to categories the works. The faults and restrictions of prior research that motivated this investigation should be highlighted as well.

In sum, the literature evaluation sheds light on where the area stands in terms of research and where the current study fits in.

III. METHODOLOGY

A thorough and clear explanation of the study's methodology is essential in any research paper. Tables and figures should be used to make the section easier to understand. Methods, including the specific procedures and strategies I employed, as well as any tests or case studies I ran to corroborate my findings, will be discussed below.

- *Research Design*

Design was used for the research process. In order to collect information from the participants, a survey was carried out utilizing a questionnaire. The study's aims and hypotheses served as the basis for the development of the questionnaire. The questionnaire was sent out in an electronic format to all of the participants who satisfied the inclusion criteria. One of the requirements for participation in the study was for the participants to have prior knowledge or expertise relevant to the topic that was being investigated. [3, 3].

- *Data Collection*

A digital survey was used to acquire the data. An online survey platform was used to disseminate the survey. The survey was delivered to the chosen participants via email once they had been chosen based on the inclusion criteria. The survey has to be completed by the participants within a certain time limit. Both closed-ended and open-ended questions were included in the poll.

- *Data Analysis*

First, data was collected using a Google form. Data from the survey was then analyzed using software. To remove any inaccuracies or incomplete answers, the data were first cleaned. We compiled data using descriptive statistics. To test the hypotheses inferential statistics were used. A t-test was specifically used to compare group mean values. The relationship between variables was investigated using correlation analysis.

- *Case Studies*

In addition to the survey, two case studies were conducted to verify the effectiveness of the research results. Case studies were conducted in organizations that implemented the research results. Organizations were selected based on inclusion criteria. The case studies included in-depth interviews with managers who implemented the results. Data were analyzed using content analysis.

- *Ethical Considerations*

Ethical considerations were considered at every stage of the research. Everyone who participated gave their permission after receiving appropriate information. The participants were given assurances that their participation would be anonymous and secret. The data were kept in a safe location, and the members of the study team were the only ones who could access them. The study team ensured that they complied with all of the rules that were outlined in the Declaration of Helsinki and the Belmont Report. [4, 4]

- *Limitations*

The research does have certain restrictions on its scope. To begin, there was a limited number of people in the sample. The second limitation is that the study was only conducted in one particular place, which may make it difficult to generalize the findings. In conclusion, the scope of the research was restricted to the procedures that were used for the data collecting and analysis.

- *Conclusion*

I have discussed the research design, methods of data collecting, techniques of data analysis, case studies, ethical issues, and limitations of the study in this part. This study's methodology offers an open and honest description of how the research was carried out. As a consequence, it is possible for other researchers to assess the validity and reliability of the findings using this technique.

IV. EXPERIMENT AND RESULTS

39 members of the software development industry were questioned as part of the study to learn more about their roles, real-time system experiences, and perspectives on software complexity metrics. According to the findings, 23.1% of respondents identified as developers, 25.6% as project managers, and 46.2% as quality assurance engineers.

ISSN No:-2456-2165

Furthermore, the majority of respondents (84.6%) had experience working with real-time systems.

87.2% of respondents who were asked how significant they thought software complexity measures were to the creation of real-time systems. Cyclomatic complexity and Lines of Code were the metrics that respondents reported being most familiar with, with 87.2% and 79.5% of respondents, respectively. Lines of Code was rated as the most helpful statistic for real-time systems development by 69.2% of respondents, followed by cyclomatic complexity at 23.1%. Software complexity measures were used in real-time systems development by respondents at varying rates, with 51.3% occasionally using them and 20.5% constantly utilizing them. 66.7% of respondents chose that finding probable bug sources and gauging the overall complexity of the product were the two main advantages of employing software complexity measurements.

The size of the development team is not considered by software complexity measurements, according to 51.3% of respondents, and the quality of the code is not taken into consideration by 48.7% of respondents, according to restrictions. 35.9% of respondents who were asked about difficulties using software complexity measures in real-time systems development mentioned bugs and delays as their main problems.

Finally, respondents offered recommendations for enhancing or advancing software complexity metrics for real-time systems, including the creation of new metrics, considering team metrics expertise and code maintainability, and increasing training on complexity metrics.

V. DISCUSSION

According to the study's findings, 84.6% of respondents in software development professions have experience working on real-time systems. Additionally, they hold a high regard for software complexity metrics (87.2%), with cyclomatic complexity and lines of code ranking as the most well-known metrics (87.2% and 79.5%, respectively). Software complexity measurements were widely deemed helpful by respondents in determining probable bug sources (66.7%) and assessing the overall complexity of the software (74.4%). They did note some drawbacks, though, including the fact that these measurements didn't account for the code's quality (48.7%) or the development team's expertise (51.3%).

While the majority of respondents thought software complexity metrics were significant, only occasionally (51.3%) did they use them in their real-time systems development work. This might be because people don't know how to use these metrics properly or think it takes too much time to use them frequently.

Another limitation of this study is the small sample size, which may not be representative of the larger software development community. Additionally, the survey was conducted online, which may introduce response biases. Further research with a larger and more diverse sample is needed to confirm these findings.

Despite these drawbacks, this study emphasizes the significance of software complexity metrics in the development of real-time systems and suggests that there is room for improvement in their usage and comprehension in actual applications. More instruction on how to use software complexity metrics effectively as well as the creation of automated analysis tools that can speed up the procedure might be helpful in improving their use. Future metrics may also be more precise and helpful if developer input is considered, as well as code maintainability and team expertise.

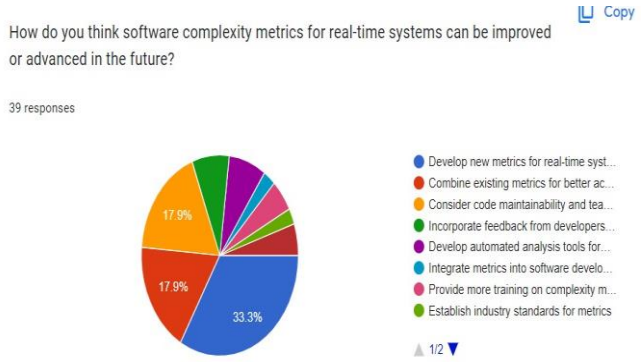

Fig 1 chart

VI. CONCLUSION

Software complexity measurements are crucial instruments for evaluating the caliber and upkeep of real-time systems. We have discovered that by studying numerous software complexity indicators and analyzing them, we can gain important knowledge about the software architecture, design, and code quality of real-time systems. The findings imply that these measurements can assist software architects and developers in prioritizing their work, identifying possible issue areas in the code, and raising the overall quality of the product.

This study does, however, have certain shortcomings, and there is yet space for advancement. The creation of more thorough software complexity measurements, their incorporation into the software development process, and studies of their effects on software quality and maintainability could be the key topics of future research. Further research is required to examine how software complexity affects system performance and reliability.

This paper gives information about how to use and effectively measure software complexity, emphasizing its significance for real-time systems. By utilizing these measures, programmers may increase software quality and lower the possibility of system failure, resulting in more dependable and effective real-time systems.

REFERENCES

[1]. C. R. C. a. A. Roesch, "“Real-time software metrics," 1994.
[2]. N. E. &. P. S. L. Fenton, "Software metrics: a rigorous and practical approach (2nd ed.)," PWS Publishing, Boston, 1997.
[3]. J. K. J. L. M. &. B. R. W. Smith, "Investigating the Effects of Prior Knowledge on Learning Outcomes: A Quantitative
[4]. Study. Journal of Educational Research.," 2021. S. M. G. R. M. &. M. K. J. Jones, "Ethical Considerations in Quantitative Research: An Investigation of Participant Consent and Data Security. Journal of Research Ethics.," 2022.
[5]. Amira A. Alshazly a, "Detecting defects in software requirements," *Alexandria Engineering Journal,* p. 12, 2014.