



# **Intelligent Code Comprehensibility Index: A Cognitive-Based Metric for Enhancing Code Review and Documentation**

Godamune G.A.P.J.  
MS24025126

A THESIS  
SUBMITTED TO  
SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY  
IN PARTIAL FULFILMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
MASTER OF SCIENCE IN INFORMATION TECHNOLOGY  
(ENTERPRISE APPLICATION DEVELOPMENT)

December 2025

I certify that I have read this thesis and that, in my opinion, it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

---

Dr. Junius Anjana

Approved for MSc. Research Project:

---

MSc in IT Programme Co-ordinator, SLIIT

Approved for MSc:

---

Head of Graduate Studies, FoC, SLIIT

# DECLARATION

This is to certify that the work is entirely my own and not of any other person, unless explicitly acknowledged (including citation of published and unpublished sources). The work has not previously been submitted to the Sri Lanka Institute of Information Technology or any other institution in any form for assessment or any other purpose.



Sign: .....

Godamune G.A.P.J.

Date: .....01/12/2025.....

# ABSTRACT

## **Intelligent Code Comprehensibility Index: A Cognitive-Based Metric for Enhancing Code Review and Documentation**

Godamune G.A.P.J.

MSc. in Information Technology (EAD)

**Supervisor:** Dr Junius Anjana

December 2025

As software systems become increasingly complex, developers face more challenging tasks in understanding, maintaining, and evolving code. Traditional software metrics like Lines of Code, Cyclomatic Complexity, and Halstead metrics provide structural insights but often fail to capture the cognitive aspects of code comprehension. This paper introduces the Intelligent Code Comprehensibility Index, a new multi-dimensional metric framework based on Cognitive Load Theory. The Intelligent Code Comprehensibility Index assesses code comprehensibility by examining three key dimensions: Structural Complexity, Documentation Quality, and Naming Quality. Each dimension targets specific cognitive loads, Intrinsic, Extraneous, and Germane, by including syntactic metrics for semantic alignment and drawing on empirical research from software engineering and neuroscience. The proposed framework aims to offer a more comprehensive and cognitively aligned method for evaluating and improving source code understandability, thereby boosting developer productivity and code quality.

## **ACKNOWLEDGEMENT**

I am expressing my sincere gratitude to Dr Junius Anjana and Dr Dilshan De Silva for their invaluable guidance, support, and encouragement throughout this research. Their expertise and constructive feedback have been instrumental in shaping the direction and success of this study. Their deep understanding of the subject matter and willingness to share their knowledge greatly enhanced the quality of the research. This work would not have reached its current form without their unwavering mentorship and dedication. I am deeply grateful for their time, patience, and insights, which have been vital in helping me achieve the aims of this thesis.

# TABLE OF CONTENTS

DECLARATION .....	2
ABSTRACT .....	3
ACKNOWLEDGEMENT .....	4
TABLE OF CONTENTS .....	5
List of Figures.....	9
List of Tables .....	10
List of Algorithms.....	11
Chapter 1 Introduction.....	12
1.1 Background.....	12
1.2 Problem Statement.....	13
1.3 Research Gap and Proposed Solution.....	14
1.3.1 Critical Limitations of Current Approaches .....	15
1.3.2 The Integrated Solution: ICCI .....	15
1.4 Research Objectives and Contributions.....	16
1.4.1 Research Objectives.....	16
1.4.2 Research Contributions.....	17
1.5 Research Questions.....	18
1.6 Significance of the Study.....	19
1.7 Scope and Limitations .....	20
Chapter 2 Related Work .....	21
2.1 Traditional Software Metrics .....	21
2.1.1 Cyclomatic Complexity .....	21
2.1.2 Lines of Code and Size-Based Metrics.....	22
2.1.3 Halstead Complexity Metrics .....	22
2.1.4 Comparative Analysis of Traditional Metrics .....	23
2.2 Cognitive Load Theory in Software Engineering.....	23
2.2.1 Theoretical Foundations .....	23
2.2.2 Empirical Studies on Cognitive Load in Programming.....	24
2.2.3 Behavioural Metrics for Cognitive Load.....	24
2.3 Code Documentation and Comment Quality.....	25
2.3.1 Comment Quality Assessment Frameworks.....	25
2.3.2 Code-Comment Alignment Studies .....	25
2.4 Machine Learning Applications in Code Analysis.....	26
2.4.1 Deep Learning for Code Intelligence .....	26
2.4.2 Reinforcement Learning in Code Generation.....	26
2.5 Automated Code Review and Refactoring Systems.....	27
2.5.1 Interactive Refactoring Approaches .....	27
2.5.2 Existing Code Review Tools .....	27

2.6 Empirical Studies on Code Comprehension .....	28
2.6.1 Identifier Naming Studies .....	28
2.6.2 Consistency and Convention Studies .....	28
2.7 Neuroscience Contributions to Program Comprehension .....	29
2.7.1 Brain Imaging Studies .....	29
2.7.2 Eye-Tracking Studies .....	29
2.8 Research Gap .....	30
2.8.1 Limitations of Existing Approaches .....	30
2.8.2 ICCI's Positioning in the Research Landscape .....	31
Chapter 3 Methodology .....	32
3.1 Research Design .....	32
3.1.1 Research Approach .....	32
3.2 Research Strategy .....	32
3.2.1 Theoretical Framework Development .....	32
3.2.2 Metric Design and Implementation .....	32
3.2.3 Machine Learning Integration .....	33
3.2.4 Empirical Validation .....	33
3.3 Conceptual Framework .....	35
3.4 ICCI Framework Design .....	36
3.5 Component Weight Determination .....	37
3.5.1 Structural Complexity (30% weight) .....	37
3.5.2 Documentation Quality (30% Weight Allocation) .....	38
3.5.3 Naming Quality (40% weight) .....	39
3.6 Component Architecture and Score Calculation .....	40
3.6.1 Source Code Parsing .....	40
3.6.2 Multi-Dimensional Component Analysis .....	41
3.6.3 Scoring and Normalisation .....	42
3.6.4 Weighted Aggregation .....	44
3.6.5 Adaptive Weight Learning .....	45
3.7 Structural Complexity Component .....	47
3.7.1 Cyclomatic Complexity Calculation .....	47
3.7.2 Nesting Depth Analysis .....	48
3.7.3 Method Length Assessment .....	50
3.8 Documentation Quality Component .....	51
3.8.1 Comment Coverage Analysis .....	51
3.8.2 Code-Comment Alignment Assessment .....	53
3.9 Naming Quality Component .....	55
3.9.1 Variable and Parameter Name Analysis .....	55
3.9.2 Method Name Evaluation .....	56

3.9.3 Class-Method Context Evaluation .....	58
3.9.4 Naming Consistency Analysis .....	59
Chapter 4 Results .....	60
4.1 Evaluation Overview .....	60
4.1.1 Participant Demographics .....	60
4.1.2 Evaluation Methodology .....	61
4.2 RQ1: Multi-Dimensional Metric Effectiveness .....	61
4.2.1 Component Analysis for Weight Derivation .....	61
4.2.2 Baseline Configuration: Theoretical Weights (30/30/40) .....	62
4.2.3 Adjusted ICCI Performance (65/25/15 Weights) .....	64
4.2.4 Performance Summary .....	65
4.2.5 RQ1: Summary .....	66
4.3 RQ2: Semantic Alignment Enhancement .....	67
4.3.1 Documentation Quality Assessment .....	67
4.3.2 Model Performance Metrics .....	69
4.3.3 RQ2: Summary .....	70
4.4 RQ3: Adaptive Learning Effectiveness .....	70
4.4.1 Weight Adjustment Outcomes .....	70
4.4.2 Prediction Accuracy Improvements .....	71
4.4.3 RQ3: Summary .....	73
Chapter 5 Discussion .....	74
5.1 Interpretation of Findings .....	74
5.1.1 Multi-Dimensional Integration Value .....	74
5.1.2 Semantic Analysis Advantages .....	76
5.1.3 Adaptive Learning Impact .....	78
5.2 Comparison with Existing Research .....	80
5.2.1 Position ICCI Results Relative to Scalabrino et al. (2019) Findings .....	80
5.2.2 How ICCI Addresses Limitations Identified in Literature .....	81
5.2.3 Agreement/Disagreement with Neurological Studies (Peitek et al.) .....	81
5.3 Theoretical Implications .....	82
5.3.1 Validation of CLT-Based Approach to Comprehensibility .....	82
5.3.2 Cognitive Load Components Mapped to Code Characteristics .....	83
5.4 Practical Implications .....	83
5.4.1 Integration into Development Workflows .....	83
5.4.2 Code Review Enhancement .....	84
5.4.3 Refactoring Prioritization .....	84
5.4.4 Educational Applications .....	85
5.5 Limitations .....	85
5.5.1 Methodological Limitations .....	85

5.5.2 Technical Limitations .....	87
Chapter 6 Conclusion .....	89
6.1 Research Summary .....	89
6.2 Contributions .....	89
6.3 Limitations .....	90
6.4 Future Work .....	91
6.5 Closing Statement .....	92
References .....	93
Appendices .....	95
Appendix A: Dataset Specification .....	95
Appendix B: Survey Instrument .....	97

# List of Figures

Figure 3-1: Conceptual Diagram of ICCI.....	34
Figure 3-2: ICCI Components Architecture Diagram .....	36
Figure 3-3: ICCI Score Calculation Flow.....	44
Figure 4-1: Comparison of CodeBERT and Traditional Documentation Quality Assessment.....	68

# List of Tables

Table 1-1: Comparison of Existing Metrics with ICCI .....	15
Table 1-2: ICCI Mapping with Cognitive Load Theory.....	15
Table 2-1: Strengths and Limitations of Traditional Metrics .....	23
Table 3-1: Final ICCI Score Mapped to an Interpretable Comprehensibility Category.....	44
Table 3-2: Cyclomatic Complexity Scoring Thresholds .....	48
Table 3-3: Nesting Depth Scoring Thresholds .....	49
Table 3-4: Method Length Scoring Thresholds.....	51
Table 4-1: Dataset selection distribution .....	60
Table 4-2: ICCI Component Correlations with Human Ratings .....	61
Table 4-3: Component Regression Performance.....	62
Table 4-4: Correlation Analysis of Baseline ICCI and Traditional Metrics.....	62
Table 4-5: Ordinal Logistic Regression of Baseline ICCI and Traditional Metrics.....	63
Table 4-6: Baseline Classification Accuracy.....	63
Table 4-7: Correlation Analysis of Adjusted ICCI and Traditional Metrics .....	64
Table 4-8: Ordinal Logistic Regression of Adjusted ICCI and Traditional Metrics .....	64
Table 4-9: Classification Performance Comparison.....	65
Table 4-10: Baseline vs Adjusted ICCI - Comparison .....	65
Table 4-11: Semantic Alignment Ordinal Regression Model Comparison.....	67
Table 4-12: Classification Performance Comparison.....	69
Table 4-13: Weight Improvement Comparison.....	70
Table 4-14: Model Fit Comparison .....	71
Table 4-15: Correlation Improvements .....	72
Table 4-16: Prediction Error Analysis.....	72
Table 6-1: Repositories used from GitHub.....	95
Table 6-2: Methods Extracted from GitHub Repositories.....	96
Table 6-3: Participation Demographics .....	97

# List of Algorithms

Algorithm 3-1: Cyclomatic Complexity Calculation .....	47
Algorithm 3-2: Nesting Depth Calculation.....	49
Algorithm 3-3: Method Length Calculation .....	50
Algorithm 3-4: Comment Coverage Calculation.....	52
Algorithm 3-5: Code-Comment Alignment with CodeBERT .....	53
Algorithm 3-6: Code-Comment Alignment with spaCy .....	54
Algorithm 3-7: Variable and Parameter Naming Calculation .....	55
Algorithm 3-8: Method Naming Calculation .....	57
Algorithm 3-9: Class-Method Naming Calculation .....	58
Algorithm 3-10: Naming Consistency Calculation .....	59