



A New Approach to Quantifying Software Code Complexity

L. H. A. N. N. Buddhadasa
(MS23002456)

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

December 2024

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. Dilshan De Silva

Approved for MSc. Research Project:

MSc. 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is entirely my own and not of any other person, unless explicitly acknowledged (including citation of published and unpublished sources).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 in any form to the Sri Lanka Institute of Information Technology or to any other institution for assessment for any other purpose.

Sign:

L. H. A. N. N. Buddhadasa

Date:2024.11.11.....

ABSTRACT

A New Approach to Quantifying Software Code Complexity

Nirmani Buddhadasa

MSc. in Information Technology

Supervisor: Dr. Dilshan De Silva

December 2024

This research presents a novel approach to quantifying software code complexity through the development of a new metric and a web-based tool using Python. Traditional complexity metrics often fall short in addressing modern software challenges such as concurrency, dynamic memory management, and exception handling. The goal is to create a paradigm-independent metric that combines conventional factors with new dimensions for a more comprehensive assessment of code complexity. The tool enables practical application of this metric with an intuitive interface for in-depth code analysis. It includes features like a secure login system, an interactive dashboard for complexity evaluation, and review sections offering detailed feedback on Python code. Data is collected from industry evaluations and Python repositories on platforms like GitHub and Kaggle, ensuring relevance and robustness. The effectiveness of the new metric and tool is assessed by comparing them with traditional complexity metrics, supported by insights from professional testers. Results demonstrate improved accuracy, adaptability, and flexibility in capturing complex software behaviors often overlooked by traditional metrics. The tool consistently outperforms metrics like Cyclomatic Complexity and Weighted Composite Metric in evaluating Python code complexity, providing more nuanced insights into modern challenges such as concurrency and exception handling. The anticipated outcomes highlight the enhanced accuracy and practical relevance of the new metric for assessing complex software systems. The tool serves as a valuable resource for developers, offering deeper insights into Python code complexity and supporting informed decisions in software design, optimization, and maintenance.

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to my supervisor, Dr. Dilshan De Silva, at the Sri Lanka Institute of Information Technology (SLIIT) for his unwavering guidance and support throughout my research. His insightful feedback, encouragement, and valuable advice were crucial to the completion of this project. I am truly thankful for his patience and dedication in helping me navigate through the challenges of my research.

I would also like to extend my sincere thanks to SLIIT for providing the resources and a conducive environment that enabled me to carry out this work successfully.

Lastly, I am deeply grateful to my parents for their constant encouragement, love, and support throughout this journey. Without their belief in me, this accomplishment would not have been possible.

TABLE OF CONTENTS

DECLARATION	ii
ABSTRACT	iii
ACKNOWLEDGEMENT	iv
TABLE OF CONTENTS	v
List of Figures	viii
List of Tables.....	x
Chapter 1 Introduction.....	1
1.1 Problem Background	1
1.2 Proposed Solution	2
1.3 Research Question	3
1.4 Thesis Statement	3
1.5 Aim.....	4
1.6 Objectives.....	4
Chapter 2 Literature Review	5
2.1 Overview of Code Complexity Metrics	5
2.2 Limitations with Existing Metrics.....	23
2.3 Factors Not Included in Existing Metrics	26
2.4 Factors That Should Be Included.....	29
2.5 Research Gap	31
Chapter 3 Methodology.....	34
3.1 Hypothesis	34
3.2 Scope of Research.....	36
3.3 Research Design	37
3.4 New Code Complexity Metric: CodeSculpt Complexity Index (CSI)	39
3.4.1 Name Selection: CodeSculpt Complexity Index (CSI).....	39
3.4.2 Proposed Calculation Method	40
3.4.3 Comparison With existing Metrics.....	48
3.5 Validation Metric Using Weyuker’s Method.....	50
3.6 New Tool: Code Complexity Checker	51
3.6.1 Tool Overview.....	52
3.6.2 Functionality and Components.....	52
3.6.3 High Level Architecture	52
3.6.4 Data Flow Diagram	53
3.6.5 Flow Chart.....	54
3.6.6 Database Design.....	54

3.7 Data Gathering	54
3.8 Resource Requirements.....	56
Chapter 4 Development.....	57
4.1 Development Environment	57
4.1.1 Programming Languages.....	57
4.1.2 Frameworks and Libraries	58
4.1.3 Data Handling and Code repositories	60
4.1.4 Local Development Environment.....	60
4.2 Code Walkthrough	61
4.2.1 Directory Structure	61
4.2.2 Key Functions and Classes.....	62
4.2.3 Data Flow	63
4.2.4 User Interaction and Output Display	64
4.3 Implementation.....	64
4.3.1 Setup.....	64
4.3.2 User Registration Page	66
4.3.3 Login Page.....	67
4.3.4 Dashboard.....	67
4.3.5 Submission History Page.....	68
4.3.6 Report Generation	68
4.4 User Interface	70
4.4.1 User Experience Design	70
4.4.2 Dashboard Features	70
4.4.3 Visual Elements.....	72
4.4.4 Additional Features	72
Chapter 5 Evaluation of Results.....	73
5.1 Evaluation Criteria.....	73
5.2 Test Results of New Code Complexity Metric	74
5.2.1 Purpose of the New Metric.....	74
5.2.2 Manual Calculation Methodology	79
5.3 Comparison with Existing Metrics	92
5.4 Test Results of Code Complexity Tool.....	97
5.5 Industry Feedback Analysis (Questionnaire Results)	98
5.6 Code Samples and Metric Results	100
5.7 Summary of Results.....	129
Chapter 6 Discussion.....	137
6.1 Implications	137

6.2 Limitations	138
Chapter 7 Conclusion	140
7.1 Summary of Findings.....	140
7.2 Contribution	140
7.3 Expert Opinion.....	141
7.4 Future Work	141
Bibliography.....	143
Appendix	148
Appendix 1: Python Code of the Cyclomatic Complexity and New Metric.....	148
Appendix 2: Screenshots of the Code Complexity Checker.....	168
Appendix 3: Flow Chart Diagram for Code Complexity metric (CSI) and Tool.....	169
Appendix 4: Questionnaire.....	172
Appendix 5: Expert Opinion	172

List of Figures

Figure 3. 1: High Level Architecture.....	53
Figure 3. 2: Data Flow Diagram.....	53
Figure 3. 3: ER Diagram	54
Figure 3. 4: Flow Chart Diagram for thee CSI - CFC Calculation	169
Figure 3. 5: Flow Chart Diagram for the CSI - FCS Calculation	170
Figure 3. 6: Flow Chart Diagram for the Code Complexity Checker Tool	171
Figure 4. 1: Registration Page	67
Figure 4. 2: Login Page	67
Figure 4. 3: Dashboard Page	68
Figure 4. 4: Submission History Page	68
Figure 4. 5: Report Generation.....	69
Figure 4. 6: Report Generation PDF.....	69
Figure 4. 7: Dashboard Code Submit and Results Part.....	71
Figure 4. 8: Heatmap and Other Results Showing	71
Figure 4. 9: Dashboard Report Generation.....	72
Figure 4. 10: Settings	168
Figure 4. 11: Message Center.....	168
Figure 4. 12: Activity Log Page	168
Figure 4. 12: Activity Log Page	168
Figure 5. 1: Code Sample for Manual Calculation.....	80
Figure 5. 2: Tool Output for the Figure 5.1 mentioned code.....	98
Figure 5. 3: Questionnaire	100
Figure 5. 4: Tool Out Put for Code Snippet 01	102
Figure 5. 5: Google Chart for Code Snippet 01.....	102
Figure 5. 6: Pie Chart for Code Snippet 01	102
Figure 5. 7: Code Snippet 1.....	103
Figure 5. 8: Tool Out Put for Code Snippet 02	105
Figure 5. 9: Google Chart for Code Snippet 02.....	106
Figure 5. 10: Pie Chart for Code Snippet 02	106
Figure 5. 11: Code Snippet 02.....	107
Figure 5. 12: Tool Out Put for Code Snippet 03	108
Figure 5. 13: Google Chart for Code Snippet 03.....	109
Figure 5. 14: Pie Chart for Code Snippet 03	109
Figure 5. 15: Code Snippet 03.....	110
Figure 5. 16: Tool Out Put for Code Snippet 04	111
Figure 5. 17: Google Chart for Code Snippet 04.....	112
Figure 5. 18: Pie Chart for Code Snippet 04	112
Figure 5. 19: Code Snippet 04.....	113
Figure 5. 20: Code Snippet 05.....	115
Figure 5. 21: Tool Out Put for Code Snippet 05	115
Figure 5. 22: Google Chart for Code Snippet 05.....	116
Figure 5. 23: Pie Chart for Code Snippet 05	116
Figure 5. 24: Tool Out Put for Code Snippet 06	118
Figure 5. 25: Google Chart for Code Snippet 06.....	118
Figure 5. 26: Pie Chart for Code Snippet 06	118
Figure 5. 27: Code Snippet 06.....	119
Figure 5. 28: Tool Out Put for Code Snippet 07	120
Figure 5. 29: Google Chart for Code Snippet 07.....	121

Figure 5. 30: Pie Chart for Code Snippet 07	121
Figure 5. 31: Code Snippet 07.....	121
Figure 5. 32: Pie Chart for Code Snippet 08	123
Figure 5. 33: Google Chart for Code Snippet 08.....	123
Figure 5. 34: Tool Out Put for Code Snippet 08	123
Figure 5. 35: Code Snippet 08.....	125
Figure 5. 36: Tool Out Put for Code Snippet 09	126
Figure 5. 37: Google Chart for Code Snippet 09.....	127
Figure 5. 38: Pie Chart for Code Snippet 09	127
Figure 5. 39: Code Snippet 15.....	128
Figure 5. 40: Downloaded Questionnaire Results CSV File	172
Figure 5. 41: Downloaded Expert Feedback Results CSV File.....	172

List of Tables

Table 3. 1 Comparison with Existing Metrics	49
Table 5. 1: Decision Points of the Cyclomatic Complexity.....	94
Table 5. 2: Weighted Composite Metric	95
Table 5. 3: Results of Code Snippet 01	101
Table 5. 4: Results of Code Snippet 02	105
Table 5. 5: Results of Code Snippet 03	108
Table 5. 6: Results of Code Snippet 04	111
Table 5. 7: Results of Code Snippet 05	114
Table 5. 8: Results of Code Snippet 06	117
Table 5. 9: Results of Code Snippet 07	120
Table 5. 10: Results of Code Snippet 08	122
Table 5. 11: Results of Code Snippet 09	126
Table 5. 12: Results Summary of All Code Snippets	129