

**AN ANALYSIS OF GOF SOFTWARE DESIGN
PATTERNS ON SOFTWARE MAINTAINABILITY IN
MICRO SERVICES**

R. M. H. S. B. Rathnayake
209370G

Degree of Master of Science

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

June 2023

**AN ANALYSIS OF GOF SOFTWARE DESIGN
PATTERNS ON SOFTWARE MAINTAINABILITY IN
MICRO SERVICES**

R. M. H. S. B. Rathnayake
209370G

Thesis submitted in partial fulfillment of the requirements for the degree
Master of Science in Computer Science

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

June 2023

DECLARATION OF THE CANDIDATE AND SUPERVISOR

I declare that this is my own work, and this thesis/dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature:

Date: 26/06/2023

Name: R.M.H.S.B.Rathnayake.

The supervisor/s should certify the thesis/dissertation with the following declaration.

The above candidate has carried out research for the Masters thesis/dissertation under my supervision. I confirm the declaration made above by the student is true and correct.

Signature of the supervisor:

Date: 26/06/2023

Name of the supervisor: Eng. Prof. Indika Perara.

Head - Dept. of Computer Science & Engineering, Faculty of Engineering

University of Moratuwa, Moratuwa 10400, Sri Lanka.

ACKNOWLEDGEMENT

I express my sincere gratitude to my supervisor Eng. Prof. Indika Perara for the guidance he provided me throughout this research. Their knowledge, expertise and encouragement helped me to complete this research.

I would also like to thank the companies that helped to gather data for this research. Without their help this research wouldn't have been possible.

Also, I would like to thank my colleagues and friends who helped me with gathering data and for providing me with technical advice regarding the tools and technologies used in this project.

Also, I would like to thank my family for their support in completing this research.

Abstract

The purpose of this paper is to identify how gang of four design patterns impact the maintainability of micro services-based systems. Design patterns were introduced as solutions to common problems that occur in programming. These are supposed to improve the maintainability of a system by improving the code quality. But with modern programming languages, frameworks, and integrated development environments, whether these patterns serve their purpose is a question that is not fully addressed. This paper proposes a tool that can be used to identify whether use of a specific design pattern by a specific developer for a particular micro service-based project can improve its maintainability or not. To do this a model has been created by analyzing enterprise micro service-based applications and gathering data from developers who were involved in the development of those projects. This data is used to create models for maintainability metrics coupling, lack of cohesion, duplication and cyclomatic complexity. This tool will help to decide whether the system is more maintainable with or without the use of selected design patterns. This helps better decision making in deciding how to write new code or refactoring existing code. Results of this research have shown that lack of cohesion is not affected by developer experience, design patterns or the language used in enterprise micro service-based applications. Cyclomatic complexity was only affected by the language used. Also, use of certain design patterns decreased the coupling in the system. But some of the design patterns caused duplications to be increased. So, the results showed that use of design patterns can have a negative and positive impact on the maintainability of a microservice depending on the design pattern used. This research also emphasizes the importance of code review process and code quality analysis automation.

Keywords: GOF design patterns, micro services, software maintainability

TABLE OF CONTENTS

Declaration of The Candidate and Supervisor	i
Acknowledgement.....	ii
Abstract	iii
Table of Contents	iv
List of Figures	viii
List of Tables.....	x
List of Abbreviations.....	xii
List of Appendices	xiii
1. Introduction	1
1.1 Software maintainability	1
1.2 GOF Design patterns.	1
1.3 Micro services	2
1.4 Background	3
1.5 Problem in brief.....	5
1.6 Research Motivations	6
1.7 Aim and Objectives	6
1.7.1 Objectives.	6
1.8 Overview of the proposed solution.	7
1.9 Chapter synopsis.....	8
2. Related Work	9
2.1 A Controlled Experiment in Maintenance Comparing Design Patterns to Simpler Solutions.	9
2.2 Maintainability Index	11
2.3 SIG maintainability model	12
2.4 MOOD metrics	13
2.5 Coupling and cohesion (towards a Valid Metrics Suite for Object-Oriented Analysis and Design).....	14
2.6 Myth or Reality Analyzing the effect of Design Patterns of Software Maintainability	15
2.7 What Do We Know about the Effectiveness of Software Design Patterns?	15
2.8 Impact of design patterns on software maintainability.	15
2.9 An application of Bayesian network for predicting object-oriented software maintainability.....	16

2.10 A Comparative Literature Survey of Design Patterns Impact on Software Quality	17
2.11 An Empirical Investigation on the Impact of Design Pattern Application on Computer Game Defects	17
2.12 An empirical study on the evolution of design patterns [19].	17
2.13 Prediction in Multiple Regression [20]	18
2.14 Research Gap.....	18
2.15 Chapter synopsis.....	19
3. Proposed Solution	20
3.1 Identifying maintainability parameters.	20
3.2 Maintainability metrics.....	21
3.3 Design patterns used in the analysis.....	22
3.4 Data analysis process.....	22
3.5 Maintainability prediction prototype.....	24
3.6 Tools used in the research.	24
3.6.1 Sonar Cloud	24
3.6.2 CodeMR.....	24
3.6.3 Google Forms	25
3.6.4 Excel	25
3.7 Chapter synopsis.....	25
4. Conceptual Evaluation	26
4.1 Data collection and conversion	26
4.1.1 Code parameters data collection	26
4.1.2 Code parameters data conversion.	27
4.1.3 Design pattern density data collection	29
4.1.4 Developer experience and Language data collection.	29
4.2 Initial Data analysis and preparation.	30
4.2.1 Design pattern density	30
4.2.2 Duplication code metric data	33
4.2.3 Coupling code metric data.	34
4.2.4 Cyclomatic complexity code metric data.....	35
4.2.5 Lack of Cohesion code metric data.....	36
4.3 Regression analysis	37
4.3.1 Coupling introduction	37

4.3.2	Coupling regression analysis.	38
4.3.3	Coupling regression analysis results.	40
4.3.4	Coupling best line fit analysis.	43
4.3.5	Duplication introduction.	44
4.3.6	Duplication regression analysis.	45
4.3.7	Duplication regression analysis results.	47
4.3.8	Duplication best line fit analysis.	50
4.3.9	Lack of Cohesion introduction.	52
4.3.10	Lack of cohesion regression analysis.	53
4.3.11	Lack of cohesion regression analysis results.	54
4.3.12	Cyclomatic Complexity introduction.	60
4.3.13	Cyclomatic complexity regression analysis.	60
4.3.14	Cyclomatic complexity regression analysis results.	63
4.3.15	Cyclomatic complexity best line fit analysis.	64
4.4	Chapter synopsis.	65
5.	Implementation.	66
5.1	High level architecture of the proposed validation prototype.	66
5.2	User Interface.	67
5.2.1	Low level design of the UI application.	68
5.2	API Server.	69
5.2.1	API specification of the API server.	69
5.2.2	Low level design of the API server.	73
5.2.3	Metric calculation process in API server.	74
5.3	Database.	79
5.3.1	Database schema.	79
5.3.2	Database schema initialization.	79
5.3.3	Data insertion to the database.	80
5.4	Chapter synopsis.	82
6.	Results and Discussion.	83
6.1	Reasoning for design pattern density results.	83
6.2	Reasoning for coupling analysis results.	84
6.3	Reasoning for duplication analysis results.	86
6.4	Reasoning for lack of cohesion analysis results.	88

6.5 Reasoning for cyclomatic complexity analysis results.....	89
6.6 Prototype maintainability prediction results.....	91
6.7 Overall impact of design patterns in maintainability.	92
6.8 Chapter synopsis.....	92
7. Conclusion	93
7.1 Achievement of objectives	93
7.1.1 Identify whether the implementation of Gang of Four (GoF) design patterns contributes to an enhancement in maintainability within systems based on microservices architecture.	93
7.1.2 Develop a tool that can predict the maintainability using maintainability parameters.....	94
7.2 Recommendations	95
7.3 New skills obtained during the research.....	95
7.4 Challenges and Difficulties	95
7.5 Future improvements.....	96
7.6 Closing remarks.....	97
Reference List	98
Appendix-A.....	103
Appendix-B	106

LIST OF FIGURES

Figure 1	Maintainability parameter generation process in summary.....	22
Figure 2	Metrics Calculation process.....	23
Figure 3	Design pattern density chart	31
Figure 4	Design pattern knowledge of developer's chart	32
Figure 5	Duplication per line of code for micro services chart.....	34
Figure 6	Coupling per LOC for micro services chart.	35
Figure 7	Cyclomatic complexity per LOC for micro services chart.....	36
Figure 8	Lack of Cohesion per LOC for micro services chart.....	37
Figure 9	Coupling residual plot.	40
Figure 10	Factory design pattern usage chart for coupling.....	41
Figure 11	Facade design pattern density chart for coupling.	41
Figure 12	Average experience for developer chart for coupling.	42
Figure 13	Strategy pattern density chart for coupling.....	42
Figure 14	Factory method pattern density line plot for coupling.	43
Figure 15	Avg experience per developer line fit plot for coupling.....	43
Figure 16	Facade pattern density line fit plot for coupling.....	44
Figure 17	Strategy pattern density line fit plot for coupling.....	44
Figure 18	Duplication residuals plot.....	48
Figure 19	Facade pattern density for duplication.....	48
Figure 20	Strategy pattern usage for duplication.	49
Figure 21	Observer pattern density for duplication.	49
Figure 22	Builder pattern usage for duplication.	50
Figure 23	Facade pattern usage line fit plot for duplication.	51
Figure 24	Strategy pattern density line fit plot.....	51
Figure 25	Observer pattern density line fit plot for duplication.....	52
Figure 26	Builder pattern density line fit plot for duplication.	52
Figure 27	Avg experience per developer for lack of cohesion.	56
Figure 28	Lines of code for lack of cohesion.....	56
Figure 29	Factory method pattern density for lack of cohesion.	57
Figure 30	Facade pattern density for lack of cohesion.	57
Figure 31	Observer pattern density for lack of cohesion.	58
Figure 32	Strategy pattern density for lack of cohesion.	58
Figure 33	Builder pattern density for lack of cohesion.....	59
Figure 34	Language for lack of cohesion.....	59
Figure 35	Factory and builder pattern for lack of cohesion.	60
Figure 36	Language for cyclomatic complexity.	64
Figure 37	Language line for plot for cyclomatic complexity.	65
Figure 38	Component diagram for validation tool.....	66
Figure 39	Sequence diagram for the modules.....	67
Figure 40	React component diagram for UI module.	68
Figure 41	User interface of the validation tool.	69
Figure 42	Class diagram of the API server.	73
Figure 43	Flow diagram of the metrics calculation process.	75

Figure 44 Entity relationship diagram for the database.	79
Figure 45 Design pattern usage chart.	83
Figure 46 Design pattern knowledge chart.	84
Figure 47 Coupling residual plot.	85
Figure 48 Duplication residuals plot.	87
Figure 49 Lack of cohesion scatter plot without lines of code.	88
Figure 50 Lack of cohesion scatter plot for LOC.	89
Figure 51 Language for cyclomatic complexity.	90
Figure 52 Developer responses on the impact of design patterns.	92

LIST OF TABLES

Table 1	Controlled experiment in maintenance comparing design patterns to simpler solutions expectation and result.	10
Table 2	Classification of design patterns used in this model.	22
Table 3	Tools used for data collection.	23
Table 4	Collected parameter values.	26
Table 5	Converted parameter values.	27
Table 6	Final parameter values for analysis.	28
Table 7	Design pattern density table.	29
Table 8	Developer experience and language used.	29
Table 9	Design pattern knowledge of developers.	31
Table 10	Duplication per line of code per micro service table.	33
Table 11	Coupling per LOC for micro services table.	34
Table 12	Cyclomatic complexity per LOC for micro services table.	35
Table 13	Lack of Cohesion per LOC for micro service table.	36
Table 14	Variable data for first analysis for coupling.	38
Table 15	P-values for first analysis for coupling.	38
Table 16	Variable values for second analysis for coupling.	39
Table 17	P-values for second analysis for coupling.	39
Table 18	Regression statistics for coupling.	40
Table 19	Coefficient values for coupling.	42
Table 20	Data for first regression analysis for duplication.	45
Table 21	P-values for first regression analysis for duplication.	46
Table 22	Data for second regression analysis for duplication.	46
Table 23	P-values of second regression analysis for duplication.	47
Table 24	Regression statistics for duplication.	47
Table 25	Coefficients for duplication.	50
Table 26	Data for first regression analysis of lack of cohesion.	53
Table 27	P-values from the first analysis for lack of cohesion.	54
Table 28	Regression statistics for the lack of cohesion analysis.	54
Table 29	Data for second analysis for lack of cohesion.	54
Table 30	P-values from the second analysis of lack of cohesion.	55
Table 31	Regression statistics for the second analysis of lack of cohesion.	55
Table 32	Data for first analysis of cyclomatic complexity.	60
Table 33	P-values from the first analysis of cyclomatic complexity.	61
Table 34	Data for second analysis of cyclomatic complexity.	61
Table 35	P-values for second analysis of cyclomatic complexity.	62
Table 36	Data for third analysis of cyclomatic complexity.	62
Table 37	P-values for third analysis of cyclomatic complexity.	63
Table 38	Regression statistics for cyclomatic complexity.	63
Table 39	Coefficients for cyclomatic complexity.	64
Table 40	Predicted metric change for test services.	91
Table 41	Predicted maintainability change for test services.	91
Table 42	Actual metric change for test service.	91

Table 43 Actual maintainability change for test services.	91
Table 44 Design pattern maintainability impact.	94
Table 41 Maintainability metric references.	94

LIST OF ABBREVIATIONS

Abbreviation	Description
AHEF	Attributed hiding effectiveness factor
AIF	Attribute inheritance factor
ALT	Alternative version without patterns
API	Application programming interface
Avg	Average
CAM	Cohesion among methods
CBO	Coupling between objects
CC	Cyclomatic complexity
CF	Coupling factor
COM	Percentage of comments per module
GOF	Gang of four
HTML	Hypertext markup language
HTTP	Hypertext transfer protocol
HV	Halstead volume
IDE	Integrated development environment
IIF	Internal inheritance factor
LCOM	Lack of cohesion methods
LOC	Average number of lines per module
LOC	Lines of code
MI	Maintainability index
MIF	Method inheritance factor
MOOD	Matrices for object-oriented design
OHEF	Operation hiding effectiveness factor
PAT	Pattern version
PF	Polymorphism factor
PPF	Parametric polymorphism factor
P-value	Probability value
RFC	Response for a class
UI	User interface

LIST OF APPENDICES

Appendix	Description	Page
Appendix - A	Survey for data collection from developers	103