

**SHARING AND PRESERVING CODING BEST
PRACTICES THROUGH PROGRAMMER DATA
ANALYTICS**

Samiththa Bashani

(168210M)

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

June 2018

SHARING AND PRESERVING CODING BEST PRACTICES THROUGH PROGRAMMER DATA ANALYTICS

Jasing Pathirana Samiththa Bashani

(168210M)

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

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

June 2018

DECLARATION

I declare that this is my own work and this thesis 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.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books)

Signature:

Date:

J.P. Samiththa Bashani

The above candidate has carried out research for the Masters thesis under my supervision.

Name of the supervisor: Dr. Indika Perera

Signature of the supervisor:

Date:

Abstract

Sharing and preserving coding best practices among the developers are becoming an important objective of software development life cycle. Because violations on coding best practices may lead to catastrophic events which are costly and time consuming. There has been numerous researches done in order to mitigate the issues related to bad coding practices. One of the most challenging tasks towards mitigating this is to identify the skill level of the developers, coding patterns and likelihood for bad coding practices. The widely used methods for this are conducting one on one interview with the developers and review developers work.

This particular research tried to contribute to the field of software architecture by analyzing the feasibility of using machine data to identify the developer coding patterns and related data and provide a mechanism to enhance the skills of a developer. By doing that it makes sure an organization can share and preserve the coding best practices within an organization.

This research focused on developing a parsing mechanism to collect those data from various file formats and types. For this research scope it focused on the static code analysis tool called FindBugs and log data. A successful parser of logs formatted in XML has been developed. A central data storage architecture has been developed in order to capture data from various sources which are different from each other.

Collected data analyzed to generate information about the developers' pattern in doing mistakes and coding styles. To prove that analyzing programmer data for a significant period can predict their abilities and weaknesses an evaluation has been carried out. The evaluation compare data from developer spot interviews with developers' log analyzed data. With those comparisons it identified log data results can match the interview results in an 80% success rate.

ACKNOWLEDGEMENT

I would like to express my profound gratitude and deep regards to my supervisor Dr. Indika Perera for his exemplary guidance, valuable feedback and constant encouragement throughout the duration of the project. His valuable suggestions were of immense help throughout this work. Also I am grateful for the support and advice given by Dr. Malaka Walpola, by encouraging this research.

Finally I would like to thank the academic and nonacademic staff of Department of Computer Science and Engineering and colleague of MSC'16 for the support and encouragement given.

TABLE OF CONTENTS

DECLARATION.....	i
Abstract	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS.....	iv
LIST OF FIGURES	vii
LIST OF TABLES	viii
LIST OF ABBREVIATIONS.....	ix
Chapter 1 INTRODUCTION	1
1.1 Background.....	2
1.2 Problem Statement	3
1.3 Motivations to Solve the Problem.....	4
1.4 Proposed Solution	4
1.5 Research Objectives	5
1.6 Overview of the Document	5
Chapter 2 LITERATURE REVIEW	7
2.1 Code Quality	8
2.1.1 Code Quality to Measure Quality of the System	9
2.1.2 Improving Code Quality	10
2.1.3 Improving Code Quality Using Gamification	10
2.1.4 Improving Code Quality Using Code Reviews and Analysis	14
2.2 Coding Best Practices	15
2.2.1 Importance of Best Coding Practices.....	15
2.3 Code Review and Code Analysis.....	17
2.3.1 Static Code Analysis.....	17
2.3.2 Importance and Benefits of Static Code Analysis	17
2.3.3 Manual Review	19
2.3.4 Automated Static Code Analysis	21
2.3.5 Source Code Analysis Tools.....	21
2.3.6 FindBugs.....	24
2.3.7 Static Code Analysis for Other Researches	26

2.3.8 Static Code Analysis with Classification Methods.....	26
2.3.9 Source Code Analysis to Remove Security Vulnerabilities in Java Socket Programs	28
2.4 Eclipse IDE.....	31
2.4.1 Eclipse Plug-ins	31
Chapter 3 METHODOLOGY.....	32
3.1 Research Scope for the Solution Architecture	33
3.2 Solution Architecture	34
3.3 Evaluation Methodology	35
Chapter 4 SOLUTION ARCHITECTURE AND IMPLEMENTATION.....	37
4.1 Analyzing Different Formats of Machine Generated Data.....	38
4.2 Reading XML Files.....	39
4.2.1 JAXB for XML Parsing.....	39
4.2.2 Java Document Object Model for XML Parsing.....	41
4.2.3 Reading JSON Files.....	41
4.3 Solution Architecture	42
4.3.1 Log Collector	43
4.3.2 Persistence DB.....	43
4.3.3 Collector API.....	43
4.3.4 Data Analyzer	44
4.3.5 Statistics Database	45
4.4 View Layer	45
4.5 Using Collected Statistics To Preserve Coding Best Practices	46
Chapter 5 EVALUATION.....	47
5.1 Collect Machine Data	48
5.2 Identify Potential Types of Information to Be Generated	49
5.3 Evaluate Collected Machine Data and Derive the Information Required.....	50
5.4 Conduct One on One Interviews	53
5.5 Comparison of the Derived Datasets	54
Chapter 6 CONCLUSION	61
6.1 Research Contribution.....	62
6.2 Future Work and Conclusion	62

REFERENCES	64
APPENDIX A	67
APPENDIX B	70

LIST OF FIGURES

Figure 2-1: The SIG Quality Model maps source code measurements onto ISO/IEC 9126 quality characteristics.....	10
Figure 2-2: Types of Game thinking.....	11
Figure 2-3: Conceptual architecture for gamification in SDLC.....	13
Figure 2-4: Badge pop up.....	14
Figure 2-5: Six categories of achievements	14
Figure 2-6: Comparison of costs to fix defects found at different stages of the development lifecycle	19
Figure 2-7: Flow of types of reviews that increase formality.	20
Figure 2-8: Architectural overview of the Collaborative Code Review Plug-in	27
Figure 2-9: Classification Process.....	28
Figure 2-10: Issues Panel and Code Editor displaying Details of a Specific Security Issue.	29
Figure 3-1: Static Code Analysis Process	33
Figure 3-2 Proposed System Architecture	35
Figure 4-1: Marshalling in JAXB	39
Figure 4-2: Unmarshalling in JAXB	40
Figure 4-3 : Solution Architecture	42
Figure 4-4 : Plug-in View1	46
Figure 4-5 : Plug-in View2	46
Figure 5-1: Comparison chart of bug instances	52
Figure 5-2 : Comparison of java language related best practices knowledge.....	54
Figure 5-3 : Comparison of bug instances related to basic language features	55
Figure 5-4 : Comparison of the bug instances related to other java related feature sections.....	56

LIST OF TABLES

Table 2-1: Static code analysis tools	22
Table 4-1: Bug Categories	44
Table 5-1 : Selected categories for the evaluation	49
Table 5-2: Overall data collected for 8 weeks	52
Table 5-3: Developer rankings in various categories.....	53
Table 5-4: Number of bug instances in different java language feature sections	55
Table 5-5 : Rated comparison of interview results and log analysis results	60

LIST OF ABBREVIATIONS

Abbreviation	Description
SCA	Source Code Analyzer
SIG	Software Improvement Group
SDLC	Software Development Life Cycle
SDK	Software Development Kit
IDE	Integrated Development Environment