UNIVERSITI PUTRA MALAYSIA

IMPROVING CONSISTENCY OF UML AND ITS IMPLEMENTATION USING REVERSE ENGINEERING APPROACH

VASANTHI A/P KALIAPPAN

FSKTM 2018 38

IMPROVING CONSISTENCY OF UML AND ITS IMPLEMENTATION USING REVERSE ENGINEERING APPROACH

By

VASANTHI A/P KALIAPPAN

Thesis Submitted to the School of Graduate Studies, University Putra Malaysia, in Fulfilment of the Requirements for the Degree of Master of Computer Science

January, 2018

TABLE OF CONTENTS

ABSTRACT v
ABSTRAK vi
ACKNOWLEDGEMENTS vii
DECLARATION FORM viii
LIST OF TABLES ix
LIST OF FIGURES x
LIST OF ABBREVIATIONS xii
CHAPTER 1: INTRODUCTION 1
1.1 Introduction 1
1.2 Research Background and Motivation 1
1.3 Problem Statement 2
1.4 Project Objectives 3
1.5 Project Contributions 4
1.6 Scope of Project 4
1.7 Dissertation Outline 5
CHAPTER 2: LITERATURE REVIEW 7
2.1 Introduction 7
2.2 UML Models in Inconsistency Management 7
2.3 UML Model Transformation 10
2.4 C# Programming Language 13
2.5 Reverse Engineering of Source Code 14
2.6 Consistency Checking 16
2.6.1 Horizontal Consistency Checking 17
2.6.2 Vertical Inconsistency Checking 21
2.7 Related Work in Inconsistency Management 23
2.8 Summary 24
CHAPTER 3: PROJECT METHODOLOGY 25
3.1 Introduction 25
3.2 Research Methodology 25
3.2.1 Theoretical Study 25
3.2.2 Identify and Analyse Requirements 26
3.2.3 Design Prototype 26
3.2.4 Prototype Development 27
3.2.5 Validate Prototype 27
3.2.6 Draw Conclusion 27
3.3 Project Requirements 27

3.3.1 Software Requirements 27
3.3.2 Hardware Requirements 28
3.4 Summary 28
CHAPTER 4: DESIGN AND IMPLEMENTATION 29
4.1 Introduction 29
4.2 Prototype Design 29
4.2.1 Architecture of UCCCT 29
4.2.2 Design Pattern 30
4.2.3 Flowchart of UCCCT 31
4.2.4 Use Case Diagram of UCCCT 31
4.3 Overall Workflow 33
4.3.1 Step 1: Processing of XMI of UML Diagrams 34
4.3.2 Step 2: Reverse Engineering of Source Code 42
4.3.3 Step 3: Perform Consistency Checking 46
4.3.4 Step 4: Display Consistency Checking Result 50
4.4 Implementation 51
4.4.1 Interface Design 51
4.5 Summary 54
CHAPTER 5: EVALUATION AND RESULT 55
5.1 Introduction 55
5.2 Evaluation of UCCCT 55
5.3 Experiment Design 56
5.4 Result Analysis 57
5.4.1 Effectiveness 57
5.4.2 Usability 58
5.5 Summary 61
CHAPTER 6: CONCLUSION AND FUTURE WORK 62
6.1 Introduction 62
6.2 Overview 62
6.3 Future Work 62
6.4 Conclusion 64
REFERENCES 65
APPENDICES 68
APPENDIX A: UCCCT INCONSISTENCY DETECTION PROGRAM 68
APPENDIX B: TEST CASE - C# PROGRAM FOR ATM SYSTEM 76
APPENDIX C: TEST CASE - UML DESIGN MODELS FOR ATM SYSTEM 87

ABSTRACT

Software development deals with various changes and evolution that cannot be avoided and it is an important activity in software life cycle because development processes are vastly incremental and iterative. In Model Driven Engineering, inconsistency between model and its implementation has huge impact on the development process in terms of added cost, time and effort. The later the inconsistencies are found, it could add more cost to the project. Thus, this project aim to propose a mechanism and to develop a tool that can improve consistency between UML design models and its C# implementation using reverse engineering approach. A list of informal consistency rules are set to check vertical and horizontal consistencies between structural (class diagram) and behavioural (sequence diagram and use case diagram) UML models against the implemented C# source code. The work deals with reverse engineering of source code using .NET Reflection API and parsing of UML XMI file using C# XMLReader. The inconsistency found between design diagram and source code are presented in textual description and visualized in tree view structure. The project is evaluated via end user evaluation. The contribution of this project is to aid software developers to maintain design models consistency in a faster and correct way and to guide them to take measures to not let design models and source code drift apart.

ABSTRAK

Perubahan dan evolusi di dalam pembangunan perisian tidak dapat dielakkan dan ia merupakan aktiviti penting dalam kitaran hayat perisian kerana proses pembangunan perlu ditambah baik secara berulang. Di dalam pembangunan berasaskan model, model yang tidak konsisten dengan pelaksanaannya mempunyai kesan besar terhadap proses pembangunan dari segi kos tambahan, masa dan tenaga. Sewaktu projek segera diperlukan dalam jangkamasa pendek, salah faham konsep dan kecuaian pengaturcara boleh menyebabkan model and program mudah menjadi tidak konsisten. Inkonsistensi yang dijumpai pada fasa akhir kitar hayat pembangunan perisian akan menambah kos berlebihan kepada projek. Oleh itu, projek ini bertujuan untuk mencadangkan mekanisme dan membangunkan applikasi yang boleh meningkatkan konsistensi di antara model UML dan pelaksanaan kod sumber menggunakan pendekatan kejuruteraan balikan. Peraturan tidak formal digunakan untuk memeriksa konsistensi di antara model dan kod sumber C#. UML di dalam format XMI diproses mengunakan C# XMLReader dan .NET Rekfleksi API digunakan untuk pendekatan kejuruteraan balikan. Inkonsistensi yang dijumpai akan dipaparkan kepada pengguna aplikasi melalui keterangan teks dan digambarkan dalam struktur paparan pokok. Projek ini dinilai menggunakan penilaian pengguna akhir. Projek ini dijangka membantu pemaju perisian untuk mengekalkan konsistensi model reka bentuk dengan cara yang lebih pantas dan betul dan membimbing mereka untuk mengambil langkah-langkah untuk tidak membiarkan model reka bentuk dan kod sumber tidak konsisten.

ACKNOWLEDGEMENTS

All thanks and praise goes to God for His will and for bestowing me with health, time and interest to further my study.

Many thanks to Dr. Norhayati Mohd Ali for being my supervisor during master degree who provided assistance, guidance and constructive comments during the process of completing this project.

Thanks to all my lecturers and friends for their support, commitment, knowledge sharing and working together throughout the study in accomplishing this Master Degree.

Lastly, countless thanks to my mother Madam Nacheammah, sisters and friends for their never ending love, moral support and guidance.

DECLARATION FORM

I hereby confirm that:

- This thesis is my original work.
- Quotations, illustrations and citations have been duly referenced.
- This thesis has not been submitted previously or concurrently for any other degree at any other institutions.
- Intellectual property from the thesis and copyright of thesis are fully-owned by University Putra Malaysia, as according to the University Putra Malaysia (Research) Rules 2012; written permission must be obtained from supervisor and the office of Deputy
- Vice-Chancellor (Research and Innovation) before thesis is published (in the form of written, printed or in electronic form) including books, journals, modules, proceedings, popular writings, seminar papers, manuscripts, posters, reports, lecture notes, learning modules or any other materials as stated in the University Putra Malaysia (Research) Rules 2012.
- There is no plagiarism or data falsification/fabrication in the thesis, and scholarly integrity is upheld as according to the University Putra Malaysia
- (Graduate Studies) Rules 2003 (Revision 2012-2013) and the University Putra
- Malaysia (Research) Rules 2012. The thesis has undergone plagiarism detection software.

Signature: _____

Date:

Name and Matric No: VASANTHI A/P KALIAPPAN, GS43875

Signature: _____

DR NORHAYATI MOHD ALI (Supervisor)

LIST OF TABLES

Table	Page
Table 2.1: Description of UML diagrams	9
Table 2.2: Summary of literature review	23
Table 3.1: List of software components required	28
Table 4.1: Type of UML elements extracted from XMI file	36
Table 4.2: Meta data extracted from class diagram	38
Table 4.3: Meta data extracted from sequence diagram	39
Table 4.4: Meta data extracted from use case diagram	40
Table 4.5: Meta data extracted from source code	44
Table 5.1: Experiment results	58
Table 5.2: Questionnaire for usability testing	59

LIST OF FIGURES

Figure	Page
Figure 2.1: Overview of UML diagram	8
Figure 2.2: Horizontal inconsistency rules between design models	19
Figure 2.3: Overview of model consistency management	20
Figure 2.4: Horizontal consistency rules for multi view diagrams	21
Figure 3.1: Project Methodology	25
Figure 4.1: Architecture of UCCCT prototype	30
Figure 4.2: Class diagram for Singleton design pattern	31
Figure 4.3: Flowchart for UCCCT prototype	32
Figure 4.4: Use case diagram for UCCCT prototype	32
Figure 4.5: Overall UCCCT process flow	33
Figure 4.6: Steps to process XMI file	34
Figure 4.7: XMI file exported from modelling tool for a class diagram	35
Figure 4.8: Meta model of topological class diagram	37
Figure 4.9: Meta model of topological sequence diagram	39
Figure 4.10: Meta model of topological use case diagram	40
Figure 4.11: Tree View structure for design class diagram	41
Figure 4.12: Tree View structure for design sequence diagram	41
Figure 4.13: Tree View structure for design use case diagram	42
Figure 4.14: Framework to reverse engineer source code	42
Figure 4.15: Dependency relationship	45

Figure 4.16: Inheritance relationship 45

Figure 4.17: Association relationship 46

Figure 4.18: Inconsistency between class diagram and sequence diagram 49

Figure 4.19: Main screen of UCCCT Prototype 51

Figure 4.20: Browse C# project folder 51

Figure 4.21: Browse for XMI file 52

Figure 4.22: Consistency checking result for class diagram 52

Figure 4.23: Consistency checking result for sequence diagram 52

Figure 4.24: Consistency checking result for use case diagram 53

Figure 4.25: Textual description of inconsistency 53

Figure 4.26: Print consistency checking result 53

Figure 4.27: View UCCCT consistency rules 54

Figure 5.1: Result for Usability Questions 59

Figure 5.2: Result for usability points 60

Figure 5.3: Result for value of the tool questions 60

Figure 5.4 Result for overall value 61

LIST OF ABBREVIATIONS

UML	Unified Modelling Language
PIM	Platform Independent Models
MDD	Model Driven Development
MDA	Model Driven Architecture
UCCCT	UML-Code Consistency Checking Tool
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language Transformations
XMI	XML Metadata Interchange

CHAPTER 1

INTRODUCTION

1.1 Introduction

This chapter introduces research background and motivation of this project. It also presents problem statements, project objectives, methodology and expected results and contributions of this project. Lastly the chapter concludes with thesis organization.

1.2 Research Background and Motivation

Software life cycle deals with various changes either in software operating environment or requirements. Software evolution cannot be avoided and it is an important activity in software life cycle because development processes are vastly incremental and iterative. Three explicit maintenance activities in software evolution are corrective to fix defect, adaptive to adapt new technologies and environment, and perfective to enhance and improve software quality. In software development, inconsistencies between architectural instruments and the implemented source code might occur due to erroneous implementation of the design architecture or the separate and uncontrolled changes or amendments in the code (Selim Ciraci, Hasan Sozer & Bedir Tekinerdogan, 2012).

To plan and repair these inconsistencies, software developers have to interfere their workflow further to reinvestigate the model changes that contribute to these inconsistencies. Other than fixing inconsistencies, software

developers also have to fix other model changes that were dependent on the erroneous model elements (Alexander Egyed, 2011).

In times where development schedule and timeline are tight or urgently required projects, manual inconsistency detection and fixing may easily breach model consistency conformance due to errors and mistakes made by human or wrong understanding of the model. According to Michael John Decker, Kyle Swartz, Michael L. Collard & Jonathan I. Maletic (2016), the manual recovery of UML class diagrams is a time consuming and expensive operation, which led industries lack of interest in maintenance activities. Also, their study found that most automatic reverse-engineering tools perform poorly. The tools mostly focused on producing simple class diagrams whereby design abstractions were not represented properly and correctly.

In such scenarios, checking consistency between a design model and its implementation is much required to make sure that function of models are implemented as they should be during various changes in software lifetime. Thus, consistency checking can also help in the understanding models, by implementing its design properties which helps developers to use model driven design approaches more effectively.

1.3 Problem Statement

Inconsistencies between UML model and source code could occur due to various changes implemented during the project's lifetime at source code level but design models were not updated accordingly due to constraints like time, money, resources and separate and uncontrolled evolution (Selim Ciraci, Hasan Sozer & Bedir Tekinerdogan, 2012).

Based on IEEE 2016 Programming Language Spectrum rating, despite being one of the most popular object oriented language among software developers, it is observed that recent researches in inconsistency management has more attention for Java and C++ compared to C#. This has been studied and synthesized in a systematic critique conducted by Raja Sehrab Bashira, Sai Peck Lee, Saif Ur Rehman Khan, Victor Chang & Shahid Farid (2016). Furthermore, it was found that most researches in existing literatures focused more on class, state chart and sequence diagrams compared to use case diagram. Combination of class diagram, use case diagram and sequence diagram are not explored much. The study also reveals that majority of literature for model inconsistencies were done using forward engineering. In contrast to the vertical consistency problems, horizontal consistency problems were more emphasized in studies and researches.

Therefore, this project aim to find a mechanism and to create a tool that can improve consistency between class diagram, use case diagram and sequence diagram and its C# implementation using reverse engineering approach. Informal consistency rules will be adopted to detect and diagnose vertical and horizontal inconsistencies.

1.4 Project Objectives

The objective of this project is to propose a mechanism to detect inconsistencies between structural (class diagram) and behavioural (use case diagram and sequence diagram) UML models against its C# source code implementation using reverse engineering approach. A prototype tool will be

designed and developed to implement the proposed mechanism. The prototype tool will be evaluated via end user evaluation.

1.5 Project Contributions

The main contribution of this research is to propose a mechanism and to develop a prototype tool that can improve consistency between structural (class diagram) and behavioural (use case diagram and sequence diagram) UML models and its C# implementation using reverse engineering approach. A consistency checker tool will be developed to

i) Read UML design class diagram, sequence diagram and use case diagram.

ii) Read C# source code and extract implemented class model

iii) Detect vertical and horizontal inconsistencies between (i) and (ii) using informal consistency rules.

iv) Generate a textual description and tree view visualization of detected inconsistencies.

The outcome of this project work is expected to assist developers to use model driven design approaches more effectively. It is also expected to fill the gaps in model inconsistency management as mentioned in problem statement.

1.6 Scope of Project

This study aims to aid software developers to maintain design models consistency in a faster and correct way to be in line with the source code implementation. The consistency checking is between code implementation in Visual Studio and UML models. Application scope will be C# source code that

is free from any syntax error. The target users of proposed tool are software developers.

1.7 Dissertation Outline

This thesis is structured in accordance to standard thesis outline. It starts with the introduction chapter and ends with conclusion chapter.

In Chapter 2, detailed review of literature is made. Literatures related to UML models, vertical and horizontal consistency checking rules and reverse engineering methods of source code into models are reviewed. This chapter further discuss related works in research area of concern.

Chapter 3 describes the methodology used in this research. It presents an explanation of vertical and horizontal consistency rules of class diagram, sequence diagram and use case diagram. The chapter also discusses about inconsistency detection logic and approaches. This chapter further discuss the research phases and activities in detail.

Chapter 4 introduces the design and implementation of consistency checker tool in detail. Metadata extraction from UML design model and source code are explained in detail. Identification of vertical and horizontal inconsistency are described with support of algorithm used in this project. Some important line of codes in the implementation program were also discussed.

Chapter 5 presents testing and evaluation of consistency checker tool. Tool testing and evaluation has been conducted using 2 case studies. The main focus of testing to identify inconsistencies between UML models and

source code by applying consistency rules on the metadata extracted from both UML model and source code.

Chapter 6 provides conclusion of this research. This chapter outlines limitation of research and some possible future works.

REFERENCES

A. Ananda Rao, T. V. Rajini Kanth, G. Ramesh (2016). “A Model Driven Framework for Automatic Detection and Tracking Inconsistencies”. *Journal of Software, Volume 11, Number 6 (pp. 538 – 553).*

A.Elmounadi, N.Berbiche, F.Guerouate, N.Sefiani (2017). “Eclipse JDT-Based Method For Dynamic Analysis Intergration In Java Code Generation Process”. *Journal of Theoretical and Applied Information Technology Vol.95. No. 24.*

Abilio G. Parada, Eliane Siegert, Lisane B. De Brisolara (2011). “Generating Java code from UML class and sequence diagrams”. *Brazilian Symposium on Computing System Engineering (SBESC) (pp. 99-101).*

Alexander Egyed (2011). “Automatically Detecting and Tracking Inconsistencies in Software Design Models”. *IEEE Transactions of Software Engineering, Vol. 37, No. 2.*

Alexander Reder, Alexander Egyed (2013). “Determining the Cause of a Design Model Inconsistency”. *IEEE Transactions of Software Engineering, Vol. 39, No. 11.*

Andrew Sutton, Jonathan I. Maletic (2005). “Mappings for Accurately Reverse Engineering UML Class Models from C++”. *WCRE Proceedings of the 12th Working Conference on Reverse Engineering (pp. 175 – 184).*

Chikofsky, E. J. and Cross, J. H. (1990), “Reverse Engineering and Design Recovery: A Taxonomy”. *IEEE Software, Vol. 7, No. 1 (pp. 13-17).*

Collard, M. L., Decker, M. J., and Maletic, J. I. (2013). “srcML: An Infrastructure for the Exploration, Analysis, and Manipulation of Source Code: A Tool Demonstration”. *29th IEEE International Conference Proceedings on Software Maintenance (ICSM) (pp. 516-519).*

Harshal D. Gurad , V.S.Mahalle (2014). “Transformation of UML Sequence Diagram To Java Code”. *The International Journal of Pure and Applied Research in Engineering and Technology, Vol.2, No.8 (pp. 703-710).*

Hector M. Chavez, Wuwei Shen (2016). “An Approach to Checking Consistency between UML Class Model and Its Java Implementation”. *IEEE Transactions of Software Engineering, Vol. 42, No. 4.*

http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2016

http://www.omg.org/spec/XMI

http://www.uml-diagrams.org

https://docs.microsoft.com/en-us/dotnet/csharp

Huy, T., Faiz, U. M., & Uwe, Z. (2015). “A graph-based approach for containment checking of behavior models of software systems”. *Proceedings of the 2015 IEEE 19th International Conference on Enterprise Distributed Object Computing (pp. 84-93).*

Huzar, Z., Kuzniarz, L., Reggio, G., & Sourrouille, J. L. (2005). “Consistency Problems in UML-Based Software Development”. *UML Modeling Languages and Applications (pp. 1–12).*

Laszlo Angyal, Laszlo Lengyel, and Hassan Charaf (2008). “A Synchronizing Technique for Syntactic Model-Code Round-Trip Engineering”. *15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems.*

Lene Nielsen (2012). “The usability expert's fear of agility: an empirical study of global trends and emerging practices”. *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design (pp. 261-264).*

M. Osman, M. Chaudron, P. Van Der Putten, T. Ho-Quang (2014). “Condensing reverse engineered class diagrams through class name based abstraction.” *Information and Communication Technologies (WICT) 2014 Fourth World Congress (pp. 158-163).*

Michael John Decker, Kyle Swartz, Michael L. Collard, Jonathan I. Maletic (2016). “A Tool for Efficiently Reverse Engineering Accurate UML Class Diagrams”. *IEEE International Conference on Software Maintenance and Evolution.*

Misbhauddin, M., & Alshayeb, M. (2015). “UML model refactoring: a systematic literature review”. *Journal of Empirical Software Engineering, Vol. 20 Issue (1). (pp. 206–251).*

Mohammadreza Sharbaf, Bahman Zamani, Behrouz Tork Ladani (2015). “Towards automatic generation of formal specifications for UML consistency verification”. *International Conference on Knowledge-Based Engineering and Innovation (KBEI) (pp. 860-865).*

N. Cuong and X. Qafmolla (2011). “Model transformation in web engineering and automated model driven development”. *International Journal of Modelling and Optimization Vol. 1 No. 1 (pp. 7-12).*

R Alroobaea, PJ Mayhew (2014). “How many participants are really enough for usability studies?”. *Science and Information Conference (SAI) (pp. 48-56).*

Raja Sehrab Bashira, Sai Peck Lee, Saif Ur Rehman Khan, Victor Chang, Shahid Farid (2016). “UML models consistency management: Guidelines for software quality Manager”. *36th International Journal of Information Management (pp. 883–899).*

Selim Ciraci, Hasan Sozer, Bedir Tekinerdogan (2012). “An Approach for Detecting Inconsistencies between Behavioral Models of the Software Architecture and the Code”. *IEEE 36th International Conference on Computer Software and Applications.*

Spanoudakis, G., & Zisman, A. (2001). “Inconsistency management in software engineering: survey and open research issues”. *Handbook of SofQaftware Engineering and Knowledge Engineering, Vol. 1. (pp. 329–380).*

Van Cam Pham, Ansgar Radermacher, Sebastien Gerard, Shuai Li (2017). “Bidirectional Mapping between Architecture Model and Code for Synchronization”. *IEEE International Conference on Software Architecture.*