



AIN SHAMS UNIVERSITY

FACULTY OF ENGINEERING
Computer Engineering and Systems

VALIDATION AND VERIFICATION OF UML MODELS

A Thesis submitted in partial fulfillment of the requirements of the degree of
Master of Science in Electrical Engineering
(Computer Engineering and Systems)

by

Azza Shawki Othman

Bachelor of Science in Electrical Engineering
Computer Engineering and Systems department

Faculty of Engineering, Ain Shams University, year 2008

Supervised By

Prof. Hoda K. Mohamed

Dr. Islam A. M. El Maddah

Cairo - (2018)



AIN SHAMS UNIVERSITY
FACULTY OF ENGINEERING
Computer and Systems

Validation and Verification of UML models

by
Azza Shawki Othman
Bachelor of Science in Electrical Engineering
(Computer Engineering and Systems)
Faculty of Engineering, Ain Shams University, 2018

Examiners' Committee

Name and Affiliation

Signature

Prof. Nawal Ahmed Elfishawy
Computer and Systems , Monfya University

.....

Prof. Hany Mohamed Kamal Mahdy
Computer and Systems Ain Shams University

.....

Prof. Hoda Korashy Mohamed
Computer and Systems , Ain Shams University

.....

Date:

Statement

This thesis is submitted as a partial fulfilment of Master of Science in Electrical Engineering, Faculty of Engineering, Ain shams University.

The author carried out the work included in this thesis, and no part of it has been submitted for a degree or a qualification at any other scientific entity.

Student name

Azza Shawki Othman

Signature

.....

Date: 19 September 2018

Researcher Data

Name : Azza Shawki Othman
Date of birth : 8/1/1985
Place of birth : Jeddah. KSA
Last academic degree : Bachelor of Engineer
Field of specialization : Electrical / Computer and control
University issued the degree : Assuit University
Date of issued degree : July 2008
Current job : Computer Engineer at creative Group

Thesis Summary

Recently, the Software designs process has clearly become much more sophisticated, thus the integrity of software system designs has gradually become difficult to be verified. For this purpose, the unified modelling language (UML) has been produced to be a standard language for designing software.

Verification and validation of UML models are necessary because some of UML models exhibited behaviours are neither wanted nor expected by the developer, this causes severe problems during runtime. This step is usually done by a model checker.

There are considerable numbers of model checking tools. However, this is usually accomplished by experts who have a good awareness of the model checker language and its construction. That's why it is necessary for a model validator to work along with the model designer. Thereby achieving the concern of separation accomplishes this step. In this thesis, a bridge has been constructed to tighten the gap between design and analysis processes through the USE2ALLOY tool. This tool can transform a model produced by the designer to a model suitable for conducting automated analysis by the validator.

This thesis presents a study of USE2ALLOY with OCL constraints. The translation allows analysis of UML models via Alloy to identify consistencies in those UML models and search for their instances with minimum effort from the designer. A comparison between this work and other related works has been done. Also, a conclusion has drawn and suggestions produced for future work.

Key words: UML; class diagram; OCL Model; Constraint; Alloy analyzer; Signature; USE tool; USE2ALLOY

Acknowledgment

First and above all, I praise and thank the God, for providing me this opportunity and granting me the capability to proceed successfully and complete this work.

This thesis appears in its final form due to the help and guidance of several people. I would therefore like to offer my sincere thanks to all of them.

Prof. Dr. Hoda K., my cordial thanks for accepting me as a M.A. student, thank for your warm encouragement, thoughtful guidance, critical comments, and correction of the thesis.

My sincere thanks and special appreciation also go to my advisor Dr. Islam El Maddah, for the trust, the insightful discussion, offering valuable advice, for your support during the whole period of the study, and especially for your patience and guidance during the writing process.

I would like to extend my sincerest thanks and appreciation to my friend Alaa Hamed, who helped me to accomplish this study, thank for your love, your supporting, encouragement and thank for giving me strength to reach.

Finally, I would like to sincerely thank my dear parents for their love, understanding, and support to finish my academic degree. I dedicate all my success to them.

September 2018

Table of Contents

Thesis Summary

Acknowledgments

List of Figures

List of Tables

List of Abbreviations

Chapter1	General Introduction and Problem statement	1
1.1	General Introduction	1
1.2	Problem Statement	2
1.2.1	The Quest for Software Correctness	2
1.2.2	Systems suffer from errors in their designs	2
1.3	Contributions	3
1.3.1	Why we need this transformation	4
1.3.2	Transformation requirements	4
1.4	Thesis Arrangement	5
Chapter2	Background: Modelling, UML Models and Model Checker	6
2.1	System life cycle	6
2.2	Modelling	7
2.2.1	UML Model	9
2.3	The Object Constraint Language (OCL)	10
2.3.1	Why we use OCL	11
2.3.2	Invariants	11
2.3.2.1	Object Navigation	12
2.3.2.2	Collections	12
2.3.2.3	Collection Operations	14
2.3.2.4	Boolean-Valued expressions	14
2.3.3	Pre-post conditions	14
2.4	Verification and validation	14
2.4.1	What is System Validation	14
2.4.2	What is System Verification	14
2.4.3	Difference between Verification and Validation	14
2.4.4	Software Verification Techniques	15

2.4.5	Formal Methods.....	16
2.4.5.1	Formal Verification Techniques.....	16
2.5	Model Checker	17
2.5.1	What is Model checking.....	17
2.5.2	Counter example.....	18
2.5.3	Important of model checker.....	19
2.5.4	Type of model checker.....	20
2.5.4.1	Model Design.....	20
2.5.4.2	Testing Strategies.....	23
2.5.5	Testing vs. Model Checking.....	25
2.5.6	Requirement Property.....	26
2.5.7	Evaluation of Model Checkers.....	27
2.5.8	Comparison of model checker.....	29
2.6	RELATED WORKS	32
2.7	Comparison &Evaluation of UML\OCL Tools	34
Chapter3	USE Tool and Alloy Tool	37
3.1	USE Tool	37
3.1.1	Why USE Tool?	37
3.1.2	USE Model-Analyzer.....	38
3.1.3	Disadvantage of USE Model-Analyzer.....	38
3.1.4	Specifying a UML Model with USE Tool.....	39
3.1.5	Working with USE.....	40
3.1.6	Example.....	41
3.1.7	Running example with USE tool.....	44
3.2	Alloy Model Checker	47
3.2.1	Why Alloy Model checker?	47
3.2.2	Alloy Model-Analyzer.....	49
3.2.2.1	Advantages of Using Alloy.....	49
3.2.2.2	KodKod and SAT solver.....	50
3.2.3	Specifying a UML Model with Alloy Model Checker.....	51
3.2.4	Example.....	53
3.3	USE Model-Analyzer VS. Model-Alloy Analyzer	58
3.4	Shortcomings in OCL	58
Chapter4	Proposed method to translate USE to Alloy	62
4.1	Understanding Alloy syntax	62
4.1.1	Navigation expression of Alloy.....	63

4.1.2	SET Theory.....	63
4.1.3	Ordering.....	63
4.1.4	Sequence\Lists.....	64
4.2	Mapping Class from USE TO Alloy.....	64
4.2.1	Converting Class.....	65
4.2.2	Converting Associations.....	65
4.2.3	Role name.....	66
4.2.4	Mapping USE OCL to Alloy.....	67
4.2.5	Mapping Operations to Alloy.....	68
	4.2.5.1 Parameterize & None Parameterize operation.....	68
4.2.6	Mapping Pre-Post condition to Alloy.....	69
4.3	USE2ALLOY Interface.....	69
Chapter5 USE2ALLOY–CASE STUDY		
5.1	Static Analysis-CASE STUDY 1	71
5.2	Dynamic Analysis-CASE STUDY 2.....	80
Chapter 6 CONCLUSION AND FUTURE WORK		
6.1	Conclusion.....	91
6.2	Future Work.....	92
	References	93
	Appendix A	99
	Arabic Summary	106

List of Figures

Figure 1.1: USE2ALOY	2
Figure 2.1: SDLC Software Development Life Cycle	6
Figure 2.2: Summary of Model Comparison Approaches [56]	8
Figure 2.3: UML Diagrams Type	10
Figure 2.4: System life cycle and error introduction detection and costs of repairmen “error hunting” [12]	17
Figure 2.5: Essential idea of model checking	18
Figure 2.6: Classification of model checker	24
Figure 2.7: Over various topics, the average of using of mode checker [46]	25
Figure 2.8: Testing [27]	25
Figure 2.9: Model checking[27]	26
Figure 3.1: Opening use tool	39
Figure 3.2: USE interface	39
Figure 3.3: Open file in USE	40
Figure 3.4: Network system	42
Figure 3.5: Classes represented in USE notation on a text file	42
Figure 3.6: Association of class diagram in USE notation	43
Figure 3.7: Constraint in USE notation	44
Figure 3.8: GUI for USE	45
Figure 3.9: Class diagram in USE tool	45
Figure 3.10: Object diagram in USE tool	46
Figure 3.11: USE Analysis Log file	46
Figure 3.12: USE class invariant	46
Figure 3.13: Evaluation browser shows the false reasons	47

Figure 3.14: Signature definition	51
Figure 3.15: Field definition	51
Figure 3.16: Fact definition	52
Figure 3.17: Function definition	52
Figure 3.18: Predicate definition	52
Figure 3.19: Assertion definition	53
Figure 3.20: Node1 is its own successor	54
Figure 3.21: Nodes do not belong to any Queue	55
Figure 3.22: One node contains a cycle	56
Figure 3.23: Node0 belongs to two different queues	57
Figure 3.24: Undefined operation in OCL	58
Figure 3.25: Unnecessary complexity in OCL	59
Figure 3.26: USE can't manipulate them directly	60
Figure 3.27: OCL expressions hard to read	61
Figure 4.1: Reserved word	62
Figure 4.2: Navigation Expression	63
Figure 4.3: Options for Set	63
Figure 4.4: Options for ordering	63
Figure 4.5: Options for Sequence\Lists	64
Figure 4.6: Converting Classes from USE to Alloy	65
Figure 4.7: Converting Association from USE to Alloy	65
Figure 4.8: Converting Association from USE to Alloy	65
Figure 4.9: Converting Invariant from USE to Alloy	68
Figure 4.10: Converting Operations from USE to Alloy	68
Figure 4.11: Converting Operations with no entries from USE to Alloy	68
Figure 4.12: Converting Pre/Post condition from USE to Alloy	69

Figure 4.13:USE2ALOY	69
Figure 5.1: Opening USE specification	72
Figure 5.2: Make transformation	72
Figure 5.3: Executing Alloy model	76
Figure 5.4: Alloy analysis for the model	76
Figure 5.5: Instance “run example” for the model	77
Figure 5.6: Press “Next” to get other instances	77
Figure 5.7: Another instance for same model system	78
Figure 5.8: Projection command	78
Figure 5.9: Instance with projecting Int	79
Figure 5.10: Metamodel for Network System in Alloy	79
Figure 5.11: The metamodel for Network system	80
Figure 5.12: Putting system in Alloy to make analysis	86
Figure 5.13: Analysis of model with Alloy	86
Figure 5.14: Instance of the model	87
Figure 5.15: System at time 0, Projection over time	87
Figure 5.16:Ssystem at Time 1	88
Figure 5.17: System at Time 2	88
Figure 5.18: System at Time 3	89
Figure 5.19: System at Time 4	89
Figure 5.20:Tthe system at Time5	90
Figure 5.21: Analysis after fixing problem	90