



Al-Azhar University,  
Faculty of Science (Girls),  
Department of Mathematics.

## **Improvement of Formal Methods Approach Using Pattern Perspective in Real Time System Applications**

A thesis submitted for

***Ph.D. Degree***

in

***Computer Science***

***by***

**Enas El-Sayed Mohamed El-Sharawy**

Department of Mathematics, Faculty of Science (Girls),  
Al-Azhar University.

***Under Supervision of:***

**Prof. Dr.**

**Ahmed Kamel El-Khouly**

Al-Azhar University,  
Faculty of Science (Girls),  
Department of Mathematics,  
(Mathematics section)

**Prof. Dr.**

**Gaber Ahmed El-Sharawy**

Al-Azhar University,  
Faculty of Science (Girls),  
Department of Mathematics,  
(Computer science section)

**Dr.**

**Eman Karam El-Sayed**

Al-Azhar University,  
Faculty of Science (Girls),  
Department of Mathematics,  
(Computer science section)

Cairo, 2014

## **ACKNOWLEDGEMENT**

Prayerful thanks, at first to our Merciful God who gives me everything I have.

I wish to express my sincere gratitude and deepest thanks to Prof. Dr.A.K El-kholy and G.A. El Sharawy, for their active supervision, and constructive criticism during the progress of this thesis.

I'm greatly indebted to Dr. Eman Karam El-Sayed for providing me most of the Guidance needed for the completion of this work.

My thanks also go to the Department of Mathematics and faculty members; for their help and support.

My vocabulary utterly fails in expressing my accolade to my revered parents who brought me to this stage. I fall short of words for the moral support extended by them. I also gratefully acknowledge the help and moral support rendered by all family members for perusing further studies.

My thanks also extended to my husband and my sweet daughter, for their continuous encouragement, their patience, interest and helpful cooperation which made this investigation possible.

Thank giving is not merely a customary word; I realize that it is of immense significance too.

Grateful and indebted  
(Enas El-Sharawy)

## Abstract

The systematic approach (formal method) is a correct valid path to convert software requirement to executable code. Each Formal Method with Proofs FMP have theorem behind it as B theorem with Event-B formal method. By formal method we can build large computerized complex systems. That is by three techniques: refinement, decomposition and generic instantiation. In this thesis, we integrate refinement and generic instantiation techniques in Event-B formal method. Practically, we used RODIN platform and its database tools of the Static Checker, the Proof Obligation Generator, the Prover and the Translator from B model to a programming language.

The main aim of this work is rising the degree of automation and increasing efficiency of theorem proving technique to decrease the cost of such developments depends directly on them. And also increase correctness and validation degree. All of them with stable reusability. So we propose using automatic theorem provers known as SMT-solvers with Event-B pattern. The benefit of this approach is to have strong validation pattern. This approach enhances the pattern reusability with high degree of automatic proving and low effort. Then, we propose a new refinement of ATM UML model by agentless data collection approach and convert it to be Event-B pattern. That convert is more suitable for mathematician. That convert reduce the proof obligation.

Finally, we discuss a logical correct path to translate Event-B pattern to C# language code. That is for both models and algorithms.

**Keywords:** Formal method, Event-B, Proof Obligations, UML-B Pattern, Event-B Pattern, ATM model, SMT-Solver, agentless data collection approach, Code Generation.

# Contents

Subject	page
<b>CHAPTER 1: INTRODUCTION</b>	
1.1 Introduction	1
1.2 Thesis Motivation and Contribution	2
1.3 Outcomes and Thesis Organization	4
<b>CHAPTER 2: FORMAL METHODS</b>	
2.1 Introduction	8
2.2 Overview of Modeling	9
2.3 Definition of Formal Methods	10
2.4 Refinement Approach	12
2.5 Formal Method language	13
2.5.1 VDM	13
2.5.2 Z method	14
2.5.3 B-Method	15
2.5.4 CSP Language	15
2.5.5 Action Systems	16
2.5.6 Ontology	17
2.5.7 Event-B Language	17
2.6 A Comparison between Formal Method Languages	18
<b>Chapter 3: The Event-B Modeling Notation and Proof Obligation Rules</b>	
3.1 Introduction	19
3.2 Event-B components: Machines and Contexts	20
3.2.1 Machine and context relationships	21
3.2.2 Context structure	23
3.2.3 Context example	24
3.2.4 Machine Structure	25
3.2.5 Machine Example	27
3.2.6 Events	28
3.2.7 Actions	29
3.2.8 Examples of events	31
3.3 Machine Refinement in Event-B	34
3.4 Proof Obligation Rules	36
3.4.1 Introduction	36
3.4.2 Invariant Preservation Proof Obligation Rule: INV	37

Subject	page
3.4.3 Feasibility Proof Obligation Rule: FIS	39
3.4.4 Guard Strengthening Proof Obligation Rule: GRD	41
3.4.5 The guard merging proof obligation rule: MRG	42
3.4.6 Simulation proof obligation rule: SIM	43
3.4.7 The Numeric Variant Proof Obligation Rule: NAT	45
3.4.8 The Finite Set Variant Proof Obligation Rule: FIN	45
3.4.9 The variant proof obligation rule: VAR	46
3.4.10 The Non-Deterministic Witness Proof Obligation Rule: WFIS	49
3.4.11 The Theorem Proof Obligation Rule: THM	50
3.4.12 The Well-Definedness Proof Obligation Rule: WD	50
<b>Chapter 4: Integration of Automatic Theorem Provers in Event-B Patterns</b>	
4.1 Introduction	51
4.2 Design Pattern	52
4.3 Satisfiability Modulo Theory (SMT)	54
4.4 Related Works	55
4.5 The Proposal Approach	56
4.6 CASE STUDIES	56
4.6.1 Overview of a Binary Search	56
4.6.2 Overview of Minimum Algorithm	67
4.7 Result Analysis	70
<b>Chapter 5 : The Refinement Validation of ATM Model By Using UML-B</b>	
5.1 Introduction	79
5.2 Overview	80
5.2.1 UML-B Background	80
5.2.2 Refinement Approach in UML-B	83
5.2.3 Agentless Data Collection Validation	83
5.2.4 ATM System	84
5.3 Literature Review	85
5.4 The Proposal Approach	86
5.5 ATM Case Study	88
5.6 System Generalization	97
5.6.1 Question/Response protocol : Initial Model Event-B	98
5.6.2 Question/Response protocol First Refinement Event-B	99
5.6.3 Pattern Similarity of Protocol	100
5.6. Result analysis	101

Subject	page
<b>Chapter 6 : Event-B Pattern to High Quality Automatic Implementation</b>	
6.1 Introduction	105
6.2 Translation Philosophy	107
6.3 Generation Phases	109
6.3.1 The Rewrite Phase	109
6.3.2 The Translation Phase	111
6.3.3 The Build Phase	115
6.4 Case Studies	116
6.4.1 Automatic C# Code Generation from Minimum Search Algorithm	116
6.4.2 Automatic C# Code Generation from ATM Pattern Model	119
6.5 Result Analysis	122
<b>Chapter 7 : CONCLUSION AND FURTHER WORK</b>	125
<b>Publications</b>	127
<b>REFERENCES</b>	128
<b>ARABIC SUMMARY</b>	

## LIST OF FIGURES

<b>Figure No.</b>	<b>page</b>
Figure 3.1 Machine And Context Component	20
Figure 3.2. Machine And Context Relationship	21
Figure 3.3 Examples of correct visibilities	22
Figure 3.4 Context structure	24
Figure 3.5 Context example	25
Figure 3.6. Context example (one per clause)	25
Figure 3.7 Machine structure	26
Figure 3.8 Machine example	27
Figure 3.9 Machine example as presented in this thesis	27
Figure 3.10. Event structure	28
Figure 3.11(a) Events associated with machine m_0a	31
Figure 3.11(b) Another machine m_0b	32
Figure 3.11(c) Machine m_1a refines machine m_0a	33
Figure 3.11(d) Machine m_1b refines machine m_0b	33
Figure 3.12 abstract event ea and concrete event ec	34
Figure 3.13 (a) Proof Obligation of feasible concrete event	35
Figure 3.13 (b) Proof Obligation of the guard of event ea	35
Figure 3.13 (c) Proof Obligation of the guard of event ec	35
Figure 3.14 Event assumption	36
Figure 3.15 Event assumption for rule INV	37
Figure 3.16 Rule of proof obligations that called evt/inv/INV	37
Figure 3.17 (a) Example of simplification of PO “initialization / inv1 / INV:”	38
Figure 3.17 (b) Example of simplification of PO “search / inv1 / INV	38
Figure 3.18 Event refinement assumption	39
Figure 3.19 Rule of proof obligations of refined invariant	39
Figure 3.20 Event assumption for rule FIS	40
Figure 3.21 Rule of proof obligations that called evt/act/FIS	40
Figure 3.22 Example of evt/act/FIS PO.	40
Figure 3.23 Event assumption for ruler GRD.	41
Figure 3.24 Rule of proof obligations that called evt/grd/GRD	41
Figure 3.25 Example of evt/grd/GRD PO.	42
Figure 3.26 Event assumptions for rule MRG	42
Figure 3.27 Rule of proof obligations that called evt/MRG	43
Figure 3.28 Event assumptions for rule SIM	43
Figure 3.29 Rule of proof obligations that called evt/act/SIM	44
Figure 3.30 Example of evt/act/SIM PO.	44

<b>Figure No.</b>	<b>page</b>
Figure 3.31 Event assumption for rule NAT	45
Figure 3.32 Rule of proof obligations that called evt/NAT	45
Figure 3.33 Event assumptions for rule FIN	46
Figure 3.34 Rule of proof obligations that called evt/FIN	46
Figure 3.35 Convergent Event assumption for rule VAR	47
Figure 3.36 Rule of proof obligations of Convergent Event that called evt/VAR if the variant is numeric	47
Figure 3.37 Rule of proof obligations of Convergent Event that called evt/VAR if the variant is a finite	47
Figure 3.38 anticipated Event assumption for rule VAR	48
Figure 3.39 Rule of proof obligations of anticipated Event that called evt/VAR if the variant is numeric	48
Figure 3.40 Rule of proof obligations of anticipated Event that called evt/VAR if the variant is a finite	48
Figure 3.41 Event assumption for rule WFIS	49
Figure 3.42 Rule of proof obligations that called evt/x/WFIS	49
Figure 4.1 Initial model of Binary search algorithm	57
Figure 4.2 First refinement of Binary search algorithm	59
Figure 4.3 Second refinement of Binary search algorithm	59
Figure 4.4 development Specification P0 of design pattern for binary search	60
Figure 4.5 Pattern refinement P1 of design pattern for binary search	60
Figure.4.6: The First step	62
Figure.4.7: The Second step	62
Figure.4.8: The Third step	63
Figure 4.9: The fourth step	63
Figure 4.10: The fifth step	63
Figure 4.11 : The screen shot of the proof before the SMT proof, the button is active and the status is not proved.	65
Figure 4.12: The screen shot of the proof after a successful proof, the button has been dis-activated and the status is “proved”.	66
Figure 4.13 Initial model of minimum algorithm	67
Figure 4.14 First Refinement of minimum algorithm	68
Figure 4.15 Development Specification P0 of design pattern for minimum algorithm.	69
Figure 4.16 Pattern refinement P1 of design pattern for minimum algorithm.	70
Figure 4.17 RODIN platform statistics for binary search model without using pattern and SMT-Solver	71
Figure 4.18: RODIN platform statistics for binary search model using SMT-Solver without using pattern	71



<b>Figure No.</b>	<b>page</b>
Figure 4.19: RODIN platform statistics for binary search model using pattern without SMT-Solver	71
Figure 4.20: RODIN platform statistics for binary search model using pattern and SMT-Solver	72
Figure 4.21: RODIN platform statistics of POs for the minimum model without using pattern and SMT-Solver	75
Figure 4.22 : RODIN platform statistics of POs for the minimum model using SMT-Solver without using pattern	75
Figure 4.23: RODIN platform statistics of POs for the minimum model using pattern without SMT-Solver	75
Figure 4.24 : RODIN platform statistics of POs for the minimum model using pattern and SMT-Solver	76
Figure 5. 1 UML-B Diagrams	81
Figure 5.2 Generated Event-B specification of machine1	82
Figure 5.3 Transactions of ATM System	85
Figure 5.4 The proposed methodology diagram	87
Figure 5.5 ATM Package Diagram	89
Figure 5.6 UML-B specification of ATM abstract machine	89
Figure. 5.7. UML-B specification of ATM First Refinement	91
Figure 5.8. UML-B specification of ATM Second Refinement	93
Figure 5.9 The seventh refinement of UML-B state machine	95
Figure 5.10 Event-B pattern specification for ATM	96
Figure 5.11 Event-B seventh refinement pattern for ATM	97
Figure 5.12 Question/Response protocol Event-B Initial Model	98
Figure 5.13 (a) Question/Response protocol Event-B first refinement	99
Figure 5.13 (b) Question/Response protocol Event-B first refinement	99
Figure 5.13 (c) Question/Response protocol Event-B first refinement	100
Figure 6.1 EB2C# translation of user action.	111
Figure 6.2 Example of rewritten event (inc)	116
Figure 6.3. Example of Event translation for inc. event	117
Figure 6.4 Example of mini event and derived C# function	117
Figure 6.5 Example of generated event-calling function for minimum search algorithm	118
Figure 6.6 Example of translation header of minimum C# derived code	118
Figure 6.7 Example of calling function environment for minimum search algorithm	119
Figure 6.8 Example of rewritten event-B event (ejectCardWithCash)	120

<b>Figure No.</b>	<b>page</b>
Figure 6.9 Event translation for Transaction event	120
Figure 6.10 Example of generated event-calling function	121
Figure 6.11 Part of translation header of ATM derived C# code	122
Figure 6.12. RSM report screen shot for automatically generated C# code of minimum algorithm	123
Figure 6.13. RSM report screen shot for C# minimum algorithm's code obtained from C# programming language	123
Figure 6.14. RSM report screen shot for automatically generated C# code of ATM model	124
Figure 6.15. RSM report screen shot for C# ATM code obtained from C# programming language	124

## LIST OF TABLES

Subject	page
Table 4.1: The Proof Obligation (POs) statistical analysis for the binary search model	72
.	
Table 4.2: The POs statistics for the minimum model	76
Table 5.1: The POs of ATM seven machines refinement	101
Table 5.2: The POs of ATM refinements models after applying the proposed phases	102
Table 6.1: Supported Event-B syntax	109

## LIST OF CHARTS

Chart	Page
Chart 4.1: The Proof Obligation (POs) for the Binary search model	74
Chart 4.2: The POs for the minimum model	78
Chart.5.1 The POs for ATM model before and after Applying SMT-Solve	103
Chart 5.2: The POs for ATM model before and after Applying SMT-Solver and event-b pattern	104

## *List of Abbreviations*

<b>FM</b>	Formal Method
<b>FMP</b>	Formal Method with Proofs
<b>RODIN</b>	Rigorous Open Development environment for complex systems INdustry day
<b>VDM</b>	Vienna Development Method
<b>CSP</b>	Communicating Sequential Processes
<b>KLOC</b>	1000 Line Of Code
<b>PO</b>	Proof Obligation
<b>SMT-SOLVER</b>	Satisfiability Modulo Theory- Solver
<b>INV</b>	Invariant Preservation Proof Obligation Rule
<b>FIS</b>	Feasibility Proof Obligation Rule:
<b>GRD</b>	Guard Strengthening Proof Obligation Rule
<b>MRG</b>	The Guard Merging Proof Obligation Rule
<b>SIM</b>	Simulation Proof Obligation Rule
<b>NAT</b>	The Numeric Variant Proof Obligation Rule
<b>FIN</b>	The Finite Set Variant Proof Obligation Rule
<b>VAR</b>	The Variant Proof Obligation Rule
<b>WFIS</b>	The Non-Deterministic Witness Proof Obligation Rule
<b>THM</b>	The Theorem Proof Obligation Rule
<b>BSA</b>	Bank Service Agent
<b>ATM</b>	Auto Teller Machine
<b>UML-B</b>	Unified Modelling Language - B
<b>EB2C#</b>	Event-B to C#
<b>API</b>	Application Programming Interface

# Chapter 1: Introduction

## 1.1 Introduction

Formal Methods are mathematically based modeling techniques used to specify and verify hardware and software systems. Z method, VDM (Vienna Development Method), B-Method (also known as classical B) and Event-B are among the most recent formal methods [5, 8].

Event-B is an evolution of the classical B. Event-B uses the concept of Refinement in modeling [31]. Event-B modeling starts with an abstract specification of a system. Details are added during refinement steps in order to arrive at a more detailed model. The mathematical language of Event-B is based on set theory and first order logic. Based on the Event-B language, a set of proofs can be produced and discharged for each Event-B model. RODIN is an open source, extensible and integrated modeling tool supporting Event-B. This tool is not only used as a modeling environment, but also provides an integrated environment for proving properties of models [9, 6, 26, 59].

Formal modelling is not only constructing descriptions, but also proving some properties about the formal models. RODIN provides an integrated environment for both modelling and proving. Extensibility of RODIN makes it easy for new features to be added to it. During recent years, some Eclipse based plug-ins were developed and added to RODIN: SMT-Solver as prover, ProB [71] as an animator, UML-B [72] as graphical environment provider, B2C# as a translator from B to C# code and, B2Latex [70] as a translator from B to Latex, have been developed and added to RODIN. Recently Event-B has been applied to developing industrial cases. However building models of large and complex systems results in large and complex models and difficult proofs. Some techniques

such as Pattern approach, SMT-Solver, UML and Event-B can help to solve this difficulty.

This thesis focuses on pattern as an approach for modeling and proving large and complex systems using Event-B. This approach enables developers to reuse the proved model in Event-B. Refinement in Event-B is too general. It does not explicitly show all of the relations between behaviors of the abstract model, called abstract events, and the behaviors of the refinement, called the concrete events. This approach is also capable of showing an overall structure of several refinement levels. Therefore it provides an effective way to handle complex development.

On the other hand, providing patterns and UML-B, makes the modelling of large systems more manageable. Using UML-B and patterns, we can achieve reusability and visualization in Event-B development.

Finally, we use code generation approach to generate C# code from Event-B model. When we do this conversion, we achieve the validation of C# code and guarantee that the code fulfills the requirement of model. To achieve reusability, correctness and visualization, the used tool has been developed as a plug-in tool for the RODIN platform.

## **1.2 Thesis Motivation and Contribution**

The intent of this work is to give some insights on modeling and formal modeling. These activities are supposed to be performed *before* undertaking the effective coding of a computer system, so that the system in question will be correct by construction.

We will also understand how it is possible to detect the presence of inconsistencies in our models just by the fact that some proofs cannot be done. The failure of the proof will provide us with some helpful clues about what is wrong or insufficiently defined in our model. We will use