

**COMBINING SEMI-FORMAL AND FORMAL NOTATIONS
IN SOFTWARE SPECIFICATION: AN APPROACH TO
MODELLING TIME-CONSTRAINED SYSTEMS**

by

Sergiu-Mihai Dascalu

**Submitted
in partial fulfillment of the requirements
for the degree of**

DOCTOR OF PHILOSOPHY

Major Subject: Computer Science

at

DALHOUSIE UNIVERSITY

Halifax, Nova Scotia

September, 2001

© by Sergiu-Mihai Dascalu, 2001

Dalhousie University
Faculty of Computer Science

The undersigned hereby certify that they have examined, and recommend to the Faculty of Graduate Studies for acceptance, the thesis entitled “Combining Semi-Formal and Formal Notations in Software Specification: An Approach to Modelling Time Constrained Systems” by Sergiu-Mihai Dascalu in partial fulfillment of the requirements for the degree of the Doctor of Philosophy.

Dated : _____

Supervisor:

Dr. PETER HITCHCOCK

External Examiner:

Dr. GREGORY BUTLER

Examiners:

Dr. PETER BODORIK

Dr. TREVOR SMEDLEY

Dr. WILLIAM PHILLIPS

Dalhousie University
Faculty of Computer Science

DATE: September 14, 2001

AUTHOR: Sergiu-Mihai Dascalu
TITLE: Combining Semi-Formal and Formal Notations in Software
Specification: An Approach to Modelling Time-Constrained Systems
MAJOR SUBJECT: Computer Science
DEGREE: Doctor of Philosophy
CONVOCATION: October, 2001

Permission is herewith granted to Dalhousie University to circulate and to have copied for non-commercial purposes, at its discretion, the above thesis upon the request of individuals or institutions.

Signature of Author

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

The author attests the permission has been obtained for the use of any copyrighted material appearing in this thesis (other than brief excerpts requiring only proper acknowledgement in scholarly writing), and that all such use is clearly acknowledged.

**To my daughter Diana, my wife Alexandra,
and my parents Niculina and Vasile**

Table of Contents

List of Tables.....	x
List of Figures.....	xi
List of Abbreviations	xv
Acknowledgements.....	xvi
Abstract	xviii
1 Introduction.....	1
1.1 Three Paradigms and a View of the Field.....	1
1.1.1 The First Paradigm or Objects as Conquerors.....	1
1.1.2 The Resilient Field of Real-Time Applications.....	2
1.1.3 The Second Paradigm or Formalisation as a Controlling Factor.....	3
1.1.4 The Third Paradigm or The Power of Pictures	5
1.2 Motivations.....	6
1.2.1 Effectiveness and Simplicity	6
1.2.2 Capability of Tackling Complex Tasks	6
1.2.3 Early Detection of Errors.....	7
1.2.4 Powerful Combination of Paradigms.....	7
1.2.5 Understandability and Practicality	7
1.2.6 Ease of Communication.....	7
1.2.7 Expressiveness and Modernity.....	8
1.2.8 Rigor and Precision	8
1.2.9 Refinement	8
1.3 Challenges.....	8
1.3.1 Efficient Combination of Techniques and Notations	9
1.3.2 Approaching Time-Constrained Systems from an Object-Oriented Perspective... 	10
1.3.3 Developing Mechanisms for Formalisation of Graphical Representations.....	10
1.3.4 Rigorous Treatment of Temporal Constraints.....	11
1.3.5 Provision for User Acceptance.....	11
1.3.6 Tool Support.....	12
1.3.7 Capability of Extension.....	12
1.4 Notes on Terminology	13
1.5 The Proposed Approach.....	14
1.6 Overview of the Thesis.....	17
1.7 Chapter Summary.....	18

2	Background: Context and Concepts.....	19
2.1	Introduction.....	19
2.2	Research Space and Topic Location.....	20
2.3	On Specifying Real-Time Systems.....	22
2.3.1	Characteristics of Real-Time Systems.....	22
2.3.2	Focus On Time.....	29
2.4	Brief Immersion in Object-Orientation.....	30
2.4.1	On Objects and Their Modelling Power.....	30
2.4.2	Object-Orientation in the Real-Time Domain.....	33
2.5	On The Importance of Graphical Notations.....	34
2.6	Formal Notations in Software Development.....	36
2.6.1	Alexander's Definition of a Formal System.....	36
2.6.2	Classifications and Examples of Formal Methods.....	38
2.6.3	Advantages and Disadvantages of Formal Methods.....	40
2.6.4	Formal Techniques within the Software Development Process.....	43
2.6.5	A New Trend: Lighter Use of Formal Methods.....	43
2.7	Chapter Summary.....	44
3	Background: Notations.....	45
3.1	Introduction.....	45
3.2	Z and Flavours of Z.....	45
3.2.1	The Z Notation.....	46
3.2.1.1	Sets, Types, and Predicates.....	47
3.2.1.2	Relations, Functions, and Sequences.....	50
3.2.1.3	Schemas and Schema Calculus.....	55
3.2.2	Z Variants and Tools.....	58
3.2.3	A Glance at Z++.....	61
3.3	On UML and Its Capability of Dealing with Time.....	62
3.3.1	A Bird's Eye View on UML.....	63
3.3.2	UML Support for Modelling Real-Time Systems.....	76
3.3.3	The UML Promise.....	82
3.4	Chapter Summary.....	87
4	Related Work.....	88
4.1	Introduction.....	88
4.2	Integration of Semi-formal and Formal Notations in Software Specification.....	89
4.3	Semi-formal/Formal Integrations of Notations Not Involving Z.....	92
4.4	Semi-formal/Formal Integrations of Notations Involving Variants of Z.....	95
4.5	Closely Related Approaches.....	96
4.5.1	Jia's Augmented Object-Oriented Modeling Language.....	96
4.5.2	Noe and Hartum's Extension of Rational Rose 98.....	99
4.5.3	Blending Octopus and Z.....	101
4.5.4	Headway System's RoZeLink.....	103

4.5.5	Object Z and UML	104
4.6	Modalities of Specifying Temporal Constraints in Z.....	106
4.6.1	Time Refinement in Z.....	107
4.6.2	The Quartz Alternative.....	108
4.6.3	Andy Evans' Approach.....	109
4.6.4	RTOZ	111
4.6.5	TCOZ.....	112
4.6.6	Other Approaches	114
4.6.7	The Z++ Alternative.....	116
4.7	Chapter Summary.....	116
5	Formal Specification of Temporal Constraints.....	118
5.1	Introduction.....	118
5.2	Dasarathy's Classification of Temporal Constraints	119
5.3	On the Rigorous Specification of Temporal Constraints.....	123
5.4	Real-Time Logic (RTL).....	124
5.4.1	The Event-Action Model.....	124
5.4.2	RTL Concepts and Notations	125
5.5	Using RTL in Z++	127
5.5.1	Lano's Key Extensions to RTL	128
5.5.2	Events.....	128
5.5.3	Terms.....	129
5.5.4	Formulae.....	130
5.5.5	Abbreviations.....	130
5.5.6	Axioms.....	131
5.6	Chapter Summary.....	131
6	Translations Between UML and Z++: Formalisation and Deformalisation ..	132
6.1	Introduction.....	132
6.2	Preliminary Remarks.....	133
6.3	Formalisation of UML Class Diagrams in Z++.....	136
6.3.1	Rules for Developing Well-Formed Class Diagrams.....	136
6.3.1.1	Rules for Class Diagrams.....	138
6.3.1.2	Rules for Classes.....	139
6.3.1.3	Rules for Relationships.....	144
6.3.2	Translation Principles for Class Diagrams.....	146
6.3.2.1	Translation of Types.....	147
6.3.2.2	Translation of Attributes	149
6.3.2.3	Translation of Operations.....	150
6.3.2.4	Translation of Classes.....	151
6.3.2.5	Translation of Relationships	152
6.3.3	Algorithm for Formalising Class Diagrams (AFCD)	154
6.3.3.1	AFCD Input.....	155
6.3.3.2	AFCD Output.....	157

6.3.3.3	AFCD Pseudocode.....	158
6.4	Formalisation of UML State Diagrams in Z++	171
6.4.1	Constraints on the Contents of State Diagrams.....	171
6.4.2	Translation Principles for State Diagrams.....	174
6.4.2.1	General Principles and Terminology	174
6.4.2.2	Translation of States.....	176
6.4.2.3	Translation of Transitions	179
6.4.3	Algorithm for Formalism State Diagrams (AFSD)	183
6.4.3.1	AFSD Input.....	183
6.4.3.2	AFSD Output.....	185
6.4.3.3	AFSD Pseudocode.....	185
6.4.4	Example of Formalising a State Diagram	189
6.5	Deformalisation: From Z++ Specifications to UML Representations.....	193
6.5.1	Principles of Deformalisation	193
6.5.1.1	Assigning Types	193
6.5.1.2	Generating Attributes for UML Classes.....	194
6.5.1.3	Generating Operations for UML Classes.....	195
6.5.1.4	Generating UML Classes.....	196
6.5.1.5	Generating Relationships.....	197
6.5.1.6	Generating State Diagrams.....	198
6.5.2	Outline of the Algorithm for Deformalisation (ADF).....	199
6.6	Notes on the Application of Formalisation and Deformalisation Algorithms.....	202
6.7	Chapter Summary.....	204
7	A Procedural Frame.....	205
7.1	Introduction.....	205
7.2	Modelling Focus.....	206
7.3	Artefacts.....	207
7.4	Activities	210
7.4.1	Conventions in the Diagrammatic Representation of the Procedural Frame	210
7.4.2	Simplifications in the Diagrammatic Representation of the Procedural Frame... ..	212
7.4.3	Stages and Steps.....	212
7.4.4	The Regular Sequence of Modelling Activities.....	215
7.4.5	Alternative Flows of Modelling Activities.....	217
7.5	Chapter Summary.....	219
8	An Application: The Case of the Elevator System.....	221
8.1	Introduction.....	221
8.2	On the Elevator Case Study.....	222
8.3	The Problem	223
8.3.1	General Requirements for the Elevator System.....	224
8.3.2	Temporal Constraints for the Elevator System	225
8.3.3	Coverage of Dasarthy Constraints by the Elevator's Timing Requirements.....	227
8.4	The Modelling Solution.....	228

8.4.1	Definition of Use Cases.....	228
8.4.2	Elaboration of Scenarios.....	229
8.4.3	Construction of the Class Diagram.....	233
8.4.4	Specification of Sequence Diagrams	234
8.4.5	Elaboration of Class Compounds.....	236
8.4.6	Formalisation through the AFCD and the AFSD	240
8.4.7	Enhancement of the Formal Specification.....	249
8.5	Chapter Summary.....	252
9	Towards an Integrated Environment: A Prototype for Harmony	253
9.1	Introduction.....	253
9.2	General Principles.....	254
9.3	Overall Organisation.....	257
9.4	The Project Pane	259
9.5	The UML Space.....	262
9.6	The Z++ Space	263
9.7	Other Features.....	266
9.8	Chapter Summary.....	268
10	Conclusions.....	269
10.1	Introduction.....	269
10.2	Summary Comparison with Closely Related Approaches	269
10.3	Main Contributions.....	271
10.4	Other Contributions.....	272
10.5	More On the Limitations of the Proposed Approach.....	273
10.6	A Look Forward	274
	Bibliography	276
	Appendix A: Summary Overview of Z++.....	295
A.1	BNF Syntax of the Z++ Class Declaration.....	295
A.2	Invocation of Operations.....	297
A.3	Notes on Semantics.....	298
A.4	Extending and Restructuring the Specification	300
A.5	Translation to Standard Z	301
	Appendix B: Java Implementation of the AFCD.....	302
B.1	Contents of the Program Listing.....	302
B.2	Program Listing.....	303
	Appendix C: Harmony's User Interface.....	359

List of Tables

Table 2.I	Classification of Research Approaches Based on Domains of Exploration	20
Table 3.I	A Summary of Sequences	54
Table 3.II	UML Things (Model Elements)	65
Table 3.III	UML Relationships	66
Table 3.IV	UML Extension Mechanisms	67
Table 3.V	Types of Events in UML	76
Table 3.VI	UML Markings and Expressions for Time and Location	81
Table 4.II	Examples of Semi-Formal/Formal Integrations Not Involving Z	94
Table 5.I	Examples of Semi-formal/Formal Integrations Involving Z	97
Table 7.I	Abbreviations for Modelling Artefacts	209
Table 8.I	Correspondence between ELS Timing Requirements and Dasarathy Constraints	227
Table 10.I	Summary Comparison with Closely Related Approaches	271
Table B.I	Contents of the FCD Program	303
Table C.I	Menus of Menu Bar	360
Table C.II	Shortcut Buttons and Their Equivalent Menu Options	361
Table C.III	File Menu Items	362
Table C.IV	Other Icons for Artefacts	363
Table C.V	Edit Menu Items	364
Table C.VI	View Menu Items	365
Table C.VII	UML Menu Items	366
Table C.VIII	Z++ Menu Items	367
Table C.IX	Tools Menu Items	368
Table C.X	Windows Menu Items	369
Table C.XI	Help Menu Items	370
Table C.XII	UML Toolbox Items	371
Table C.XIII	Items of the Z++ Symbol Box	372

List of Figures

Fig 2.1	Domains of Research Space and Topic Location	21
Fig 2.2	Hard, Firm, and Soft Real-Time Systems	23
Fig 3.1	Partial Z Description of a Robot Arm	57
Fig 3.2	General Form of Z++ Class Declaration	61
Fig 3.3	Snapshot of Rational Software Corporation's Rational Rose	64
Fig 3.4	The 4+1 Architectural Views and UML Diagrams That Express Them	69
Fig 3.5	Overview of the Automatic Camshaft Testing System (ACTS)	70
Fig 3.6	Example of Use Case Diagram: Excerpt from ACTS Specification	72
Fig 3.7	Example of Class Diagram: Excerpt from ACTS Specification	73
Fig 3.8	Example of Object Diagram: Excerpt from ACTS Specification	74
Fig 3.9	Example of Sequence Diagram: Excerpt from ACTS Specification	75
Fig 3.10	Example of Signal: Excerpt from ACTS Specification	77
Fig 3.11	UML Symbols for Asynchronous and Synchronous Communication	79
Fig 3.12	Example of Statechart Diagram: A 2-Speed DC Motor for ACTS Axis X	80
Fig 4.1	First Zoom-In on The Research Space	93
Fig 4.2	Second Zoom-In on The Research Space	95
Fig 4.3	Jia's AML-based Approach	98
Fig 4.4	Noe and Hartum's Approach	100
Fig 4.5	The Octopus and Z Integration Approach	102
Fig 4.6	The RoZeLink Tool	103
Fig 4.7	The UML/Object-Z Combination	105
Fig 6.1	The Top-Level FCD Procedure	159
Fig 6.2	The CheckCDSyntax Procedure	159
Fig 6.3	The CheckRelationships Procedure	161
Fig 6.4	Alternative CheckRelationships Procedure	161
Fig 6.5	The CheckAcrossCD Procedure	162

Fig 6.6	The CheckClasses Procedure	162
Fig 6.7	The CDTranslate Procedure	164
Fig 6.8	The TranslateClasses Procedure	164
Fig 6.9	The TranslateClass Procedure	164
Fig 6.10	The TranslateAttributes Procedure	165
Fig 6.11	The TranslateAttribute Procedure	165
Fig 6.12	The Translate Operations Procedure	166
Fig 6.13	The TranslateOperation Procedure	166
Fig 6.14	The ProcessOpParams Procedure	167
Fig 6.15	The ProcessOpReturn Procedure	167
Fig 6.16	The PlaceZPPAttributes Procedure	168
Fig 6.17	The PlaceZPPOperations Procedure	168
Fig 6.18	The TranslateRelationships Procedure	169
Fig 6.19	The TranslateAggregation Procedure	169
Fig 6.20	The TranslateAssociation Procedure	170
Fig 6.21	General Form of A State Transition	172
Fig 6.22	The SDTranslate Procedure	186
Fig 6.23	The TranslateStates Procedure	187
Fig 6.24	The TranslateState Procedure	187
Fig 6.25	The TranslateTransitions Procedure	188
Fig 6.26	The ProcessCallTrans Procedure	188
Fig 6.27	The GenerateTransitOperation Procedure	189
Fig 6.28	The WriteHistoryPredicates Procedure	189
Fig 6.29	DCMotor State Diagram from the ACTS	190
Fig 6.30	Z++ Class DCMotor Generated by the AFSD	191
Fig 6.31	The AFD Procedure	200
Fig 6.32	The TranslateZPPClass Procedure	201
Fig 6.33	The GenerateUMLClass Procedure	201
Fig 7.1	The Procedural Frame	211
Fig 7.2	Regular Sequence of Modelling Activities	216
Fig 7.3	An Example of Irregular Flow of Modelling Activities	218

Fig 8.1	ELS Use Case Diagram	228
Fig 8.2	ELS Scenario: Outside Request A	230
Fig 8.3	ELS Scenario: Outside Request B	231
Fig 8.4	ELS Scenario: Outside Request C	232
Fig 8.5	ELS Scenario: Inside Request A	232
Fig 8.6	ELS Class Diagram: Initial Structure	233
Fig 8.7	ELS Sequence Diagram: Outside Request A	235
Fig 8.8	ELS Class Button	237
Fig 8.9	ELS State Diagram for the Button Class	237
Fig 8.10	ELS Class Elevator	237
Fig 8.11	ELS State Diagram for the Elevator Class	238
Fig 8.12	ELS Class Diagram with Attributes and Operations Attached to Classes	239
Fig 8.13	ELS Z++ Specification Generated by the AFCD	241
Fig 8.14	ELS Z++ Class Button Updated by the AFSD	246
Fig 8.15	ELS Z++ Class Elevator Updated by the AFSD	247
Fig 9.1	Harmony's Look	258
Fig 9.2	The View Menu	258
Fig 9.3	The Project Pane	260
Fig 9.4	The New Model Element Selector	260
Fig 9.5	Harmony with New Project Just Created.....	261
Fig 9.6	The UML Menu	262
Fig 9.7	A UML Toolbox	263
Fig 9.8	Harmony's Z++ Menu	264
Fig 9.9	The Z++ Symbol Box	265
Fig 9.10	Harmony Specific Buttons	266
Fig 9.11	The Legend Pane	267
Fig C.1	The Harmony Window	359
Fig C.2	Harmony's Menu Bar	360
Fig C.3	Harmony's Main Toolbar	361
Fig C.4	Harmony's File Menu	362

Fig C.5	Harmony's New Model Element Selector	363
Fig C.6	Harmony's Edit Menu	364
Fig C.7	Harmony's View Menu	365
Fig C.8	Harmony's UML Menu	366
Fig C.9	Harmony's Z++ Menu.....	367
Fig C.10	Harmony's Tools Menu	368
Fig C.11	Harmony's Window Menu	368
Fig C.12	Harmony's Help Menu	369
Fig C.13	Harmony's UML Toolboxes	370
Fig C.14	Harmony's Z++ Symbol Box	371
Fig C.15	Harmony's Legend Pane: COMP and CD Symbols	373
Fig C.16	Examples of Harmony Messages	374
Fig C.17	Harmony's About Message	374

List of Abbreviations

ACTS	Automatic Camshaft Testing System
ADF	Algorithm for Deformalisation
AFCDD	Algorithm for Formalising Class Diagrams
AFSD	Algorithm for Formalising State Diagrams
CASE	Computer-Aided Software Engineering
CD	Class Diagram
CLS	Class Specification
CLSTD	Class State Diagram
COMP	Class Compound
CSP	Communicating Sequential Processes
DFD	Data Flow Diagrams
ERD	Entity Relationship Diagrams
IDE	Integrated Development Environment
ISE	Integrated Specification Environment
GUI	Graphical User Interface
ELS	The Elevator System
OMG	Object Management Group
OMT	Object Modelling Technique
OO	Object-Oriented
OOA	Object-Oriented Analysis
OOD	Object-Oriented Design
OOSE	Object-Oriented Software Engineering
OOZ	Object-Oriented Variant of Z
ROOM	Real-Time Object Oriented Modelling
RT	Real-Time
RTL	Real-Time Logic
RTS	Real-Time Systems
SC	Scenario
SCG	Scenario Group
SQD	Sequence Diagram
SQDG	Sequence Diagram Group
TCS	Time-Constrained Systems
TL	Temporal Logic
UC	Use Case
UCD	Use Case Diagram
UML	Unified Modelling Language
ZPPC	Z++ Class
ZSPEC	Z++ Specification

Acknowledgements

First of foremost I would like to express my deepest gratitude to my thesis supervisor, Dr. Peter Hitchcock, who guided my research with infinite wisdom and patience, as well as with unabated confidence in my ability of completing the work presented here. I am much indebted to Dr. Hitchcock for introducing me to the rigorous, admirable world of formal notations, and for spending countless hours with me guiding the shaping of the approach described in this thesis.

I would also like to express my heartfelt gratitude to the members of my examining committee, Dr. Peter Bodorik, Dr. Trevor Smedley, and Dr. William Phillips, for their guidance and support throughout the course of my studies. In particular, I would like to thank Dr. Bodorik for supervising my earlier coursework and research, to express my gratitude to Dr. Smedley for opening for me the realm of visual notations, and to thank Dr. Phillips for his advice and encouragement.

I would also like to extend my thanks to the external examiner, Dr. Gregory Butler, from Concordia University, Montreal, for his thorough review and constructive critique of the thesis. Dr. Butler has pointed out with great clarity both the merits and the limitations of our thesis, his observations about the latter being extremely helpful for us to better evaluate our work and to define its future directions.

I would also like to express my gratitude to the former School of Computer Science and the former Technical University of Nova Scotia for the financial support I received in the form of scholarships and awards during my doctoral studies. I would like to thank both the former School of Computer Science at TUNS and the current Faculty of Computer Science at Dalhousie University, in particular to their directors, Dr. Jonathan Barzikai and, respectively, Dr. Jacob Slonim, for trusting me with teaching appointments as lecturer for the courses Software Development with Ada, Software Engineering, Database Management Systems, and Computer Organization for Electrical Engineers. Teaching these courses offered me the privilege of being constantly stimulated by the inquiring minds of my students and helped me consolidate the foundation of the present work.

Among the students who steadily motivated and encouraged me in my work Paul Evans, Greg Power, Chad Seward, Patrick Lee, Silvano Da Ros, Carmen Wong, Fernand Boudreau, Stephen Nickerson, Atreya Basu, Samer Mansour, and Jas Singh are the ones to whom I am especially grateful. Many thanks are due as well to my graduate colleagues Dawn Jutla, Eve Rosenthal, Marcel Karam, Krys Gawetski, and Arun Sood who provided me, in various phases of my studies, with a friendly and stimulating environment.

I am deeply indebted to Dr. Traian Ionescu, my former team leader and head of department at the Polytechnic University of Bucharest, Romania, who mentored my earlier professional steps with wisdom, goodwill, and grace. To him and his wife, Mrs. Mihaela Ionescu, I owe special thanks for their guidance and friendship throughout many years.

I was very fortunate to work with and be guided by several other senior professors at the Polytechnic University of Bucharest. I would like to take this opportunity and thank Dr. Radu Dobrescu, Dr. Sergiu Iliescu, and Dr. Aurelian Stanescu for giving me, years ago, through their personal examples, an excellent definition of the word academia.

To my longtime friend, Cristian Mierlea, I would like to say thank you for many things but especially for teaching me, many years ago, that in essence the computers are an extraordinary combination of mathematics and literature. And for inducing me, in those high school years, to value them dearly. For their constant support and encouragement throughout the years my heartfelt thanks go as well to my friend Mike Cailean and to his wife, Rodica Cailean. I would also like to express my thanks to a special friend, close to my heart and mind, Jorge Luis Borges, the great Argentine writer, presently in Heaven, in all probability narrating one of his fabulous stories. The beauty and finesse of his writings lifted my spirit in countless nights.

I have many special thanks for my parents, my wife, and my daughter. Words can hardly express my gratitude for them, but I trust they know how deeply I have appreciated their support throughout my studies. My daughter Diana grew up with a student father, quietly accepting to have me away from her in so many occasions. Finally, I would like to particularly thank my wife for her unfailing understanding, extended sacrifice, and unconditional love. I would like her to know that without her support this thesis would never have happened.

Abstract

This thesis is about the integration of semi-formal, graphical representations with formal notations within a modelling approach aimed at the construction of time-constrained systems (TCS). We believe that the two types of notation, graphical (semi-formal) and, respectively, formal, can efficiently complement each other and provide the basis for a software specification approach that can be both rigorous and practical. Although many authors have envisaged the advantages of combining informality with formality in software construction, there are very few reports that address the problem within the context of object-orientation and project its solution over the canvas of TCS modelling.

The pillars of our approach are the following: the combination of formal and semi-formal notations for specification purposes, the integration into an object-oriented approach of modelling capabilities that target properties of TCS, the elaboration of detailed algorithms for UML to Z++ translations, and the proposal of a procedural frame for effective and reliable development of TCS. Principles and an outline of an algorithm for the reverse translation, from Z++ to UML, are also included in the approach.

While the graphical notation employed is a subset of the UML, the formal notations used are Lano's Z++, an object-oriented variant of Z, and Jahanian and Mok's Real Time Logic. Both structural and dynamic aspects of the system are considered and a new modelling element denoted class compound is proposed.

From a methodological point of view, after several UML-based modelling steps are completed the formalisation process can take place, the result being a formal specification derived from the graphical representations obtained in the earlier steps. The integrated, semi-formal and formal model of the system can be subsequently enhanced while the designed translation mechanisms allow changes in the graphical representations to be reflected into the formal specifications as well as modifications of the formal specifications to be fed back into the diagrammatic descriptions of the system.

A case study, an Elevator System, is included in the thesis to illustrate the application of the proposed approach and the GUI-centred design of Harmony, an integrated specification environment intended to support the approach, is also presented.

Although we believe the proposed approach offers a viable solution for modelling software systems, it has nevertheless a number of limitations that need be pointed out. Firstly, the translation of UML constructs is restricted to a subset of the notation, and the treatment of state diagrams is confined to sequential, non-composite executions (composite states and aspects related to concurrency are not covered). Secondly, although timing constraints can be attached to structural UML constructs in the regular way, we have not tackled their mechanised translation to Z++, and there is a limited incorporation of such constraints in the state diagrams considered. Thirdly, the formal language employed, Z++, is currently lacking in supporting tools, which could be an impediment to the use of the proposed approach in industrial applications. Fourthly, for the formalisation algorithms a set of rules for well-formedness and a set of principles for translation are given without using meta-models for UML and Z++/RTL, yet the use of these meta-models would have probably allowed a more concise and precise description of the algorithms.

However, our belief is that, through future work, the above limitations can be overcome and our proposal can thus become a stronger contender in the landscape of object-oriented approaches for modelling TCS.