

Enabling Component-Based Model Transformations with QVT

by

Li Dan

Doctor of Philosophy in Software Engineering

2013



Faculty of Science and Technology

University of Macau



Enabling Component-Based Model Transformations with QVT

by

Li Dan

SUPERVISOR: Prof. Xiaoshan Li

CO-SUPERVISOR: Dr. Zhiming Liu

Department of Computer and Information Science

Doctor of Philosophy in Software Engineering

2013

Faculty of Science and Technology

University of Macau



Author's right 2013 by

LI Dan





Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisors, Prof. Xiaoshan Li and Dr. Zhiming Liu, for giving me the chance to start my PhD study, and for their guidance and help over the past four years. Also, I would like to thank Prof. Gong Zhiguo, Dong Mingchui, Xu Qiwen, and Robert P. Biuk-Aghai of the University of Macau, and Prof. Anders P. Ravn of Aalborg University, Prof. T.H. Tse of the University of Hong Kong, Prof. Shaoying Liu of Hosei University, for having taken time and efforts to review my thesis and to serve on the examination committees of my thesis proposal and oral defense.

Particularly, I am grateful to Dr. Volker Stolz of the University of Oslo, who has helped me a lot during my PhD study. I would also like to express my thanks to my colleagues Mr. Liang Zhao, Ruzhen Dong, Xiaoliang Wang, Jiaqi Zhu, Jian Zhu, Yuanzhang Chang, Danning Li, Ms. Shuling Wang, Lijie Yang, Prof. Mingyi Zhang, and all the colleagues of UNU-IIST, University of Macau, Institute of Software of CAS, and Guizhou Academy of Sciences, for their cooperation, discussion and friendly advice that are of great help in my work. Special thanks to UNU-IIST and general services staff Ms. Wendy Hoi, Alice Pun and Kitty Chan for their administrative support these years.

I am greatly indebted to my family - my parents, my wife and my son, for their patience, love and endless support in all these years I was away from home.

Chapter 2 & 3 of this thesis are partly based on the literature [46; 74; 134; 135; 148] and [38; 39; 82; 97; 112; 181], respectively. I would like to acknowledge and thank all the authors of these publications.

Abstract

Model Driven Architecture (MDA) has become a mainstream framework for *model-driven* and *component-based* development. To achieve the faster and less costly software development promised by MDA, automating the development process with tools is critical, but not easy. One reason is that MDA lacks a precise and concrete development methodology. On the other side, stepwise refinement is a formal mechanism which facilitates the development of systems correct-by-construction by incrementally applying the refinement rules. But the limitation of available refinement rules and the lack of integrated support tools make the method hard to scale up to complex models.

The research presented in this thesis aims at exploring techniques to automate the MDA development process through application of the refinement mechanism as model transformations. We use the *refinement for Component and Object Systems* (rCOS) in the research, as the formal theory supports component-based modeling, and includes a comprehensive refinement calculus. We define a UML profile that augments rCOS with standard UML elements and diagrams, implement a set of the refinement rules as model transformations, and integrate them into a CASE tool. Thus the full component-based development process can be driven by interactively applying the refinement transformations. Particularly, we present a transformation from object-oriented models to component models, so the OO design and component-based techniques can be seamlessly combined in the development process.

Automation of these transformations places a high demand on transformation tools. We adopt the graphical notation of *QVT Relations*, an intuitive and powerful language for defining model transformations. But currently there is no practical support tool available. We propose an approach to implement QVT transformations as executable XSLT programs. A tool, named QVTR-XSLT, has been developed to support the approach. The tool has a graphical editor for specifying a QVT transformation with the graphical notation, and a code generator which automatically translates the transformation to XSLT programs. However, the QVTR-XSLT tool marks a useful attempt to bring the OMG standard language to practice.

Declaration

I declare that this thesis has not been previously submitted for a degree or other qualification at this or any other higher education institution. To the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference or acknowledgement is made.

I also acknowledge that I have read and understood the Rules on Handling Student Academic Dishonesty and the Regulations of the Student Discipline of the University of Macau.



Contents

Contents	iv
List of Figures	x
1 Introduction	1
1.1 General Background	4
1.2 Specific Background	17
1.3 Research Goals and Objectives	19
1.4 Research Methodology and Design	21
1.5 Potential Contributions	23
1.6 Organization of the Thesis	25
1.7 Statement of Originality	28
2 Model-Driven Development	30
2.1 Model Driven Architecture (MDA)	31
2.1.1 Models of MDA	32
2.1.2 Development process of MDA	34
2.2 Meta Object Facility (MOF)	35
2.2.1 Essential MOF (EMOF)	37
2.2.2 XML Metadata Interchange (XMI)	38
2.3 Unified Modeling Language (UML)	39
2.3.1 UML diagrams	40
2.3.2 UML profiles	48
2.4 Object Constraint Language (OCL)	51
2.5 Model Transformations and QVT	53
2.6 Component-Based Development (CBD)	60
2.6.1 Component definition	62

2.6.2	Component composition	64
2.6.3	Component diagram	65
2.6.4	Component interaction protocol	66
2.6.5	Component-based development process	67
3	Refinement for Component and Object Systems – rCOS	69
3.1	UTP - Background of rCOS	70
3.1.1	Programs as designs	72
3.1.2	Refinement of designs	73
3.1.3	Reactive designs	74
3.2	Object-Orientation of rCOS	75
3.2.1	Syntax of rCOS	76
3.2.2	Semantics of rCOS	79
3.2.3	Object-oriented refinements of rCOS	82
3.3	Component Model of rCOS	85
3.3.1	Close components and refinement	86
3.3.2	Open components and composition	88
3.3.3	Processes and coordination	91
3.4	Development Method of rCOS	92
3.4.1	Requirements modeling	94
3.4.2	Object-oriented design	97
3.4.3	Component-based design	98
3.4.4	Platform specific design and implementation	100
3.5	Tool Support for rCOS	101
4	QVTR-XSLT: A Tool for Graphical Notation of QVT Relations	102
4.1	State of the Art	108
4.1.1	Graph transformation languages and tools	109
4.1.2	QVT-like languages and tools	111

CONTENTS

4.1.3	Efforts to support graphical notation of QVT Relations	115
4.1.4	XSLT related approaches	116
4.2	QVT Relations and XSLT	117
4.2.1	QVT Relations	117
4.2.2	XPath	122
4.2.3	XSLT	124
4.2.4	Comparison of QVT Relations and XSLT	127
4.3	OCL to XPath	129
4.3.1	Mapping OCL types	130
4.3.2	Mapping OCL expressions	131
4.3.3	Mapping OCL operations	134
4.4	QVT Relations to XSLT	136
4.4.1	Mapping source domain to selection pattern	137
4.4.2	Creating variable declarations	139
4.4.3	Generating construction instructions	140
4.4.4	Translating relation calls	141
4.5	Support Semantics of QVT Relations	142
4.5.1	Transformation direction and check-before-enforce semantics	142
4.5.2	Relation domains	143
4.5.3	Rule scheduling	144
4.5.4	Pattern matching	146
4.5.5	Object creation using patterns	147
4.5.6	Transformation traceability	148
4.5.7	Transformation composition	149
4.5.8	In-place transformation	151
4.6	Extensions to QVT Relations	153
4.6.1	Transformation parameter	153
4.6.2	Relation alias name and priority	155

4.6.3	Conditional relation call	157
4.7	Model Query with QVT-R Graphical Notation	157
4.7.1	Syntax of the query facility	158
4.7.2	Semantics of the query facility	161
4.7.3	Mapping domain patterns to XPath path expressions	162
4.7.4	Generating XSLT functions for queries	164
4.8	Tool Design and Implementation	165
4.8.1	A UML profile for QVT Relations	165
4.8.2	QVT-R graphical editor	173
4.8.3	XSLT code generator	176
4.8.4	A collection of predefined transformations	177
4.9	Case Studies	178
4.9.1	Simple UML to simple RDBMS transformation	178
4.9.2	Model querying examples	184
4.10	Summary	190
5	A UML Profile for rCOS	193
5.1	A Metamodel of rCOS	194
5.1.1	Types and classes of rCOS	194
5.1.2	Components of rCOS	196
5.2	A UML Profile for rCOS	198
5.2.1	Mapping rCOS Concepts to UML	199
5.2.2	rCOS modeling architecture	201
5.2.3	Class and interface modeling	202
5.2.4	Component modeling	204
5.2.5	rCOS sequence diagram	209
5.2.6	State machine diagram	212
5.2.7	Visual diagram modeling	214

CONTENTS

5.2.8	Profile constraints	216
5.2.9	Profile implementation	217
5.3	Supporting rCOS Development	220
5.3.1	Requirements modeling	220
5.3.2	Object-oriented design	222
5.3.3	Component-based model design	223
5.3.4	Code generation	224
5.4	Related Work	224
5.5	Summary	227
6	Enabling Model Transformations of rCOS	228
6.1	From Requirements Models to Design Models	232
6.1.1	From sequence diagrams to protocol state machines	234
6.1.2	Labeling protocol state machines	241
6.1.3	Generation of controller classes	243
6.1.4	An example : clinic reception	246
6.2	From OO Design- to Component-Based Models	250
6.2.1	Selection of object lifelines	251
6.2.2	Validating the lifeline selection	253
6.2.3	Generating a component from selected lifelines	255
6.2.4	Implementation	256
6.3	Case Study	258
6.3.1	Requirements modeling	258
6.3.2	Object-oriented design	262
6.3.3	Component-based design	267
6.4	Related Work	277
6.5	Summary	281

7	Conclusions and Future Work	283
7.1	Conclusions	283
7.2	Limitations of Current Study	290
7.3	Perspectives for Future Work	293
	References	295
	Appendix	311
A	EBNF for OCL used in QVTR-XSLT tool	312
A.1	Basic definitions	312
A.2	Expression definitions	313
B	Algorithms for OO- to component sequence diagram transformation	314
B.1	Calculate strongly connected lifelines	314
B.2	Generate component from selected object lifelines	314
B.3	Generate required interface	316
	Curriculum Vitae	317

List of Figures

1.1	Research methodology (adapted from [127])	22
1.2	Dependency relationships of the major chapters	28
2.1	The four modeling layers of MOF	36
2.2	Extract of the EMOF metamodel	37
2.3	A use case diagram	42
2.4	A class diagram for the POS system	44
2.5	A state diagram for class <i>Sale</i>	45
2.6	A sequence diagram for operation <i>enterItem</i>	47
2.7	Model transformation	54
2.8	QVT Architecture (from [134])	57
3.1	Basic operations of UTP	72
3.2	Syntax of rCOS OO language (RCOSP)	76
3.3	rCOS development methodology	93
4.1	Overall process of using QVTR-XSLT tool	105
4.2	Comparison of transformation approaches and tools	114
4.3	QVT Relations metamodel	118
4.4	Example QVT-R relation in graphical notation	119
4.5	An excerpt of XML data model	122
4.6	XSLT metamodel	124
4.7	Trace class model	148
4.8	In-place transformation example - relation PullUpAttribute.	152
4.9	Abstract syntax of QVT-R extensions	153
4.10	Definition of additional transformation parameters.	154
4.11	Metamodel of the model query	159

4.12	Example of the model query	161
4.13	The UML profile for QVT-R	166
4.14	The <i>MetaModel</i> package of QVT-R profile.	169
4.15	The stereotype for mapping UML to XMI	169
4.16	A stereotype for in-place transformation.	170
4.17	The toolbar for the QVT-R graphical editor	174
4.18	Validating the transformation model	175
4.19	The XSLT code generator	176
4.20	Simple UML metamodel	180
4.21	Simple RDBMS metamodel	180
4.22	The transformation model	180
4.23	The root relation	180
4.24	Class to table	181
4.25	Attribute to column	181
4.26	Association to foreign key	181
4.27	Primitive attribute to column	181
4.28	Complex attribute to column	182
4.29	Attribute in super class to column	182
4.30	Target metamodel	185
4.31	Top-level relation	185
4.32	Display query result	185
4.33	Count number of classes	186
4.34	Classes with number of associations	186
4.35	Maximal depth of inheritance	188
4.36	Number of super classes	188
4.37	Find elements by name	189
4.38	Getting attribute names	189

LIST OF FIGURES

5.1	Types package of rCOS metamodel	195
5.2	Class package of rCOS metamodel	195
5.3	Components package of rCOS metamodel	197
5.4	Mapping from rCOS concepts to UML	199
5.5	Overall structure of rCOS model	201
5.6	<i>Class</i> and <i>Interface</i> models	203
5.7	Modeling rCOS component	205
5.8	Realizing component composition	206
5.9	An example of component composition	208
5.11	rCOS sequence diagram	209
5.12	Protocol state machine	212
5.13	Metamodel of diagram interchange (DI)	215
5.14	Definitions of rCOS profile	218
5.15	Requirements modeling in the rCOS modeler	221
5.16	OO design in rCOS modeler	222
5.17	Component-based design in the rCOS modeler	223
6.1	Starting the <i>SD</i> to <i>PSM</i> transformation	235
6.2	Rule for <i>message occurrence</i>	236
6.3	Applying rule for <i>message occurrence</i>	236
6.4	Applying rule for <i>alt</i>	237
6.5	Applying rule for <i>break</i>	237
6.6	Applying rule for <i>loop</i>	238
6.7	Applying rule for <i>opt</i>	238
6.8	Applying rules for <i>par</i> and <i>critical</i>	240
6.9	Applying rule for <i>ref</i>	240
6.10	Labeled <i>PSM</i> from Fig. 6.7	242
6.11	Labeled <i>PSM</i> from Fig. 6.8	242

6.12	Use case and component for clinic reception	246
6.13	Interface and controller class for component <i>Reception</i>	246
6.14	System sequence diagram for interface <i>ReceptDesk</i>	247
6.15	Protocol state machine for interface <i>ReceptDesk</i>	248
6.16	A relation to generate a component lifeline	257
6.17	Domain class diagram of CoCoME example	259
6.18	Use case <i>ProcessSale</i>	259
6.19	Component <i>ProcessSale</i>	259
6.20	Sequence diagram for <i>CInter</i>	260
6.21	Interface of <i>ProcessSale</i>	260
6.22	Protocol state machine for <i>CInter</i>	261
6.23	The controller class for interface <i>CInter</i>	262
6.24	Interactively applying expert pattern	263
6.25	The object sequence diagram after applying expert pattern	264
6.26	The object sequence diagram after OO refinement	265
6.27	Design class diagram of CoCoME example	265
6.28	Select lifelines to become a component	268
6.29	Validation error message	269
6.30	The sequence diagram after the first transformation	270
6.31	Component diagram after first transformation	270
6.32	Definitions of component <i>COM_proce</i>	271
6.33	Component sequence diagram for component <i>ProcessSale</i>	271
6.34	Component diagram for <i>ProcessSale</i>	272
6.36	SD of <i>COM_store</i>	273
6.37	PSM of <i>COM_store</i>	273
6.38	PSM for the required interface of component <i>COM_proce</i>	273
6.39	Apply component composition	274
6.40	Result of component composition	275

LIST OF FIGURES

6.41	Sequence diagram for component <i>COM_pay_SH</i>	275
6.42	PSM for component <i>COM_pay_SH</i>	276
6.43	Optimized component diagram for <i>ProcessSale</i>	277

