To my family
# CONTENTS IN BRIEF

## PART I

**FUNDAMENTALS**

1. **INTRODUCTION**  3

2. **MODEL-DRIVEN ENGINEERING FOUNDATIONS**  9

3. **ONTOLOGY FOUNDATIONS**  21

4. **MARRYING ONTOLOGY AND MODEL-DRIVEN ENGINEERING**  44

---

**CONCLUSION OF PART I**

## PART II

**THE TWOUSE APPROACH**

5. **THE TWOUSE CONCEPTUAL ARCHITECTURE**  65

6. **QUERY LANGUAGES FOR INTEGRATED MODELS**  78

7. **THE TWOUSE TOOLKIT**  86

---

**CONCLUSION OF PART II**

## PART III

**APPLICATIONS IN MODEL-DRIVEN ENGINEERING**

8. **IMPROVING SOFTWARE DESIGN PATTERNS WITH OWL**  99

9. **MODELING ONTOLOGY-BASED INFORMATION SYSTEMS**  112
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>ENABLING LINKED DATA CAPABILITIES TO MOF COMPLIANT MODELS</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td>CONCLUSION OF PART III</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONCLUSION OF PART IV</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>MODEL-DRIVEN SPECIFICATION OF ONTOLOGY TRANSLATIONS</td>
<td>141</td>
</tr>
<tr>
<td>12</td>
<td>AUTOMATIC GENERATION OF ONTOLOGY APIs</td>
<td>156</td>
</tr>
<tr>
<td>13</td>
<td>USING TEMPLATES IN OWL ONTOLOGIES</td>
<td>171</td>
</tr>
<tr>
<td>14</td>
<td>CONCLUSION</td>
<td>187</td>
</tr>
</tbody>
</table>
# CONTENTS

LIST OF FIGURES  xv

LIST OF TABLES  xix

FOREWORD  xxi

PREFACE  xxiii

ACRONYMS  xxvii

## PART I

## FUNDAMENTALS

1  INTRODUCTION  3

1.1 Motivation  3
1.2 Research Questions  5

2  MODEL-DRIVEN ENGINEERING FOUNDATIONS  9

2.1 Introduction  9
2.2 Model-Driven Engineering Structure  9
  2.2.1 Models  11
  2.2.2 Metamodels  11
  2.2.3 Modeling Languages  13
  2.2.4 Model Transformations  17
  2.2.5 Query Languages  17
2.3 Technical Spaces  19
2.4 Conclusion  20

3  ONTOLOGY FOUNDATIONS  21

3.1 Introduction  21
3.2 Ontology  22
  3.2.1 Ontology Modeling  22
3.3 The Ontology Web Language  24
  3.3.1 OWL 2 Syntax  24
  3.3.2 OWL 2 Semantics  27
  3.3.3 World Assumption and Name Assumption  27
CONCLUSION OF PART I

PART II
THE TWOUSE APPROACH

5 THE TWOUSE CONCEPTUAL ARCHITECTURE 65
5.4 Metamodeling Architecture 70
5.4.1 The TwoUse Metamodel 70
5.5 Syntax 72
5.5.1 UML Profile for OWL 72
5.5.2 Pure UML Class Diagrams 75
5.5.3 Textual Notation 75
5.6 Conclusion 77

6 QUERY LANGUAGES FOR INTEGRATED MODELS 78

6.1 Introduction 78
6.2 Combining Existing Approaches 78
6.3 Querying Ontologies Using OWL Syntax: SPARQLAS 80
6.3.1 SPARQLAS Concrete Syntax 80
6.3.2 SPARQLAS Metamodel 81
6.3.3 Transformation from SPARQLAS to SPARQL 81
6.4 Querying Integrated Models: SPARQLAS4TwoUse 82
6.5 Conclusion 84

7 THE TWOUSE TOOLKIT 86

7.1 Introduction 86
7.2 Use Case Descriptions 87
7.3 A Generic Architecture for MDE and Ontology Engineering 87
7.3.1 Core Services 88
7.3.2 Engineering Services 89
7.3.3 Front-End 90
7.4 Instantiating the Generic Model-Driven Architecture: The TwoUse Toolkit 90
7.5 Conclusion 93

CONCLUSION OF PART II

PART III
APPLICATIONS IN MODEL-DRIVEN ENGINEERING

8 IMPROVING SOFTWARE DESIGN PATTERNS WITH OWL 99

8.1 Introduction 99
8.2 Case Study 100
8.2.1 Applying the Strategy Pattern 100
8.2.2 Extending to the Abstract Factory 101
8.2.3 Drawbacks 103
8.3 Application of the TwoUse Approach 104
8.3.1 OWL for Conceptual Modeling 104
8.3.2 TwoUse for Software Design Patterns: The Selector Pattern 105
11.3 Application of the TwoUse Approach 145
   11.3.1 Concrete Syntax 145
   11.3.2 Metamodels 146
   11.3.3 Model Libraries 148
   11.3.4 Semantics 148
   11.3.5 Ontology Translation Process 148
   11.3.6 Implementation 149
11.4 Examples 150
11.5 Analysis 153
11.6 Related Work 154
11.7 Conclusion 155

12 AUTOMATIC GENERATION OF ONTOLOGY APIS 156

   12.1 Introduction 156
   12.2 Case Study 158
   12.3 Application of the TwoUse Approach 161
      12.3.1 Key Domain Concepts 161
      12.3.2 agogo Concrete Syntax by Example 163
      12.3.3 Implementation 166
   12.4 Analysis 167
   12.5 Related Work 169
   12.6 Conclusion 170

13 USING TEMPLATES IN OWL ONTOLOGIES 171

   13.1 Introduction 171
   13.2 Case Study 172
   13.3 Application of the TwoUse Approach 174
      13.3.1 Extending the OWL Metamodel with Templates 174
      13.3.2 Semantics of Templates 177
      13.3.3 Notations for Templates in OWL 179
      13.3.4 Query Templates 180
   13.4 Analysis 181
      13.4.1 Limitations 182
   13.5 Related Work 182
   13.6 Conclusion 183

CONCLUSION OF PART IV

14 CONCLUSION 187

   14.1 Contributions 187
   14.2 Outlook 189
      14.2.1 Ongoing Research 189
APPENDIX A 191

A.1 EBNF Definition of the Concrete Textual Syntax for TwoUse 191
A.2 EBNF Grammar of SPARQLAS Functional Syntax 192
A.3 EBNF Grammar of SPARQLAS Manchester Syntax 197
A.4 SPARQLAS Metamodel 202
A.5 Ecore to OWL: Translation Rules 204

APPENDIX B 206

B.1 Use Cases 206
  B.1.1 Design Integrated Models 206
  B.1.2 Design Integrated UML Class Diagram 206
  B.1.3 Design Integrated Ecore Model 207
  B.1.4 Specify SPARQLAS4TwoUse Query Operations 207
  B.1.5 Transform to OWL 207
  B.1.6 Compute Alignments 208
  B.1.7 Browse 208
  B.1.8 Explain Axioms 209
  B.1.9 Query UML Class-Based Models 209
  B.1.10 Query OWL Ontologies 209
  B.1.11 Design Ontology Engineering Services 209
  B.1.12 Design Ontology API 210
  B.1.13 Design Ontology Translation 210
  B.1.14 Design Ontology Template 210
  B.1.15 Generate Service 211
B.2 Connecting Use Cases with Requirements 211

REFERENCES 212

INDEX 226
LIST OF FIGURES

1.1 Context of the Book. 6
2.1 Main Concepts of Megamodel. 10
2.2 Notion of RepresentationOf in Megamodel. 11
2.3 Notion of ConformsTo in Megamodel. 12
2.4 Layered Architecture. 12
2.5 EMOF Classes. 13
2.6 Ecore Structure. 14
2.7 Structure, Semantics, and Syntax of the UML Language. 16
2.8 UML Class Diagram of an E-Shop System. 18
2.9 MOF Technical Space. 19
3.1 Semantic Web Stack Covered in This Chapter. 22
3.2 E-Shop Example with Description Logic Syntax. 23
3.3 Closing the Domain of E-Shop with OWL Axioms. 30
3.4 OWL Class Descriptions of the OMG OWL Metamodel [114]. 35
3.5 OWL Properties of the OMG OWL Metamodel [114]. 36
3.6 RDF S Properties of the OMG OWL Metamodel [114]. 36
3.7 OWL Class Descriptions of the NeOn Metamodel. 37
3.8 OWL Properties of the NeOn Metamodel. 37
3.9 OWL Class Descriptions of the OWL 2 Metamodel. 38
3.10 OWL Properties of the OWL 2 Metamodel. 39
3.11 Snippets of the SWRL Metamodel and the Connections with the OWL Metamodel. 39
3.12 Snippets of the SPARQL Metamodel. 40
3.13 The Description Logics Technical Space. 41
3.14 Relation between the EBNF Technical Space and the Description Logics Technical Space. 42
3.15 Model-Driven Viewpoint of Ontology Technologies. 42
4.1 Marrying MMTS and OTS. 45
4.2 Comparing UML Class Diagrams, OWL-DL, OWL 2, and DL-Lite. 46
4.3 Snippet of the Feature Model of Bridging OTS and MMTS. 48
4.4 Organization of Features According to Technical Space. 53
4.5 Checking Consistency of UML Models. 54
4.6 Feature Model Configuration for Model Checking. 55
4.7 Mapping between Two Models Ma and Mb. 56
4.8 Feature Model Configuration for Model Enrichment. 57
4.9 Ontology Modeling with UML Profile. 58
4.10 Feature Model Configuration for Ontology Modeling. 59
5.1 The TwoUse Conceptual Architecture. 69
5.2 Adapting the OWL Class for UML Class-Based Modeling. 71
5.3 The OWL 2 Metamodel Adapted for the UML Class-Based Metamodel — the TwoUse Metamodel. 73
5.4 UML Class Diagram Profiled with UML Profile for OWL and TwoUse Profile. 74
6.1 Existing Approaches for Querying Models. 79
6.2 Variables in the SPARQLAS Metamodel. 81
6.3 Composing the SPARQLAS Metamodel and the TwoUse Metamodel. 83
6.4 Snapshot of the Running Example. 84
6.5 Positioning SPARQLAS4TwoUse among Existing Approaches. 85
7.1 Use Case for a Generic Architecture for MDE and Ontology Engineering. 88
7.2 A Generic Architecture for MDE and Ontology Engineering. 89
7.3 The TwoUse Toolkit. 91
7.4 TwoUse Toolkit Snapshot: Explanation Service. 92
7.5 TwoUse Toolkit Snapshot: View Inferred Class Hierarchy. 92
7.6 Instantiation of the Generic Architecture: The TwoUse Toolkit. 93
8.1 Application of the Strategy Pattern in the Running Example. 101
8.2 Drawbacks of the Strategy Pattern. 102
8.3 Strategy and Abstract Factory Patterns with Configuration Object. 102
8.4 UML Sequence Diagram of Strategy and Abstract Factory Patterns with Configuration Object. 103
8.5 Domain Design by a UML Class Diagram Using a UML Profile for OWL. 105
8.6 Profiled UML Class Diagram of an Ontology-Based Solution. 106
8.7 Profiled UML Class Diagram with the Strategy Pattern. 107
8.8 Sequence Diagram of an OWL-Based Solution. 108
8.9 Structure, Participants, and Collaborations in the Selector Pattern. 109
9.1 UML Class Diagram and Sequence Diagram of KAT Algorithms. 114
9.2 UML Class Diagram of KAT. 118
9.3 Excerpt of a KAT model (M1). 120
9.4 Snapshot of KAT (M0). 122
10.1 Development Life Cycle of the TwoUse Toolkit. 125
10.2 Snippets of Use Case Diagram from TwoUse Toolkit. 126
10.3 Snippets of BPMN Diagram from TwoUse Toolkit. 126
10.4 Snippets of Component Diagram from TwoUse Toolkit. 127
10.5 Snippet of BPMN metamodel and UML metamodel for Use Cases. 129
10.6 Mapping Ecore and OWL. 131
11.1 Ontology Mapping Challenge for the Running Example. 143
11.2 Abstraction vs. Expressiveness. 145
11.3 Example of a Translation Rule. 146
11.4 Fragment of the ATL Metamodel. 147
11.5 Snippet of the Package Type and Package Expressions of the OCL Metamodel. 147
11.6 Ontology Translation Process. 149
11.7 Screenshot of MBOTL. 150
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1</td>
<td>Limitations of Current Approaches.</td>
<td>157</td>
</tr>
<tr>
<td>12.2</td>
<td>Ontology and API for the Semantic Annotation Pattern.</td>
<td>158</td>
</tr>
<tr>
<td>12.3</td>
<td>Snippet of the <em>agogo</em> Metamodel.</td>
<td>162</td>
</tr>
<tr>
<td>12.4</td>
<td>Architecture of the <em>agogo</em> Approach.</td>
<td>166</td>
</tr>
<tr>
<td>12.5</td>
<td>Screenshot of <em>agogo</em> Implementation.</td>
<td>167</td>
</tr>
<tr>
<td>13.1</td>
<td>Modeling the Running Example with OMG UML Profile for OWL and UML Profile for SWRL.</td>
<td>175</td>
</tr>
<tr>
<td>13.2</td>
<td>Metamodel for Ontology Templates.</td>
<td>176</td>
</tr>
<tr>
<td>13.3</td>
<td>The Template Binding Realization Algorithm.</td>
<td>178</td>
</tr>
<tr>
<td>13.4</td>
<td>Modeling the Running Example with OWL 2 Graphical Syntax.</td>
<td>179</td>
</tr>
<tr>
<td>13.5</td>
<td>Ontology Development with Templates.</td>
<td>180</td>
</tr>
<tr>
<td>A.1</td>
<td>SPARQLAS Metamodel</td>
<td>203</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.1</td>
<td>Syntax of Class Expression Axioms</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Syntax of Object Property Axioms</td>
<td>25</td>
</tr>
<tr>
<td>3.3</td>
<td>Syntax of Data Property Axioms</td>
<td>25</td>
</tr>
<tr>
<td>3.4</td>
<td>Syntax of Assertions</td>
<td>26</td>
</tr>
<tr>
<td>3.5</td>
<td>Syntax of Class Expressions</td>
<td>26</td>
</tr>
<tr>
<td>3.6</td>
<td>Syntax of Data Ranges</td>
<td>27</td>
</tr>
<tr>
<td>3.7</td>
<td>Semantics of Class Expression Axioms</td>
<td>27</td>
</tr>
<tr>
<td>3.8</td>
<td>Semantics of Object Property Axioms</td>
<td>28</td>
</tr>
<tr>
<td>3.9</td>
<td>Semantics of Data Property Axioms</td>
<td>28</td>
</tr>
<tr>
<td>3.10</td>
<td>Semantics of Assertions</td>
<td>28</td>
</tr>
<tr>
<td>3.11</td>
<td>Semantics of Class Expression</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>OTS and MMTS: Comparable Features</td>
<td>45</td>
</tr>
<tr>
<td>5.1</td>
<td>Correlating Building Blocks with Requirements</td>
<td>70</td>
</tr>
<tr>
<td>5.2</td>
<td>Mapping between the UML Profile for OWL (Hybrid Diagram) and the TwoUse Metamodel</td>
<td>75</td>
</tr>
<tr>
<td>6.1</td>
<td>Evaluation of SPARQLAS Expressions According to the Running Example Snapshot</td>
<td>84</td>
</tr>
<tr>
<td>9.1</td>
<td>Specifying KAT with Description Logic Syntax</td>
<td>116</td>
</tr>
<tr>
<td>9.2</td>
<td>Evaluation of SPARQLAS Expressions According to the KAT Snapshot</td>
<td>121</td>
</tr>
<tr>
<td>10.1</td>
<td>TwoUse Measurement</td>
<td>132</td>
</tr>
<tr>
<td>11.1</td>
<td>Satisfying Ontology Translation Requirements</td>
<td>153</td>
</tr>
<tr>
<td>12.1</td>
<td>Comparison of Size between agogo and the Current COMM API in Two Cases</td>
<td>168</td>
</tr>
<tr>
<td>12.2</td>
<td>Correlating agogo Requirements with Quality Attributes</td>
<td>169</td>
</tr>
<tr>
<td>B.1</td>
<td>Mapping Use Cases and Requirements</td>
<td>211</td>
</tr>
</tbody>
</table>
Software modeling is in a schizophrenic situation. On the one hand, it is targeted towards the development of completely formal systems, i.e., executable code. On the other hand, the tools dominating in software modeling are typically drawing tools prepared with specific graphical icons. This dichotomy implies that the targeted meaning of a software model is limited in its use towards human understanding and communication only.

This dichotomy is reconciled when software is enriched with formulae specifying the functionality of the code. This is an exciting branch in software engineering, however, for the time being, this is a very labor-intensive exercise that can only be applied for smaller scale systems with particular value, e.g., strong safety requirements.

The above-explained dichotomy is also reduced when software models are exploited in model-driven development for the semi-automatic derivation of more formal models, e.g., executable code (stubs). In such model-driven development the meaning of a model is implicitly defined by mapping it into a (more), formal model. This (more) formal model, however, is exclusively oriented towards operational semantics, it does not bear any semantic meaning for issues like organization and modularization of software models.

Hence, what is obviously missing is a stronger notion of meaning for software models themselves. A meaning that is not only accessible to human interpretation, but that can be operationalized on the software model alone and not only on one view of a software model but on different sublanguages that together constitute a software modeling framework.

In this book, Fernando Silva Parreiras makes a major step towards realizing such meaning for software models. With his methodology TwoUSE—Transforming and Weaving Ontologies and UML for Software Engineering—he combines the established routines of current-day software modelers with the most recent technology for reasoning over large and complex models, i.e., ontology technology.

Ontology technology, based on the family of description logics dialects, has thrived over the last 15 years, coming from small formal systems where it was hardly possible to manage 102 entities in one model to systems that reason over 105 entities—and growing. It is the core target of ontology technologies to model classes, their relationships, and their instances in a versatile manner that still leads to a decidable logical language, which can (mostly) be reasoned about for models that do not appear in the worst case, but in practice. Hence, ontology technology is ideally suited to be carried over to the world of software models.
Such a step seems to be incremental at first sight. This, however, is not the case. The reason is that it is not sufficient to come up with a single mapping, e.g., from UML class diagrams to an ontology language, because the range of software models is ranging much farther and what is needed is a methodology with example cases and best practices rather than an ad hoc development.

Fernando Silva Parreiras has accomplished such a methodology with TwoUse. And this methodology has become influential even before this book could be published. First, the EU project MOST—Marrying Ontology and Software Technologies—running from February 2008 to April 2011 has relied heavily on Fernando’s TwoUse methodology and has taken it as a major source of inspiration for further developing best practices for using ontology technologies in software development. Second, his work has become pivotal for other researchers in our lab—and beyond—who have been building on the integration of software models and ontologies and have further refined it, most notably Tobias Walter and Gerd Gröner.

Finally, the development of TwoUse has been a major accomplishment, because its development has been off the beaten path between the software modeling and the ontology technology communities and staying within neither. At the same time, advising Fernando and charting unexplored research terrain with him has become one of my most beloved research experiences of the last years—intellectually and personally—one that I would not want to miss by any means.

Steffen Staab
Koblenz, Germany

April 2012
The audience for this book embraces computer science graduate students, researchers, advanced professionals, practitioners, and implementers in the areas of software engineering, knowledge engineering, and artificial intelligence, interested in knowing the possibilities of using semantic web technologies in the context of model-driven software development or in enhancing knowledge engineering process with model-driven software development.

For the knowledge engineering community, the advent of ontology engineering required adapting methodologies and technologies inherited from software engineering to an open and networked environment. With the advances provided by model-driven software development, the semantic web community is keen on learning what the benefits are of disciplines like metamodeling, domain-specific modeling, and model transformation for the semantic web field.

For software engineering, declarative specification is one of the major facets of enterprise computing. Because the Ontology Web Language (OWL) is designed for sharing terminologies, interoperability, and inconsistency detection, software engineers will welcome a technique that improves productivity and quality of software models. This book is relevant for researchers who work in the field of complex software systems using model-driven technology and for companies that build large-scale software like enterprise software offerings, data-warehousing products, and software product lines.

HOW TO READ THIS BOOK

In Part I, we present the fundamental concepts and analyze state-of-the-art approaches. Chapters 2 and 3 describe the concepts and technologies around MDE and ontologies, respectively. In Chapter 4, we present the commonalities and variations of both paradigms, analyze existing work in this area, and elicit the requirements for an integrated solution.

Part II describes the role of MDE techniques (DSL, model transformation, and metamodeling) and ontology technologies (reasoning services, query answering) in an integrated approach. In Chapters 5 and 6, we describe the conceptual architecture of our approach. Chapter 7 presents the TwoUse Toolkit—the implementation of the conceptual architecture.

We use the TwoUse Toolkit to realize case studies from the model-driven engineering and ontology engineering domains. Part III assembles case studies that use our approach at the modeling level and at the language level. Chapter 8 analyzes the application of TwoUse in software design patterns, and in Chapter 9 we present
the application of TwoUse in ontology-based information systems. Chapter 10 describes the usage of TwoUse to support software developers in integrating software languages.

Part IV presents an analysis of employing our approach in ontology engineering services. We address the need for multiple languages for ontology mapping in Chapter 11. Chapter 12 presents a domain-specific language for specifying ontology APIs. Chapter 13 uses templates for encapsulating complexity of ontology design patterns.

COMMUNICATIONS OF THIS BOOK

We have communicated the research presented in this book through conference papers, a journal paper, conference tutorials, conference demonstrations, and bachelor/master theses. In the following, we list the publications according to the chapters covering the respective contributions.


Applications in MDE:
* Software Languages
* Ontology-Based Inf. Systems
* Software Design Patterns

Applications in Ontology Engineering
* Generation of Ontology APIs
* Ontology Translation Language
* Ontology Templates

TwoUseToolkit
Architecture and Services

The TwoUseApproach
Structure, Querying, Notations

Fundamentals
MDE foundations, ontology foundations, commonalities, and variations

Roadmap of This Book.


We presented parts of this work in the following tutorials:


