Contents

I Motivation 25

1 Aligning organizational goals and technological infrastructure with model-driven software development 25

1.1 The vision: software made in a way everyone can understand 25

1.2 Describing organizations with enterprise models 26

1.3 Enterprise information systems for supporting organizational tasks 29

1.4 Business–IT alignment with methodical support 32

1.5 Domain-specific software engineering approaches 34

1.6 Deriving requirements towards enterprise information systems from enterprise models 35

1.7 Structure of this work 36

2 An overview example: Online web-shop 37

II Approach 45

3 Concepts and terminology 45

3.1 Modeling languages, meta-models and model instances 45

3.2 Model transformations 46

3.3 Validity checks 51

3.4 Business process models and workflow models 52

3.5 Resources and information objects 54

3.6 Perceived type-instance blurring 57

4 Requirements towards an enterprise model driven engineering approach for enterprise information systems 60

5 Enterprise models for model-driven software engineering 67

5.1 Organization theory concepts in enterprise modeling languages 67

5.1.1 Actors 67

5.1.2 Resources 68

5.1.3 Interactions 69

5.1.4 Business processes 69
5.1.5 Strategy 71
5.2 Model-driven software engineering as an act of interpretation 71
5.2.1 Conceptual vagueness in domain-specific modeling languages and models 71
5.2.2 Incorporating semi-formal interpretation transformations into model-driven software engineering with domain-specific models 73
5.3 Related research and existing approaches 73
5.3.1 Model-driven architecture (MDA) 73
5.3.2 Rational Unified Process (RUP) 75
5.3.3 Domain-specific software engineering 76
5.3.4 Enterprise architecture 79
5.3.5 Business process model execution 79
5.3.6 Analyses of business process models 81
5.3.7 Incorporating actor and resource models into software engineering 82
5.3.8 Strategic models for software engineering 83
5.3.9 Process-centered software engineering environments (PCSEEs) . 84
5.3.10 Self-referential enterprise information systems 87
5.4 Deficiencies of existing approaches and contributions by the proposed method 88

III A Domain-Specific Method for Model-Driven Software Engineering with Enterprise Models 93

6 Method constituents 93
6.1 Overview 93
6.1.1 Internal enterprise model representation language 93
6.1.2 Implementation strategies and mapping model 94
6.1.3 Model transformations 94
6.1.4 Validity checks 94
6.1.5 APIs 94
6.1.6 Code generation templates 95
6.1.7 Tooling support 95
6.1.8 Overview on the methodical procedure 95

6.2 Models and modeling languages 98
6.2.1 Enterprise models and their internal representation 99
6.2.2 Mapping model 108
6.2.3 Implementation strategy models and corresponding modeling languages 118
6.3 Model transformations 119
6.3.1 Adapter transformation for enterprise models 120
6.3.2 Mapping model initialization transformation 121
6.3.3 Artifact generation and alternative approaches 123
6.4 Validity checks 124
6.4.1 Validity check for enterprise models 125
6.4.2 Validity check for the mapping model 126
6.5 Domain APIs for EIS 127

7 Applying the method 131

7.1 Applying the method to enterprise information system development . . . 131
7.1.1 Step 1: Create and edit enterprise models 131
7.1.2 Step 2: Transform enterprise models to a internal representation 132
7.1.3 Step 3: Check validity of the enterprise model representation . . 133
7.1.4 Step 4: Initialize or update the mapping model and the implementation strategy models 133
7.1.5 Step 5: Manually edit the mapping model and the implementation strategy models 134
7.1.6 Step 6: Check validity of the mapping model and the implementation strategy models 135
7.1.7 Step 7: Generate deployable artifacts 135
7.2 Configuring the method to be used with a specific enterprise modeling language 136
7.2.1 Step 1: Identify language concepts equivalent in EML and EEM . 137
7.2.2 Step 2: Implement transformation rules for equivalent language concepts 137
7.2.3 Step 3: Formulate hints to express other EEM concepts in EML . 137
7.2.4 Step 4: Implement transformation rules for other language concepts via hints 139
7.3 Configuring the method for specific target architectures 140
7.3.1 Step 1: Conceptualize a target architecture API 141

7.3.2 Step 2: Implement the target architecture API 142
7.3.3 Step 3: Meta-model architecture-specific process-step implementation strategy types 142
7.3.4 Step 4: Meta-model architecture-specific event implementation strategy types 143
7.3.5 Step 5: Meta-model architecture-specific actor implementation strategy types 144
7.3.6 Step 6: Meta-model architecture-specific resource implementation strategy types 145
7.3.7 Step 7: Meta-model architecture-specific information resource implementation strategy types 146
7.3.8 Step 8: Meta-model architecture-specific sequence implementation strategy types 148
7.3.9 Step 9: Conceptualize hints at choosing default implementation strategies 148
7.3.10 Step 10: Implement hints at choosing default implementation strategies 149
7.3.11 Step 11: Implement code generation templates for generic implementation strategies 150
7.3.12 Step 12: Implement code generation templates for architecture-specific implementation strategies 152

8 Design of a prototypical enterprise information system 162

8.1 General architectural design considerations 163
8.1.1 Coordination in a distributed environment 163
8.1.2 Realizing data storages 165
8.1.3 Automatically executed process-steps 166
8.2 User interface sketch 166
8.2.1 Process instance management functionality 167
8.2.2 Process-step editor functionality 168
8.2.3 Information access and document editing functionality 168
8.2.4 Manual task handling functionality 169
8.2.5 Decision functionality 169
8.2.6 Communication functionality 170
8.2.7 Project specific functionality 171
8.3 Abstract domain API 171

8.3.1 Front-end API interfaces 172
8.3.2 Back-end API interfaces 175

9 Example implementation strategies 180
9.1 Implementation strategies for process-members 180
9.1.1 Interactive process-steps 180
9.1.2 Additional high-level process-member implementation strategies . 184
9.1.3 Automatic process-steps 186
9.1.4 Event implementation strategies 189
9.1.5 Sequence implementation strategies 190
9.2 Implementation strategies for actors 196
9.3 Resource implementation strategies 197
9.3.1 Information resource implementation strategies 198
9.3.2 Software resource implementation strategies 202
9.3.3 Physical resource implementation strategies 204
9.3.4 Resource access implementation strategies 205

IV Applying the Method: Prototypical Design and Implementation 213

10 Example scenario of a BPEL-orchestrated SOA target application architecture 213
10.1 Application scenario in the food supply chain domain 213
10.2 Domain-specific language for supply chain modeling 214
10.3 A distributed service oriented architecture (SOA) 216
10.4 Implementation strategy meta-model for a SOA platform 216
10.5 Executable BPEL workflow 220
10.6 Overall implemented example 225

11 MEMO enterprise models for developing JSP web applications 228
11.1 Adapting the MEMO enterprise modeling method 228
11.1.1 The MEMO language family as input enterprise modeling languages 228
11.1.2 MEMOCENTERNG as editor application 228
11.1.3 Adapter transformation to configure the method for the MEMO language family 229

11.2 Configuring a JSP web-application target architecture 230

11.2.1 Example implementation strategy meta-model for a JSP web-application platform 230

12 Code generation and tooling support 233

12.1 Deriving executable artifacts 233

12.2 Code generation templates 234

12.3 Requirements towards tooling support 235

12.4 Enterprise modeling with the MEMOCENTERNG platform 236

12.5 Tooling on top of the ECLIPSE MODELING FRAMEWORK (EMF) 237

12.5.1 Mapping model editor 237

12.5.2 Implementation strategy meta-modeling with ECORE 238

12.5.3 Model-to-model transformations with the XTEND language . . . 238

12.5.4 Model-checking with the CHECK language 240

12.5.5 Code generation with XPAND templates 240

12.5.6 GUI components to invoke the transformation steps in the method 242

12.5.7 EEM model editor for test purposes 243

V Reflection 245

13 Evaluation 245

14 Remaining Work 255

A Example software artifacts 257

A.1 Source code packages of the provided examples 257

A.1.1 Package `de.gulden.modeling.seem.eem` 257

A.1.2 Package `de.gulden.modeling.seem.generator` 258

A.1.3 Package `de.gulden.modeling.seem.generator.memo` . . . 258

A.1.4 Package `de.gulden.modeling.seem.generator.ui` 258

A.1.5 Package `de.gulden.modeling.seem.mapping` 258

A.1.6 Package `de.gulden.modeling.seem.workflow` 258

A.1.7 Package `de.gulden.modeling.seem.architecture.web` . . 259

A.1.8 Package `de.gulden.server.xmldb` 259

A.1.9 Package `org.rescueit.modeling.targetarchitecture` . . 259

A.1.10 Package `org.rescueit.modeling.workflow` 259
A.1.11 Package `de.gulden.modeling.seem.api.web` 260
A.1.12 Package `de.gulden.modeling.seem.example.webshop` . . 260
A.1.13 Package `webshop` 260
A.2 Example artifacts overview 260
A.3 Introductory example artifacts 264
A.3.1 Adaptation to the MEMO enterprise modeling languages 265
A.3.2 Mapping model handling 270
A.3.3 Code generation 278
A.3.4 Generated example artifact 290
A.3.5 Modeling conventions to incorporate additional semantics into the enterprise models 292
A.4 Artifacts of the comprehensive example 294
A.4.1 Adaptation to a domain-specific supply chain modeling language 294
A.4.2 Mapping model handling 296
A.4.3 Code generation 300
A.4.4 Input model and generated example artifacts 305