

Methods to Create, Retrieve and Apply Cross-Domain Problem Solutions

Von der Fakultät für Ingenieurwissenschaften,
Abteilung Informatik und Angewandte Kognitionswissenschaft
der Universität Duisburg-Essen
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften

genehmigte Dissertation

vorgelegt von

Alexander Fülleborn

aus

Enschede (Niederlande)

1. Gutachter: Prof. Dr. Maritta Heisel
2. Gutachter: Prof. Dr. Eduardo B. Fernandez

Tag der mündlichen Prüfung: 15.06.2016

Berichte aus der Informatik

Alexander Fülleborn

**Methods to Create, Retrieve and
Apply Cross-Domain Problem Solutions**

A Problem-Oriented Pattern Management Approach

Shaker Verlag
Aachen 2016

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <http://dnb.d-nb.de>.

Zugl.: Duisburg-Essen, Univ., Diss., 2016

Copyright Shaker Verlag 2016

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the publishers.

Printed in Germany.

ISBN 978-3-8440-4609-0

ISSN 0945-0807

Shaker Verlag GmbH • P.O. BOX 101818 • D-52018 Aachen

Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9

Internet: www.shaker.de • e-mail: info@shaker.de

To Sandy and Linus

Contents

1	Introduction	1
1.1	Thesis Goals and Contributions	2
1.2	Structure	7
I	Basics	9
2	Basic Concepts, Terminology and Notations	11
2.1	Patterns	11
2.2	Analysis Patterns	12
2.3	Design Patterns	14
II	Overview and Concepts of Problem-oriented Pattern Management (ProPMan)	17
3	ProPMan Terminology	19
3.1	Overview about artifacts used in <i>ProPMan</i>	20
3.2	Problem-Context and Solution Patterns	21
3.2.1	Problem-Context Patterns	23
3.2.2	Solution Patterns	24
3.2.3	Problem-context and solution patterns as common representation for abstracted analysis and design patterns	25
3.3	Problem-bearing Models - to be improved	26
3.4	Problem-Context Model	27
3.5	Solution Model	28
3.6	Pattern Provider	30
3.7	Pattern Consumer	30

4	Overview of ProPMan Processes	31
4.1	<i>ProPMan</i> Process for Improving UML Analysis or Design Models . . .	31
4.1.1	Generate problem-context pattern	33
4.1.2	Search for appropriate solution using pattern retrieval	33
4.1.3	Develop appropriate own solution	33
4.1.4	Generate solution pattern	34
4.1.5	Apply pattern	34
4.2	<i>ProPMan</i> Process for Extending Incomplete Patterns	35
4.2.1	ProPExt Step 1: <i>Create ProPMan solution pattern for pattern from the literature</i>	36
4.2.2	ProPExt Step 2: <i>Create problem-context model and refactor it to solution model</i>	36
4.2.3	ProPExt Step 3: <i>Transform domain-specific problem-context and solution models to cross-domain patterns</i>	36
4.2.4	ProPExt Step 4: <i>Validate generated problem-context and solution patterns</i>	37
5	Overview of ProPMan Tool	39
5.1	Technical Basics	39
5.2	UML Profile for Problem-oriented Pattern Management	40
5.2.1	UML Model Extension «ProblemContextModel»	41
5.2.2	UML Model Extension «SolutionModel»	41
5.2.3	UML Model Extension «ProblemContextPattern»	42
5.2.4	UML Model Extension «SolutionPattern»	42
5.2.5	UML Class Extension «ProblemStatement»	42
5.2.6	UML Dependency Extension «ProblemBearing»	44
5.3	Tool Realization	45
5.4	Supported UML Model Elements	46
III	Supporting Pattern Providers	49
6	Method for Problem-oriented Pattern Generation (<i>ProPGen</i>)	51
6.1	ProPGen process <i>Generate problem-context pattern</i>	53

6.1.1	ProPGen process <i>Generate problem-context pattern</i> : Overview	53
6.1.2	Step 1: Analyze problem in current domain-specific analysis or design model	54
6.1.3	Step 2: Add problem information to current domain-specific analysis or design model	55
6.1.4	Step 3: Transform domain-specific problem-context model to cross-domain problem-context pattern	59
6.1.4.1	<i>Determine relevant UML elements</i>	59
6.1.4.2	<i>Rename relevant UML elements</i>	62
6.1.5	ProPGen process <i>Generate problem-context pattern</i> : Summary	64
6.2	ProPGen process <i>Generate solution pattern</i>	66
6.2.1	ProPGen process <i>Generate solution pattern</i> : Overview	66
6.2.2	Step 1: Add information about refactoring source and target dependency between problem-context and solution model . . .	68
6.2.3	Step 2: Transform domain-specific solution model to cross- domain solution pattern	70
6.2.3.1	<i>Determine relevant UML elements</i>	71
6.2.3.2	<i>Rename relevant UML elements</i>	74
6.2.3.3	Derive problem-context pattern to solution pattern transformation (Pcp2Sp) trace model	77
6.2.4	ProPGen process <i>Generate solution pattern</i> : Summary	78
6.3	ProPGen: Contribution and Future Work	79
6.3.1	Contribution	79
6.3.2	Future Work	80
7	Method for Problem-oriented Pattern Extension (<i>ProPExt</i>)	81
7.1	ProPExt process <i>Extend pattern from the literature</i>	83
7.1.1	ProPExt process <i>Extend pattern from the literature</i> : Overview	83
7.1.2	Step 1: Create ProPMan solution pattern for pattern from the literature	87
7.1.3	Step 2.1: Choose suitable example from expert domain	87
7.1.4	Step 2.2: Create/modify problem-bearing model	87
7.1.5	Step 2.3/3.1: Generate problem-context pattern	88
7.1.6	Step 2.4: Refactor problem-context to solution model	88

7.1.7	Step 2.5/3.2: Generate solution pattern	88
7.1.8	Step 4.1: Decide whether modeled and generated solution patterns are congruent	89
7.1.9	Step 4.2: Decide if not congruent patterns should be stored as a variant	89
7.1.10	Step 4.3: Adjust modeled solution pattern from the literature	90
7.1.11	Step 4.4: Adjust element names in generated problem-context pattern to solution pattern from the literature	90
7.1.12	Step 4.5: Store generated problem-context and modeled solution pattern as a pair to the ProPMan pattern library	90
7.2	Decide whether modeled and generated solution patterns are congruent: Refining sub-process	91
7.2.1	Step 1: Determine difference between number of classes in modeled and generated solution pattern	94
7.2.2	Step 2: Check: Additional classes have the role of a CLIENT that only give contextual background information and are not essential for the solution?	94
7.2.3	Step 3: Exclude classes that have the role of a CLIENT that only give contextual background information and are not essential for the solution from further comparisons	95
7.2.4	Step 4: Determine difference between number of generalizations in modeled and generated pattern	95
7.2.5	Step 5: Iterate next class in generated solution pattern	96
7.2.6	Step 6: Determine generalization roles of generated solution pattern class	96
7.2.7	Step 7: Read next class in modeled solution pattern with generalization roles equal to generated solution pattern	96
7.2.8	Step 8: Read next class in modeled solution pattern that does not participate in any generalization relationship	97
7.2.9	Step 9: Determine attributes and methods of current class in modeled solution pattern that only give contextual/sample background information and are not essential for the solution	97

7.2.10	Step 10: Exclude attributes and methods in modeled solution pattern that only give contextual/sample background information and are not essential for the solution	98
7.2.11	Step 11: Determine difference between number of attributes and methods of generated solution pattern being instantiations of analogous elements in modeled solution pattern	98
7.2.12	Step 12: Check if further class in generated solution pattern exists	99
7.2.13	Step 13: Check if further class in modeled solution pattern exists	99
7.3	ProPExt: Contribution	99
8	Method Validation ProPGen and ProPExt	101
8.1	Create patterns: <i>Analysis pattern</i> from business object <i>Invoice</i>	102
8.1.1	Introduction to example for requirements on business object <i>Invoice</i>	102
8.1.2	ProPGen process <i>Generate problem-context pattern</i> Step 1: Analyze problem in current domain-specific analysis or design model	103
8.1.3	ProPGen process <i>Generate problem-context pattern</i> Step 2: Add problem information to current domain-specific analysis or design model	104
8.1.4	ProPGen process <i>Generate problem-context pattern</i> Step 3: Transform domain-specific problem-context model to cross-domain problem-context pattern	104
8.1.5	Develop appropriate own solution: Refactor problem-context model of Invoice business object	107
8.1.6	ProPGen process <i>Generate solution pattern</i> Step 1: Add information about refactoring source and target dependency between problem-context and solution model	107
8.1.7	ProPGen process <i>Generate solution pattern</i> : Step 2: Transform domain-specific solution model to cross-domain solution pattern	109
8.2	Extend patterns: Structural design pattern <i>Adapter</i>	111

8.2.1	ProPEXt Step 1: Create ProPMan solution pattern from the literature	111
8.2.2	ProPEXt Step 2.1: Choose suitable example from expert domain	111
8.2.3	ProPEXt Step 2.2: Create/modify problem-bearing model . . .	112
8.2.4	ProPEXt Step 2.3/3.1: Generate problem-context pattern . . .	112
8.2.4.1	ProPGen process <i>Generate problem-context pattern</i> Step 1: Analyze problem in current domain-specific analysis or design model	112
8.2.4.2	ProPGen process <i>Generate problem-context pattern</i> Step 2: Add problem information to current domain- specific analysis or design model	113
8.2.4.3	ProPGen process <i>Generate problem-context pattern</i> Step 3: Transform domain-specific problem-context model to cross-domain problem-context pattern . . .	114
8.2.5	ProPEXt Step 2.4: Refactor problem-context to solution model	116
8.2.6	ProPEXt Step 2.5/3.2: Generate solution pattern	117
8.2.6.1	ProPGen process <i>Generate solution pattern</i> : Step 1: Add information about refactoring source and target dependency between problem-context and solution model	117
8.2.6.2	ProPGen process <i>Generate solution pattern</i> : Step 2: Transform domain-specific solution model to cross- domain solution pattern	118
8.2.7	ProPEXt Step 4.1: Decide whether modeled and generated solution patterns are congruent	121
8.2.8	ProPEXt Step 4.4: Adjust element names in generated problem- context pattern to solution pattern from the literature	125
8.2.9	ProPEXt Step 4.5: Store generated problem-context pattern and modeled solution pattern as a pair to the ProPMan pat- tern library	126
8.2.10	Validation of ProPEXt/ProPGen processes for <i>Adapter</i> : Sum- mary	127
8.3	Extend patterns: Creational design pattern <i>Singleton</i>	128

8.3.1	ProPEXt Step 1: Create ProPMan solution pattern for pattern from the literature	128
8.3.2	ProPEXt Step 2.1: Choose suitable example from expert domain	128
8.3.3	ProPEXt Step 2.2: Create/modify problem-bearing model . . .	129
8.3.4	ProPEXt Step 2.3/3.1: Generate problem-context pattern . . .	129
8.3.4.1	ProPGen process <i>Generate problem-context pattern</i> Step 1: Analyze problem in current domain-specific analysis or design model	129
8.3.4.2	ProPGen process <i>Generate problem-context pattern</i> Step 2: Add problem information to current domain- specific analysis or design model	130
8.3.4.3	ProPGen process <i>Generate problem-context pattern</i> Step 3: Transform domain-specific problem-context model to cross-domain problem-context pattern . . .	131
8.3.5	ProPEXt Step 2.4: Refactor problem-context to solution model	134
8.3.6	ProPEXt Step 2.5/3.2: Generate solution pattern	134
8.3.6.1	ProPGen process <i>Generate solution pattern</i> : Step 1: Add information about refactoring source and target dependency between problem-context and solution model	135
8.3.6.2	ProPGen process <i>Generate solution pattern</i> : Step 2: Transform domain-specific solution model to cross- domain solution pattern	135
8.3.7	ProPEXt Step 4.1: Decide whether modeled and generated solution patterns are congruent	138
8.3.8	ProPEXt Step 4.2: Decide if not congruent patterns should be stored as a variant	140
8.3.9	ProPEXt Step 2.2 (Second iteration): Create/modify problem- bearing model	141
8.3.10	ProPEXt Step 2.3/3.1 (Second iteration): Generate problem- context pattern	141
8.3.11	ProPEXt Step 2.4 (Second iteration): Refactor problem-context to solution model	143

8.3.12	ProPExt Step 2.5/3.2 (Second iteration): Generate solution pattern	144
8.3.13	ProPExt Step 4.1 (Second iteration): Decide whether modeled and generated solution patterns are congruent	145
8.3.14	ProPExt Step 4.4: Adjust element names in generated problem-context pattern to solution pattern from the literature	148
8.3.15	ProPExt Step 4.5: Store generated problem-context pattern and modeled solution pattern as a pair to the ProPMan pattern library	148
8.3.16	Validation of ProPExt/ProPGen processes for <i>Singleton</i> : Summary	149
8.4	Extend patterns: Behavioural design pattern <i>Observer</i>	150
8.4.1	ProPExt Step 1: Create ProPMan solution pattern for pattern from the literature	150
8.4.2	ProPExt Step 2.1: Choose suitable example from expert domain	151
8.4.3	ProPExt Step 2.2: Create/modify problem-bearing model . . .	151
8.4.4	ProPExt Step 2.3/3.1: Generate problem-context pattern . . .	152
8.4.4.1	ProPGen process <i>Generate problem-context pattern</i> Step 1: Analyze problem in current domain-specific analysis or design model	152
8.4.4.2	ProPGen process <i>Generate problem-context pattern</i> Step 2: Add problem information to current domain-specific analysis or design model	153
8.4.4.3	ProPGen process <i>Generate problem-context pattern</i> Step 3: Transform domain-specific problem-context model to cross-domain problem-context pattern . . .	154
8.4.5	ProPExt Step 2.4: Refactor problem-context to solution model	154
8.4.6	ProPExt Step 2.5/3.2: Generate solution pattern	156
8.4.6.1	ProPGen process <i>Generate solution pattern</i> : Step 1: Add information about refactoring source and target dependency between problem-context and solution model	156

8.4.6.2	ProPGen process <i>Generate solution pattern</i> : Step 2: Transform domain-specific solution model to cross- domain solution pattern	156
8.4.7	ProPEXt Step 4.1: Decide whether modeled and generated solution patterns are congruent	157
8.4.8	ProPEXt Step 4.4: Adjust element names in generated problem- context pattern to solution pattern from the literature	162
8.4.9	ProPEXt Step 4.5: Store generated problem-context and mod- eled solution pattern as a pair to the ProPMan pattern library	163
8.4.10	Validation of ProPEXt/ProPGen processes for <i>Observer</i> : Sum- mary	164
8.5	Validation of ProPGen and ProPEXt: Summary	164

IV Supporting Pattern Consumers 167

9 Method for Cross-Domain Problem-oriented Pattern Retrieval (*Pro- PRet*) 169

9.1	Process <i>Search for appropriate solution using pattern retrieval</i>	170
9.1.1	Select retrieval request problem-context pattern to be used as search criterion	173
9.1.2	Start ProPPret retrieval function by selecting appropriate con- text menu entry	174
9.1.3	Retrieve patterns	174
9.1.4	Present retrieval results	175
9.2	Concept of model-to-model comparison via graph matching	177
9.2.1	Problem-context pattern graph (PCP graph)	177
9.2.2	Problem-context pattern similarity metric	179
9.3	Retrieve patterns: Refining sub-process	184
9.3.1	Step 1: Provide all right problem-context patterns from pat- tern library	187
9.3.2	Step 2: Read next <i>right</i> problem-context pattern	187
9.3.3	Step 3: Create tuples of nodes for PCP graphs	188
9.3.4	Step 4: Generate delta «ProblemBearing» nodes to relevant PCP graph	193

9.3.5	Step 5: Permute and calculate similarity of PCP graph node mappings	193
9.3.6	Step 6: Calculate node mapping with highest graph edit similarity score	195
9.3.7	Step 7: Add highest graph edit similarity score to current retrieval result	195
9.3.8	Step 8: Provide retrieval result data	196
9.3.9	Step 9: Add current retrieval result to result list	197
9.3.10	Step 10: Sort retrieval result list and calculate ranks	198
9.4	Permute and calculate similarity of PCP graph node mappings: Refining sub-process	199
9.4.1	Permutation algorithm according to C.T. Fike	200
9.4.2	Example for Permutation of right PBS nodes	201
9.4.3	Permutation of right PBS nodes for Salary Statement Application example: Overview	209
9.4.4	Step 1: Exchange current position of right PBS node in set and position from given input and check if current position in right PBS node set is lower than number of right PBS nodes	209
9.4.5	Step 2: Exchange right PBS node value at current position and value at given position	211
9.4.6	Step 3: Do further positions exist / end of set is not reached?	212
9.4.7	Step 4: Permute and calculate similarity of PCP graph node mappings	213
9.4.8	Step 5: Create mapping and similarity metrics for current permutation	213
9.4.9	Step 6: Reverse previous exchange of right PBS node value at current position and value at given position	214
9.4.10	Step 7: Increment counter for current position in set by 1	215
9.4.11	Step 8: Check current position in set $<$ number of nodes in set	215
9.5	Create mapping and similarity metrics for current permutation: Refining sub-process	216
9.5.1	Details on data objects used in the process	217
9.5.2	Details on process steps	220

9.6	Discussion concerning restriction to one « <i>ProblemStatement</i> » class in a problem-context pattern	230
9.7	Contribution and Future Work	236
10	Method for Problem-oriented Pattern Instantiation (<i>ProPIns</i>)	239
10.1	Process <i>Apply pattern</i>	241
10.1.1	Select retrieval result from ProPRet result list of retrieved patterns (Step 1)	243
10.1.2	Start ProPIns instantiation function by pushing button <i>Instantiate Solution pattern from selected line</i> (Step 2)	247
10.1.3	Instantiate retrieval result solution pattern to abstract retrieval request solution model (Step 3)	248
10.1.4	Instantiate abstract retrieval request solution model to domain-specific solution model (Step 4)	252
10.1.5	Present domain-specific solution model in project with instantiated solution patterns (Step 5)	255
10.2	Instantiate retrieval result solution pattern to abstract retrieval request solution model: Refining sub-process	257
10.2.1	Create empty target abstract solution model	259
10.2.2	Determine type of retrieval result problem-context pattern model element	261
10.2.3	Instantiate « ProblemBearing » dependency supplier element	263
10.2.4	Instantiate class as type of « ProblemBearing » dependency supplier element	264
10.2.5	Instantiate added new model element from source retrieval result pattern as new model element to target abstract retrieval request solution model	264
10.2.6	Instantiate container class of « ProblemBearing » supplier attribute or method element	267
10.2.7	Check if further model element exists	267
10.2.8	Calculate difference between number of existing and solved problems in retrieval request problem-context pattern	268
10.2.9	Copy « ProblemStatement » class with EXISTING prefix from retrieval request problem-context pattern to abstract solution model	268

10.2.10 Copy «ProblemStatement» class with DELETED prefix from retrieval request problem-context pattern to abstract solution model	269
10.2.11 Determine and copy model elements related to unsolved prob- lems with EXISTING prefix from left retrieval request problem- context pattern to abstract solution model	270
10.3 Instantiate abstract retrieval request solution model to domain-specific solution model: Refining sub-process	273
10.3.1 Create empty left retrieval request domain-specific solution model	274
10.3.2 Determine left retrieval request problem-context model and Pcm2PcpTrafoTrace model	274
10.3.3 Instantiate existing and deleted elements from left retrieval request abstract to domain-specific solution model	275
10.3.4 Instantiate new elements from left retrieval request abstract to domain-specific solution model	277
10.4 Contribution	279

V Tool Support 281

11 ProPMan Tool Support 283

11.1 Developer Documentation	283
11.1.1 Requirements	283
11.1.2 Architecture	286
11.1.2.1 Eclipse Platform: Architecture Overview	286
11.1.2.2 ProPMan Tool: Architecture Overview	288
11.1.2.3 <i>ProPMan Core Functions</i> Plug-in	289
11.1.2.4 <i>ProPMan Supplemental Functions</i> Plug-in	304
11.1.2.5 <i>SimMetrics Functions</i> Plug-in	309
11.1.3 Implementation	310
11.1.3.1 Development Platform	310
11.1.3.2 Epsilon Framework	311
11.1.3.3 Naming Conventions	331
11.1.3.4 Plug-In Registration	332

11.1.3.5	<i>ProPMan Core Functions</i> Plug-in	334
11.1.3.6	<i>ProPMan Supplemental Functions</i> Plug-in	427
11.2	End User Documentation	441
11.2.1	ProPMan Graphical User Interface Elements	441
11.2.1.1	ProPMan Pre-defined Projects in the Eclipse Project Explorer	441
11.2.1.2	ProPMan Context Menus for *.uml Files	444
11.2.1.3	ProPMan View <i>ProPRet result list of retrieved patterns</i>	446
11.2.2	How to Run ProPMan Processes	448
11.2.2.1	Use and extend ProPMan customizing	448
11.2.2.2	Use of ProPMan functions	448
VI	Related Work and Conclusions	453
12	Related Work and Conclusions	455
12.1	Related Work on Tool Support for Patterns	455
12.2	Conclusions	458
VII	Appendix	461
A	Details on Running Examples	463
A.1	Example Salary Statement Application	463
A.2	Example SAP System Messages Viewer Application	466
A.2.1	Problem-context model	467
A.2.2	Conditions and fitting problem-context model elements for Pcm2Pcp transformation	468
A.2.3	Source and renamed target model elements	469
A.2.4	Problem-context pattern	469
B	ProPMan Pattern Library	471
B.1	ProPMan newly created analysis pattern from business object <i>Invoice</i>	472
B.2	ProPMan extended structural design pattern <i>Adapter</i>	473
B.3	ProPMan extended creational design pattern <i>Singleton</i>	474
B.4	ProPMan extended behavioural design pattern <i>Observer</i>	475

C ProPIns refining sub-processes - continued 477

C.1	Instantiate «ProblemBearing» dependency supplier element: Refining sub-process	477
C.1.1	Check if appropriate analogous «ProblemBearing» dependency supplier element exists in source retrieval request problem-context pattern	479
C.1.2	Determine appropriate analogous «ProblemBearing» dependency supplier element in source retrieval request problem-context pattern	481
C.1.3	Check if «ProblemBearing» dependency supplier element is copied to solution pattern by retrieval result Pcp2Sp transformation	482
C.1.4	Copy appropriate analogous «ProblemBearing» dependency supplier element with the EXISTING prefix to target abstract retrieval request solution model	483
C.1.5	Copy appropriate analogous «ProblemBearing» dependency supplier element AND «ProblemBearing» dependency itself with the DELETED prefix to target abstract retrieval request solution model	484
C.1.6	Increment counter for number of solved problems	484
C.2	Instantiate class as type of «ProblemBearing» dependency supplier element: Refining sub-process	485
C.2.1	Check if appropriate analogous class as type of «ProblemBearing» dependency supplier element exists in source retrieval request problem-context pattern	487
C.2.2	Determine appropriate analogous class as type of «ProblemBearing» dependency supplier element in source retrieval request problem-context pattern	488
C.2.3	Check if class as type of «ProblemBearing» dependency supplier element is copied to solution pattern by retrieval result Pcp2Sp transformation	488

C.2.4	Create appropriate analogous class as type of «ProblemBearing» dependency supplier element with the EX- ISTING prefix in target abstract retrieval request solution model	489
C.2.5	Copy appropriate analogous class as type of «ProblemBearing» dependency supplier element with the DELETED prefix to target abstract retrieval request solution model	490
C.3	Instantiate added new model element from source retrieval result pat- tern as new model element to target abstract retrieval request solution model: Refining sub-process	491
C.3.1	Check type of new model element added to right retrieval result solution pattern	493
C.3.2	Instantiate new attribute or method from source retrieval re- sult pattern as new model element to target abstract retrieval request solution model	493
C.3.3	Copy added new class from retrieval result solution pattern with the NEW prefix to retrieval request abstract solution model	494
C.3.4	Add transformation trace dependency to Sp2SmTrafo Trace model with new retrieval result class as client and re- trieval request class as supplier	495
C.3.5	Instantiate new generalization from source retrieval result pat- tern as new model element to target abstract retrieval request solution model	495
C.4	Instantiate new attribute or method from source retrieval result pattern as new model element to target abstract retrieval request solution model: Refining sub-process	496
C.4.1	Determine container class name of new attribute or method in retrieval result solution pattern	499
C.4.2	Check in Pcp2Sp Trafo Trace Model if retrieval result con- tainer class exists in retrieval result problem-context pattern	500
C.4.3	Check if container class of retrieval result solution pattern is also a «ProblemBearing» supplier attribute or method con- tainer class or a type of a «ProblemBearing» supplier at- tribute in retrieval result problem-context pattern	501

C.4.4	Determine appropriate analogous class related to «ProblemBearing» supplier attribute or method in retrieval request problem-context pattern	503
C.4.5	Check if retrieval request problem-context pattern container class exists with EXISTING prefix in retrieval request abstract solution model	504
C.4.6	Copy container class from retrieval request problem-context pattern with EXISTING prefix to retrieval request abstract solution model	504
C.4.7	Check if retrieval result solution pattern container class with NEW prefix exists in retrieval request abstract solution model	505
C.4.8	Copy container class from retrieval result solution pattern with NEW prefix to retrieval request abstract solution model	506
C.4.9	Copy added new attribute or method from retrieval result so- lution pattern with the NEW prefix to retrieval request ab- stract solution model	507
C.4.10	Add transformation trace dependency to Sp2SmTrafo Trace model with retrieval result attribute or method class as client and retrieval request attribute or method as supplier . .	507
C.4.11	Assign copied container class to copied attribute or method in retrieval request abstract solution model	509
C.4.12	Add transformation trace dependency to Sp2SmTrafoTrace model with retrieval result container class as client and re- trieval request container class as supplier	509
C.5	Instantiate new generalization from source retrieval result pattern as new model element to target abstract retrieval request solution model: Refining sub-process	512
C.5.1	Copy added new generalization from retrieval result solution pattern with the NEW prefix to retrieval request abstract so- lution model	515
C.5.2	Set role of class in generalization to 'general'	515
C.5.3	Determine name for class with specified role in new general- ization in retrieval result solution pattern	515

C.5.4	Check in Pcp2SpTrafoTrace model if retrieval result class with specified role exists in retrieval result problem-context pattern	516
C.5.5	Check if retrieval result solution pattern class with specified role is also «ProblemBearing» supplier class OR type of a supplier attribute OR container of supplier attribute/method in retrieval result problem-context pattern	518
C.5.6	Determine appropriate analogous class with specified role in retrieval request problem-context pattern	519
C.5.7	Check if class with prefix for EXISTING elements exists in retrieval request abstract solution model	520
C.5.8	Copy class with specified role from retrieval request problem-context pattern with EXISTING prefix to retrieval request abstract solution model	521
C.5.9	Create new Sp2SmTrafoTrace model dependency with retrieval result class as client and retrieval request class as supplier	522
C.5.10	Check if class with prefix NEW exists in retrieval request abstract solution model	522
C.5.11	Copy class with specified role from retrieval result solution pattern with NEW prefix to retrieval request abstract solution model	523
C.5.12	Create new Sp2SmTrafoTrace model dependency with retrieval result class as client and retrieval request class as supplier	523
C.5.13	Add name of class with specified role in retrieval request abstract solution model to value for role in new generalization	524
C.5.14	Check value of specified role	524
C.5.15	Set role of class in generalization to 'specific'	525
C.5.16	Create new Sp2SmTrafoTrace model dependency with retrieval result generalization as client and retrieval request generalization as supplier	525
C.6	Instantiate container class of «ProblemBearing» supplier attribute or method element: Refining sub-process	527
C.6.1	Read next «ProblemBearing» dependency supplier attribute or method of container class from the retrieval result problem-context pattern	529

C.6.2	Determine appropriate analogous «ProblemBearing» dependency supplier attribute or method from the retrieval request problem-context pattern	529
C.6.3	Check if container class of «ProblemBearing» dependency supplier attribute or method from retrieval request problem-context pattern exists already with EXISTING prefix in retrieval request abstract solution model	530
C.6.4	Copy container of «ProblemBearing» dependency supplier attribute or method with EXISTING prefix from retrieval request problem-context pattern to abstract solution model . . .	530
C.6.5	Add transformation trace dependency to Sp2SmTrafoTrace model with retrieval result container class as client and retrieval request container class as supplier	530
C.6.6	Check if further «ProblemBearing» dependency supplier attribute or method exists in container class	531
C.7	Instantiate existing and deleted elements from left retrieval request abstract to domain-specific solution model: Refining sub-process . . .	531
C.7.1	Read next domain-specific model element from left retrieval request problem-context model	534
C.7.2	Determine left retrieval request cross-domain model element from Pcm2PcpTrafoTrace model	535
C.7.3	Determine prefix value for left retrieval request cross-domain model element from abstract solution model	538
C.7.4	Copy left retrieval request domain-specific problem-context model element to domain-specific solution model	540
C.7.5	Do not copy left retrieval request domain-specific problem-context model element to domain-specific solution model . . .	542
C.7.6	Copy remaining existing element from left retrieval request domain-specific problem-context model to solution model . . .	543
C.7.7	Check if further element exists in left domain-specific problem-context model	544
C.8	Instantiate new elements from left retrieval request abstract to domain-specific solution model: Refining sub-process	545

C.8.1	Read next element with the <i>@New_element@</i> prefix from the left retrieval request abstract solution model	547
C.8.2	Check type of model element with <i>@New_element@ prefix</i> . . .	550
C.8.3	Copy new attribute or method in left abstract retrieval request solution model to domain-specific solution model	550
C.8.4	Determine prefix for container of abstract model element . . .	551
C.8.5	Assign abstract container to attribute or method	552
C.8.6	Determine domain-specific container class	552
C.8.7	Assign domain-specific container class to attribute or method . . .	554
C.8.8	Copy new class in left abstract retrieval request solution model to domain-specific solution model	555
C.8.9	Instantiate new generalization in left abstract retrieval request solution model to domain-specific solution model	555
C.9	Instantiate new generalization in left abstract retrieval request solution model to domain-specific solution model: Refining sub-process . . .	557
C.9.1	Copy new generalization in left abstract retrieval request solution model to domain-specific solution model	558
C.9.2	Set role of class in generalization to 'general'	559
C.9.3	Determine prefix for class with specified role in new generalization	559
C.9.4	Determine domain-specific class	560
C.9.5	Assign determined domain-specific class with specified role to domain-specific generalization	561
C.9.6	Assign class with specified role from abstract to domain-specific generalization	562
C.9.7	Check value of specified role	563
C.9.8	Set role of class in generalization to 'specific'	563
C.9.9	Check if further element with the <i>@New_element@</i> prefix exists in left retrieval request abstract solution model	563
D	Developer Documentation - continued	565
D.1	ProPMan Core Functions Plug-in: Classes	565
D.2	Epsilon EOL module <i>ProPGenGlobalOperations.eol</i> : Selected Listings	566
D.3	Epsilon ETL module <i>Pcm2PcpTrafo.etl</i> : Selected Additional Listings	567
D.4	Epsilon ETL module <i>Sm2SpTrafo.etl</i> : Selected Additional Listings . .	568

D.5	Epsilon EOL module <i>GetValueForStereotypePropPcmOfSmSrc.eol</i> . .	576
E	End User Documentation - continued	577
E.1	Installation Guide	577
E.1.1	Java Installation	577
E.1.2	ProPMan Software Package Installation	579
E.2	Step-to-step Sample Scenario	584
E.2.1	Pattern Provider Activities: Create domain-specific problem-context model	584
E.2.1.1	[Step 1] Copy project with domain-specific problem-bearing model	586
E.2.1.2	[Step 2] Apply ProPMan UML Profile to copy of domain-specific problem-bearing model	589
E.2.1.3	[Step 3] Add problem statement to domain-specific problem-bearing model	593
E.2.1.4	[Step 4] Add «ProblemStatement» stereotype property values for <i>problem</i> and <i>context</i> to domain-specific problem-bearing model	595
E.2.1.5	[Step 5] Add dependencies with stereotype «ProblemBearing» to domain-specific problem-bearing model	600
E.2.2	Pattern Provider Activities: Create cross-domain problem-context pattern	603
E.2.3	Pattern Provider Activities: Create domain-specific solution model	607
E.2.4	Pattern Provider Activities: Create cross-domain solution pattern	614
E.2.5	Pattern Consumer Activities: Create domain-specific problem-context model	616
E.2.5.1	[Step 1] Copy project with domain-specific problem-bearing model	617
E.2.5.2	[Step 2] Apply ProPMan UML Profile to copy of domain-specific problem-bearing model	621
E.2.5.3	[Step 3] Add problem statement to domain-specific problem-bearing model	625

E.2.5.4	[Step 4] Add ProblemStatement stereotype property values for <i>problem</i> and <i>context</i> to domain-specific problem-bearing model	628
E.2.5.5	[Step 5] Add dependencies with stereotype « ProblemBearing » to domain-specific problem-bearing model	632
E.2.6	Pattern Consumer Activities: Create cross-domain problem-context pattern	634
E.2.7	Pattern Consumer Activities: Perform pattern retrieval	636
E.2.8	Pattern Consumer Activities: Perform pattern instantiation	637

Bibliography	641
---------------------	------------

List of Figures

1.1	Overview of thesis goal: Software engineering process supported by managing analysis and design patterns	2
1.2	Overview of thesis goal and contributions: Supported software engineering process and building blocks of our <i>Methodology for Problem-oriented Pattern Management</i>	4
1.3	Overview <i>ProPMan</i> Methodology: Approach, supported processes, software engineer roles	5
3.1	Overview: UML meta model of UML artifacts used in this thesis	20
3.2	Example: Problem-context pattern	23
3.3	Example: Solution pattern	25
3.4	Example: Problem-bearing model - to be improved	26
3.5	Example: Problem-context model	27
3.6	Example: Solution model	29
4.1	BPMN process model: Refined support process <i>Improve domain-specific UML analysis or design model</i> and assigned <i>ProPMan</i> methods	32
4.2	BPMN process model: <i>ProPMan</i> process for extending incomplete patterns - <i>ProPExt</i>	35
5.1	Overview: <i>ProPMan</i> UML meta model extensions	41
5.2	Overview - Tool Realization	45
6.1	Overview of the method for generating problem-oriented patterns, <i>ProPGen</i>	52
6.2	BPMN process model: <i>ProPGen</i> process <i>Generate problem-context pattern</i>	53
6.3	Example for problem-bearing model - to be improved	55

6.4	Contents of <i>problem</i> properties file	57
6.5	Contents of <i>context</i> properties file	57
6.6	Example for problem-context model	58
6.7	Example: UML elements of a problem-context model that meet abstraction conditions in order to be considered in the problem-context pattern	60
6.8	Example: First abstraction step of the problem-context model to problem-context pattern transformation	61
6.9	Salary Statement Application example: First abstraction level problem-context pattern before renaming after first problem-context model to problem-context pattern transformation abstraction step	62
6.10	ProPGen renaming concept	63
6.11	Example for problem-context pattern generation: Second abstraction step to finalize transformation by renaming of UML elements	63
6.12	Salary Statement Application example: Second final abstraction level problem-context pattern after renaming	64
6.13	BPMN process model: <i>ProPGen</i> process <i>Generate solution pattern</i>	66
6.14	Example: Self-developed solution model to Salary Statement Application problem-context model	69
6.15	Example Salary Statement Application: Name of refactored problem-context model as value of the solution model stereotype property <i>pcm</i>	70
6.16	Example: UML elements of a solution model that each meet at least one abstraction condition in order to be considered in the solution pattern	72
6.17	Example: First abstraction step of the solution model to solution pattern transformation	73
6.18	Salary Statement Application example: First abstraction level solution pattern before renaming after first solution model to solution pattern transformation abstraction step	74
6.19	Example for solution pattern generation: Second abstraction step to finalize transformation by renaming of UML elements	75
6.20	Example <i>Salary Statement Application</i> : Solution pattern	76
6.21	Problem-context pattern to solution pattern transformation trace model as post processing result of solution pattern generation	77

7.1	Overview of the method for extending patterns from the literature, <i>ProPExt</i>	82
7.2	BPMN process model: <i>ProPMan process for extending incomplete patterns - ProPExt</i>	84
7.3	BPMN process model: Steps 1 to 2.4 of <i>ProPExt</i> process <i>Extend pattern from the literature</i>	85
7.4	BPMN process model: Steps 2.5 to 4.5 of <i>ProPExt</i> process <i>Extend pattern from the literature</i>	86
7.5	BPMN process model: Steps 1 to 4 of <i>ProPExt</i> sub-process <i>Decide whether modeled and generated solution patterns are congruent</i>	92
7.6	BPMN process model: Steps 5 to 13 of <i>ProPExt</i> process <i>Decide whether modeled and generated solution patterns are congruent</i>	93
8.1	Requirements document created by the <i>Sales and Distribution</i> de- partment	102
8.2	Analysis model for business object <i>Invoice</i>	103
8.3	<i>Invoice</i> business object: Problem-context model as result of Step 2 in the ProPGen <i>Generate problem-context pattern</i> process	104
8.4	<i>Invoice</i> business object: Problem-context pattern as result of ProP- Gen Step 3	106
8.5	<i>Invoice</i> business object: Solution model as result of refactoring the problem-context model	107
8.6	Stereotype property value <i>pcm</i> maintained in solution model for In- voice business object as result of Step 1 in ProPGen process <i>Generate solution pattern</i>	108
8.7	<i>Invoice</i> business object: Solution pattern as result of ProPGen Step 2	110
8.8	<i>Adapter</i> : Solution pattern modeled from literature pattern as result of ProPExt Step 1	111
8.9	<i>Adapter</i> : Problem-bearing model as result of ProPExt Step 2.2	112
8.10	<i>Adapter</i> : Problem-context model as result of Step 2 in the ProPGen <i>Generate problem-context pattern</i> process	114
8.11	<i>Adapter</i> : Problem-context pattern as result of ProPGen Step 3 (as well as of ProPExt Step 2.3/3.1)	116
8.12	<i>Adapter</i> : Solution model as result of ProPExt Step 2.4	117

8.13	<i>Adapter</i> : Stereotype property value <i>pcm</i> maintained in solution model for Drawing Editor Application as result of Step 1 in ProPGen process <i>Generate solution pattern</i>	118
8.14	<i>Adapter</i> : Solution pattern as result of ProPGen Step 2 (as well as of ProPExt Step 2.5/3.2)	120
8.15	<i>Adapter</i> : Overview of solution pattern modeled from the literature compared to solution pattern generated from Drawing Editor Application example as result of ProPExt Step 4.1	124
8.16	<i>Adapter</i> : Problem-context pattern as result of renaming according to Step 4.4	126
8.17	<i>Singleton</i> : Solution pattern modeled from literature pattern as result of ProPExt Step 1	128
8.18	<i>Singleton</i> : Problem-bearing model as result of ProPExt Step 2.2 . . .	129
8.19	<i>Singleton</i> : Problem-context model as result of Step 2 in the ProPGen <i>Generate problem-context pattern</i> process	131
8.20	<i>Singleton</i> : Problem-context pattern as result of ProPGen Step 3 (as well as of ProPExt Step 2.3/3.1)	133
8.21	<i>Singleton</i> : Solution model as result of ProPExt Step 2.4	134
8.22	<i>Singleton</i> : Stereotype property value <i>pcm</i> maintained in solution model for Maze Game Application as result of Step 1 in ProPGen process <i>Generate solution pattern</i>	135
8.23	<i>Singleton</i> : Solution pattern as result of ProPGen Step 2 (as well as of ProPExt Step 2.5/3.2)	137
8.24	<i>Singleton</i> : Overview of solution pattern modeled from the literature compared to solution pattern generated from Maze Game Application example as result of ProPExt Step 4.1	139
8.25	<i>Singleton</i> : Modified/adjusted problem-bearing model as result of the second iteration of ProPExt Step 2.2	141
8.26	<i>Singleton</i> : Problem-context model as result of Step 2 (Second iteration) in the ProPGen <i>Generate problem-context pattern</i> process . . .	142
8.27	<i>Singleton</i> : Adjusted problem-context pattern as result of ProPGen Step 2.3/3.1	143
8.28	<i>Singleton</i> : Solution model as result of ProPExt Step 2.4, based on adjusted problem-context model	143

8.29	<i>Singleton</i> : Adjusted solution pattern as result of the second iteration of ProPGen Step 2 (as well as of ProPExt Step 2.5/3.2)	144
8.30	<i>Singleton</i> : Overview of solution pattern modeled from the literature compared to solution pattern generated from adjusted Maze Game Application example as result of second iteration of ProPExt Step 4.1 after adjusting the generated solution pattern	147
8.31	<i>Singleton</i> : Problem-context pattern as result of renaming according to Step 4.4	148
8.32	<i>Observer</i> : Solution pattern modeled from literature pattern as result of ProPExt Step 1	150
8.33	<i>Observer</i> : Problem-bearing model as result of ProPExt Step 2.2 . . .	151
8.34	<i>Observer</i> : Problem-context model as result of Step 2 in the ProPGen <i>Generate problem-context pattern</i> process	153
8.35	<i>Observer</i> : Problem-context pattern as result of ProPGen Step 3 (as well as of ProPExt Step 2.3/3.1)	154
8.36	<i>Observer</i> : Solution model as result of ProPExt Step 2.4	155
8.37	<i>Observer</i> : Solution pattern as result of ProPGen Step 2 (as well as of ProPExt Step 2.5/3.2)	157
8.38	<i>Observer</i> : Overview of solution pattern modeled from the literature compared to solution pattern generated from Salary Statement Application example as result of ProPExt Step 4.1	161
8.39	<i>Observer</i> : Problem-context pattern as result of renaming according to Step 4.4	163
9.1	Overview of the method for retrieving problem-oriented patterns, <i>ProPRet</i>	169
9.2	BPMN process model: Overview of ProPMan processes for improving domain-specific UML analysis or design models - containing Step 2 <i>Search for appropriate solution using pattern retrieval</i>	171
9.3	BPMN process model: <i>ProPRet</i> process <i>Search for appropriate solution using pattern retrieval</i>	172
9.4	Example for result of Step 1: Selected left problem-context pattern to be used as search criterion	174
9.5	Example for result of Step 2: Selected appropriate context menu entry for ProPRet retrieval function	174

9.6	Mockup with excerpt from external ProPRet result list presented to pattern consumers in the graphical user interface	176
9.7	<i>ProPRet</i> - Example for abstracting a problem-context pattern in UML notation to a problem-context pattern in graph representation, a <i>PCP graph</i>	178
9.8	<i>ProPRet</i> retrieval approach: Example used for explaining the metrics to determine similarity of different graphs	181
9.9	BPMN process model: Steps 1 to 5 of <i>ProPRet</i> sub-process <i>Retrieve patterns</i>	185
9.10	BPMN process model: Steps 6 to 10 of <i>ProPRet</i> sub-process <i>Retrieve patterns</i>	186
9.11	Step 3: <i>Problem</i> and <i>context</i> stereotype property values of left problem-context pattern for SAP System Messages Viewer Application	188
9.12	Step 3: Contents of <i>problem</i> properties file	189
9.13	Step 3: Contents of <i>context</i> properties file	189
9.14	Step 3: <i>Problem</i> and <i>context</i> stereotype property values of right problem-context pattern	189
9.15	Step 3: «ProblemBearing» first supplier attribute in left problem-context pattern	190
9.16	Step 3: «ProblemBearing» second supplier property in left problem-context pattern	191
9.17	Step 3: «ProblemBearing» supplier property in right problem-context pattern	191
9.18	Step 3: «ProblemBearing» supplier method in right problem-context pattern	192
9.19	Step 7: Excerpt of attributes from class <i>RetrievalResult</i> relevant to be presented in the graphical user interface	195
9.20	Mockup with external ProPRet result list presented to pattern consumers in the graphical user interface	198
9.21	BPMN process model: Steps 1 to 5 of <i>ProPRet</i> sub-process <i>Permute and calculate similarity of PCP graph node mappings</i>	199
9.22	BPMN process model: Steps 6 to 8 of <i>ProPRet</i> sub-process <i>Permute and calculate similarity of PCP graph node mappings</i>	200

9.23	BPMN process model: Steps 1 to 3 of <i>ProPRet</i> sub-process <i>Create mapping and similarity metrics for current permutation</i>	216
9.24	BPMN process model: Steps 4 to 6 of <i>ProPRet</i> sub-process <i>Create mapping and similarity metrics for current permutation</i>	217
9.25	BPMN data object <i>PcpGraphMappingCollection</i> as used in process <i>Create mapping and similarity metrics for current permutation</i> - expressed as UML class	218
9.26	BPMN data object <i>PcpGraphMapping</i> as used in process <i>Create mapping and similarity metrics for current permutation</i> - expressed as UML class	219
9.27	BPMN data object <i>MappingLineItem</i> as used in process <i>Create mapping and similarity metrics for current permutation</i> - expressed as UML class	220
9.28	Example: Retrieval using problem-context patterns with several problem statements	231
9.29	Example: Retrieval using problem-context patterns with one problem statement	232
9.30	BPMN process model: <i>ProPGen</i> process <i>Generate problem-context pattern</i>	233
9.31	Problem-context model with two problems	234
9.32	First projection: Problem-context model with one problem	235
9.33	Second projection: Problem-context model with one problem	235
10.1	Overview of the method for instantiating problem-oriented patterns, ProPIns	239
10.2	BPMN process model: Overview of ProPMan processes for improving domain-specific UML analysis or design models containing Step 5 <i>Apply pattern</i>	241
10.3	BPMN process model: <i>ProPIns</i> process <i>Apply pattern</i>	242
10.4	Running example <i>Salary Statement Application</i> used for ProPIns: Excerpt of ProPRet result list with relevant entry for <i>Salary Statement Application</i> patterns as external result list presented in the graphical user interface (mockup)	244

10.5	Running example <i>Salary Statement Application</i> used for ProPIs: Selected entry and activated button <i>Instantiate Solution pattern from selected line</i>	245
10.6	Running example <i>Salary Statement Application</i> used for ProPIs: retrieval <i>result</i> problem-context pattern	245
10.7	Running example <i>Salary Statement Application</i> used for ProPIs: retrieval <i>result</i> solution pattern	246
10.8	Running example <i>SAP System Messages Viewer Application</i> used for ProPIs: retrieval <i>request</i> problem-context pattern	246
10.9	ProPIs example: Excerpt from the highest score mapping between <i>Salary Statement Application</i> and <i>SAP System Messages Viewer Application</i> problem-context patterns	248
10.10	SAP System Messages Viewer Application: Abstract retrieval request solution model resulting from the process <i>Instantiate retrieval result solution pattern to abstract retrieval request solution model</i>	250
10.11	<i>SAP System Messages Viewer Application</i> : Instantiated domain-specific retrieval request <i>concrete Solution model</i>	253
10.12	SAP System Messages Viewer Application: Project with instantiated domain-specific retrieval request solution model	256
10.13	BPMN process model: Steps 1 to 6 of <i>ProPIs</i> sub-process <i>Instantiate retrieval result solution pattern to abstract retrieval request solution model</i>	257
10.14	BPMN process model: Steps 7 to 11 of <i>ProPIs</i> sub-process <i>Instantiate retrieval result solution pattern to abstract retrieval request solution model</i>	258
10.15	SAP System Messages Viewer Application: Folder with instantiated abstract retrieval request solution model	260
10.16	Retrieval result <i>Salary Statement Application</i> : Problem-context pattern with model elements relevant for instantiation - annotated with the appropriate next steps	262
10.17	Retrieval result <i>Salary Statement Application</i> : Solution pattern with model elements relevant for instantiation - annotated with the appropriate next step	262

10.18	Running example <i>SAP System Messages Viewer Application</i> used for ProIns: Resulting abstract retrieval request solution model after processing all relevant elements according to sub-process <i>Instantiate new attribute or method from source retrieval result pattern as new model element to target abstract retrieval request solution model</i> . . .	265
10.19	BPMN process model: <i>ProPIns</i> sub-process <i>Instantiate abstract retrieval request solution model to domain-specific solution model</i> . . .	273
11.1	Eclipse Software Development Kit Architecture Overview	287
11.2	Architecture of the <i>ProPMan tool</i> feature as the new Eclipse tool to support our ProPMan Methodology	288
11.3	Overview about the <i>ProPMan Core Functions</i> plug-in component . .	289
11.4	Class diagram of the ProPGen Pcm2Pcp Transformation (Pcm2Pcp Trafo) component	291
11.5	Sequence diagram for <i>Pcm2PcpExtendedHandler</i> : Objects and messages for the Pcm2Pcp Trafo	292
11.6	Class diagram of the ProPGen Sm2Sp Transformation (Sm2Sp Trafo) component	293
11.7	Sequence diagram for <i>Sm2SpExtendedHandler</i> : Objects and messages for the Sm2Sp Trafo	294
11.8	Class diagram of the <i>ProPRet Retrieval Engine</i> component	296
11.9	Class diagram of the ProPRet <i>ProPRetExtendedHandler</i> Controller class	297
11.10	Sequence diagram for <i>ProPRetExtendedHandler</i> : Objects and messages for a ProPRet Pattern Retrieval run	298
11.11	Class <i>ResultListTableView</i> : Excerpt of attributes	299
11.12	Class <i>ResultListTableView</i> : Excerpt of methods	300
11.13	Class <i>RetrievalResult</i> : Attributes	301
11.14	Class <i>RetrievalResult</i> : Methods	302
11.15	Sequence diagram for <i>RetrievalResult</i> : Objects and messages for the ProPIns instantiation procedure	303
11.16	Class diagram: Overview about the <i>ProPMan Supplemental Functions</i> plug-in component classes	304
11.17	Class <i>PcpGraphMappingCollection</i> : Excerpt of attributes and methods	305
11.18	Class <i>PcpGraphMapping</i> : Excerpt of attributes and methods	306

11.19	Class <i>MappingLineItem</i> : Excerpt of attributes and methods	307
11.20	Class <i>PsPropsProvider</i> : Attributes and Methods	308
11.21	Class <i>PcpNodeSimCalculator</i> with dependency on <i>SimMetrics Functions</i> plug-in	308
11.22	Component <i>SimMetrics Functions</i> plug-in containing <i>Levenshtein</i> class for string edit distance calculation	309
11.23	Class <i>ProPManCoreFunctionsPlugIn</i> and its integration into the Eclipse plug-in class hierarchy	333
11.24	<i>ProPMan Core Functions</i> plug-in: Overview about package structure	334
11.25	<i>ProPGen Pcm2Pcp Transformator</i> component: Packages	335
11.26	Sequence diagram for <i>Pcm2PcpExtendedHandler</i> : Objects and messages for the Pcm2Pcp Trafo	337
11.27	<i>ProPGen Sm2Sp Transformator</i> component: Packages	350
11.28	Sequence diagram for <i>Sm2SpBaseHandler</i> : Objects and messages for the Sm2Sp Trafo	351
11.29	<i>ProPRet Retrieval Engine</i> component: Packages	361
11.30	Sequence diagram for <i>ProPRetExtendedHandler</i> : Objects and messages for preparing the models that need to be compared in the ProPRet Pattern Retrieval run	363
11.31	Sequence diagram for <i>RetrievalResult</i> : Objects and messages for preparing the models that need to be compared in the ProPIns instantiation run	390
11.32	<i>ProPMan Supplemental Functions</i> plug-in: Overview about package structure	427
11.33	<i>propret</i> package: Classes	427
11.34	Eclipse Project Explorer with projects that are pre-defined by ProPMan	441
11.35	Project <i>99_ProPMan Transformation Trace Models</i> with its subfolders	443
11.36	Expanded subfolder for problem-context model to pattern transformation trace models	443
11.37	Expanded subfolder for solution model to pattern transformation trace models	444
11.38	Expanded subfolder for problem-context to solution pattern transformation trace models	444

11.39	ProPMan context menu extensions for its ProPGen and ProPRet pattern management functions	445
11.40	Entries in the ProPMan context menu extension for its two ProPGen pattern management functions	445
11.41	Entry in the ProPMan context menu extension for its ProPRet retrieval function	446
11.42	Papyrus view <i>ProPRet result list of retrieved patterns</i>	446
11.43	Activation of the Papyrus view <i>ProPRet result list of retrieved patterns</i>	447
11.44	Maintaining <i>problem</i> properties files in the ProPMan customizing . .	448
11.45	Activity <i>Create domain-specific problem-context model</i> as preparatory step before running the Pcm2Pcp Trafo function	449
11.46	Activity <i>Create domain-specific solution model</i> as preparatory step before running the Sm2Sp Trafo function	450
A.1	Running example <i>Salary Statement Application</i> : Program Steps and Graphical User Interface	463
A.2	Running example <i>Salary Statement Application</i> : Problem-bearing model that needs to be improved due to design deficiencies	464
A.3	Running example <i>SAP System Messages Viewer Application</i> : Graphical user interface	466
A.4	SAP System Messages Viewer Application: Domain-specific problem-context model created by pattern consumers	467
A.5	SAP System Messages Viewer Application: Cross-domain problem-context pattern created by pattern consumers and used as search criterion for ProPRet	469
B.1	<i>Invoice</i> business object: Problem-context pattern	472
B.2	<i>Invoice</i> business object: Solution pattern	472
B.3	<i>Adapter</i> : Problem-context pattern	473
B.4	<i>Adapter</i> : Solution pattern	473
B.5	<i>Singleton</i> : Problem-context pattern	474
B.6	<i>Singleton</i> : Solution pattern	474
B.7	<i>Observer</i> : Problem-context pattern as result of ProPGen Step 3 (as well as of ProPExt Step 2.3/3.1)	475

B.8	<i>Observer</i> : Solution pattern as result of ProPGen Step 2 (as well as of ProPExt Step 2.5/3.2)	475
C.1	BPMN process model: <i>ProPIns</i> sub-process <i>Instantiate «ProblemBearing» dependency supplier element</i>	478
C.2	ProPIns example: Extract from the highest score mapping between <i>Salary Statement Application</i> and <i>SAP System Messages Viewer Application</i> problem-context patterns	480
C.3	ProPIns example: Excerpt of the <i>Salary Statement Application</i> transformation trace model for <i>@abstraction_of@employeeDet:EmployeeDetail[1]</i>	482
C.4	BPMN process model: <i>ProPIns</i> sub-process <i>Instantiate class as type of «ProblemBearing» dependency supplier element</i>	486
C.5	ProPIns example: Excerpt of the <i>Salary Statement Application</i> transformation trace model for class <i>@abstraction_of@EmployeeDetail</i>	489
C.6	BPMN process model: <i>ProPIns</i> sub-process <i>Instantiate added new model element from source retrieval result pattern as new model element to target abstract retrieval request solution model</i>	492
C.7	BPMN process model: Steps 1 to 5 and Step 7 of the <i>ProPIns</i> sub-process <i>Instantiate new attribute or method from source retrieval result pattern as new model element to target abstract retrieval request solution model</i>	497
C.8	BPMN process model: Steps 6 and 8 to 12 of the <i>ProPIns</i> sub-process <i>Instantiate new attribute or method from source retrieval result pattern as new model element to target abstract retrieval request solution model</i>	498
C.9	Running example <i>SAP System Messages Viewer Application</i> used for ProPIns: Resulting abstract retrieval request solution model after processing all relevant elements according to sub-process <i>Instantiate new attribute or method from source retrieval result pattern as new model element to target abstract retrieval request solution model</i>	511
C.10	BPMN process model: Steps 1 to 5 and 10 to 12 of <i>ProPIns</i> sub-process <i>Instantiate new generalization from source retrieval result pattern as new model element to target abstract retrieval request solution model</i>	513

C.11 BPMN process model: Steps 6 to 9 and 13 to 16 of <i>ProPI</i> ns sub-process <i>Instantiate new generalization from source retrieval result pattern as new model element to target abstract retrieval request solution model</i>	514
C.12 BPMN process model: Steps 1 and 2 of <i>ProPI</i> ns sub-process <i>Instantiate container class of «ProblemBearing» supplier attribute or method element</i>	527
C.13 BPMN process model: Steps 3 to 6 of <i>ProPI</i> ns sub-process <i>Instantiate container class of «ProblemBearing» supplier attribute or method element</i>	528
C.14 BPMN process model: Steps 1 to 3 of <i>ProPI</i> ns process <i>Instantiate existing and deleted elements from left retrieval request abstract to domain-specific solution model</i>	532
C.15 BPMN process model: Steps 4 to 7 of <i>ProPI</i> ns process <i>Instantiate existing and deleted elements from left retrieval request abstract to domain-specific solution model</i>	533
C.16 SAP System Messages Viewer Application: Domain-specific problem-context model	534
C.17 BPMN process model: Steps 1 to 4 and 8 and 9 of <i>ProPI</i> ns process <i>Instantiate new elements from left retrieval request abstract to domain-specific solution model</i>	545
C.18 BPMN process model: Steps 5 to 7 and 10 of <i>ProPI</i> ns process <i>Instantiate new elements from left retrieval request abstract to domain-specific solution model</i>	546
C.19 Running example <i>SAP System Messages Viewer Application</i> used for <i>ProPI</i> ns: Resulting abstract retrieval request solution model according to the process <i>Instantiate retrieval result solution pattern to abstract retrieval request solution model</i>	548
C.20 BPMN process model: Steps 1 to 3 of <i>ProPI</i> ns process <i>Instantiate new generalization in left abstract retrieval request solution model to domain-specific solution model</i>	557
C.21 BPMN process model: Steps 4 to 8 of <i>ProPI</i> ns process <i>Instantiate new generalization in left abstract retrieval request solution model to domain-specific solution model</i>	558

D.1	Class <i>PcpGraphMappingCollection</i>	565
D.2	Class <i>PcpGraphMapping</i> : Attributes	566
E.1	Oracle download area for Java JDK and JRE	578
E.2	Example: Downloaded file for Java JDK	578
E.3	Start of installation wizard for Java	579
E.4	Example: Create a file directory folder on the target computer to prepare copy of ProPMan Software Package files	579
E.5	Example: Open file directory folder containing files to be copied from DVD to target computer	580
E.6	Example: Select and copy all files of the ProPMan Software Package from DVD to target computer	580
E.7	Example: Paste the files and folders of the ProPMan Software Pack- age copied from DVD into target directory	581
E.8	Absolute path of the Java Virtual Machine to be changed in the file <code>eclipse.ini</code>	581
E.9	Adjustment of default Eclipse workspace: Start of Eclipse executable program	582
E.10	Adjustment of default Eclipse workspace: Error after start of Eclipse executable program	582
E.11	Adjustment of default Eclipse workspace: Error after confirming the first error dialog message	582
E.12	Adjustment of default Eclipse workspace: Dialog message for chang- ing path to Eclipse workspace	583
E.13	Adjustment of default Eclipse workspace: Opened Eclipse application after adjusting the default Eclipse workspace	583
E.14	Activity <i>Create domain-specific problem-context model</i> as refinement of the Step 1 of the activity <i>Generate problem-context pattern</i>	584
E.15	Activity <i>Copy project with domain-specific problem-bearing model</i> as refinement process of the Step 1 of the activity <i>Create domain-specific problem-context model</i>	586
E.16	Pattern Provider activities: Domain-specific model <i>SalaryStatemen- tApplication</i> that needs to be refactored	587
E.17	Pattern Provider activities: Select and copy domain-specific model <i>SalaryStatementApplication</i> that needs to be refactored	587

E.18	Pattern Provider activities: Paste copied domain-specific model to ProPMan specific modeling folder	588
E.19	Pattern Provider activities: Optional step of renaming copy of domain-specific model	588
E.20	Pattern Provider activities: Result of optional step of renaming copy of domain-specific model	588
E.21	Activity <i>Apply ProPMan UML Profile to copy of domain-specific problem-bearing model</i> as refinement of the Step 2 of the Activity <i>Create domain-specific problem-context model</i>	589
E.22	Pattern Provider activities: Open domain-specific model	590
E.23	Pattern Provider activities: Show domain-specific model	590
E.24	Pattern Provider activities: Select domain-specific model in the Papyrus Model Explorer in order to apply a profile	591
E.25	Pattern Provider activities: Start of function to apply a profile	591
E.26	Pattern Provider activities: Screen to pick a profile to the selected elements area	591
E.27	Pattern Provider activities: Screen to confirm application of a selected profile	591
E.28	Pattern Provider activities: Screen to show details (name, location and version) about an applied profile	592
E.29	Pattern Provider activities: Screen to assign stereotype «ProblemContextModel» to a problem-bearing model	592
E.30	Pattern Provider activities: Screen to apply stereotype «ProblemContextModel» to a problem-bearing model	592
E.31	Activity <i>Add problem statement to domain-specific problem-bearing model</i> as refinement of the Step 3 of the Activity <i>Create domain-specific problem-context model</i>	593
E.32	Pattern Provider activities: Adding a new class from the Papyrus Palette View	594
E.33	Pattern Provider activities: Show the newly created class in the Papyrus model diagram	594
E.34	Pattern Provider activities: Start adding the «ProblemStatement» stereotype to the newly created class	594

E.35	Pattern Provider activities: Adding the « ProblemStatement » stereotype to the newly created class by picking it from the area with available stereotypes	595
E.36	Steps 1-9 of Activity <i>Add «ProblemStatement» stereotype property values for problem and context to domain-specific problem-bearing model</i> as refinement of the Step 4 of the activity <i>Create domain-specific problem-context model</i>	595
E.37	Steps 10-18 of Activity <i>Add «ProblemStatement» stereotype property values for problem and context to domain-specific problem-bearing model</i> as refinement of the Step 4 of the activity <i>Create domain-specific problem-context model</i>	596
E.38	Pattern Provider activities: Result of adding the « ProblemStatement » stereotype to the newly created class with empty values for properties <i>problem</i> and <i>context</i>	597
E.39	Pattern Provider activities: Select and start to open properties file for predefined values for property <i>problem</i>	597
E.40	Pattern Provider activities: Content of properties file for predefined values for property <i>problem</i>	597
E.41	Pattern Provider activities: Copy value from properties file for predefined values for property <i>problem</i>	598
E.42	Pattern Provider activities: Paste value from properties file for predefined values for property <i>problem</i> to field in stereotype properties area	598
E.43	Pattern Provider activities: Select and open properties file with predefined context properties values	598
E.44	Pattern Provider activities: Show content of properties file with predefined values for property <i>context</i>	598
E.45	Pattern Provider activities: Paste value from properties file for predefined values for property <i>context</i> to field in stereotype properties area	599
E.46	Pattern Provider activities: Completed new class with stereotype « ProblemBearing » and maintained values for properties <i>problem</i> and <i>context</i>	599

E.47 Activity <i>Add dependencies with stereotype «ProblemBearing» to domain-specific problem-bearing model</i> as refinement of the Step 5 of the activity <i>Create domain-specific problem-context model</i> : Steps 1 to 4 . . .	600
E.48 Activity <i>Add dependencies with stereotype «ProblemBearing» to domain-specific problem-bearing model</i> as refinement of the Step 5 of the activity <i>Create domain-specific problem-context model</i> : Steps 5 to 10 . . .	600
E.49 Pattern Provider activities: Add new dependency from new class with «ProblemStatement» stereotype to relevant other class	601
E.50 Pattern Provider activities: Result of adding new dependency from new class with «ProblemStatement» stereotype to relevant other class	601
E.51 Pattern Provider activities: Change stereotype of new dependency to «ProblemBearing»	601
E.52 Pattern Provider activities: Result of changing stereotype of new dependency to «ProblemBearing»	602
E.53 Pattern Provider activities: Change supplier of «ProblemBearing» dependency to the property <i>employeeDet</i>	602
E.54 Pattern Provider activities: Change supplier of «ProblemBearing» dependency to the operation <i>EmployeeSelector</i>	603
E.55 Pattern Provider activities: Context menu selection and start of the problem-context model to pattern transformator function, the <i>Pcm2Pcp Trafo</i>	603
E.56 Pattern Provider activities: Progress indicator of Pcm2Pcp Trafo job	603
E.57 Pattern Provider activities: Opening the ProPMan Pattern Library project	604
E.58 Pattern Provider activities: Start to open the generated problem-context pattern	604
E.59 Pattern Provider activities: Open the generated problem-context pattern within the Model Explorer	604
E.60 Pattern Provider activities: Navigation into the project <i>99_ProPMan Transformation Trace Models</i>	605
E.61 Pattern Provider activities: Navigation into the folder <i>Pcm2Pcp Trafo Trace Models</i> within the project <i>99_ProPMan Transformation Trace Models</i>	605

E.62	Pattern Provider activities: Start to open the generated Pcm2Pcp transformation trace model from the folder <i>Pcm2Pcp Trafo Trace Models</i>	605
E.63	Pattern Provider activities: Opened generated Pcm2Pcp transformation trace model	605
E.64	Activity <i>Create domain-specific solution model</i> as refinement of the Step 1 of the activity <i>Generate solution pattern</i> : Steps 1 to 4	607
E.65	Activity <i>Create domain-specific solution model</i> as refinement of the Step 1 of the activity <i>Generate solution pattern</i> : Steps 5 to 8	607
E.66	Pattern Provider activities: Select and copy problem-context model that needs to be refactored	608
E.67	Pattern Provider activities: Paste copied problem-context model that needs to be refactored into the same project	608
E.68	Pattern Provider activities: Rename copied problem-context model to solution model name for refactoring	608
E.69	Pattern Provider activities: Renamed solution model for refactoring	609
E.70	Pattern Provider activities: Open renamed solution model for refactoring	609
E.71	Pattern Provider activities: Show opened renamed solution model for refactoring	609
E.72	Pattern Provider activities: Open stereotype properties view to change stereotype of solution model copied from problem-context model	610
E.73	Pattern Provider activities: Remove currently assigned «ProblemContextModel» stereotype from solution model	610
E.74	Pattern Provider activities: Add new stereotype «SolutionModel» to solution model	610
E.75	Pattern Provider activities: Newly added stereotype of solution model	610
E.76	Pattern Provider activities: Added value for stereotype property <i>pcm</i> of solution model	611
E.77	Pattern Provider activities: Refactoring by adding new classes	611
E.78	Pattern Provider activities: Result of adding new classes	611
E.79	Pattern Provider activities: Refactoring by adding operations to newly created classes	612

E.80	Pattern Provider activities: Completed result of refactoring before elimination of problem information	612
E.81	Pattern Provider activities: Elimination of problem statement and problem-bearing dependencies after completion of the refactoring . .	613
E.82	Pattern Provider activities: Completed solution model	613
E.83	Pattern Provider activities: ProPMan context menu for ProPGen Transformations with function <i>Transform selected solution model to solution pattern (Sm2Sp Trafo)</i>	614
E.84	Pattern Provider activities: Contents of the generated solution pattern shown within the Model Explorer	614
E.85	Activity <i>Create domain-specific problem-context model</i> as refinement of the Step 1 of the activity <i>Generate problem-context pattern</i> . . .	616
E.86	Activity <i>Copy project with domain-specific problem-bearing model</i> as refinement process of the Step 1 of the activity <i>Create domain-specific problem-context model</i> : Steps 1 to 4	617
E.87	Activity <i>Copy project with domain-specific problem-bearing model</i> as refinement process of the Step 1 of the activity <i>Create domain-specific problem-context model</i> : Steps 5 to 9	618
E.88	Pattern Consumer activities: Domain-specific model <i>SAPSystemMessagesViewerApplication</i> that needs to be refactored	619
E.89	Pattern Consumer activities: Select and copy domain-specific model <i>SAPSystemMessagesViewerApplication</i> that needs to be refactored .	619
E.90	Pattern Consumer activities: Paste copied domain-specific model to ProPMan specific modeling folder	620
E.91	Pattern Consumer activities: Optional step of renaming copy of domain-specific model	620
E.92	Pattern Consumer activities: Result of optional step of renaming copy of domain-specific model	620
E.93	Activity <i>Apply ProPMan UML Profile to copy of domain-specific problem-bearing model</i> as refinement of the Step 2 of the Activity <i>Create domain-specific problem-context model</i> : Steps 1 to 7	621
E.94	Activity <i>Apply ProPMan UML Profile to copy of domain-specific problem-bearing model</i> as refinement of the Step 2 of the Activity <i>Create domain-specific problem-context model</i> : Steps 2 to 15	622

E.95	Pattern Consumer activities: Show domain-specific model	622
E.96	Pattern Consumer activities: Select domain-specific model in the Papyrus Model Explorer in order to apply a profile	623
E.97	Pattern Consumer activities: Start of function to apply a profile	623
E.98	Pattern Consumer activities: Screen to pick a profile to the selected elements area	623
E.99	Pattern Consumer activities: Screen to confirm application of a selected profile	623
E.100	Pattern Consumer activities: Screen to show details (name, location and version) about an applied profile	624
E.101	Pattern Consumer activities: Screen to assign stereotype «ProblemContextModel» to a problem-bearing model	624
E.102	Pattern Consumer activities: Screen to apply stereotype «ProblemContextModel» to a problem-bearing model	624
E.103	Activity <i>Add problem statement to domain-specific problem-bearing model</i> as refinement of the Step 3 of the Activity <i>Create domain-specific problem-context model</i> : Steps 1 to 7	625
E.104	Activity <i>Add problem statement to domain-specific problem-bearing model</i> as refinement of the Step 3 of the Activity <i>Create domain-specific problem-context model</i> : Steps 8 to 13	626
E.105	Pattern Consumer activities: Adding a new class from the Papyrus Palette View	626
E.106	Pattern Consumer activities: Show the newly created class in the Papyrus model diagram	627
E.107	Pattern Consumer activities: Start adding the «ProblemStatement» stereotype to the newly created class	627
E.108	Pattern Consumer activities: Adding the «ProblemStatement» stereotype to the newly created class by picking it from the area with available stereotypes	627
E.109	Activity <i>Add «ProblemStatement» stereotype property values for problem and context to domain-specific problem-bearing model</i> as refinement of the Step 4 of the activity <i>Create domain-specific problem-context model</i>	628

E.110	Pattern Consumer activities: Result of adding the «ProblemStatement» stereotype to the newly created class with empty values for properties <i>problem</i> and <i>context</i>	629
E.111	Pattern Consumer activities: Select and start to open properties file for predefined values for property <i>problem</i>	629
E.112	Pattern Consumer activities: Content of properties file for predefined values for property <i>problem</i>	629
E.113	Pattern Consumer activities: Copy value from properties file for predefined values for property <i>problem</i>	630
E.114	Pattern Consumer activities: Paste value from properties file for predefined values for property <i>problem</i> to field in stereotype properties area	630
E.115	Pattern Consumer activities: Select and open properties file with predefined context properties values	630
E.116	Pattern Consumer activities: Show content of properties file with predefined values for property <i>context</i>	630
E.117	Pattern Consumer activities: Paste value from properties file for predefined values for property <i>context</i> to field in stereotype properties area	631
E.118	Pattern Consumer activities: Completed new class with stereotype «ProblemBearing» and maintained values for properties <i>problem</i> and <i>context</i>	631
E.119	Activity <i>Add dependencies with stereotype «ProblemBearing» to domain-specific problem-bearing model</i> as refinement of the Step 5 of the activity <i>Create domain-specific problem-context model</i>	632
E.120	Pattern Consumer activities: Add new dependency from new class with «ProblemStatement» stereotype to relevant other class	633
E.121	Pattern Consumer activities: Result of adding new dependency from new class with «ProblemStatement» stereotype to relevant other class	633
E.122	Pattern Consumer activities: Change stereotype of new dependency to «ProblemBearing»	633
E.123	Pattern Consumer activities: Result of changing stereotype of new dependency to «ProblemBearing»	634

E.124	Pattern Consumer activities: Change supplier of «ProblemBearing» dependency to the property <i>lo_t100a_info</i>	634
E.125	Pattern Consumer activities: Change supplier of «ProblemBearing» dependency to the operation <i>lo_t100_details</i>	634
E.126	Pattern Consumer activities: Context menu selection and start of the problem-context model to pattern transformer function, the <i>Pcm2Pcp Trafo</i>	635
E.127	Pattern Consumer activities: Progress indicator of Pcm2Pcp Trafo job	635
E.128	Pattern Consumer activities: Start to open the generated problem- context pattern	635
E.129	Pattern Consumer activities: Start pattern retrieval function on se- lected problem-context pattern as search criterion	636
E.130	Pattern Consumer activities: Result list of pattern retrieval function on selected problem-context pattern as search criterion	636
E.131	Pattern Consumer activities: Instantiate a selected pattern from re- trieval result list via the function <i>Instantiate Solution Pattern from selected line</i>	637
E.132	SAP System Messages Viewer Application: Folder with instantiated domain-specific retrieval request solution model	638
E.133	SAP System Messages Viewer Application: Instantiated domain-specific retrieval request solution model	639

List of Tables

3.1	Mapping of analysis pattern documentation sections to problem-context and solution patterns	21
3.2	Mapping of design pattern documentation sections to problem-context and solution patterns	22
6.1	Enumeration of general, re-occurring <i>problems</i> identified during our research work	56
6.2	Enumeration of general, re-occurring <i>contexts</i> identified during our research work	57
6.3	Conditions a problem-context model element needs to fulfill to be considered in problem-context pattern	59
6.4	Conditions a solution model element alternatively needs to fulfill to be considered in solution pattern	71
8.1	<i>Invoice</i> business object : Conditions and fitting problem-context model elements for transformation to problem-context pattern	105
8.2	<i>Invoice</i> business object: Fitting problem-context model elements renamed in problem-context pattern	106
8.3	<i>Invoice</i> business object: Conditions and fitting solution model elements for transformation to solution pattern	109
8.4	<i>Invoice</i> business object: Fitting solution model elements renamed in solution pattern	110
8.5	Drawing Editor Application: Conditions and fitting problem-context model elements for transformation to problem-context pattern	115
8.6	Drawing Editor Application: Fitting problem-context model elements renamed in problem-context pattern	115

8.7	Drawing Editor Application: Conditions and fitting solution model elements for transformation to solution pattern	119
8.8	Drawing Editor Application: Fitting solution model elements renamed in solution pattern	120
8.9	Adapter: Single steps and results according to process <i>Decide whether modeled and generated solution patterns are congruent</i>	121
8.10	Adapter: Single steps and results according to process <i>Decide whether modeled and generated solution patterns are congruent</i> Second Iteration	122
8.11	Adapter: Single steps and results according to process <i>Decide whether modeled and generated solution patterns are congruent</i> Third Iteration	123
8.12	Maze Game Application: Conditions and fitting problem-context model elements for transformation to problem-context pattern	132
8.13	Maze Game Application: Fitting problem-context model elements renamed in problem-context pattern	133
8.14	Maze Game Application: Conditions and fitting solution model elements for transformation to solution pattern	136
8.15	Maze Game Application: Fitting solution model elements renamed in solution pattern	137
8.16	Singleton: Single steps and results according to process <i>Decide whether modeled and generated solution patterns are congruent: First iteration</i>	138
8.17	Singleton: Single steps and results according to process <i>Decide whether modeled and generated solution patterns are congruent</i> for adjusted generated solution pattern	145
8.18	Observer: Single steps and results according to process <i>Decide whether modeled and generated solution patterns are congruent: First iteration</i>	157
8.19	Observer: Single steps and results according to process <i>Decide whether modeled and generated solution patterns are congruent: Second iteration</i>	159
8.20	Observer: Single steps and results according to process <i>Decide whether modeled and generated solution patterns are congruent: Third iteration</i>	159
8.21	Observer: Single steps and results according to process <i>Decide whether modeled and generated solution patterns are congruent: Fourth iteration</i>	160

9.1	Result of Step 3: Example for contents of the BPMN data object <i>Internal result list</i> - represented as enumeration within the ProPRet tool and sorted according to the calculated ranks	175
9.2	Example for result of Step 1: Set of right problem-context patterns .	187
9.3	Overview about resulting PCP Graph string tuples according to Step 3	192
9.4	PCP graph node mappings for running examples	194
9.5	Example for retrieval result as result of Step <i>Add highest graph edit similarity score to current retrieval result</i>	196
9.6	Example for retrieval result as result of Step <i>Provide retrieval result data</i>	197
9.7	Example for retrieval result list as result of Step <i>Add current retrieval result to result list</i>	197
9.8	Overview of all permutations for the right PBS nodes <i>c</i> , <i>c.a</i> , <i>c.m</i> .	201
9.9	Details of all permutations for the right PBS nodes <i>c</i> , <i>c.a</i> , <i>c.m</i> ac- cording to process illustrated by Figures 9.21 and 9.22	202
9.10	Permutation of right PBS nodes for Salary Statement Application example: Overview	209
9.11	Example for Step <i>Exchange current position of right PBS node in set and position from given input and check if current position in right PBS node set is lower than number of right PBS nodes</i> for the first iteration of the containing process	210
9.12	Example for Step <i>Exchange current position of right PBS node in set and position from given input and check if current position in right PBS node set is lower than number of right PBS nodes</i> for the second iteration of the containing process	211
9.13	Example for Step <i>Exchange right PBS node value at current position and value at given position</i> executed within the first iteration of the containing process	212
9.14	Example for Step <i>Exchange right PBS node value at current position and value at given position</i> executed within the second iteration of the containing process	212
9.15	PCP graph node mappings for running examples	214
9.16	Example for Step <i>Reverse previous exchange of right PBS node value at current position and value at given position</i> for first iteration . . .	214

9.17	Example for Step <i>Reverse previous exchange of right PBS node value at current position and value at given position</i> for second iteration	215
9.18	Example for a newly created instance of the <i>PcpGraphMapping</i> class with initial attribute values	221
9.19	Example for instance of the <i>PcpGraphMapping</i> class: Provided attribute values as result of Step 2	222
9.20	Example for a newly created instance of the <i>MappingLineItem</i> class with initial attribute values	223
9.21	Example for the instance of the <i>PcpGraphMapping</i> class from the preceding process Step 2 with an added newly created <i>MappingLineItem</i> instance	224
9.22	Example for added PS nodes to instance of the <i>MappingLineItem</i> class	225
9.23	First instance of the <i>PcpGraphMapping</i> class with its two assigned mapping line items	226
9.24	Example for added PBS nodes to the first instance of the <i>MappingLineItem</i> class of the first <i>PcpGraphMapping</i> instance	226
9.25	Example for added PBS nodes to the second instance of the <i>MappingLineItem</i> class of the first <i>PcpGraphMapping</i> instance	227
9.26	Second instance of the <i>PcpGraphMapping</i> class with its two assigned mapping line items	227
9.27	Example for added PBS nodes to the first instance of the <i>MappingLineItem</i> class of the second <i>PcpGraphMapping</i> instance	228
9.28	Example for added PBS nodes to the second instance of the <i>MappingLineItem</i> class of the second <i>PcpGraphMapping</i> instance	228
10.1	SAP System Messages Viewer Application: Resulting model elements as result of Step 11	272
10.2	List of elements in the abstract SAP System Messages Viewer Application solution model and the domain-specific counterpart that are copied to the solution model	276
10.3	List of elements in the abstract SAP System Messages Viewer Application solution model and the domain-specific counterpart that is not copied to the solution model	277

10.4	SAP System Messages Viewer Application: Model elements of type <i>attribute</i> or <i>method</i> and the @New_element@ prefix with derived container classes in the domain-specific solution model	278
10.5	SAP System Messages Viewer Application: Model elements of type <i>class</i> and the @New_element@ prefix copied from the abstract solution model to the domain-specific solution model	278
10.6	SAP System Messages Viewer Application: Model elements of type <i>generalization</i> and the @New_element@ prefix instantiated from the abstract solution model to the domain-specific solution model	279
11.1	Type system of EOL	314
11.2	Excerpt of operations of EOL types used for ProPMan Tool	315
A.1	SAP System Messages Viewer Application : Conditions and fitting problem-context model elements for transformation to problem-context pattern	468
A.2	SAP System Messages Viewer Application: Fitting problem-context model elements renamed in problem-context pattern	469
C.1	Result of Step 2 for each attribute/method in the SAP System Messages Viewer Application abstract retrieval request solution model . .	494
C.2	New generalization elements in the SAP System Messages Viewer Application abstract solution model as result of Step 5	496
C.3	Salary Statement Application retrieval result solution pattern: New attributes and methods and their assigned container elements as result of Step 1 - to be instantiated to the retrieval request abstract solution of the SAP System Messages Viewer Application	499
C.4	Salary Statement Application: Extract from contents of the Pcp2SpTrafoTrace model with source and target elements as well as result of Step 2 . . .	501
C.5	Result of Step 3 for each container class in the Salary Statement Application retrieval result problem-context pattern	502
C.6	Result of Step 4: Appropriate analogous classes related to the supplier attribute or method in the SAP System Messages Viewer Application abstract retrieval request solution model	503

C.7	Result of Step 6: SAP System Messages Application retrieval request problem-context pattern container classes copied with EXISTING prefix to abstract solution model	505
C.8	Result of Step 8: Salary Statement Application retrieval result solution pattern container classes copied with NEW prefix to SAP System Messages Application abstract solution model	507
C.9	Sp2SmTrafoTrace model as result of Step 10 for all relevant attributes and methods	508
C.10	Result of Steps 9 and 11 for each attribute/method in the SAP System Messages Viewer Application abstract retrieval request solution model	509
C.11	Extract from Sp2SmTrafoTrace model as result of Step 12 for all relevant container classes	510
C.12	First iteration result of Step 4 for each class in the Salary Statement Application retrieval result solution pattern that has the 'general' role in the new generalization elements	517
C.13	Second iteration result of Step 4 for each class in the Salary Statement Application retrieval result solution pattern that has the 'specific' role in the new generalization elements	518
C.14	SAP System Messages Viewer Application: List of model elements within the domain-specific problem-context model	535
C.15	Result of Step 2 <i>Determine left retrieval request cross-domain model element from Pcm2PcpTrafoTrace model</i> for each model element in the domain-specific problem-context model of the SAP System Messages Viewer Application that has an abstract counterpart in the problem-context pattern	537
C.16	List of <i>remaining existing</i> model elements in the domain-specific problem-context model of the SAP System Messages Viewer Application without an abstract counterpart in the problem-context pattern	538
C.17	Result of Step 3 for each model element in the domain-specific problem-context model of the SAP System Messages Viewer Application: List of elements in the abstract solution model and their prefix value . . .	539

C.18	Result of Step 4 for each model element in the domain-specific problem-context model of the SAP System Messages Viewer Application: List of elements in the abstract solution model and the domain-specific counterpart that is copied to the solution model	541
C.19	Result of Step 5 for each model element in the domain-specific problem-context model of the SAP System Messages Viewer Application: List of elements in the abstract solution model and the domain-specific counterpart that is not copied to the solution model	542
C.20	List of <i>remaining existing</i> model elements in the domain-specific problem-context model of the SAP System Messages Viewer Application copied to the solution model without any changes	544
C.21	Model elements with the @New_element@ prefix in the SAP System Messages Viewer Application abstract solution model as result of Step 1549	
C.22	SAP System Messages Viewer Application: Model elements of type <i>attribute</i> or <i>method</i> and the @New_element@ prefix copied from the abstract solution model to the domain-specific solution model as result of Step 3	551
C.23	SAP System Messages Viewer Application: Model elements of type <i>attribute</i> or <i>method</i> and the @New_element@ prefix with container classes assigned from the abstract solution model to the domain-specific solution model as result of Step 5	552
C.24	SAP System Messages Viewer Application: Model elements of type <i>attribute</i> or <i>method</i> and the @New_element@ prefix with container classes having the @Existing_element@ prefix in the abstract solution model	553
C.25	SAP System Messages Viewer Application: Model elements of type <i>attribute</i> or <i>method</i> and the @New_element@ prefix with derived container classes in the domain-specific solution model	554
C.26	SAP System Messages Viewer Application: Model elements of type <i>class</i> and the @New_element@ prefix copied from the abstract solution model to the domain-specific solution model as result of Step 8	555

C.27 SAP System Messages Viewer Application: Model elements of type <i>generalization</i> copied to the domain-specific solution model as result of Step 9	556
C.28 SAP System Messages Viewer Application: Model elements of type <i>generalization</i> and the @New_element@ prefix copied from the abstract solution model to the domain-specific solution model	559
C.29 SAP System Messages Viewer Application: Model elements of type <i>generalization</i> and the @New_element@ prefix copied from the abstract solution model to the domain-specific solution model and used to determine a domain-specific class	561
C.30 SAP System Messages Viewer Application: Model elements of type <i>generalization</i> copied to the domain-specific solution model with determined domain-specific classes assigned as result of Step 5	561
C.31 SAP System Messages Viewer Application: Model elements of type <i>generalization</i> copied to the domain-specific solution model with determined domain-specific classes assigned as result of Step 6	562

Abstract

In this thesis, we present an approach to enable a reuse of problem solutions across expert domains. We focus on analysis and design patterns, for which we provide a methodology to generate and retrieve these kinds of patterns by tool-supported processes. A guiding principle is *problem orientation*: The problem that needs to be solved is always expressed within a dedicated model that contains this problem in the relevant context. This kind of model is then used to derive the search criterion for retrieving appropriate solutions. The basic idea is that business analysts or software engineers can remain on their domain-specific expert level, such as e.g. a specific business or technical domain, whilst they create *cross-domain* artifacts to allow the retrieval of a more general solution that is applicable to more than one expert domain. To realize our methodology by tool support, we use model-to-model transformations and comparisons. We illustrate the applicability of our approach by a case study for analysis patterns in the accounting and sales business domains as well as by a case study for design patterns in the human resources business and a technical domain. Furthermore, we provide a method to extend existing analysis and design patterns in order to make them accessible by our cross-domain pattern retrieval method.

Analysis patterns, design patterns, problem derivation, model abstraction, cross-domain documentation, problem-context patterns, solution patterns, UML models, domain-specific UML models, problem-bearing models, graph transformation, graph matching, model to model transformation, model to model comparison, software pattern search, software pattern retrieval

Zusammenfassung

In dieser Arbeit präsentieren wir einen Ansatz zur fachgebietsübergreifenden Wiederverwendung von Problemlösungen. Wir fokussieren Analyse- und Entwurfsmuster, für die wir eine Methodologie zur Verfügung stellen, die die Erzeugung und Suche solcher Muster über werkzeuggestützte Prozesse erlaubt. Ein wesentliches Prinzip dabei ist die *Problemorientierung*: Ein zu lösendes Problem wird stets in einem eigens vorgesehenen Modell ausgedrückt, das dieses Problem im relevanten Kontext enthält. Dieses Modell wird dann für die Ableitung des Kriteriums für die Lösungssuche benutzt. Die grundlegende Idee dabei besteht darin, dass Business Analysten oder Softwareingenieure einerseits gedanklich auf der Ebene ihres Fachgebietes, wie beispielsweise ein betriebswirtschaftliches oder technisches, bleiben können, während sie gleichermaßen *fachgebietübergreifende* Artefakte erzeugen, die eine Suche nach allgemeingültigen Lösungen ermöglichen, welche sich auf mehr als ein Fachgebiet anwenden lassen. Zur Realisierung unserer Methodik mit einem Softwarewerkzeug nutzen wir Modell-zu-Modell-Transformationen und Modell-zu-Modell-Vergleiche. Wir veranschaulichen unseren Ansatz sowohl durch eine Fallstudie für Analysemuster im Bereich des Rechnungswesens und des Verkaufs im Bereich von betriebswirtschaftlichen Domänen, als auch durch ein Fallbeispiel für Entwurfsmuster im Bereich der Personalwirtschaft und einer technischen Domäne. Desweiteren stellen wir eine Methode zur Erweiterung existierender Analyse- und Entwurfsmuster bereit, um diese für unsere Suchemethode zugreifbar zu machen.

Analysemuster, Entwurfsmuster, Problemableitung, Modellabstraktion, domänenübergreifende Dokumentation, Problem-Kontext Muster, Lösungsmuster, UML Modelle, domänenspezifische UML Modelle, Problemtragende Modelle, Graphentransformation, Graphensuche, Modell-zu-Modell-Transformation, Modell-zu-Modell-Vergleich, Software Pattern Suche, Software Pattern Retrieval

Acknowledgements

First, I am sincerely and heartily grateful to my advisor, Prof. Maritta Heisel, who did a splendid job in supervising my work. Especially with respect to the fact that in my role as an external doctoral candidate, I am sure that this dissertation would not have been possible without her continuous support, understanding and encouragement. Maritta worked with me on most of the topics, had good ideas for improvement and always provided constructive and early feedback.

Many thanks also to Eduardo B. Fernandez, who was already involved in early phase of our research, for his thorough and valuable feedback. It is a great honor for me to have Eduardo as second assessor.

Especially concerning valuable feedback, I would like to express a lot of thanks to Marittas working group, namely to Azadeh Alebrahim, Kristian Beckers, Isabel Côté, Stephan Fassbender, Nazila Golmohammadi, Denis Hatebur, Rene Meis, Holger Schmidt, Nelufar Ulfat-Bunyadi, Ina Wentzlaff as well as to the external doctoral candidates Thomas Frese and Thomas Rottke, who gave me very valuable feedback and ideas in the numerous review sessions (aka “Promi workshops”) at the University Duisburg-Essen. I would also like to thank Klaus Meffert for our successful joint research work and a number of inspiring discussions. Furthermore, many thanks to Thomas Santen for his valuable input especially on the ProPEXt chapter.

Special thanks to my wife, Sandra, and my son, Linus, for their extraordinary patience and support during my work on this thesis throughout the years. Without their encouragement and “back-office” services this project would never have been possible.