

Contents

I Problem Statement

1	Introduction	19
1.1	Motivation	19
1.2	Contribution of our Work	23
1.3	Overview	24
2	Problem Statement	25
2.1	Use Cases	25
2.1.1	Stud.IP	25
2.1.1.1	System Overview	26
2.1.1.2	A Spotlight on <i>Stud.IP</i> 's Notification System	27
2.1.1.3	Software-Technological Analysis	27
2.1.1.4	Essential Cognitions	35
2.1.2	InfoWiss	38
2.1.2.1	System Overview	38
2.1.2.2	A Spotlight on <i>InfoWiss</i> ' Event-Handling	39
2.1.2.3	Software-Technological Analysis	40
2.1.2.4	Essential Cognitions	44
2.1.3	Further Use Cases in Brief	45
2.2	Inferred Requirements	46
2.2.1	Semantic Requirements	47
2.2.2	Technical Requirements	48
2.2.3	Software-Engineering Requirements	49
2.3	Summary	50
3	Existing Approaches	51
3.1	Classification Scheme for Publish/Subscribe Systems	51

3.2	Overview of Existing Approaches	57
3.2.1	Research Projects	57
3.2.1.1	Applications	57
3.2.1.2	Meta-Modelling and Model Transformation	58
3.2.1.3	Subscription Specification	58
3.2.1.4	Event Matching	60
3.2.1.5	Matching Optimization	61
3.2.1.6	Event Detection	61
3.2.1.7	Data Storage	63
3.2.1.8	Subscription and Notification Storage	63
3.2.1.9	Publication Interface	63
3.2.1.10	Notification Delivery	63
3.2.1.11	Security	64
3.2.2	Commercial Systems	64
3.2.3	Comparison Against Requirements	65
3.3	Summary	66

II The Non-Invasive Approach

4	Solution Overview	69
4.1	Motivation for the Generative Approach	69
4.2	The Declarative-Generative Approach	72
4.2.1	Central Data Storage	73
4.2.2	Information System's Data Model as a Basis	73
4.2.3	Generic Meta-Model of Arbitrary Data Models	73
4.2.4	Enhancement of the Meta-Model by a Generic Event-Handling Meta-Model	74
4.2.5	Enrichment of the Data Model	74
4.2.6	Transformation of the Enriched Data Model into Event-Handling Code	75
4.3	Genericity and Dimensions of Abstraction	75
4.4	Details on the Architecture's Components	77
4.4.1	Models	77
4.4.2	Event-Handling Data Access Layer	78
4.4.3	Event Detection	78
4.4.4	Scenario Monitoring	78
4.4.5	Publish/Subscribe Component	79

4.4.6	Optimizer/Generator	79
4.4.7	Legacy Application	79
4.5	Dynamic View on the System's Lifecycle	79
4.5.1	Designtime Lifecycle	80
4.5.2	Runtime Lifecycle	81
4.6	Summary	82
5	Data and Event Models	83
5.1	Representation of Data Model and System Instance	83
5.1.1	Data Model	83
5.1.2	System Instance	85
5.2	Event-Handling Constructs and Formal Event Model	87
5.2.1	Subscribers	88
5.2.2	Subscribables	88
5.2.3	Observed Attributes	88
5.2.4	Implicit Subscriptions	89
5.2.5	Event-Propagating Associations	90
5.2.6	Explicit Subscriptions	91
5.3	Overlays	91
5.4	Graph Representations for Data Model, Overlay and Instance	92
5.4.1	Graph Representation of Data Model	93
5.4.2	Graph Representation of System Instance	93
5.4.3	Graph Representation of Overlays	94
5.4.4	Path Descriptions	94
5.4.5	Path Instances	95
5.5	Interpretation	97
5.5.1	Definitions	97
5.5.2	Subscribers to Inform about Updates	99
5.5.2.1	Implicit Subscribers	99
5.5.2.2	Explicit Subscribers	100
5.6	Real-Life Example	102
5.6.1	Sample Scenario	102
5.6.2	Formal Representation	104
5.6.3	Interpretation of the Event-Handling Specification	106
5.7	Discussion on Design Decisions	110
5.7.1	Handling Inheritance	110
5.7.2	Non-Transitive Implicit Subscriptions	110
5.7.3	Benefit of Overlays	111

5.8	Summary	112
6	Generic Trigger Generation	113
6.1	Overview of the Generation Algorithm	113
6.2	Generation Algorithm in Detail	114
6.3	Sample Generation Process	120
6.4	Qualitative Analysis	122
6.4.1	Correctness	122
6.4.2	Avoidance of Event Cascades	125
6.4.3	Discussion on Cycles	125
6.5	Summary	126
7	Generic Optimization Strategy	127
7.1	Optimization Idea	127
7.1.1	Computation of Matching Paths	128
7.1.2	Optimization by Precomputing Matching Paths	128
7.2	Strategy for the Usage of Event Propagation Indices	129
7.3	Cost Model	131
7.3.1	Costs of Event Propagation Indices	132
7.3.2	Costs of Paths	132
7.4	Probabilities	133
7.5	Expected Costs of Path Descriptions	135
7.6	Probability Models	136
7.6.1	Heuristic Probability Model	136
7.6.2	Design-time Probability Model	136
7.6.3	Empirical Probability Model	137
7.6.4	Comparison of Probability Models	137
7.7	Cost Models	138
7.7.1	Heuristic Cost Model	138
7.7.2	DBCOSTModel	142
7.7.3	Empirical Cost Model	142
7.8	Sample Comparison of Indexing Alternatives	143
7.8.1	Scenario 1	145
7.8.2	Scenario 2	145
7.8.3	Scenario 3	146
7.8.4	Scenario 4	147
7.8.5	Empirical Validation of the Results	148
7.8.6	Sophisticated Index Maintenance Algorithms	149

7.8.7	Results	150
7.9	Determining the Optimal Index Usage	150
7.10	Estimated Behaviour Depending on Path Length	152
7.10.1	Highly Connected Object Graphs	153
7.10.2	Sparsely Connected Object Graphs	154
7.10.3	Consequences	155
7.11	Summary	155

III The Model Driven Implementation for Active Databases

8	Technology Selection	159
8.1	Motivation for Technology Selection	159
8.2	Model Driven Architecture	160
8.2.1	MDA in General	160
8.2.1.1	The MDA Development Life Cycle	160
8.2.1.2	Automation of the Transformation Steps	162
8.2.1.3	Building Blocks of MDA	163
8.2.2	AndroMDA	167
8.3	Active Database Technology	168
8.4	Materialized Views	170
8.5	Summary	172
9	Reference Architecture and Implementation	173
9.1	Substantiating the Abstraction Layers	173
9.1.1	UML Profiles and MDA	174
9.1.2	Transforming Models into Triggers	175
9.1.2.1	Using Materialized Views for Optimization Purposes	175
9.1.2.2	Cost Model for Relational Databases	176
9.2	System Architecture Components in Detail	176
9.2.1	Event-Handling Data Access Layer	176
9.2.2	Event Processing using Triggers	178
9.2.2.1	Database Tables to Monitor	178
9.2.2.2	Determining Relevant Subscribers using Triggers . .	184
9.2.2.3	Example Without Optimization	185
9.2.2.4	Processing Notification Entries	190
9.2.2.5	Optimization	191
9.2.2.6	Inheritance Revisited	199

9.3 Summary	200
10 From UML Models to Optimized Triggers	201
10.1 UML Profile	201
10.1.1 Object-Relational Mapping	202
10.1.2 Event-Handling Profile Elements	204
10.2 Model Template	206
10.3 Metafacades and Transformation Helper Classes	206
10.3.1 AndroMDA Base Metafacades	207
10.3.2 Event-Handling Metafacades	210
10.3.3 Transformation Helper Classes	212
10.4 MDA Transformations	215
10.5 Correctness	217
10.6 Comprehensive Sample Model	218
10.7 Summary	231
IV Résumé	
11 Critical Evaluation	235
11.1 Evaluation of the Prototypic Implementation	236
11.1.1 Software Quality According to ISO 9126	236
11.1.1.1 Functionality	236
11.1.1.2 Reliability	237
11.1.1.3 Usability	237
11.1.1.4 Efficiency	238
11.1.1.5 Maintainability and Portability	238
11.1.2 A Detailed View on Performance and Scalability	239
11.1.2.1 Performance and Scalability in General	239
11.1.2.2 Benefit of Optimization	246
11.1.2.3 Real-Life Scenario	249
11.1.2.4 Overall Performance Results	253
11.1.3 Applicability in Distributed Environments	255
11.1.4 Technological Alternatives	256
11.1.4.1 Alternatives to MDA	257
11.1.4.2 Alternatives to Database Triggers	257
11.1.4.3 Alternatives to Materialized Views	258

11.2	Rating of the Event-Handling Concept	258
11.2.1	Fulfilment of Basic Postulated Requirements	259
11.2.1.1	Functional Adequacy	259
11.2.1.2	Technical / Architectural Adequacy	262
11.2.1.3	Software-Technological Adequacy	263
11.2.2	Technological and Conceptual Alternatives	264
11.3	Review of the Generative Approach in General	265
11.3.1	Assets and Drawbacks	265
11.3.2	Factors for the Successful Usage of the Generative Paradigm	266
11.4	Summary	268
12	Contribution and Open Ends	269
12.1	Contribution of Our Work	269
12.2	Open Ends	271
12.2.1	Technological Improvements	272
12.2.2	Conceptual Improvements	273
12.2.3	Visionary Ideas	274
12.3	Summary	275
A	Bibliography	277
B	List of Figures	293
C	Listings	297