Entwurf einer Richtlinien-Datenbank für ein Umweltinformationssystem mit dem OPAQUE-Modell

Diplomarbeit
von
Thomas Zurek

Betreuer:
Dipl.-Inform. Stefan M. Lang

Lehrstuhl für Systeme der Informationsverwaltung
Prof. Dr. Peter C. Lockemann

Institut für
Programmstrukturen und Datenorganisation
Fakultät für Informatik
Universität Karlsruhe
76128 Karlsruhe

27. Dezember 1993

Diese Arbeit wurde von mir selbständig angefertigt. Alle verwendeten Literaturstellen und sonstigen Quellen sind im Literaturverzeichnis aufgeführt. Andere Hilfsmittel habe ich nicht verwendet. Ich versichere dies ausdrücklich mit der nachstehenden Unterschrift.

Karlsruhe, 27. Dezember 1993

Für meine Eltern und meine Schwester.

Kurzfassung

Umweltinformationssysteme (UIS) sind eines von vielen Mitteln, die komplexen Beziehungen der vom Menschen geschaffenen Technosphäre und der natürlichen Umwelt zu beobachten und zu kontrollieren. Ziel ist es dabei, die Informationsflüsse aus unterschiedlichen Quellen für die verschiedenen Interessenten aus Politik, Verwaltung, Wirtschaft etc. schnell und anwenderfreundlich aufzubereiten und weiterzugeben. Diese Komplexität spiegelt sich unmittelbar in sehr komplexen Systemarchitekturen bestehender UIS wider, die sich aus zahlreichen Komponenten zusammensetzen.

In dieser Arbeit wird eine Komponente eines UIS entworfen, die dazu dienen soll, die in zahlreichen Gesetzen und anderen Richtlinien in Form von Grenz-, Leit-, Richtwerten u.ä. festgehaltenen Regelungen in einer Datenbank zu halten und die Informationen anwenderspezifisch aufzubereiten und zu verarbeiten.

Als logisches Datenmodell für den Entwurf wird das Kontextmodell OPAQUE gewählt. Es soll einen den oben genannten Anforderungen genügenden Datenbankentwurf unterstützen. OPAQUE verinnerlicht einerseits die Vorteile objektorientierter Modellierungsansätze und andererseits Möglichkeiten, flexibel auf kurz- und langfristige Änderungen und Entwicklungen zu reagieren.

In einem statischen Entwurf wird zunächst die komplexe Struktur der Diskurswelt modelliert. Dazu werden Gemeinsamkeiten innerhalb der Richtlinien identifiziert und in einer Objektmodellierung auf Basis der Object Modeling Technique dargestellt. Diese Modellierung wird danach in die OPAQUE-EM Notation umgesetzt.

Auf Grundlage der statischen Modellierung wird ein Entwurf entwickelt, der das dynamische Verhalten der Komponenten im Objektmodell beschreibt und der sich geradlinig und beinahe mechanisch aus dem Modell ableiten läßt. Diese Geradlinigkeit hat ihre Ursache in einer sehr einfachen, systemnahen und prozedural ausgelegten Konzeption, die die Verantwortung im Umgang mit dem komplexen Anwendungsszenario an den Benutzer weiterreicht. Dies führt zu einer zwar umfangreichen, aber einfachen Implementierung und einem Anfragemodell, das den Bedürfnissen unerfahrener Benutzer allerdings nicht gerecht wird.

In einem zweiten dynamischen Entwurf wird den Nachteilen des ersten begegnet, indem einerseits ein deskriptiver Ansatz für das Anfragemodell gewählt wird und andererseits das Suchmodells in einer Weise konzipiert wird, daß es Veränderungen und Erweiterungen am Datenbasisschema nicht nur toleriert, sondern automatisch in den Suchprozeß integriert, falls die Erweiterungen bestimmten Mindestanforderungen genügen.

Die Arbeit schließt mit einer Auswertung des entwickelten Entwurfs und einem Vergleich zu einem Entwurf, der vom FZI Karlsruhe auf Grundlage des relationalen Datenmodells erstellt wurde. Bei einer Vernachlässigung der Unterschiede, die aufgrund unterschiedlicher Entwurfsentscheidungen zustande kamen, zeigt sich, daß gerade die Vorteile einer kontextsensitiven Modellierung zu einer deutlich vergrößerten Flexibilität, Anpaßbarkeit und Erweiterbarkeit geführt haben.

Vorwort

Ich möchte an dieser Stelle denjenigen danken, die mich während der sechs Monate unterstützt haben, in denen diese Arbeit entstand. Der Dank gilt ganz besonders Stefan, der immer wieder Teile der Ausarbeitung durchlas, mit mir darüber diskutierte und mir wertvolle Hinweise und Ratschläge gab. Außerdem war es schön, daß Diskussionen und Aktivitäten auch über den Themenkreis dieser Arbeit hinausgingen.

Einen Dank möchte ich auch an Ralf Kramer vom FZI richten, der mir zahlreiche seiner Arbeitspapiere oder Abbildungen zugänglich gemacht hat, so daß ich den Vergleich zu 'seinem' Entwurf ziehen konnte, was diese Arbeit sehr bereichert hat.

Darüberhinaus möchte ich all denen für ihr Verständnis danken, die mich während dieser Zeit in der einen oder anderen Form begleitet haben und ertragen mußten, daß ich manchmal gestreßt, schlecht gelaunt und kurz angebunden war. Das gilt insbesondere für meine Familie und meine Freunde. Vielen Dank für die Geduld, die ihr mit mir hattet.

Karlsruhe, den 27. Dezember 1993

Inhaltsverzeichnis

1 Einführung 15

1.1 Motivation . . . 15

1.2 Aufgabenstellung . . . 16

1.3 Gliederung der Arbeit . . . 16

2 Einführung in die Diskurswelt 19

2.1 Überblick . . . 19

2.2 Zusammenfassung . . . 21

3 Grundlagen 23

3.1 Datenbankentwurf . . . 23

3.1.1 Informationsbedarfsanalyse . . . 24

3.1.2 Konzeptueller Entwurf . . . 25

3.1.3 Logischer Entwurf . . . 26

3.1.4 Datenbankentwurf für OPAQUE . . . 27

3.2 Object-Modeling-Technique . . . 28

3.2.1 Objektorientierung und objektorientierter Entwurf . . . 29

3.2.2 Überblick über OMT . . . 30

3.2.3 Objektmodell . . . 31

3.2.4 Dynamisches Modell . . . 35

3.2.5 Funktionales Modell . . . 37

3.3 OPAQUE . . . 39

3.3.1 Einführender Vergleich: OPAQUE – klassische Objektorientierung . . . 39

3.3.2 Statische OPAQUE-Konzepte . . . 45

3.3.3 Dynamische OPAQUE-Konzepte . . . 52

3.4 OPAQUE-EM . . . 54

3.4.1 Idee einer Entwurfsmethode für OPAQUE 54
3.4.2 Kontextschablonen 55
3.4.3 Kristallschablonen und Klassen 56
3.4.4 Rollen und Kardinalitäten 57
3.4.5 Vererbungsbeziehungen 58
3.5 Transformation von OMT nach OPAQUE-EM 59
3.5.1 Vorgehensweise 59
3.5.2 Transformation von Objektklassen 59
3.5.3 Transformation von Assoziationen 59
3.5.4 Transformation von Aggregationen 60
3.5.5 Transformation von Vererbungsbeziehungen 61

4 Analyse der Diskurswelt 65
4.1 Datenanalyse 65
4.1.1 Problematik nicht einheitlicher Richtlinienstrukturen 66
4.1.2 Vereinheitlichung und Definition der Begriffswelt 67
4.2 Benutzerprofile 71
4.2.1 Beispiele von Benutzern 71
4.2.2 Benutzergruppen 73

5 Statischer Entwurf 77
5.1 Gliederung des Entwurfs 77
5.2 Ein OMT-Objektmodell 77
5.2.1 Richtlinien und Richtlinienregelungen 77
5.2.2 Institutionen, Meßobjekte und Gebietsbereiche 79
5.2.3 Vererbungshierarchie der Richtlinien 80
5.2.4 Richtlinienregelungs-Aggregat 83
5.2.5 Richtlinienwert-Aggregat 87
5.2.6 Benutzer und Anfragen 88
5.3 Ein OPAQUE-EM-Objektmodell 90
5.3.1 Richtlinien und Richtlinienregelungen 90
5.3.2 Institutionen, Meßobjekte und Gebietsbereiche 91
5.3.3 Vererbungshierarchie der Richtlinien 93
5.3.4 Richtlinienregelungs-Aggregat 95

5.3.5 Richtlinienwert-Aggregat 97
5.3.6 Benutzer und Anfragen 99
5.4 Bemerkungen zum statischen Entwurf 99

6 Vorüberlegungen zum dynamischen Entwurf 101
6.1 Erwartete Funktionalität 101
6.2 Möglichkeiten, die Funktionalität umzusetzen 101

7 Ein prozeduraler dynamischer Entwurf 103
7.1 Vorgehensweise 103
7.2 Anfrage- und Suchprimitive 104
7.3 Das Anfragemodell 105
7.3.1 Konzeption 105
7.3.2 Der Anfragealgorithmus 106
7.3.3 Eine Beispielanfrage 107
7.4 Globale Entwurfsebene 108
7.4.1 Motivation 108
7.4.2 Suchoperationen 111
7.4.3 Ausgabeoperationen 116
7.5 Klassen- und Kontextebene 118
7.5.1 Beschreibung der Operationen 118
7.5.2 Zuordnung der Operationen zu den Klassen 119
7.6 Implementierungsebene 119
7.6.1 Die Suchoperation `suche_Gebietsbereiche` 120
7.6.2 Die Ausgabeoperation für `Institution` 122
7.6.3 Die Operation `hat_Name` aus `Gebietsbereich` 122
7.6.4 Die Operation `passt_zu_Stichwort` aus `Gebietsbereich` 123
7.6.5 Die Operation `verbundene_Institutionen` der Klasse `Gebietsbereich` . 123
7.6.6 Die Operation `alle_Oberbereiche` aus `Gebietsbereich` 124
7.7 Bewertung 124

8 Ein deskriptiver dynamischer Entwurf 129
8.1 Zielrichtung und Grundideen des Entwurfs 129
8.2 Vorgehensweise 130

8.3 Das Anfragemodell 131
8.3.1 Konzeption des Anfragemodells 131
8.3.2 Der Anfragealgorithmus 132
8.3.3 Beispiel einer Anfrage 133
8.4 Das Suchmodell 144
8.4.1 Suchalgorithmus 144
8.4.2 Der Struktur- und Wertevergleich 146
8.5 Das Ausgabemodell 153

9 Auswertung des Entwurfs 155

9.1 Flexibilität gegenüber Änderungen am Objektmodell 155
9.1.1 Anfragemodell 157
9.1.2 Suchmodell 160
9.1.3 Ausgabemodell 160
9.1.4 Zusammenfassung 160
9.2 Anpaßbarkeit der Modelle 163
9.2.1 Anfragemodell 163
9.2.2 Suchmodell 163
9.2.3 Ausgabe- und andere Auswertemodelle 166
9.3 Zusammenfassung 169

10 Vergleich mit einem relationalen Entwurf 171

10.1 Ein relationaler Entwurf 171
10.1.1 Zielsetzung 172
10.1.2 Die statische Modellierung 172
10.1.3 Das Anfragemodell 177
10.1.4 Das Suchmodell 178
10.1.5 Das Ausgabemodell 180
10.2 Vergleichende Bewertung 182
10.2.1 Parallelen zum ersten dynamischen Entwurf 182
10.2.2 Eigenschaften des relationalen Entwurfs 183
10.2.3 Erweiterbarkeit 184

11 Zusammenfassung und Ausblick 187

11.1 Zusammenfassung 187
11.2 Ausblick 188

A Glossar **191**

B Struktur der Richtlinien **195**
B.1 Richtlinien zum Meßobjekt Luft 195
B.2 Richtlinien zum Meßobjekt Wasser 197
B.3 Richtlinien zum Meßobjekt Boden 199

C Konsolidierte Struktur der Richtlinien **201**
C.1 Richtlinien zum Meßobjekt Luft 201
C.2 Richtlinien zum Meßobjekt Wasser 203
C.3 Richtlinien zum Meßobjekt Boden 207

D Spezifikationen zum ersten dynamischen Entwurf **209**
D.1 Spezifikationen der Suchoperationen 209
D.2 Zuordnung der Operationen 220

Abbildungsverzeichnis

3.1 Vorgehensweise beim Datenbankentwurf 25

3.2 Vorgehensweise beim Datenbankentwurf mit OPAQUE 28

3.3 Symbole zur Darstellung von Objekten und Objektklassen, sowie deren Instanzen. 32

3.4 Darstellung einer binären Assoziation zwischen Objektklassen 32

3.5 Darstellung einer 3-fach Assoziation zwischen Objektklassen 32

3.6 Beispiel für eine Assoziation 33

3.7 Beispiel für einen Link 33

3.8 Attribute in Assoziationen 33

3.9 Vererbungshierarchie unterschiedlicher Umweltrichtlinien 34

3.10 Beispiel einer Aggregation 34

3.11 Zustandsübergangsdiagramm zum Beispiel 37

3.12 Datenflußdiagramm für Richtlinienwertanfrage 38

3.13 Aggregation von Datenflüssen 38

3.14 Datenflußdiagramm für eine Schadstoffmessung 39

3.15 Beispiel aus dem Umweltgrenzwertszenario in OMT-Modellierung 41

3.16 Objektklasse `Chemischer-Stoff` in OPAQUE 43

3.17 Objekt 'Asbest' der Klasse `Chemischer-Stoff` 43

3.18 Dynamische Erweiterung der Objektbeschreibung durch einen zusätzlichen Kontext 44

3.19 Rohstoff-Kontext wurde aus der Objektbeschreibung entfernt 45

3.20 Beispiel aus dem Umweltszenario in OPAQUE-Modellierung 46

3.21 Definition einer Kontextschablone `Richtlinienwert` in OPAQUE 49

3.22 Kristall – Kontext Anbindungen (1) 50

3.23 Kristall – Kontext Anbindungen (2) 50

3.24 Definition einer Klasse `Chemischer-Stoff` in OPAQUE 51

3.25 Definition einer Klasse in OPAQUE 51

3.26 Kommunikation zwischen zwei OPAQUE-Objekten 52

3.27 OPAQUE-EM als hybride oder als eigenständige Entwurfsmethode 55

3.28 Eine Kontextschablone in OPAQUE-EM 56

3.29 Beispiel einer Kontextschablone `Richtlinie_Ktxt` 56

3.30 Eine Kristallschablone in OPAQUE-EM 57

3.31 Anknüpfung einer Kristallschablone an eine Kontextschablone über eine Rolle . . 57

3.32 Beispiel einer Anknüpfung von Kristall- und Kontextschablonen mit Kardinalitäten 58

3.33 Vererbungsbeziehungen in OPAQUE-EM 58

3.34 Transformation einer OMT-Klasse in eine OPAQUE-EM-Klasse 60

3.35 Transformation einer OMT-Assoziation nach OPAQUE-EM 61

3.36 Transformation einer OMT-Aggregation nach OPAQUE-EM 62

3.37 Transformation einer Vererbungsbeziehung nach OPAQUE-EM 63

4.1 Tabellenausschnitt aus den EG-Luftqualitätsnormen 66

4.2 Zuordnung von Benutzern zu Benutzergruppen 75

5.1 OMT-Objektmodell – Teil 1: Richtlinien und Richtlinienregelungen 79

5.2 OMT-Objektmodell – Teil 2: Institutionen, Meßobjekte und Gebietsbereiche . . 81

5.3 OMT-Objektmodell – Teil 3: Die Richtlinien-Vererbungshierarchie 82

5.4 OMT-Objektmodell – Teil 4: Das Richtlinienregelungs-Aggregat 85

5.5 OMT-Objektmodell – Teil 5: Das Richtlinienwert-Aggregat 88

5.6 OMT-Objektmodell – Teil 6: Benutzer und Anfragen 89

5.7 OPAQUE-EM-Objektmodell – Teil 1: Richtlinien und Richtlinienregelungen . . . 90

5.8 OPAQUE-EM-Objektmodell – Teil 2: Institutionen, Meßobjekte und Gebietsbereiche 92

5.9 OPAQUE-EM-Objektmodell – Teil 3: Die Richtlinien-Vererbungshierarchie . . . 94

5.10 OPAQUE-EM-Objektmodell – Teil 4: Das Richtlinienregelungs-Aggregat 96

5.11 OPAQUE-EM-Objektmodell – Teil 5: Das Richtlinienwert-Aggregat 98

5.12 OPAQUE-EM-Objektmodell – Teil 6: Benutzer und Anfragen 99

7.1 Zusammensetzung einer komplexen Anfrage aus Anfrageprimitiven 105

7.2 Die OPAQUE Objektklasse `Gebietsbereich` 109

7.3 Dezentrale Lösung: Verteilung der Suchoperationen auf die einzelnen Objektklassen 110

7.4 Zentrale Lösung: Sammlung aller Suchoperationen in `Anfrage`, also einer einzigen Objektklasse 110

7.5 Operation `suche_Gebietsbereiche` 113

7.6 Operation suche_Richtlinien . . . 114
7.7 Operation suche_EG-Richtlinien . . . 114
7.8 Operation suche_Nicht-EG-Richtlinien . . . 115
7.9 Operation suche_Werte . . . 115
7.10 Operation Ausgabe allgemein für eine beliebige Objektklasse . . . 116
7.11 Operation Ausgabe_Ergebnismenge . . . 117
7.12 Datenflüsse, die in einem Gebietsbereich entspringen . . . 120
7.13 Code der Suchoperation suche_Gebietsbereiche . . . 121
7.14 Code der Operation Ausgabe für eine Institution . . . 122
7.15 Code der Operation hat_Name . . . 123
7.16 Code der Operation passt_zu_Stichwort . . . 123
7.17 Code der Operation verbundene_Institutionen . . . 125
7.18 Code der Operation alle_Oberbereiche . . . 126

8.1 Ein gültiges Anfrageskelett . . . 134
8.2 Ein ungültiges Anfrageskelett . . . 134
8.3 Aufbau einer Anfrage (1) . . . 136
8.4 Aufbau einer Anfrage (2) . . . 136
8.5 Aufbau einer Anfrage (3) . . . 137
8.6 Aufbau einer Anfrage (4) . . . 137
8.7 Aufbau einer Anfrage (5) . . . 138
8.8 Aufbau einer Anfrage (6) . . . 139
8.9 Aufbau einer Anfrage (7) . . . 140
8.10 Aufbau einer Anfrage (8) . . . 141
8.11 Erstes Anfrageskelett der Beispielanfrage . . . 142
8.12 Zweites Anfrageskelett der Beispielanfrage . . . 143
8.13 Eine Implementierung des Suchalgorithmus'. . . . 145
8.14 Objektstruktur eines Anfrageskeletts . . . 146
8.15 Objektstruktur einer Datenbasis . . . 147
8.16 Eine Nachricht msg(...) an einen Kristall mit einer Umgebung von fünf Kontexten 148
8.17 Phase 1 des Struktur- Wertevergleichs . . . 150
8.18 Phase 2 des Struktur- Wertevergleichs . . . 150
8.19 Phase 3 des Struktur- Wertevergleichs . . . 150
8.20 Ein Anfrageskelett mit Zyklus . . . 151

8.21 Auflösung des Zyklus' 152
8.22 Einzelergebnisse eines Struktur- und Wertevergleichs für ein einfaches Anfrageskelett 153

9.1 Richtlinie erweitert um die Kontextschablone Genehmigungsverfahren_Ktxt . 156
9.2 Erweiterung des Objektmodells zur Aufnahme von Meßdaten 158
9.3 Erweiterung des Objektmodells zur Anbindung von Informationen über chemische Stoffe 159
9.4 Auswirkung einer Erweiterung auf die Anfragesituation aus Abb. 7.1 159
9.5 Auswirkungen einer Objektmodellerweiterung auf das Suchmodell 161
9.6 Auswirkungen einer Objektmodellerweiterung auf das Ausgabemodell 162
9.7 Ein Anfrageskelett in OMT-Notation 164
9.8 Ein strenger Attributvergleich (match1) und ein toleranter (match2) 165
9.9 Benutzerdefinierte match-Operation an einem Anfrageskelett 166
9.10 Ein Exaktwertobjekt erweitert um einen Kontext der Schablone Umrechnung_Ktxt 168

10.1 Eine Entity-Relationship-Modellierung 173
10.2 Das OMT-Objektmodell in vereinfachter Form 174
10.3 Maske zur Suche nach Immissions-Richtlinienwerten 177
10.4 Eine Werteliste für das Suchattribut Parameter 178
10.5 Ergebnismaske der Immissionssuche 181
10.6 Erweiterung des Entity-Relationship-Diagramms 184
10.7 Die Relation Genehmigungsverfahren 184

D.1 Operation suche_Gebietsbereiche 210
D.2 Operation suche_Institutionen 210
D.3 Operation suche_Meßobjekte 211
D.4 Operation suche_Richtlinien 211
D.5 Operation suche_Richtlinienregelungen 212
D.6 Operation suche_EG-Richtlinien 212
D.7 Operation suche_Nicht-EG-Richtlinien 213
D.8 Operation suche_Amtsblattreferenzen 213
D.9 Operation suche_Veröffentlichungen 214
D.10 Operation suche_Parameter 215
D.11 Operation suche_Richtlinienwerte 215
D.12 Operation suche_Emissionsquellen 216

D.13 Operation `suche_Meßmethoden` 216
D.14 Operation `suche_Bemerkungen` 217
D.15 Operation `suche_Bezugszeiträume` 217
D.16 Operation `suche_Richtlinienwertnamen` 218
D.17 Operation `suche_Werte` 218
D.18 Operation `suche_Wertarten` 219
D.19 Operation `suche_Maßeinheiten` 219

Tabellenverzeichnis

3.1 Vergleich zwischen klassischer Objektorientierung und OPAQUE 47

10.1 Begriffsabgleich zwischen dem OMT-Objektmodell und der ER-Modellierung . . 175

D.1 Operationen der Klasse `Gebietsbereich` 221
D.2 Operationen der Klasse `Institution` 221
D.3 Operationen der Klasse `Meßobjekt` 221
D.4 Operationen der Klasse `Richtlinie` 222
D.5 Operationen der Klasse `EG-Richtlinie` 222
D.6 Operationen der Klasse `Nicht-EG-Richtlinie` 222
D.7 Operationen der Klasse `Amtsblattreferenz` 222
D.8 Operationen der Klasse `Veröffentlichung` 223
D.9 Operationen der Klasse `Richtlinienregelung` 224
D.10 Operationen der Klasse `Parameter` 224
D.11 Operationen der Klasse `Richtlinienwert` 225
D.12 Operationen der Klasse `Emissionsquelle` 225
D.13 Operationen der Klasse `Meßmethode` 226
D.14 Operationen der Klasse `Bemerkung` 226
D.15 Operationen der Klasse `Bezugszeitraum` 227
D.16 Operationen der Klasse `Richtlinienwertname` 227
D.17 Operationen der Klasse `Wert` 227
D.18 Operationen der Klasse `Exaktwert` 228
D.19 Operationen der Klasse `Intervallwert` 228
D.20 Operationen der Klasse `Wertart` 228
D.21 Operationen der Klasse `Maßeinheit` 229

Kapitel 1

Einführung

1.1 Motivation

Die vielfältigen Einflüsse aus Industrie, Wirtschaft und von privater Seite auf das Ökosystem haben in den letzten Jahrzehnten stark zugenommen und den Schutz der Umwelt zu einem zentralen Anliegen von Politik und Gesellschaft gemacht. Beinahe täglich führen neue Erkenntnisse, Beschlüsse und Vorfälle zu neuen Diskussionen und zwingen die Verantwortungsträger in Politik und Industrie zu Entscheidungen und zum Handeln. Dabei ist die Komplexität der Beziehungen zwischen der vom Menschen geschaffenen Technosphäre und der natürlichen Umwelt eines der Hauptprobleme.

Man versucht, diese Beziehungen durch eine große Menge an chemisch-physikalischen Indikatoren zu beschreiben. Einfache Beispiele für solche Indikatoren sind die Lufttemperatur oder der pH-Wert von Seen und Flüssen. Die Beobachtung der Indikatoren gibt Aufschluß über den Zustand des Ökosystems und auch darüber, an welchen Stellen Schutzmaßnahmen ergriffen werden müssen bzw. ob bereits durchgeführte Maßnahmen greifen.

Während Wissenschaft und Forschung das Beobachtungsnetz aus derartigen Indikatoren verfeinern und ausbauen und in Zusammenarbeit mit der Politik Umweltschutzrichtlinien ausarbeiten, müssen diese Regelungen von der Industrie und den Bürgern beachtet und umgesetzt und von Behörden, Aufsichtsämtern, Regierungspräsidien überprüft werden. Man erkennt daran, daß Umweltschutz in einer Zusammenarbeit vieler Beteiligter aus fast allen Bereichen des täglichen Lebens besteht.

Umweltinformationssysteme (UIS) sind ein Mittel, die Prozesse des Forschens, Regelns, Beobachtens, Umsetzens, Überprüfens zu unterstützen und den Informationsfluß zwischen den Beteiligten zu koordinieren, um dadurch ein effizientes und schnelles Handeln zu ermöglichen. Dabei gilt es, Informationen aus unterschiedlichen Quellen für die verschiedenen Interessenten aus Politik, Verwaltung, Wirtschaft und Gesellschaft schnell und anwenderfreundlich aufzubereiten und weiterzugeben. Zur Komplexität der Diskurswelt kommt dadurch noch eine informationstechnische hinzu. Letztere schlägt sich unmittelbar in sehr komplexen Systemarchitekturen bestehender UIS nieder, die aus zahlreichen Einzelkomponenten aufgebaut sind [May93].

In dieser Arbeit befassen wir uns mit dem Entwurf einer Komponente eines UIS, die dazu dienen soll, die in zahlreichen Gesetzen und anderen Richtlinien in Form von Grenz-, Leit-, Richtwerten u.ä. festgehaltenen Regelungen in einer Datenbank zu halten und die Informationen anwenderspezifisch aufzubereiten und zu verarbeiten. Die Komplexität der Diskurs- und der Anwenderwelt, sowie die Forderungen nach Flexibilität im Hinblick auf die Einarbeitung von Änderungen und die Adaptivität hinsichtlich einer Integration der zu entwerfenden Komponente in ein UIS stellen besondere Anforderungen an den Datenbankentwurf dar.

Das Kontextmodell OPAQUE scheint in besonderer Weise geeignet zu sein, einen den oben genannten Anforderungen genügenden Datenbankentwurf zu unterstützen. OPAQUE verinnerlicht einerseits die Vorteile objektorientierter Modellierungsansätze und andererseits Möglichkeiten, flexibel auf kurz- und langfristige Änderungen und Entwicklungen zu reagieren. Außerdem verspricht die Konzeption, dynamisches Verhalten von Objekten in Abhängigkeit von ihrer Umgebung, also ihrem *Kontext*, zu bestimmen, eine angemessene Modellierung der unterschiedlichen Ansprüche der unterschiedlichen Anwender zu ermöglichen.

Da am FZI Karlsruhe ein Entwurf für dieselbe Diskurswelt entstand, der auf dem relationalen Datenmodell beruht, können wir einen Vergleich zu dem in dieser Arbeit entwickelten Entwurf ziehen und so Aufschlüsse darüber erhalten, welche Vor- und Nachteile OPAQUE gegenüber einer herkömmlichen Modellierung auf Grundlage anderer logischer Datenmodelle hat.

1.2 Aufgabenstellung

Ziel dieser Arbeit ist ein logischer Datenbankentwurf auf Grundlage des Kontextmodells OPAQUE [Lan93] [Lau93],

- der die Informationen modelliert, die in der Grenzwertbroschüre [LR92] der Landesanstalt für Umweltschutz (LfU) aufgeführt werden,
- der die Diversität der Anwenderwelt berücksichtigt, also dem unterschiedlichen Anfrageverhalten und der unterschiedlichen Antworterwartung der Benutzer Rechnung trägt,
- der sich möglichst einfach in ein UIS einbinden läßt und
- der sich möglichst problemlos erweitern läßt.

Dabei sollen die statischen und dynamischen Konzepte OPAQUEs ausgenützt werden.

Neben der eigentlichen Modellierung soll die Arbeit Aufschlüsse über den Entwurfsprozeß mit OPAQUE liefern. Dabei ist von Interesse, inwieweit herkömmliche objektorientierte Entwurfstechniken, wie OMT, auch für OPAQUE eignen und ob sich die in [Kop93] entwickelte OPAQUE-spezifische Entwurfsmethode OPAQUE-EM bewährt. Aufschluß darüber wird auch ein Vergleich mit dem am FZI Karlsruhe durchgeführten Entwurf auf Grundlage des relationalen Datenmodells [Kra93a], [Kra93b], [HK93] und [KS93] geben.

1.3 Gliederung der Arbeit

Die Arbeit gliedert sich in die folgenden Kapitel:

- Kapitel 2 führt in die Diskurswelt der Umweltrichtlinien und der dort verwendeten Begriffe ein. Dies soll dem Leser helfen, die in den nachfolgenden Kapiteln angeführten Beispiele zu verstehen, die dem Richtlinienszenario entnommen wurden,

- Kapitel 3 gibt eine Einführung in die Grundlagen, die für eine Datenbankentwurf mit OPAQUE benötigt werden. Zunächst werden generelle Konzepte eines Datenbankentwurfs beschrieben, um danach in die Object-Modeling-Technique (OMT), das Kontextmodell OPAQUE, die Entwurfsmethode OPAQUE-EM und die Transformation von Modellierungen von OMT nach OPAQUE-EM einzuführen.

- In Kapitel 4 wird eine Analyse der Diskurswelt durchgeführt. Diese teilt sich auf in eine Analyse der Daten und eine Analyse der Benutzergruppierungen.

- In Kapitel 5 werden die statischen Strukturen der Diskurswelt modelliert. Das bedeutet, daß Objekte und Objektklassen und deren Beziehungen untereinander identifiziert werden.

- Kapitel 6 beschreibt einige Vorüberlegungen für die nachfolgenden Entwürfe der dynamischen Elemente der Richtliniendatenbank: zum einen stellt sich die Frage, welche Funktionalität erwartet wird, zum anderen, wie diese Funktionalität erreicht werden kann. Für letzteres ergeben sich zwei mögliche Ansätze, die in den Kapiteln 7 und 8 ausgearbeitet werden.

- Kapitel 7 beschreibt einen ersten Ansatz für einen dynamischen Entwurf. Dieser Ansatz ist prozedural und wird geradlinig aus einer Betrachtung des in Kapitel 5 entworfenen Objektmodells entwickelt. Bereits während des Entwurfs zeigen sich einige Nachteile dieses Ansatzes, die in einer Bewertung am Ende des Kapitels zusammengefaßt werden.

- Kapitel 8 verfolgt einen zweiten, deskriptiven Ansatz, der zudem gezielt den Schwächen des in Kapitel 7 entwickelten Entwurfs entgegenwirkt.

- In Kapitel 9 werten wir den in Kapitel 8 präsentierten dynamischen Entwurf aus.

- Kapitel 10 stellt den am FZI Karlsruhe entwickelten Entwurf einer Grenz-/Richtwertdatenbank vor, der auf Grundlage des Relationenmodells erstellt wurde. Dieser Entwurf wird dem in dieser Arbeit entwickelten verglichen und bewertet.

- Kapitel 11 gibt eine Zusammenfassung und einen Ausblick.

Kapitel 2

Einführung in die Diskurswelt

2.1 Überblick

In jedem Datenbankentwurf wird ein Ausschnitt aus der realen Welt modelliert. Dieser Ausschnitt wird oft auch als *Miniwelt* oder *Diskurswelt* bezeichnet. Die Diskurswelt, die im folgenden modelliert wird, ist der Bereich von Umweltgrenzwerten. Wir werden in diesem Abschnitt den Leser zunächst einmal informell in diese Miniwelt einführen. Dies dient in erster Linie dazu, die in den späteren Kapiteln beschrieben Entwurfsschritte besser nachvollziehen zu können. Außerdem werden in Kapitel 3 die Grundlagen des Entwurfs bereits mit Hilfe von Beispielen aus der Diskurswelt erläutert.

In der Bundesrepublik Deutschland besteht ein Umweltrecht, das auf einer Fülle von Gesetzen, Verordnungen und Vorschriften beruht, die wir im folgenden der Einfachheit halber alle als *Richtlinien* bezeichnen werden. Diese Richtlinien dienen dazu, die Funktionsfähigkeit der Natur zu erhalten und Gefahren von Mensch und Umwelt abzuwehren. Man versucht dieses Ziel zu erreichen, indem man einerseits verbindliche Grenzwerte und andererseits weitgehend unverbindliche Richt-, Leit-, Prüfwerte u.ä. festlegt, die eine Richtschnur für das Handeln darstellen. Im folgenden werden wir diese Werte zusammenfassend als *Richtlinienwerte* bezeichnen.

Richtlinien, die in der Bundesrepublik Deutschland angewendet werden, sind beispielsweise die EG-Richtlinien, die TA Luft, die WHO-Luftqualitätsleitlinien, VDI-Richtlinien, Bundesgesetze, die Trinkwasserverordnung usw. In dieser Arbeit werden nur Richtlinien berücksichtigt, die Richtlinienwerte für *Schadstoffe* oder *Parameter* innerhalb der *Meßobjekte* Luft, Wasser und Boden angeben.

Schadstoffe sind chemische Substanzen, die für Mensch, Tier und Pflanze schädlich oder auf ein Ökosystem ungünstig wirken. Die Festlegung von Richtlinienwerten für Schadstoffe ist ein wichtiges Instrument, um die Belastung der Umwelt so zu steuern, daß Gefahren für Mensch und Biosphäre weitgehend ausgeschlossen werden können. Beispiele für Schadstoffe sind Blei, Asbest, Quecksilber.

Weiterhin lassen sich Schadstoffe zu *Schadstoffgruppen* zusammenfassen. Bekannte Beispiele für derartige Gruppen sind:

- Schwermetalle (Arsen, Cadmium, Quecksilber, Blei, ...)

- Stickoxide
- Pestizide (Parathion, Dieldrin, ...)
- Fluorchlorkohlenwasserstoffe (Freon, Frigen, ...)

Neben den Werten für Schadstoffe existieren Richtlinienwerte für verschiedene andere physikalische Größen. Meist sind dies Indikatoren, die direkt oder indirekt auf eine Gefährdung schließen lassen, weil sie die chemischen und physikalischen Gleichgewichte innerhalb von Ökosystemen entscheidend beeinflußen oder diese Systeme direkt angreifen. Beispiele dafür sind:

- pH-Wert
- Temperatur[1]
- Färbung
- Transparenz
- gelöster Sauerstoff
- Salmonellen pro Liter[2]
- Fäkalcoliforme Bakterien pro 100 ml
- Darmviren

In den Richtlinien werden diese Größen zusammen mit Schadstoffkonzentrationen und -mengen als *Parameter* bezeichnet. Meistens werden Schadstoffkonzentrationen und -mengen einfach durch den Namen des zugehörigen Schadstoffes bezeichnet, was dazu führt, daß Schadstoffnamen gleichzeitig Namen von Parametern sind. Beispielsweise enthält eine EG-Wasserqualitätsnorm einen Leitwert für den Parameter Mangan von 0.05 mg/l.

Zur Messung eines Parameters in einem Meßobjekt bedarf es einer *Meßmethode*. Da die Meßmethode einen zu messenden Parameterwert beeinflußen kann, werden die Meßmethoden für bestimmte Parameter von den Richtlinien vorgeschrieben. Sie werden dort auch als Analytik, Referenzmethode, Analyse- oder Kontrollverfahren bezeichnet. Beispiele dafür sind

- Gaschromatographie[3]
- Elektrometrie[4]
- Spektroskopische Methoden
- Geschmacksüberprüfung[5]

[1]Relevant z.B. bei der Trinkwassergewinnung oder zum Schutz von Fischgewässern.
[2]Beispielsweise in der EG-Qualitätsnorm für Badegewässer.
[3]Beispielsweise zur Bestimmung der Konzentration von Kohlenwasserstoffverbindungen im Wasser.
[4]Z.B. zur Messung des pH-Werts.
[5]Z.B. zum Nachweis, daß phenolhaltige Verbindungen in Fischgewässern vorhanden sind.

Die Richtlinien lassen sich gemäß ihrer Zielsetzung oder ihres Anwendungsbereiches untergliedern. Beispielsweise lassen sie sich aufgrund einer Spezialisierung der Meßobjekte unterteilen. So gibt es für Wasser und Boden Richtlinien für

- Trink-, Oberflächen-, Grund-, Abwasser, Fließ- und Badegewässer, ...
- Schlamm, Klärschlamm, Kulturböden, ...

Eine andere Möglichkeit der Gliederung ergibt sich für die Meßobjekte Luft und Wasser. Hier unterscheidet man *Emissions-, Immissionsrichtlinienwerte* und *Qualitätsnormen.* Während Immissionsrichtlinienwerte und Qualitätsnormen etwas über den Zustand von Luft und Wasser aussagen, stehen Emissionsrichtlinienwerte in Zusammenhang mit einer Emissionsquelle, wie etwa Feuerungs-, Verbrennungsmotoren-, Schmelzanlagen oder Gießereien.

2.2 Zusammenfassung

Im Umweltrecht existieren zahlreiche *Richtlinien*, die *Richtlinienwerte* für bestimmte *Parameter* enthalten. Diese Richtlinien lassen sich unterteilen in

- Luftrichtlinien
  - Immissionsrichtlinien
  - Emissionsrichtlinien
  - Qualitätsnormen
- Wasserrichtlinien
  - Emissionsrichtlinien
  - Richtlinien für Trinkwasser
  - Richtlinien für Oberflächenwasser
  - Richtlinien für Grundwasser
  - Richtlinien für Abwasser
  - ...
- Bodenrichtlinien
  - Richtlinien für Kulturböden
  - Richtlinien für Schlamm
  - Richtlinien für Klärschlamm
  - ...

Hierbei werden Luft, Wasser und Boden als *Meßobjekte* bezeichnet. Diese Unterteilungen sind nicht notwendigerweise disjunkt und für die Modellierung nur teilweise relevant.

Ein *Parameter* ist entweder die Konzentration oder Menge eines Schadstoffes, von Bakterien, Viren etc. oder eine andere physikalische Größe, die auf eine Gefährdung schließen läßt. Zur Überprüfung eines Richtlinienwertes eines Parameters wird eine *Meßmethode* verwendet, die oft durch die betreffende Richtlinie vorgegeben wird.

Eine detaillierte zusammenfassende Darstellung der hier eingeführten Begriffe findet sich im Glossar (Anhang A), welches dem Leser auch Hilfestellung bei der zunächst verwirrenden Vielfalt von Bezeichnungen geben soll, die in den unterschiedlichen Richtlinien auftreten.

Bereits in dieser kurzen Beschreibung ist erkennbar, daß das Richtlinienszenario komplexe statische Strukturen, eine Vielfalt von Fachbegriffen, aber auch einige Gliederungselemente enthält. Wir werden darauf im Verlaufe des Entwurfs noch näher eingehen. Die oben stehende Einführung soll zunächst genügen, den Leser so weit in die Diskurswelt einzuführen, damit die in den folgenden Kapiteln angeführten Beispiele und Zusammenhänge verstanden werden können.

Kapitel 3

Grundlagen

In diesem Kapitel werden die methodischen Grundlagen für den in den folgenden Kapiteln durchgeführten Datenbankentwurf vorgestellt. Zunächst werden wir die Entwurfsschritte und die dahinterstehenden Ideen darstellen. Dies wird dem Leser gleichzeitig auch die Gliederung der folgenden Kapitel verständlich machen. Daran anschließend werden verschiedene Entwurfswerkzeuge eingeführt.

3.1 Datenbankentwurf

Wir werden in diesem Abschnitt eine Zusammenfassung der wesentlichen Schritte eines Datenbankentwurfs geben. Für eine detailliertere Beschreibung verweisen wir auf die einschlägige Literatur, wie etwa [LS87], [KS91] oder [DR90].

Wie in jedem anderen herkömmlichen Entwurf ist auch hier die Modellierung eines Systems ein zentraler Punkt. Die Erstellung eines Modells löst mehrere Probleme hinsichtlich der Entwicklung von Software bzw. einer Datenbank:

- Ein Modell kann dazu dienen, dem Auftraggeber zu erklären, welche Elemente das System beinhalten und was es einmal leisten wird. Dadurch kann der Auftraggeber noch korrigierend eingreifen, falls einige seiner Wünsche nicht berücksichtigt wurden oder dem Modell falsche Annahmen zugrundeliegen. Da sowohl der Auftraggeber als auch der Entwickler in der Regel mit der Begriffs- und Denkwelt des jeweils anderen nicht vertraut ist, gilt es, das Modell aus möglichst einfachen, allgemein verständlichen Begriffen aufzubauen, damit es von beiden ohne große Probleme verstanden werden kann.
- Ein Modell verschafft den Entwicklern einen Überblick über das System. Da die Implementierung eine Abbildung der realen Welt auf eine rechnerinterpretierbare Welt darstellt und diese Abbildung möglichst geradlinig sein soll, kann ein Modell dazu dienen, die Struktur der realen Welt zu verdeutlichen, um daraus die Struktur der Software bzw. Datenbank zu entwickeln. Darüberhinaus können Fehler und Mängel schon frühzeitig erkannt werden, bevor diese implementiert werden.
- Der wichtigste Zweck eines Modells liegt darin, sich von unnötigen Details und Redundanzen zu befreien und sich auf das Wesentliche zu konzentrieren. Dadurch reduziert sich die

Komplexität des Problems, was den Entwicklern einerseits hilft, die Miniwelt zu verstehen, und andererseits die Entwicklung auf das notwendige Maß beschränkt.

Aus diesen Gründen heraus ergibt sich auch eine Unterteilung des Modellierungs- bzw. Entwurfsprozesses in unterschiedliche Phasen. In jeder Phase steht einer der oben genannten Modellierungsgedanken im Vordergrund und es kann daher angebracht sein, in den unterschiedlichen Entwurfsphasen unterschiedliche Modelle zu verwenden.

Eine Übersicht über die Vorgehensweise bei einem Datenbankentwurf gibt Abb. 3.1:

Der erste Entwurfsschritt ist eine *Informationsbedarfsanalyse*. Hier werden Strukturen und Begriffe der Diskurswelt analysiert und die Anforderungen an das System identifiziert. Das Ergebnis dieser Analyse ist die Grundlage für den sich anschließenden *konzeptuellen Entwurf*. Hier werden mit Hilfe von Ausdrucksmitteln eines *semantischen Datenmodelles* Gegenstände und Beziehungen der Gegenstände untereinander dargestellt. Abhängig vom gewählten Datenmodell erhält man eine semiformale Beschreibung der beobachteten Umwelt, beispielsweise in Form eines ER-Diagramms[1]. Das Ergebnis ist ein *konzeptuelles Schema*, das innerhalb des *logischen Entwurfs* beinahe mechanisch in ein *logisches Schema* überführt werden kann. Das logische Schema wird mit Hilfe eines *logischen Datenmodells* dargestellt. Bekannte Beispiele hierfür sind das relationale, das Netzwerk- oder das hierarchische Modell. Ziel unseres Entwurfs ist ein logisches Schema, das auf dem Kontextmodell OPAQUE beruht. Wir werden auf einen physischen Entwurf verzichten, da wir in dieser Arbeit die Modellierungseigenschaften von OPAQUE näher untersuchen möchten, die physische Abbildung von OPAQUE-Schemata derzeit noch in der Entwicklung ist und da ein physischer Entwurf keine logischen Veränderungen mehr mit sich bringt, sondern 'nur' die Leistungsmerkmale einer Datenbankanwendung durch effiziente Speicherabbildungen, Indexstrukturen, Hashverfahren etc. verbessert.

Abb. 3.1 zeigt auf der linken Seite die Entwurfsschritte, während sich auf der rechten Seite die Namen von Datenmodellen befinden, die der entsprechenden Entwurfsphase zuzuordnen sind. OMT, OPAQUE-EM und OPAQUE werden später noch vorgestellt.

Wir werden nun die für uns zentralen drei Entwurfsschritte näher erläutern.

3.1.1 Informationsbedarfsanalyse

In Form einer Informationsbedarfsanalyse versucht man, sich einen Überblick über die zunächst unstrukturierte Diskurswelt zu verschaffen.

Dazu muß man die Diskurswelt, die man modellieren möchte, zuerst einmal vom restlichen Teil der Welt abgrenzen. In der Regel ist eine solche Eingrenzung unproblematisch, da meistens organisatorische Strukturen (wie etwa eine einzelne Abteilung, eine Organisation, ein Konzern etc.) oder gar eine räumliche Beschränkung (z.B. ein Staat, ein Bundesland, eine Ländergemeinschaft) bestehen, so daß sich dadurch bereits eine natürliche Eingrenzung ergibt. Selbstverständlich interessieren wir uns nicht für alle Aspekte dieses Weltausschnitts, sondern ganz gezielt für einige wenige. So werden wir uns bei der Modellierung des Umweltrichtlinienszenarios nicht für chemische Reaktionen der Schadstoffe, ungefährliche Stoffverbindungen oder Produkte interessieren, die Schadstoffe enthalten, sondern nur für die in den Richtlinien enthaltenen Bestimmungen über die Richtlinienwerte der dort genannten Stoffe. Diese Einschränkung mag

[1] ER = Entity-Relationship

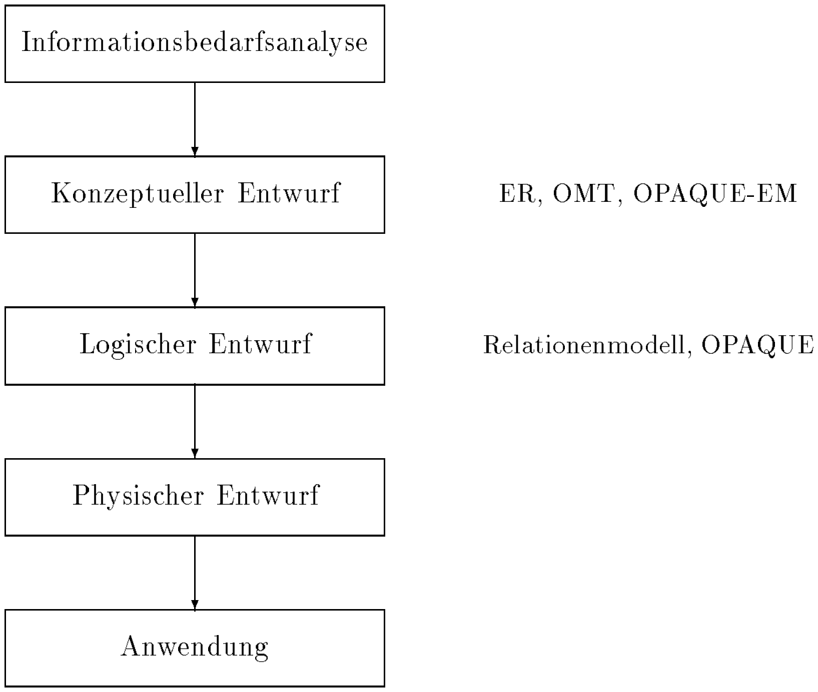


Abbildung 3.1: Vorgehensweise beim Datenbankentwurf

aufgrund des genannten Beispiels einfach und offensichtlich wirken, ist aber sehr wichtig, da sie die Komplexität des zu modellierenden Weltausschnitts reduziert.

Sobald die wesentlichen Aspekte identifiziert sind, kann man damit beginnen, diese Aspekte auf strukturelle Eigenschaften, Organisationseinheiten, Objekte, Beziehungen, Operationen, Informationsquellen etc. zu untersuchen, die wichtig sind, um die Diskurswelt zu verstehen. Dazu ist es beispielsweise notwendig, die Bedeutung verwendeter Begriffe zu klären, da sich hinter diesen meist zentrale Elemente der Modellierung, wie etwa Objekte oder Beziehungen, verbergen.

Weiterhin müssen die Anforderungen an das gewünschte System herausgearbeitet werden. Dazu kann man Anwendungsprofile der Benutzer, typische Anfragen, Anfragehäufigkeit u.ä. ermitteln. In der Regel werden auch Leistungsmerkmale festgelegt.

Das Ergebnis der Informationsbedarfsanalyse wird in einer schriftlich verfaßten Spezifikation festgehalten. Diese ist meistens umgangssprachlich formuliert und nur stellenweise formal.

3.1.2 Konzeptueller Entwurf

In dieser Entwurfsphase versucht man, die Diskurswelt mit Hilfe der Konzepte eines *semantischen Datenmodells* präzise zu beschreiben. Diese Beschreibung ist noch frei von technischen Realisierungsdetails, wie beispielsweise dem Datenmodell des zu verwendenden Datenbanksystems und insbesondere der dort zugrunde liegendenden Datendefinitionssprache (DDL). Es stellt jedoch genügend Ausdrucksmittel zur Verfügung, um die für die Datenmodellierung relevanten

Aspekte auszudrücken.

Die semantische Datenmodellierung basiert auf dem Systembegriff der allgemeinen Systemtheorie:

> Jede Miniwelt wird beschrieben als eine Menge von Gegenständen (unserer Anschauung), zwischen denen wohldefinierte Systembeziehungen bestehen.

Das bedeutet, daß eine beobachtete Umwelt mit Begriffen[2] beschrieben wird, deren Bedeutung als bekannt und erklärt vorausgesetzt wird [LS87]. Es werden also genau die Daten, die tatsächlich in der Datenbasis gespeichert werden, und die Beziehungen, die zwischen diesen Daten bestehen, identifiziert [KS91]. Insbesondere objektorientierte Datenmodellierungen gehen über diese strukturelle Beschreibung noch hinaus und ergänzen sie um eine dynamische, die insbesondere Kontrollaspekte des Systems, wie etwa Ereignisse, Zustände, Zustandsübergänge, enthält [RBP+91].

Das Resultat der semantischen Datenmodellierung ist ein *semantisches* bzw. *konzeptuelles Schema* meist in Form von Diagrammen. Dieses Schema dokumentiert die Ergebnisse der Informationsbedarfsanalyse in einer semiformalen Beschreibung.

Die bekanntesten semantischen Datenmodelle sind

- das Entity-Relationship-Modell (ER-Modell, [Che76]) und
- die Object-Modeling-Technique (OMT, [RBP+91]).

In dieser Arbeit wird OMT Anwendung finden, das wir in Abschnitt 3.2 vorstellen. Um den OPAQUE-spezifischen Aspekten gerecht zu werden, wird ergänzend zu OMT das semantische Datenmodell OPAQUE-EM (OPAQUE-Entwurfsmethode [Kop93]) verwendet (siehe Abschnitt 3.4).

3.1.3 Logischer Entwurf

Ziel des logischen Entwurfs ist ein *logisches Schema*, das von einem Rechner verarbeitet werden kann. Dieses Schema wird aus dem konzeptuellen Schema abgeleitet, das im vorherigen Entwurfsschritt gewonnenen wurde. Dem logischen Schema liegt ein *logisches Datenmodell* zugrunde. Jedes Datenbanksystem bietet ein solches Datenmodell an. Es enthält eine Reihe von Strukturkonzepten, die es dem Rechner erlauben, das entworfene logische Schema zu interpretieren. Beispiele von logischen Datenmodellen sind:

- das hierarchische Modell,
- das Netzwerk-Modell,
- das relationale Modell,
- objektorientierte Modelle.

[2] Wir verweisen an dieser Stelle noch einmal ausdrücklich auf den Abschnitt über Informationsbedarfsanalyse, in dem auf die notwendige Erklärung der Begriffswelt hingewiesen wurde.

Wir werden in unserem Entwurf OPAQUE als logisches Datenmodell verwenden, das in die Klasse der objektorientierten Modelle einzuordnen ist.

Wie wir oben erwähnt haben, ist das konzeptuelle Schema frei von Implementierungsdetails. Es enthält Informationen darüber, *welche* Daten und im objektorientierten Fall auch *welche* Operationen in der Datenbank gehalten werden sollen. Im logischen Schema wird nun gesagt, *wie* diese Daten und Operationen gespeichert bzw. implementiert werden.

Beispiel

Ein konzeptuelles Schema enthalte einen Gegenstand `Benutzer` mit den Attributen `Name`, `Vorname`. Dazu stehe mindestens noch die Operation `Benutzerdaten_ändern` zur Verfügung.

Im logischen Schema wird nun definiert, daß `Benutzer` als Objektklasse mit Variablen `Name`, `Vorname` implementiert wird. Dabei soll `Name` als `string[20]` und `Vorname` als `string[10]` abgebildet werden. Weiterhin wird eine Implementierung der Operation `Benutzerdaten_ändern` angegeben.

3.1.4 Datenbankentwurf für OPAQUE

Wir haben oben die Ideen und die Vorgehensweise bei einem Datenbankentwurf erläutert und möchten nun die Struktur eines Entwurfs für OPAQUE zusammenfassen.

OPAQUE gehört der Klasse der objektorientierten Datenmodelle an. Der Entwurf wird sich deshalb ganz gezielt am objektorientierten Paradigma ausrichten, das sich durch alle Phasen hindurchziehen wird. Dies ist ein entscheidender Vorteil des objektorientierten Ansatzes, da sich die Darstellungsebene nicht ständig ändert [KM90] [Heu93].

Eine Übersicht über den Datenbankentwurfsprozeß, wie wir ihn in dieser Arbeit durchführen werden, gibt Abb. 3.2:

- Zu Beginn wird eine Informationsbedarfsanalyse durchgeführt (Kapitel 4).
- Es folgt ein konzeptueller Entwurf, der sich in drei Phasen gliedert:

  1. einen objektorientierten Entwurf der statischen Strukturen basierend auf OMT (Kapitel 5, Abschnitt 5.2),
  2. einer Umsetzung des konzeptuellen OMT-Entwurfs in OPAQUE-EM (Kapitel 5, Abschnitt 5.3) und
  3. einer Modellierung der dynamischen Aspekte mit Hilfe von Notationen, die von OMT und OPAQUE-EM angeboten werden (Kapitel 7 und 8).

  Phase 2 ist notwendig, um die kontextsensitiven Aspekte berücksichtigen zu können. Dies ist mit Hilfe der Standardmodellierungstechniken für objektorientierte Systeme, wie OMT [RBP+91], der Methode von Coad/Yourdon [CY91] oder des Ansatzes von Booch [Boo91],

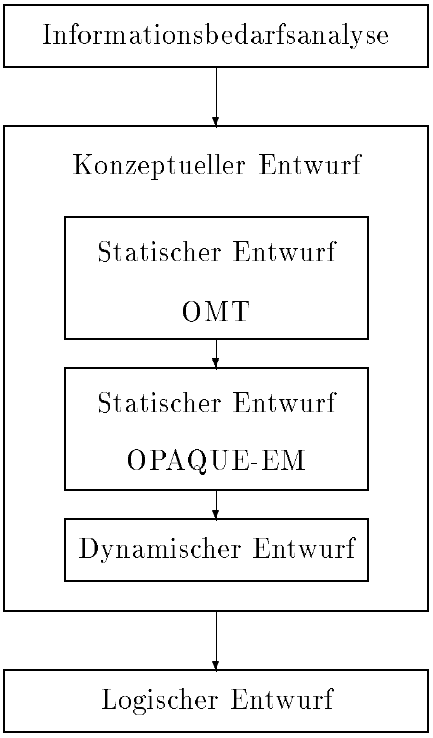


Abbildung 3.2: Vorgehensweise beim Datenbankentwurf mit OPAQUE

nicht möglich. Deshalb verwenden wir die in [Kop93] eigens für die Belange OPAQUEs entwickelten Entwurfsmethode OPAQUE-EM.

Phase 3 ist eine Besonderheit des objektorientierten Ansatzes: Objekte enthalten einerseits statische Strukturinformation und andererseits Informationen über das eigene Verhalten. Herkömmliche Datenmodelle halten nur die Strukturdaten; dynamische Aspekte werden dort außerhalb des Schemas entworfen, während sie in objektorientierten Systemen inhärenter Bestandteil des Schemas sind.

- Schließlich wird das Ergebnis des statischen und des dynamischen Entwurfs im logischen Entwurf auf das logische Datenmodell OPAQUE abgebildet. Die Abbildung eines (statischen) OPAQUE-EM-Modells auf OPAQUE ist trivial [Kop93]. Dagegen werden wir auf kritische Stellen innerhalb des dynamischen Entwurfs eingehen und deren Umsetzung in OPAQUE erläutern (Kapitel 7 und 8).

In den folgenden Abschnitten werden wir nun die genannten Entwurfswerkzeuge OMT, OPAQUE-EM und OPAQUE vorstellen.

3.2 Object-Modeling-Technique

Die Object-Modeling-Technique (OMT) ist ein Hilfsmittel für den konzeptuellen Entwurf, das es erlaubt, die Struktur und das Verhalten einer Miniwelt zu modellieren. Sie basiert auf dem objektorientierten Paradigma. Der nachfolgende Abschnitt wird die Ideen und Konzepte von OMT erläutern, die wir für unseren Entwurf benötigen. OMT wird ausführlich in [RBP$^+$91] beschrieben.

Wir werden zunächst eine Einführung in die Objektorientierung und die Besonderheiten des objektorientierten Entwurfs geben, um danach die für uns relevanten Teile von OMT einzuführen.

3.2.1 Objektorientierung und objektorientierter Entwurf

Objektorientierung

Um die in Abschnitt 3.1 genannten Aspekte der Modellierung zu erfüllen, bietet der objektorientierte Ansatz einige sehr einfache und natürliche Konzepte an. Im Mittelpunkt stehen Objekte und die Beziehungen zwischen diesen Objekten. Dieser Ansatz beruht auf dem Systembegriff der allgemeinen Systemtheorie, den wir bereits im Abschnitt über Datenbankentwurf erläutert haben. Das Objektkonzept gilt als sehr natürlich und in besonderer Weise für geeignet, Struktur und Verhalten eines Systems darzustellen. Das wird in der nachfolgenden Einführung in OMT noch deutlich werden.

Ein *Objekt* ist zunächst einmal ein natürlicher Gegenstand oder Begriff unserer Anschauung und unseres Denkens. Was man in einem speziellen Fall als Objekt betrachtet, ist problemabhängig und unterliegt der (subjektiven) Einschätzung des Beobachters. Beispiele für Objekte sind: die Richtlinie “TA Luft” vom 26. 2. 1986, der Schadstoff Aceton, die DPD-Methode als Meßverfahren u. ä.

Jedes Objekt hat eine eigene *Identität.* Dies macht die Objekte auf jeden Fall unterscheidbar, auch wenn sie sich rein äußerlich nicht unterscheiden: zwei Meßwerte mit identischem Zahlenwert für denselben Parameter und am selben Ort gemessen sind dennoch zwei unterschiedliche Objekte – z.B. weil sie zu unterschiedlichen Zeiten, von unterschiedlichen Meßgeräten oder von unterschiedlichen Personen gemessen wurden.

Man beschreibt ein Objekt, indem man sagt, was es *ist* und was es *kann*, also durch seinen *Zustand* (Attributwerte) und sein *Verhalten* (Operationen). Identität, Attribute und Operationen sind also die Bestandteile eines Objekts. Attributwerte können nur durch die vom Objekt bzw. seiner Objektklasse angebotenen Operationen geändert werden. Ein Objekt ist also ein autarkes Gebilde, das alle für seine Anwendung notwendigen Informationen enthält und nur über eine klar definierte Schnittstelle zugänglich macht.

Man faßt gleichartige Objekte in *Objektklassen* zusammen. Die Gleichartigkeit besteht darin, daß alle Objekte einer Klasse potentiell die gleichen Zustände annehmen und dasselbe Verhalten aufzeigen können. Das heißt, sie werden durch dieselben Attribute beschrieben und können dieselben Operationen ausführen. Beispiele für Objektklassen sind: die Klasse der Richtlinien, der Schadstoffe, der Meßverfahren (Die oben genannten Objekte sind Beispielinstanzen dieser Klassen).

Objektklassen können an Vererbungshierarchien[3] teilnehmen. Die *Vererbung* ist zentraler Bestandteil der Objektorientierung. Sie geht einher mit *Generalisierung* und *Spezialisierung*, die die beiden Beziehungsrichtungen bei der Vererbung bezeichnen. So läßt sich beispielsweise

[3]und manchmal auch Vererbungsheterarchien

die Klasse der Richtlinien in die der Luft-, Wasser- und Bodenrichtlinien spezialisieren. Das bedeutet, daß `Richtlinie` *Oberklasse* der *Unterklassen* `Luftrichtlinie`, `Wasserrichtlinie` und `Bodenrichtlinie` ist. Die umgekehrte Richtung bezeichnet man als Generalisierung der Unterklassen zu einer Oberklasse.

Vererbung bedeutet, daß die Unterklassen Attribute und Operationen der Oberklassen übernehmen und diese um eigene, spezifische Attribute und Operationen ergänzen. Es ist auch möglich beispielsweise die in der Oberklasse gegebene Implementierung einer Operation, die sog. *Methode*, zu überschreiben und durch eine eigene, spezifischere Methode zu ersetzen. Enthält die Klasse der Richtlinien zum Beispiel eine Operation `ausdrucken`, so wird die zugehörige Methode für den generellen Fall einer Richtlinie anders aussehen als z.B. für Wasserrichtlinien, wo beispielsweise noch Informationen über Art des Wassers (Trink-, Grund-, Abwasser o.ä.) gegeben werden. Daher wird man in der Unterklasse eine andere Methode angeben.

Objektorientierter Entwurf

Ein Problem der meisten Entwurfsmethoden besteht darin, daß mit dem Übergang in eine neue Entwurfsphase ein Wechsel von Darstellungen, Abstraktionen und Modellierungsbegriffen einhergeht. So werden beispielsweise *Gegenstands-* und *Beziehungstypen* einer ER-Modellierung auf *Relationen* innerhalb des relationalen Datenmodells abgebildet. Es besteht also kein einheitliches Modellierungsparadigma, das sich von der Analyse über die einzelnen Entwurfsphasen bis zur Implementierung erstreckt. Der ständig notwendige Wechsel der Begriffs- und Darstellungswelt ist prädestiniert dafür, Fehler zu verursachen. Außerdem wird im allgemeinen die Kommunikation erschwert, falls unterschiedliche Personen in den verschiedenen Phasen arbeiten und beispielsweise einen Fehler, der plötzlich auf einer Ebene auftritt, auf mögliche Fehler auf vorangehenden Ebenen zurückverfolgen.

Der Vorteil eines objektorientierten Entwurfs liegt nun darin, daß bereits bei der Analyse, zu einem sehr frühen Zeitpunkt, mit Begriffen und Konzepten gearbeitet wird, die sich nachher sehr einfach in eine objektorientierte Implementierung umsetzen lassen. Deshalb mag es zunächst verwirrend wirken, wenn in OMT bereits von Objektklassen u.ä. gesprochen wird, obwohl wir den konzeptuellen Entwurf frei von Implementierungsdetails halten wollten. Das liegt daran, daß Begriffe wie Objekt, Objektklassen, Vererbung, ... einerseits Strukturkonzepte des objektorientierten Entwurfs, andererseits aber auch Implementierungskonzepte sind, da sich das objektorientierte Paradigma vom Entwurf bis zur Implementierung erstreckt. Dadurch ist es nicht notwendig, die Begriffswelt von einer Phase zur anderen zu wechseln [KM90] [Heu93].

3.2.2 Überblick über OMT

OMT bietet für die Modellierung eines Systems drei Teilmodelle an:

- das Objektmodell,
- das dynamische Modell,
- das funktionale Modell

Die Idee, ein System anhand der obengenannten Modelle zu beschreiben, ist die: ein System hat einerseits einen statischen, andererseits einen dynamischen Aspekt. Da sind zum einen die Gegenstände oder Objekte und deren Beziehungen, zum anderen ein (zeitliches) Änderungsverhalten. Wir sprechen von einer statischen Struktur und einem dynamischen Verhalten des Systems. Diese beiden Komponenten werden in OMT durch das *Objektmodell* (statische Struktur) und das *dynamische Modell* und das *funktionale Modell* (dynamisches Verhalten) repräsentiert, die wir in den folgenden Abschnitten erläutern werden. Die Zweiteilung in ein dynamisches und ein funktionales Modell beim dynamischen Verhalten wirkt auf den ersten Blick etwas willkürlich. Man kann jedoch sagen, daß das dynamische Modell eher das zeitliche Kontrollverhalten aus der Sicht eines Objektes beschreibt, während das funktionale Modell Datenflüsse im Zuge von großen Prozessen darstellt, die mehrere Objekte des Systems einbeziehen und deshalb eine eher globale Betrachtung erfordern. Diese Unterscheidung soll für unsere Zwecke genügen. Genauere Informationen findet man in der Literatur [RBP+91].

3.2.3 Objektmodell

Im Objektmodell werden die Objekte[4] und die Beziehungen der Objekte untereinander dargestellt. Dazu bietet OMT eine Reihe von Symbolen. In Abb. 3.3 sehen wir die Symbole zur Darstellung von Objektklassen und Objekten. In der linken Hälfte der Abbildung ist der allgemeine Fall dargestellt, auf der rechten Seite sind Beispiele angegeben. Wir weichen in der angegebenen Notation leicht von der in [RBP+91] vorgestellten ab. Das sind nur kosmetische Änderungen, die die Übersichtlichkeit der Diagramme verbessern sollen. Manchmal spielen Attribute und/oder Operationen für die Aussage des Diagramms keine Rolle in diesem Fall werden sie einfach weggelassen (siehe z.B. Abb. 3.7).

Beziehungen zwischen Objektklassen werden als *Assoziationen* bezeichnet und durch Verbindungslinien zwischen den beiden Objektklassen dargestellt, falls es sich um eine Assoziation von zwei Klassen handelt. Sind drei oder mehr Klassen beteiligt, so verwendet man eine Raute. Jede Assoziation hat einen Namen, der an die Raute bzw. die Verbindungslinie geschrieben wird (siehe Abb. 3.4 und 3.5).

Jede Assoziation läßt sich instanziieren, indem man Objekte der an der Assoziation beteiligten Klassen in Beziehung setzt. Diese 'Ausfüllung' einer Assoziation nennt man in OMT einen *Link*. Ein Link ist eine Instanz einer Assoziation, ähnlich wie ein Objekt eine Instanz einer Objektklasse ist. Abb. 3.6 zeigt eine Assoziation `hat_Richtlinienregelung` zwischen den Klassen `Richtlinie` und `Richtlinienregelung`. Abb. 3.7 gibt ein Beispiel für einen Link dieser Assoziation.

Jede Objektklasse nimmt in einer bestimmten *Rolle* mit einer bestimmten *Kardinalität* an einer Assoziation teil. Dabei gibt die Kardinalität einer Rolle an, an wievielen Links ein Objekt der Klasse teilnimmt. In Abb. 3.7 existieren beispielsweise zwei Kardinalitätsangaben. Die Kardinalität $< 1, * >$ neben der Klasse `Richtlinie` besagt, daß eine Richtlinie mindestens an einem Link der Assoziation `hat_Richtlinienregelung` teilnimmt. Sie kann an beliebig vielen Links teilnehmen. Auf das Beispiel umgesetzt bedeutet das, daß eine Richtlinie mindestens eine Richtlinienregelung besitzt. Umgekehrt bedeutet die Kardinalität $< 1, 1 >$ neben der

[4] Eigentlich Objektklassen. Da man aber in der Regel die hinter diesen Objektklassen stehenden Objekte betrachtet, ist es üblich geworden, statt von Objektklassen nur von Objekten zu sprechen. Dies geht normalerweise aus dem Zusammenhang hervor und sollte den Leser nicht weiter verwirren.

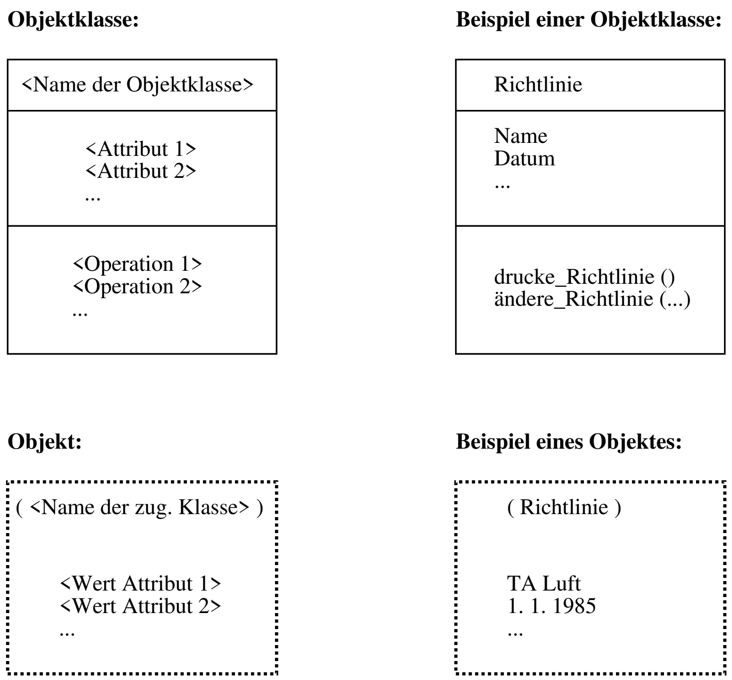


Abbildung 3.3: Symbole zur Darstellung von Objekten und Objektklassen, sowie deren Instanzen.

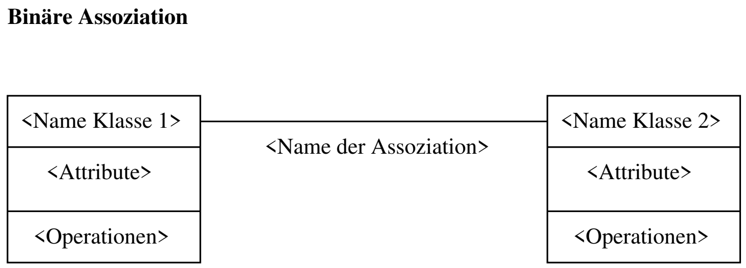


Abbildung 3.4: Darstellung einer binären Assoziation zwischen Objektklassen

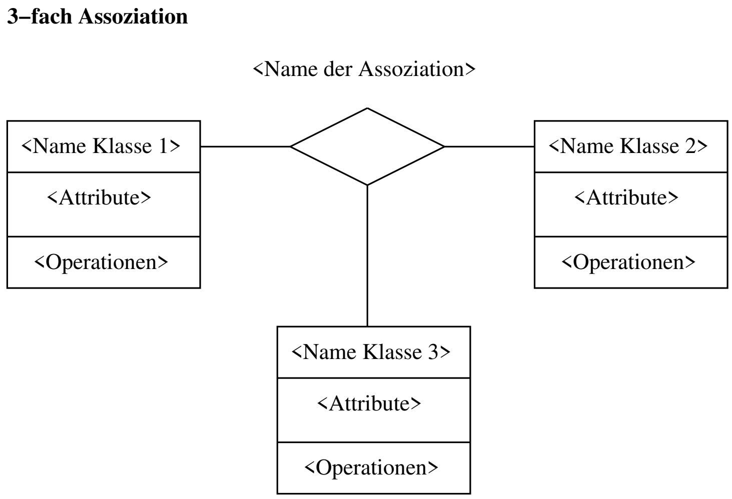


Abbildung 3.5: Darstellung einer 3-fach Assoziation zwischen Objektklassen

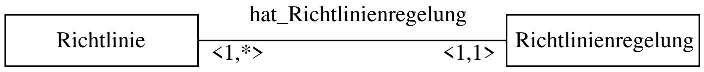


Abbildung 3.6: Beispiel für eine Assoziation

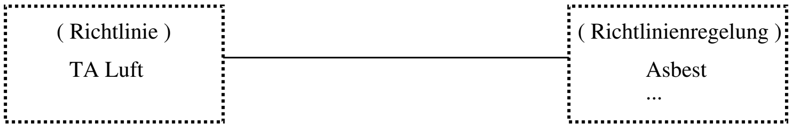


Abbildung 3.7: Beispiel für einen Link

Klasse `Richtlinienregelung`, daß eine Richtlinienregelung an *genau* einem Link der Assoziation `hat_Richtlinienregelung` teilnimmt, was der Tatsache entspricht, daß jede Richtlinienregelung in genau einer Richtlinie enthalten sein muß.

Ähnlich wie Objektklassen können auch Assoziationen Attribute haben, die die Art der Beziehung näher charakterisieren. Die zugehörige Notation findet sieht man im Beispiel aus Abb. 3.8.

Vererbungsbeziehungen zwischen Objektklassen werden wie in Abb. 3.9 beschrieben. Hier ist die aus Kapitel 2 bekannte Hierarchie der unterschiedlichen Richtlinien dargestellt. Neben dem Dreieck, das die Vererbung symbolisiert, kann noch der Name eines *Diskriminators* stehen. Das ist ein Attribut der Oberklasse, dessen Werte die Objekte unterteilen. In unserem Beispiel enthalte die Klasse `Richtlinie` ein Attribut `Meßobjekt`, dessen mögliche Werte `Luft`, `Wasser` und `Boden` seien. Ähnlich verhält es sich mit dem Attribut `Nutzung` der Klasse `Wasserrichtlinie`.

Neben allgemeinen Beziehungen (Assoziationen) und Vererbungsbeziehungen lassen sich im Objektmodell noch sogenannte *Aggregationen* darstellen. Beispiele von Aggregationen finden sich in Komponentenhierarchien. Eine Aggregation stellt in diesem Fall Objekte in eine 'ist-Bestandteil-von' Beziehung. Abb. 3.10 zeigt die Aggregation eines Richtlinienwerts: er besteht aus Objekten der Klassen `Richtlinienwertname`, `Wert`, `Maßeinheit`und manchmal noch aus einem Objekt aus `Wertart` (siehe Kardinalität $< 0, 1 >$).

OMT bietet neben den hier vorgestellten Möglichkeiten noch weitere, die wir für unsere Modellierung nicht verwenden werden und auf deren Einführung wir hier verzichten. Der interessierte Leser sei dafür auf [RBP$^+$91] verwiesen.

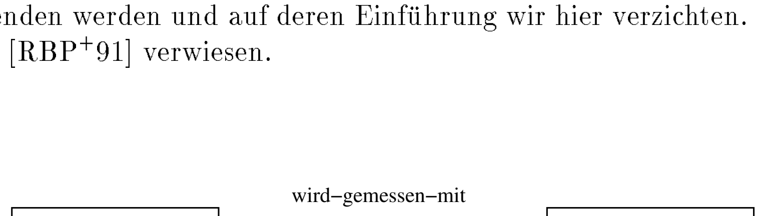


Abbildung 3.8: Attribute in Assoziationen

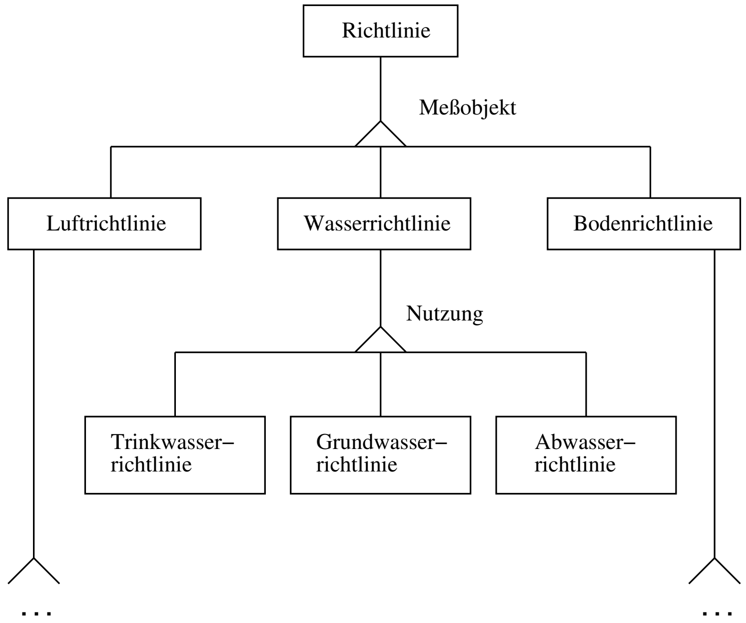


Abbildung 3.9: Vererbungshierarchie unterschiedlicher Umweltrichtlinien

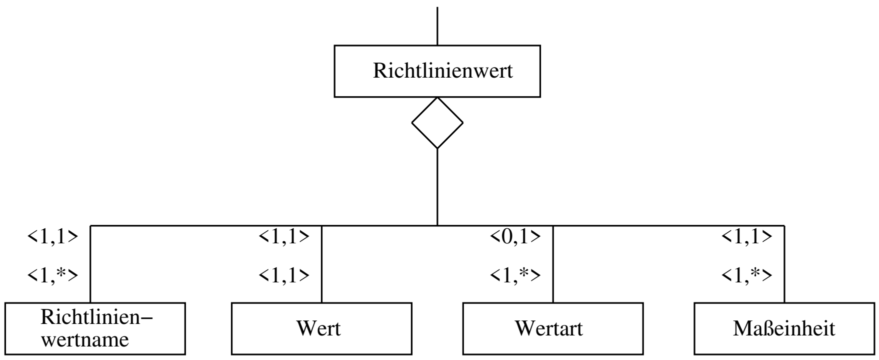


Abbildung 3.10: Beispiel einer Aggregation

3.2.4 Dynamisches Modell

Das dynamische Modell beschreibt das Verhalten eines Systems, wenn ein sogenanntes *Ereignis* auftritt. Ein Ereignis ist ein Vorgang oder Stimulus, wie z.B. das Überschreiten eines Grenzwertes oder das Inkrafttreten einer neuen Richtlinie, auf den das System reagiert. In solchen Fällen verändert ein solches Ereignis den bisherigen Zustand des Systems. Das Resultat ist ein neuer (System-) Zustand. Damit sind die drei wichtigsten Komponenten des dynamischen Modells identifiziert:

- Zustände
- Zustandsübergänge
- Ereignisse

Der *Zustand* eines Systems setzt sich aus den Einzelzuständen der am System beteiligten Objekte zusammen. Der Zustand eines Objektes ist eine abstrakte Beschreibung der momentan bestehenden Attributwerte und Links des Objekts.

Wir werden nun ein Beispiel geben, mit dessen Hilfe die genannten Begriffe und Zusammenhänge verdeutlicht werden können und das dynamische Modell zum funktionalen Modell (siehe Abschnitt 3.2.5) abgegrenzt wird. Das soll genügen, da wir in unserer Modellierung des Richtlinienszenarios auf ein dynamisches Modell (im Sinne OMTs) verzichten können, da Zustandsveränderungen der dort vorhandenen Objekte oder Ereignisse für den Entwurf keine Rolle spielen.

Beispiel

Es existiere eine Objektklasse `Luft` mit den Attributen `Ort`, `SO`$_2$`-Gehalt`, `NO`$_x$`-Gehalt`, `O`$_3$`-Gehalt`. Als Objekte dieser Klasse kann man sich etwa die Luft in Umgebung eines Heizkraftwerks, die Luft über Karlsruhe, Stuttgart, München etc. vorstellen. Es bestehe nun die Vorschrift, daß der Zustand der Luft als *kritisch* zu betrachten sei, falls der Gehalt eines der Schadstoffe SO_2, NO_x oder O_3 den zulässigen Grenzwert (GW) überschreite. Der Zustand der Luft werde als *gefährlich* betrachtet, falls die Konzentration eines weiteren Schadstoffs den Grenzwert überschreitet. Er werde so lange als gefährlich betrachtet, bis alle Schadstoffe wieder unter ihren jeweiligen Grenzwert gefallen sind.

Im Beispiel erkennen wir die genannten drei Komponenten wieder:

- Zustände:

  `normal`: Weder der SO_2-Gehalt, noch der NO_x-Gehalt, noch der O_3-Gehalt hat seinen zugehörigen GW überschritten.

  `kritisch`: Genau ein Schadstoffgehalt hat seinen zugehörigen GW überschritten.

  `gefährlich`: Mindestens zwei Schadstoffgehalten haben ihren zugehörigen GW überschritten.

- Zustandsübergänge:
  - von `normal` zu `kritisch`
  - von `kritisch` zu `normal`
  - von `kritisch` zu `gefährlich`
  - von `gefährlich` zu `normal`
- Ereignisse:
  - Ein Schadstoffgehalt überschreitet seinen zugehörigen Grenzwert.
  - Alle Schadstoffgehalte unterschreiten ihren zugehörigen Grenzwert.

Man sieht nun an dem Beispiel sehr deutlich, daß der Zustand der Luft nicht vom Wert eines Attributs abhängt, sondern von mehreren. So beschreibt der Zustand `kritisch` zusammenfassend, daß genau eine Schadstoffkonzentration ihren zugehörigen Grenzwert überschritten hat, während die anderen unter ihrem Grenzwert liegen. Weiterhin ist zu beachten, daß man von den reinen Zahlenwerten, die diese Konzentrationen darstellen, abstrahiert und sie dahingehend interpretiert, daß man eine Aussage macht, wie groß die Gefährdung z.B. für den Menschen ist. Dadurch ist es möglich, daß sich zwei Objekte derselben Klasse auch dann in demselben Zustand befinden, wenn sie nicht genau dieselben Attributwerte haben.

Abb. 3.11 zeigt nun das Verhalten von Objekten der Objektklasse `Luft` anhand eines Zustandsübergangsdiagramms. Zustände sind durch Ovale, Übergänge durch Pfeile und Ereignisse durch Beschriftung der Pfeile gekennzeichnet. Als viertes Element sind Aktionen vorhanden, die ausgeführt werden sollen, falls der entsprechende Zustandsübergang vollzogen wird bzw. falls das zum Übergang gehörende Ereignis eintritt.

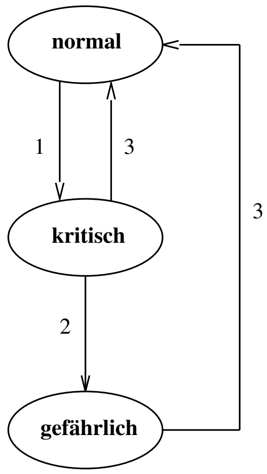
Man kann ein Zustandsübergangsdiagramm fast beliebig detailliert gestalten. So kann man einen Zustand intern in mehrere Einzelzustände unterteilen, zwischen denen ebenfalls Übergänge bestehen, so daß ein Zustand in sich ein eigenes Zustandsübergangsdiagramm enthält. Der interessierte Leser findet in zahlreichen Büchern und Artikeln aus fast allen Gebieten der Informatik Einführungen in die Thematik der Zustandsübergangsdiagramme, die einmal als endliche Automaten, manchmal als Transduktoren o.ä. bezeichnet werden.

Zusammenfassung

Das dynamische Modell erlaubt es, das Verhalten eines Systems beim Eintritt von Ereignissen darzustellen. Ereignisse beschreiben einen Zeitpunkt. Das dynamische Modell spezifiziert damit das Kontrollverhalten eines Systems.

Als Beschreibungsmittel werden Zustandsübergangsdiagramme verwendet. Sie enthalten die Zustände, die ein Objekt der zugehörigen Objektklasse einnehmen kann. Übergänge in einen anderen Zustand werden durch Ereignisse ausgelöst. Daneben können Ereignisse bestimmte Aktionen initiieren.

Idealer-, aber nicht notwendigerweise gibt es pro Objektklasse eines oder mehrere Zustandsübergangsdiagramme. Die Gesamtheit aller Diagramme ist die Beschreibung des dynamischen Systemverhaltens. Ein Problem besteht aber darin, daß die Semantik der Zustände und der Übergänge nicht formal an die Klasseneigenschaften gebunden werden können, die im Objektmodell spezifiziert wurden.



1 ein Schadstoffgehalt überschreitet seinen GW / Emissionsquellen zurückschalten
2 (weiterer) Schadstoffgehalt überschreitet seinen GW / alle Emissionsquellen abschalten
3 alle Schadstoffgehalte unterschreiten GW / Erlaubnis für Vollbetrieb erteilen

Abbildung 3.11: Zustandsübergangsdiagramm zum Beispiel

3.2.5 Funktionales Modell

Während das Objektmodell die Objektstruktur und das dynamische Modell das einzelne Objektverhalten beschreibt, gibt das funktionale Modell Informationen darüber, welche Daten mittels welcher Prozesse zwischen welchen Objekten ausgetauscht werden. Dies geschieht durch die Angabe von Datenflußdiagrammen. Es wird dabei *nicht* gesagt, wann der Datenfluß abläuft – das wird im dynamischen Modell spezifiziert.

Ein *Datenflußdiagramm* ist ein Graph, dessen Knoten Objekte, Prozesse oder Datenbehälter und dessen Kanten Datenflüsse sind. Objekte treten dabei nur als Randknoten – als Datenquelle oder -senke – auf. Rumbaugh et al. sprechen von *Aktoren*, die Daten produzieren und konsumieren; interne Knoten sind Prozesse und Datenbehälter. Wir werden nun die Begriffe und die Symbolik anhand eines Beispiels erläutern.

Abb. 3.12 zeigt ein einfaches Beispiel eines Datenflußdiagramms aus dem Umweltgrenzwertszenario. Es enthält die vier obengenannten Elemente eines solchen Diagramms:

- Aktoren bzw. Objekte (Rechtecke),
- einen Prozeß (Oval),
- Datenflüsse (Pfeile),
- einen Datenbehälter (zwei parallele Linien)

Das Diagramm stellt die Datenflüsse bei einer Datenbankanfrage nach dem Richtlinienwert für einen bestimmten Parameter dar: von den Aktoren `Parameter` und `Richtlinienwert` stammen

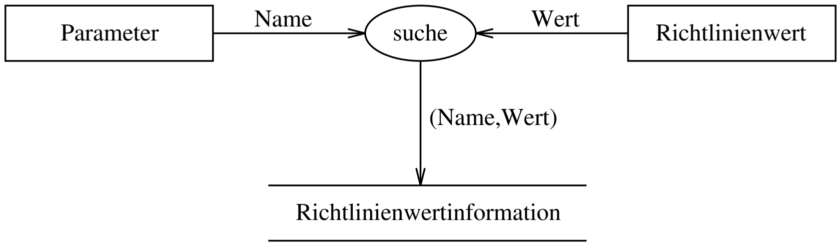


Abbildung 3.12: Datenflußdiagramm für Richtlinienwertanfrage

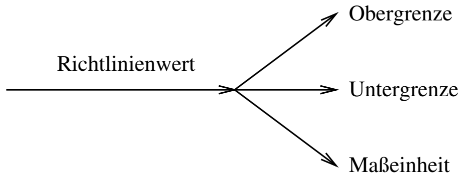


Abbildung 3.13: Aggregation von Datenflüssen

die Daten für Namen und Werte, die vom Prozeß `suche` verarbeitet und in einem Datenbehälter `Richtlinienwertinformation` abgelegt werden.

Wir erläutern nun die Elemente eines Datenflußdiagramms:

Aktor: Ein Aktor ist ein Objekt, das Daten liefert oder erhält, aber keine Daten verarbeitet – das ist allein den Prozessen vorbehalten (siehe unten). Deshalb sind Aktoren Randknoten des Datenflußgraphen, da sie entweder nur eingehende oder ausschließlich wegführende Kanten haben. Aktoren werden als Rechtecke dargestellt, die mit dem Namen der zugehörigen Objektklasse beschriftet sind.

Prozeß: Ein Prozeß verarbeitet Datenwerte. Es gibt sehr einfache, elementare Prozesse, wie beispielsweise eine Addition oder eine Multiplikation, aber auch komplexe Prozesse, wie eine Datenbankanfrage oder die Messung eines Wertes. Meist lassen sich die komplexen Prozesse in eine Reihe kleinerer Prozesse aufbrechen.

Prozesse bilden die 'Innenknoten' des Datenflußgraphen, d.h. sie haben sowohl eingehende Kanten (für die Eingabewerte), als auch wegführende Kanten (für die Ausgabewerte). Sie werden als Ovale dargestellt, die mit dem Namen des Prozesses beschriftet sind.

Datenfluß: Datenflüsse verbinden Aktoren (Objekte), Prozesse und Datenbehälter miteinander. Sie werden als beschriftete Pfeile dargestellt und repräsentieren Datenwerte oder Zwischenergebnisse der maschinellen Verarbeitung. Abb. 3.12 zeigt drei Datenflüsse, die unmittelbar einsichtig sind: die Datenwerte `Name` bzw. `Wert` 'fließen' vom Objekt `Parameter` bzw. vom Objekt `Richtlinienwert` zum Prozeß `suche`, das von diesem Prozeß ausgewählte Wertepaar `(Name, Wert)` 'fließt' zum Datenbehälter `Richtlinienwertinformation`.

Datenflüsse können aggregiert werden. Abb. 3.13 zeigt ein Beispiel: `Richtlinienwert` ist eine Aggregation der Datenflüsse `Untergrenze`, `Obergrenze` und `Maßeinheit`.

Datenbehälter: Ein Datenbehälter ist neben Objekten der einzige Ort, an dem Daten gespeichert werden können. Er dient als Zwischenspeicher. Wegen ihrer Verwandtschaft zu den

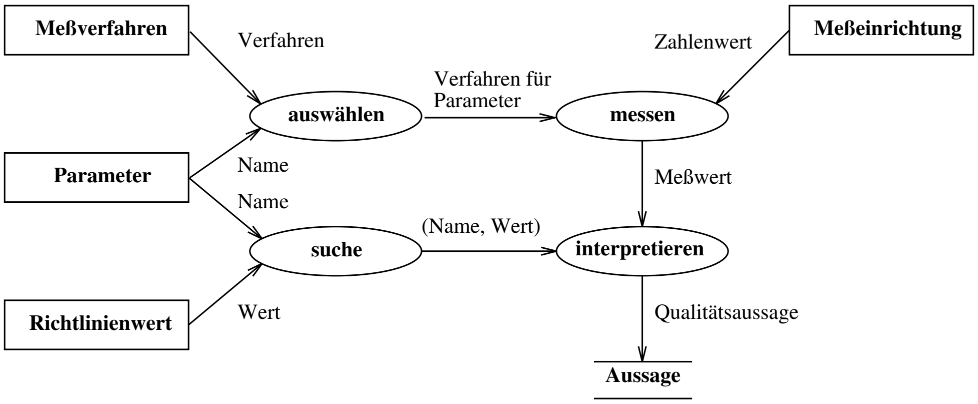


Abbildung 3.14: Datenflußdiagramm für eine Schadstoffmessung

Objekten werden sie auch *passive Objekte* genannt. Sie werden dargestellt durch zwei parallele Linien, zwischen denen der Name des Datenbehälters steht.

Abb. 3.14 zeigt ein weiteres Datenflußdiagramm. Es stellt die Datenflüße dar, wenn gemessen wird, ob ein Schadstoff (Parameter) seinen Grenzwert einhält oder nicht. Dazu benötigen wir einerseits den zugehörigen Grenzwert, den uns der Prozeß `suche` liefert, ein Meßverfahren, das durch den Prozeß `auswählen` ermittelt wird, und einen Zahlenwert, den eine Meßeinrichtung an den Prozeß `messen` liefert. Der Prozeß `interpretieren` vergleicht den gemessenen Wert mit dem von der Richtlinie vorgegebenen Wert und erstellt eine Qualitätsaussage, die in einem Datenbehälter `Aussage` abgelegt wird. Unter diesem Datenbehälter kann man sich beispielsweise einen Drucker oder einen Bildschirm vorstellen, der die Aussage abdruckt.

3.3 OPAQUE

In diesem Abschnitt werden wir das Kontextmodell OPAQUE [Lan93] [Lau93] vorstellen. Dazu führen wir zunächst einen Vergleich zur klassischen Objektorientierung durch, die wir im Rahmen von OMT in Abschnitt 3.2 besprochen haben, um Unterschiede und Gemeinsamkeiten hervorzuheben. Dies wird dem Leser eine erste Idee geben. In den darauffolgenden Abschnitten werden die statischen und dynamischen Konzepte von OPAQUE detaillierter vorgestellt. Insbesondere wird auch die Syntax der Konstrukte eingeführt.

3.3.1 Einführender Vergleich: OPAQUE – klassische Objektorientierung

OPAQUE ist ein logisches Datenmodell, das einige Konzepte des objektorientierten Paradigmas aufbricht und deshalb in strengem Sinne nicht mehr als objektorientiert bezeichnet werden kann. Wir wollen es daher *objektbasiert* nennen, um darauf hinzuweisen, daß der Begriff des Objekts zwar eine zentrale Rolle in OPAQUE spielt, daß er aber nicht exakt dem herkömmlichen Gebrauch entspricht. Um die Unterschiede zwischen klassischer Objektorientierung (siehe Abschnitt 3.2.1) und den OPAQUE-Konzepten zu motivieren, wollen wir zunächst einige Nachteile des klassischen objektorientierten Paradigmas aufzeigen, da OPAQUE genau an diesen Nachteilen ansetzen wird.

Wie wir in Abschnitt 3.2.1 gesehen haben, kann man ein Objekt als autarkes Gebilde auffassen, das eine Identität, eine Beschreibung seines Zustands (Attribute und Beziehungen) und seines Verhaltens (Operationen) enthält. Diese drei Komponenten sind von einer Art Schutzhülle umgeben und von außen nur über eine klar definierte Schnittstelle zugänglich. Dies ist die Verwirklichung des Geheimnisprinzips (information hiding) und der Forderung nach Modularisierung aus dem Software Engineering. Dies soll den Softwareentwurf, die -wartung, die Übersichtlichkeit und Erweiterbarkeit unterstützen und wird darüberhinaus als eine natürliche Modellierung empfunden: viele Szenarien enthalten voneinander unabhängige Objekte, die miteinander in fest vorgegebener Art und Weise kommunizieren und dadurch Daten austauschen.

Es gibt jedoch einige Situationen und Szenarien, in denen die strenge Einkapselung und Autarkie der Objekte eine natürliche Modellierung behindert. So reagieren die Objekte nur, wenn sie durch eine Nachricht von außen aus ihrer Untätigkeit herausgerissen werden und nur dann können sie ihren eigenen Zustand verändern. Von alleine kann ein Objekt gar nicht auf die Veränderung seiner Umgebung reagieren, da es auf einen äußeren Impuls angewiesen ist. Wir wollen dies an einem Beispiel verdeutlichen.

Beispiel

Wir haben in Kapitel 2 das Szenario der Umweltgrenzwerte schon vorgestellt: es existieren Richtlinien, die für bestimmte Parameter (u.a. sind das Konzentrationen für chemische Stoffverbindungen) Richtlinienwerte enthalten. Zu einem Richtlinienwert gehören eine Maßeinheit und manchmal auch eine Meßmethode, mit der der Richtlinienwert im konkreten Fall zu überprüfen ist. Abb. 3.15 zeigt eine mögliche Modellierung dieser Situation in OMT-Notation. Im Mittelpunkt steht ein Objekt `Richtlinienwert`, das durch Assoziationen mit den Objekten `Chemischer-Stoff`, `Maßeinheit`, `Meßmethode` und `Richtlinie` verbunden ist, um die oben erwähnten Beziehungen zu modellieren. Die Kardinalitäten spielen für das Folgende keine Rolle, sollten aber unmittelbar einsichtig sein.

`Chemischer-Stoff` enthält eine Operation `gebe-Risikostufe(Meßwert)`, die aufgrund der für den jeweiligen chemischen Stoff geltenden Richtlinienwerte einen gegebenen Meßwert einstuft. Dazu muß das jeweilige Objekt aus `Chemischer-Stoff` Nachrichten an die mit ihm verbundenen Richtlinienwertobjekte senden, um die entsprechenden Richtlinienwerte zu erfragen.

Wird nun der Richtlinienwert für einen chemischen Stoff z.B. aufgrund neuer Erkenntnisse oder politischen Drucks verändert, so werden nur Attributwerte des Richtlinienwertobjekts angepaßt. Das mit diesem Richtlinienwert in Beziehung stehende Objekt aus `Chemischer-Stoff` erfährt von dieser Änderung nichts und ändert deshalb auch nicht seinen eigenen Zustand. Andererseits kann es in der realen Wert für einen Stoff wichtige Konsequenzen haben, wenn sich seine Richtlinienwerte ändern: z.B. kann es sein, daß er in bestimmten Produkten nicht mehr verwendet werden darf oder daß er zukünftig einer besonderen Entsorgung unterliegt. Diese *implizite* Änderung ist in der klassischen Objektorientierung nur dadurch zu modellieren, daß

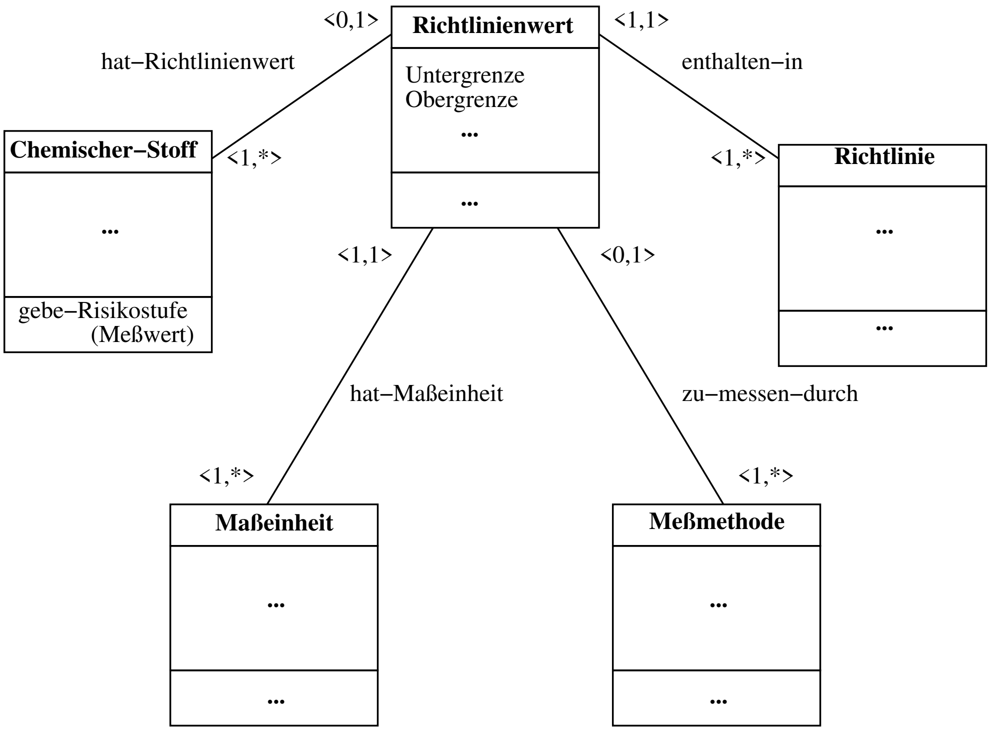


Abbildung 3.15: Beispiel aus dem Umweltgrenzwertszenario in OMT-Modellierung

mit dem Ändern der Attributwerte im Richtlinienwertobjekt *explizit* eine Nachricht an das betroffene Objekt der Klasse `Chemischer-Stoff` gesendet wird oder dieses Objekt die Änderung irgendwann *explizit* erfragt.

Eleganter und natürlicher ist dieses Problem durch Kooperation zu lösen, d.h. durch einen gemeinsamen Datenbereich, der sowohl zu einem Objekt aus der Klasse `Chemischer-Stoff` als auch zu einem aus der Klasse `Richtlinienwert` gehört. Wir werden weiter unten noch auf dieses Beispiel zurückkommen, wenn wir eine OPAQUE-Modellierung vorstellen werden. Der Leser sei an dieser Stelle trotzdem schon einmal auf Abb. 3.20 verwiesen: sie enthält die Struktur einer OPAQUE-Modellierung des genannten Beispiels. Zwei dort vorhandene (OPAQUE-) Objektklassen sind durch Ovale umschlossen, so daß deutlich zu erkennen ist, daß Objekte dieser Klassen jeweils einen eigenen privaten Bereich haben und einen Bereich, der mit dem jeweils anderen Objekt überlappt.

Eine weitere Eigenschaft des herkömmlichen Objektbegriffes, die sich manchmal nachteilig auswirken kann, ist die feste statische Bindung eines Objekts an seine Klasse, insbesondere an die in dieser Klasse definierten Attribute und Operationen. Das verhindert strukturelle und verhaltensmäßige Änderungen im Verlaufe des Lebenszyklus' des Objekts, so daß das Objekt keine Evolution durchlaufen kann. Es ist aber natürlich, daß ein Objekt im Verlaufe seines 'Lebens' sein Verhalten und seine Zustandsbeschreibung verändert oder ergänzt. Wir können dazu das oben erwähnte Beispiel eines chemischen Stoffes aufgreifen: Asbest ist ein Vertreter dieser Klasse. Vor 20 Jahren noch war Asbest ein viel verwendeter Rohstoff in der Bauwirtschaft oder im Fahrzeugbau, während man ihn heute aufgrund der in den letzten Jahren gewonnenen

Erkenntnisse fast nur noch als Schadstoff betrachtet. Während man sich bei einem Rohstoff für Preise, Lieferanten etc. interessiert, möchte man bei einem Schadstoff z.B. seine Toxizität, Entsorgungsmöglichkeiten u.ä. wissen. Das bedeutet, daß Asbest vor 20 Jahren durch andere Daten beschrieben wurde als heute. Zumindest hat sich das Interesse heutzutage auf weitere Informationen ausgeweitet. Natürlich ließe sich dies im herkömmlichen Objektmodell dadurch modellieren, daß man zusätzliche Attribute schon von vorneherein vorsieht. Das würde aber bedeuten, daß man heute schon vorhersehen müßte, was in ein paar Jahren von Interesse sein wird.

OPAQUE wirkt diesem Defizit entgegen, indem es die Identität eines Objektes von der Objektbeschreibung abtrennt und diese Beschreibungen *dynamisch* an die Identität bindet: die Identität ist in einem sogenannten *Kristall* abgelegt, an den mindestens ein *Kontext* gebunden ist, der einen Teil oder alle der ursprünglichen Attribute und Operationen enthält. Die Gesamtheit der an einem Kristall angehängten Kontexte stellen die aktuelle Beschreibung des Objektes dar.

Ein Kontext ist mindestens mit einem Kristall zu verwenden. Er kann aber auch mit mehreren verbunden sein. Um mehrere Kristalle an sich anhängen zu können, bietet ein Kontext *Rollen* an. Dadurch kann man die Kristalle dahingehend unterscheiden, welche Bedeutung sie für den Kontext haben. Dies ist insbesondere für die im Kontext enthaltenen Operationen wichtig.

Die möglichen Rollen, die Attribute und die Operationen werden nicht für jeden Kontext individuell, sondern in einer *Kontextschablone* definiert. Ein Kontext ist eine Instanz einer solchen Schablone, ähnlich wie ein Objekt die Instanz einer Objektklasse oder eine Variable eine Instanz eines Typs ist.

Abb. 3.16 zeigt ein Beispiel einer Objektklasse in OPAQUE. Sie besteht aus einem Kristallplatzhalter (schwarzer Kreis), der mit dem Namen der Objektklasse beschriftet ist. Im gezeigten Beispiel hängt der Kristallplatzhalter an zwei Kontextschablonen `Chemischer-Stoff_Ktxt` und `Rohstoff` in den Rollen `Stoff` bzw. `Rohstoff`, die neben den Verbindungslinien zwischen Schablone und Kristallplatzhalter stehen.

Abb. 3.17 zeigt eine Instanziierung der Objektklasse aus Abb. 3.16. Um diese Instanziierung als solche zu kennzeichnen, verwenden wir an Stellen, wo diese wichtig ist wie schon zuvor in OMT gestrichelte Linien. Die Abbildung greift das Beispiel von Asbest auf, das wir bereits oben erwähnt haben. Aus dem Kristallplatzhalter ist ein Kristall entstanden, der die Identität des Objektes trägt. Wir haben diese Identität mit 'Asbest' bezeichnet, wobei wir darauf hinweisen, daß diese von dem Attributwert 'Asbest' des Attributes `Name` im Kontext der Schablone `Chemischer-Stoff_Ktxt` zu unterscheiden ist. Aus den Kontextschablonen sind nun Kontexte geworden, die Attributwerte enthalten und in der von ihnen jeweils angebotenen einen Rolle den Kristall 'Asbest' aufnehmen. Der Kristall und die beiden mit ihm verbundenen Kontexte stellen ein Objekt in OPAQUE dar.

Wie wir oben schon erwähnt haben, ist es möglich, daß ein Objekt im Verlaufe seines Lebens neue oder andere Beschreibungen erfordert. Es schlüpft sozusagen in neue oder zusätzliche Rollen, was wir in OPAQUE dadurch modellieren, daß das Objekt solche Rollen in weiteren Kontexten besetzt. 'Asbest' wurde beispielsweise als krebserregender Schadstoff erkannt, dessen toxische Wirkungen nicht außer Acht gelassen werden dürfen. Deshalb wurde für ihn ein Richtlinienwert erlassen. Asbest hat sozusagen eine Rolle als Parameter eines Richtlinienwerts übernommen. In OPAQUE können wir diesem Umstand Rechnung tragen, indem wir 'Asbest'

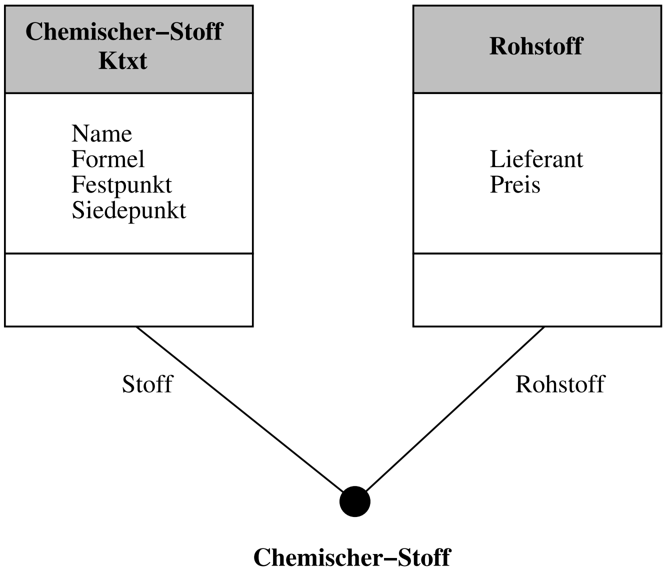


Abbildung 3.16: Objektklasse `Chemischer-Stoff` in OPAQUE

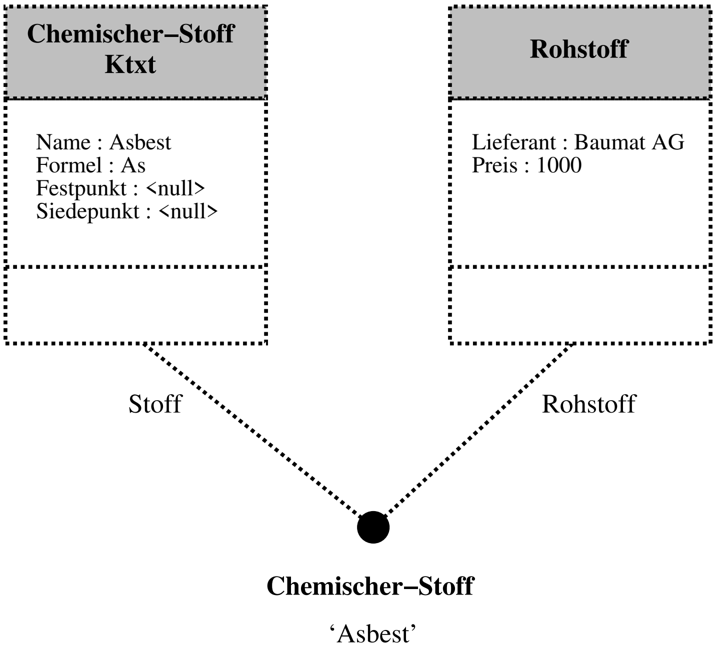


Abbildung 3.17: Objekt 'Asbest' der Klasse `Chemischer-Stoff`

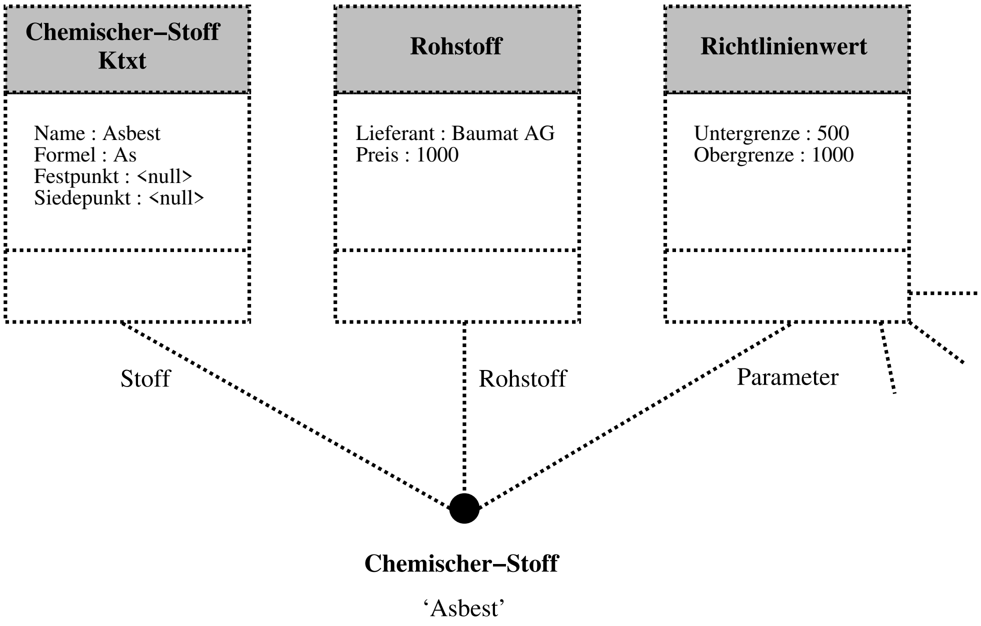


Abbildung 3.18: Dynamische Erweiterung der Objektbeschreibung durch einen zusätzlichen Kontext

durch einen zusätzlichen Kontext beschreiben (siehe Abb. 3.18).

Man kann sich z.B. auch vorstellen, daß ‘Asbest’ eines Tages seine Rolle als nennenswerter Rohstoff verliert. Das kann man in OPAQUE nachvollziehen, indem man den in den Abbildungen 3.17 und 3.18 vorhandenen Kontext `Rohstoff` wegnimmt (Abb. 3.19).

Dieses dynamische Hinzufügen und Wegnehmen von Kontexten wird in OPAQUE als *Evolution* bezeichnet. Dadurch kann man die Entwicklung eines Objekts, das Durchlaufen von mehreren Stadien im Verlaufe seines Lebens modellieren. In Zusammenhang mit dem oben erwähnten Klassenbegriff bedeutet das, daß ein Objekt seine Klasse wechseln kann. Beispielsweise gehört das in Abb. 3.18 dargestellte Objekt nicht mehr der in Abb. 3.16 dargestellten Klasse an.

Eine etwas komplexere Modellierung zeigt Abb. 3.20, die das Beispiel aus Abb. 3.15 aufnimmt und hier eine OPAQUE-Variante präsentiert. Sie enthält die vier Objektklassen

`Chemischer-Stoff`, `Maßeinheit`, `Meßmethode` und `Richtlinie`

und die fünf Kontextschablonen

`Richtlinienwert_Ktxt`, `Chemischer-Stoff_Ktxt`, `Maßeinheit_Ktxt`, `Meßmethode_Ktxt` und `Richtlinie_Ktxt`.

Von besonderem Interesse ist sicher die erstgenannte Schablone, die vier Rollen anbietet, was zu sich überlappenden Datenbereichen führt: Objekte der vier genannten Klassen überschneiden sich in Kontexten der Schablone `Richtlinienwert_Ktxt`. Dies ist durch Ovale angedeutet, die die Klassen umschließen. Durch diese Überlappungen muß beispielsweise ein Objekt der Klasse `Chemischer-Stoff` nicht mit einem anderen Objekt kommunizieren, um die zu ihm gehörenden

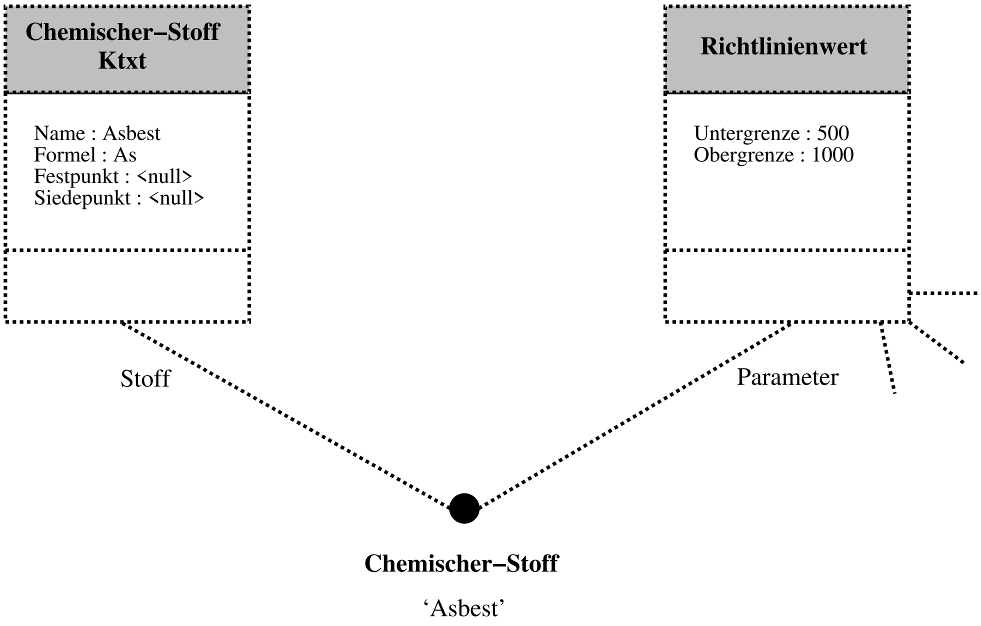


Abbildung 3.19: Rohstoff-Kontext wurde aus der Objektbeschreibung entfernt

Richtlinienwerte zu erfahren, da diese in einem Datenbereich liegen, der nicht nur zu ihm selbst, sondern auch zu anderen Objekten gehört. Die Objekte, die sich diesen Datenbereich teilen, können über ihn kooperieren.

Zusammenfassend enthält Tabelle 3.1 einen Vergleich zwischen klassischer Objektorientierung und OPAQUE.

3.3.2 Statische OPAQUE-Konzepte

In diesem Abschnitt werden die in der Einführung bereits erwähnten Strukturkonzepte, wie Kristalle, Kontexte, Kontextschablonen etc. noch einmal detailliert erläutert und an Beispielen ihre syntaktische Darstellung erklären. Dies soll für unsere Zwecke genügen; eine ausführliche Beschreibung der Syntax OPAQUEs ist in [Nau93] zu finden.

Kristalle und Objekte

Wie wir in den einführenden Beispielen schon gesehen haben, tragen *Kristalle* die Identität eines Objektes im herkömmlichen Sinn. Sie sind die einzigen referenzierbaren Einheiten in OPAQUE und über Variablen ansprechbar. Deshalb werden sie als Nachrichtenquellen und -senken sowie als Ausgangspunkt für Operationen verwendet. Kristalle selbst enthalten aber weder einen inneren Zustand noch ein Verhaltensmuster. Dies wird ihnen durch Anbindung an einen Kontext gegeben.

Ein Kristall, sowie die an ihn gebundenen Kontexte bilden ein *Objekt* im herkömmlichen Sinne. Alle Komponenten dieses Objektes (gemeint sind die Kontexte und die darin enthaltenen Attribute und Operationen) werden über den Kristall angesprochen. Er ist der Zugangspunkt

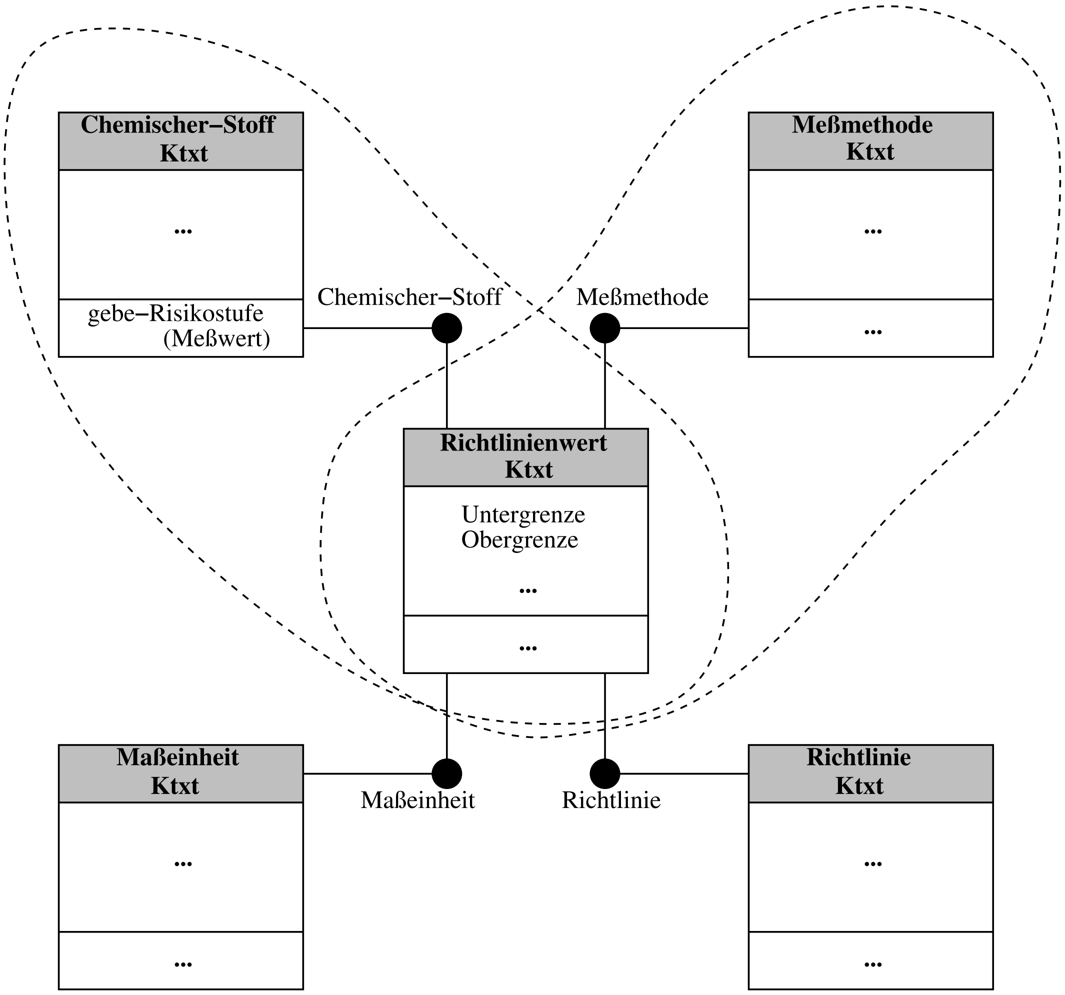


Abbildung 3.20: Beispiel aus dem Umweltszenario in OPAQUE-Modellierung

Aspekt	**Klassische Objektorientierung**	**OPAQUE**
Objektbegriff	Ein Objekt besteht aus den drei Komponenten • Identität, • Attribute, • Operationen, die von anderen Objekten nur über eine klar definierte Schnittstelle zugänglich sind.	Ein Objekt besteht aus einem *Kristall*, der Objektidentität enthält, und mindestens einem *Kontext*, der mit dem Kristall verbunden ist. Dort sind die Attribute und Operationen des Objektes enthalten.
Änderungen von Objektbeschreibungen	Ein Objekt ist *statisch* an seine Klasse gebunden. Dadurch sind die Attribute und die Operationen, durch die das Objekt beschrieben wird, fest vorgegeben. Zur Laufzeit können keine Attribute oder Operationen neu hinzugenommen bzw. entfernt werden.	Ein Kristall, der Repräsentant eines Objektes in OPAQUE, kann zur Laufzeit an zusätzliche Kontexte gebunden werden oder kann Rollen an Kontexten aufgeben, so daß seine Objektbeschreibung sich *dynamisch* ändern kann. Dadurch kann ein Objekt eine *Evolution* durchlaufen.
Kommunikation und Kooperation	Die Datenstrukturen der Objekte sind rein privat. Daher ist keine Kooperation zwischen den Objekten möglich. Informationen können nur durch Kommunikation ausgetauscht werden.	Die Datenstrukturen der Objekte sind in Kontexten abgelegt. Diese gehören nicht notwendigerweise nur zu einem Kristall (Objekt), sondern eventuell zu mehreren, an der Datenstruktur begründet interessierten Objekten. Dadurch überschneiden sich die Datenbereiche dieser Objekte in einem Überlappungsbereich, über den die beiteiligten Objekte kooperieren können. Das bedeutet, daß Teile der Datenstruktur eines Objektes nach außen kontrolliert sichtbar werden. Daneben besteht natürlich noch die Möglichkeit, durch Senden und Empfangen von Nachrichten zu kommunizieren.

Tabelle 3.1: Vergleich zwischen klassischer Objektorientierung und OPAQUE

für das Objekt. Deshalb werden die Begriffe 'Kristall' und 'Objekt' in Zusammenhang mit OPAQUE oft in gleicher Weise verwendet, d.h. wenn wir in OPAQUE von Objekten reden, dann meinen wir zunächst einmal Kristalle, über die wir auf die weiteren Objektkomponenten zugreifen können.

Da ein Kristall allein nur eine Objektidentität darstellt, gehört er auch keiner speziellen Klasse an. Wie wir beispielsweise in den Abb. 3.17 und 3.18 gesehen haben, kann ein Kristall über Rollen an zusätzlichen Kontexten teilnehmen oder auch eine Rolle und damit einen Kontext aufgeben. Eine Klassenzugehörigkeit kann man nur dem Kristall in Zusammenhang mit den an ihm hängenden Kontexten (also dem Gesamtobjekt) zuweisen, nicht aber dem Kristall selbst. Eine alleinige Teilnahme eines Kristalls am Vererbungsmechanismus ist daher auch sinnlos.

Kontexte und Kontextschablonen

Kontexte enthalten *Strukturvariable* und *Operationen* zur Struktur- und Verhaltensbeschreibung von Objekten. Ein Kontext allein enthält keine Identität, da er für sich allein keinen Sinn ergibt. Er ist deshalb immer mit mindestens einem Kristall verbunden. Dazu enthält ein Kontext einige spezielle Strukturvariable, die sogenannten *Rollenslots*, die auf Kristalle verweisen, die am Kontext in einer bestimmten *Rolle* teilnehmen. Die übrigen Strukturvariablen werden auch *Zustandsvariablen* genannt.

Man kann sich einen Kontext als eine Szene aus einem Theaterstück vorstellen: sie bietet den Rahmen für das, was ist (Zustand) und was passieren kann oder wird (Operationen). Weiterhin bietet sie einige Rollen an, die später einmal durch Schauspieler ausgefüllt werden müssen. Sie sind es, die die Beschreibungen des Autors in eine konkrete Handlung umsetzen. Die Schauspieler im Rahmen der Kontexte sind die Kristalle; die Handlungsbeschreibung ist in den Operationen festgelegt, die auf Basis des Handlungsumfeldes – dem durch die Strukturvariablen beschriebenen Zustand – ausgeführt werden.

Ein Kontext enthält also die Struktur- und Verhaltensfacette einer Situation und kann nicht unabhängig von einem Kristall existieren. Genauso wie in einer Theaterszene mehrere Schauspieler teilnehmen können (das hängt von der Szene ab), können auch mehrere Kristalle an einen Kontext gebunden sein. Wie wir oben gesehen haben, repräsentiert jeder dieser Kristalle ein Objekt, das diesen Kontext als Teil seiner Objektbeschreibung enthält. Der Kontext ist dann gleichzeitig Bestandteil mehrerer solcher Beschreibungen; die beteiligten Objekte haben also einen gemeinsamen Datenbereich, über den sie kooperieren können. In diesem Zusammenhang sei noch einmal auf Abb. 3.20 verwiesen, wo vier Objekte über einen Kontext `Richtlinienwert` kooperieren können.

In der Praxis werden in der Regel verschiedene Kontexte derselben Bauart verwendet werden. So werden wir für das Beispiel aus dem Umweltszenario mehrere Kontexte zur Beschreibung von Richtlinien, Richtlinienwerten u.ä. benötigen. Aus diesen praktischen Erwägungen und zur Vermeidung von Redundanzen, ist es sinnvoll *Kontextschablonen* anzugeben, die die in den Kontexten verwendeten Zustandsvariablen, Rollenslots und Operationen definieren. Ein Kontext entsteht dann dadurch, daß eine Kontextschablone instanziiert wird.

Abb. 3.21 gibt ein Beispiel für die Definition einer Kontextschablone in OPAQUE. Es wird eine Schablone `Richtlinienwert` mit Struktur und Operationen definiert. Sie enthält sechs Strukturvariable, wovon die ersten beiden (`untergrenze`, `obergrenze`) dazu verwendet werden, Zah-

```
define context template Richtlinienwert  {
   structure
      { Number           untergrenze, obergrenze;
        Chemischer-Stoff stoff;
        Messmethode      methode;
        Masseinheit      einheit;
        Richtlinie       richtlinie;
      }
   operations
      { Number gebe-Untergrenze ()
           { var   { }
             roles { stoff, richtlinie }
             code  { return(untergrenze) }
           }
        Number gebe-Obergrenze ()
           { var   { }
             roles { stoff, richtlinie }
             code  { return(obergrenze) }
           }
        Number gebe-Mittelwert ()
           { var   { Number mw }
             roles { stoff, richtlinie }
             code  { mw := (untergrenze + obergrenze) / 2;
                     return(mw);
                   }
           }
      }
}
```

Abbildung 3.21: Definition einer Kontextschablone `Richtlinienwert` in OPAQUE

lenwerte aufzunehmen (Typ `Number`), während die übrigen als Rollenslots dienen, um Objekte der Klassen `Chemischer-Stoff`, `Messmethode`, `Masseinheit` und `Richtlinie` aufzunehmen. Dies modelliert genau die Struktur der Kontextschablone `Richtlinienwert` aus Abb. 3.20.

Außerdem werden drei einfache Operationen definiert. Neben den temporären Variablen, die mit Hilfe des Schlüsselworts `var` deklariert werden, und dem eigentlichen Code der Operation werden mit Hilfe von `roles` diejenigen Rollen bestimmt, über die auf diese Operation zugegriffen werden darf. Logischerweise müssen die hierin genannten Rollen eine Teilmenge der unter `structure` angebotenen Rollen der Schablone sein.

Um nun einen Kontext mit Hilfe einer Kontextschablone zu generieren, muß man ein Objekt einer Klasse erzeugen, die diese Schablone in ihrer Definition enthält. Klassendefinition und Objekterzeugung werden im folgenden Abschnitt erläutert. Wichtig dabei ist, daß man erkennt, daß ein Kontext nur in Zusammenhang mit einem Objekt bzw. einem Kristall Sinn macht, da ein Kontext für sich allein keine Identität besitzt. Man kann auf ihn nur über einen der Kristalle zugreifen, die eine Rolle an ihm spielen. Ein Kontext wird deshalb über den Pfad

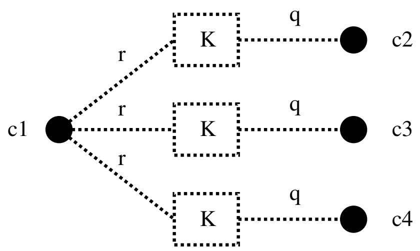


Abbildung 3.22: Kristall – Kontext Anbindungen (1)

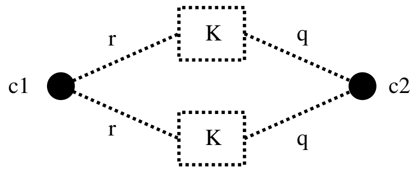


Abbildung 3.23: Kristall – Kontext Anbindungen (2)

Kristall – Rollenname – Schablonenname

angesprochen. Pfade, die auf mehrere Kontexte verweisen, sind erlaubt: Abb. 3.22 zeigt eine solche Situation. Die drei dort abgebildeten Kontexte können über den Kristall *c1* nicht eindeutig referenziert werden, da sie mit ihm über dieselbe Rolle verbunden sind und auch noch derselben Schablone entstammen. Dagegen ist jeder der Kontexte eindeutig über einen der Kristalle *c2*, *c3* und *c4* ansprechbar.

Eine ebenfalls erlaubte Situation zeigt Abb. 3.23. Die beiden dort vorhandenen Kontexte der Schablone *K* sind allerdings über keinen der beiden Kristalle *c1*, *c2* eindeutig referenzierbar. Derartige Situationen sind im allgemeinen ungünstig, und es gibt wenige praktische Beispiele, in denen eine solche Situation sinnvoll ist: es macht keinen Sinn, zwei Objekte über zwei gleichartige Datenbereiche kooperieren zu lassen.

Klassen

Ähnlich wie es sinnvoll ist, Kontextschablonen zu definieren, die man zu instanziieren kann, um einen Kontext zu erhalten, kann man in OPAQUE *Objektschablonen* bzw. *Klassen* definieren. Dadurch können leicht gleichartige Objekte, also Kristalle mit der gleichen Sammlung von beteiligten Kontexten, erzeugt werden.

Abb. 3.24 zeigt die Definition der Klasse `Chemischer-Stoff`, die in Abb. 3.16 graphisch spezifiziert wurde. Es wird prototypisch angegeben, an welchen Kontexten mit der dazugehörigen Rolle ein Objekt der Klasse teilnimmt. Das Konstrukt

`Chemischer-Stoff_Ktxt@Stoff`

besagt, daß ein Objekt der Klasse an einem Kontext der Schablone `Chemischer-Stoff_Ktxt` die Rolle `Stoff` spielt.

```
define class template Chemischer-Stoff  {
   structure
      Chemischer-Stoff_Ktxt@Stoff;
      Rohstoff@Rohstoff;
}
```

Abbildung 3.24: Definition einer Klasse `Chemischer-Stoff` in OPAQUE

```
define class template Klassenname {
   superclass
      Name der Oberklasse;
   structure
      Kontextschablonenname@Rollenname;
      Kontextschablonenname@Rollenname;
      ...
}
```

Abbildung 3.25: Definition einer Klasse in OPAQUE

Vererbungsbeziehungen können dadurch ausgedrückt werden, daß man eine in der Klassendefinition Oberklasse angibt. Dies geschieht mit Hilfe des Schlüsselwortes `superclass`. Die Unterklasse erbt dann alle Rollenanbindungen der Oberklasse und ergänzt diese eventuell mit zusätzlichen in ihrem `structure` Block. Die allgemeine Notation einer Klassendefinition sieht dann wie in Abb. 3.25 aus.

Ein neues Objekt einer Klasse kann durch

Bezeichner`.create()`

erzeugt werden, wobei der *Bezeichner* eine Referenz auf das neue Objekt enthält. Mit der Erzeugung eines Objektes entsteht ein Kristall, sowie Kontexte der in der Klassendefinition auf geführten Schablonen. Der Kristall nimmt an den Kontexten in der jeweils spezifizierten Rolle teil.

Programmaufbau in OPAQUE

Ein OPAQUE-Programm gliedert sich in die folgenden Abschnitte:

- *Deklaration globaler Variable*
- *Definition von Kontextschablonen und Klassen*
- `main`

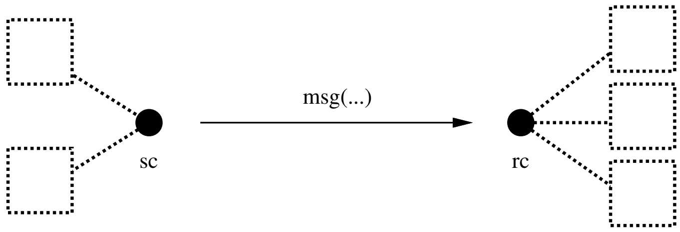


Abbildung 3.26: Kommunikation zwischen zwei OPAQUE-Objekten

- *Hauptprogramm-Block*

Dabei ist der zentrale Teil der, der die Definitionen der Kontextschablonen und der Objektklassen enthält. Er beinhaltet weitgehend der Code der Implementierung. Das Hauptprogramm dient nur dazu, die Interaktion der Objekte anzustoßen.

3.3.3 Dynamische OPAQUE-Konzepte

Im vorangegangenen Abschnitt haben wir die statischen Konzepte von OPAQUE, insbesondere auch die Deklaration von Kontextschablonen und Objektklassen erläutert. In dem nun folgenden Abschnitt wollen wir zeigen, wie nun OPAQUE-Objekte

- auf Nachrichten reagieren und
- wie sie mit zusätzlichen Kontexten verbunden oder von Kontexten getrennt werden können.

Nachrichtenmechanismus

Wir wollen hier beschreiben, was passiert, wenn ein Objekt einem anderen eine Nachricht schickt. Dies ist für uns von zentralem Interesse, da das neben der Kooperation die einzige Möglichkeit ist, Informationen zwischen Objekten auszutauschen. Während eine Kooperation nur zwischen Objekten mit sich überlappenden Kontexten möglich ist, können alle Objekte uneingeschränkt miteinander kommunizieren. Deshalb ist dies der wesentliche Interaktionsmechanismus eines OPAQUE-Programms.

Ausgangspunkt unserer Betrachtungen ist die folgende Situation: ein Objekt *sc* sendet eine Nachricht *msg* an ein Objekt *rc*. Eine solche Nachricht wird z.B. dadurch ausgelöst, daß im Code eines der Kontexte von *sc* der Aufruf

```
x.msg(...)
```

steht, wobei die Variable x auf das Objekt *rc* verweist. Abb. 3.26 zeigt die genannte Situation. Die Kristalle, die wir nach ihren zugehörigen Objekten ebenfalls *sc* und *rc* nennen, sind Nachrichtenquelle bzw. -senke. Für sich allein haben diese beiden Kristalle aber keine Möglichkeit, auf die Nachricht zu reagieren, da sie selbst nur die Objektidentität und keinerlei Verhaltensbeschreibung tragen. Letztere ist aber in den mit den Kristallen verbundenen Kontexten enthalten.

Wir werden im folgenden beschreiben, wie sich die Reaktion der Objekte auf die Nachricht mit Hilfe der in den Kontexten vorhandenen Beschreibungen synthetisieren läßt.

Zunächst einmal müssen wir die in den mit *rc* verbundenen Kontexten nach den Operationen suchen, die für eine Reaktion auf die Nachricht in Frage kommen. In diesen Kontexten suchen wir daher nach Operationen mit dem Namen *msg*. Dadurch erhalten wir u.U. eine (zunächst noch ungeordnete) Sammlung von Operationen, die als Reaktion auf die Nachricht auszuführen sind.

Da OPAQUE als System mit einer Aktivitätsbahn ausgelegt ist, ist es notwendig, die angesammelten Operationen in eine Reihenfolge zu bringen und sie nacheinander auszuführen. Zahlreiche Beispiele belegen, daß nicht jede beliebige Reihenfolge auch sinnvoll sein kann, da eine Operation z.B. ein Ergebnis einer anderen Operation verarbeiten möchte. In der Regel bestehen also einige Abhängigkeitsbeziehungen zwischen den Operationen. Diese Abhängigkeiten werden in der syntaktischen Form

Kontextname`@`*Rollenname*

im Operationscode spezifiziert. Sie bedeuten, daß das Ergebnis der Teiloperation der genannten Stelle vom System verfügbar gehalten werden, also schon berechnet sein muß. Man kann diese Abhängigkeiten als vom Programmierer eingefügte Constraints betrachten, die die Beliebigkeit der Operationsreihenfolge einschränken.

In der Regel verbleiben mehrere mögliche Ablauffolgen, von denen das Laufzeitsystem OPAQUEs eine auswählt und ausführt. Jede Operation liefert dabei ein einzelnes Ergebnis, das in einer sogenannten *Ergebnispipeline* gesammelt und abrufbar bereit gehalten wird. Die Ergebnispipeline ist eine Variable, die vom System automatisch beim Senden einer Nachricht erzeugt wird und über die die von der Nachricht betroffenen Kontexte kooperieren können. Auf die Ergebnispipeline kann mit Hilfe einer der drei folgenden Ausdrücke zugegriffen werden:

- `result.current`
- `result.before` *Kontextname*`@`*Rollenname*
- `result.after` *Kontextname*`@`*Rollenname*

Dabei verweist `result.current` auf das zuletzt abgelegte Ergebnisobjekt. Für `result.before` und `result.after` muß noch ein Bedingung spezifiziert werden. Man erhält dann eine Referenz auf das Ergebnisobjekt, das unmittelbar vor bzw. nach der genannten Bedingung in die Pipeline gelegt wurde.

Dynamisches Hinzufügen und Wegnehmen von Kontexten

Wie wir in Abschnitt 3.3.1 gesehen haben, ist es ein sehr nützliches Instrument, Kontexte dynamisch an einen Kristall binden zu können. Das bedeutet, daß man Kontexte von einem Kristall lösen, aber ihm auch neue anhängen kann. OPAQUE bietet dafür die beiden Nachrichten

- *Objekt*`.leave (`*Kontextname*`@`*Rollenname*`)`

- *Objekt*.`enter` (*Kontextreferenz*, *Rollenname*)

an, wobei *Objekt* eine Referenz auf ein Objekt, *Kontextname* der Name einer Kontextschablone, *Rollenname* der Name einer Rolle dieser Schablone und *Kontextreferenz* eine Referenz auf einen Kontext ist. Letztere kann man durch die Nachricht

Objekt.`at` (*Kontextname*`@`*Rollenname*)

erhalten, da ein Kontext eindeutig durch über das Objekt (an dem der Kontext hängt), eine Rolle (in der der Kristall am Kontext teilnimmt) und einen Schablonennamen zu referenzieren ist.

Die Nachricht

```
c.leave (k@r)
```

entfernt den Kristall `c` aus der Rolle `r` des Kontextes der Schablone `k`.

Dagegen fügt die Nachricht

```
c1.enter (c2.at(k@r2),r1)
```

den Kristall `c1` in die Rolle `r1` des Kontextes der Schablone `k` ein, an dem der Kristall `c2` in der Rolle `r2` hängt.

Bemerkung

Es ist zu erwarten, daß gerade die großen Freiräume, die die dynamischen Konzepte OPAQUEs bieten, eine flexible, auch komplexen Anforderungen genügende Modellierung der dynamischen Aspekte eines Systems erlauben. Die dafür zur Verfügung stehenden Mittel sind

- zum einen der Nachrichtenmechanismus OPAQUEs, in dem die Gesamtreaktion auf eine Nachricht aus der Umgebung eines Kristalls, also den Teilreaktionen der Kontexte heraus generiert wird, und
- zum anderen die Möglichkeiten die Umgebung eines Kristalls durch Hinzufügen und Wegnehmen von Kontexten zu verändern und dadurch flexibel zu gestalten.

3.4 OPAQUE-EM

3.4.1 Idee einer Entwurfsmethode für OPAQUE

Der Hauptunterschied zwischen dem *Kontextmodell* OPAQUE und den herkömmlichen *objektorientierten Modellen* liegt darin, daß OPAQUE erlaubt, kontextsensitive Aspekte zu beschreiben. Die für die herkömmlichen objektorientierten Modelle entwickelten Entwurfsmethoden, wie beispielsweise OMT, bieten aber keinerlei Ausdrucksmittel an, um Kontextsensitivität zu modellieren oder sichtbar zu machen. Beispielsweise hat OMT keine Möglichkeit, den

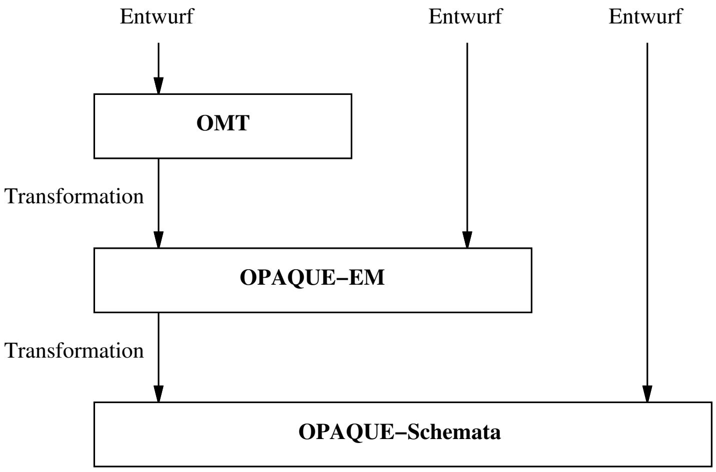


Abbildung 3.27: OPAQUE-EM als hybride oder als eigenständige Entwurfsmethode

Wechsel eines Objektes von einer Klasse in eine andere darzustellen, da dies in der klassischen Objektorientierung nicht möglich ist. Weiterhin fehlen OMT Elemente, die es erlauben, Kontexte oder verwandte Konzepte zu modellieren, die uns die konzeptuellen OMT-Schemata leicht auf logische OPAQUE-Schemata abbilden lassen. Aus den genannten Gründen entwickelte O. Kopp in [Kop93] eine für die Bedürfnisse OPAQUEs geeignete Entwurfsmethode, genannt *OPAQUE-EM*.

Abb. 3.27 [Kop93] zeigt die Konzeption OPAQUE-EMs als hybride und als eigenständige Entwurfsmethode: entweder man baut auf einer OMT-Modellierung auf oder verwendet OPAQUE-EM direkt zum Entwurf von OPAQUE-Schemata. Wir werden in dieser Arbeit die Vorgehensweise wählen, die eine Modellierung mittels OMT vorsieht, die anschließend in eine OPAQUE-EM-Modellierung transformiert wird. Diese Umsetzung erfolgt mechanisch und wird in Abschnitt 3.5 beschrieben.

Wir beschränken uns im bei der Behandlung von OPAQUE-EM und der Transformation von OMT nach OPAQUE-EM allein auf den statischen Aspekt, d.h. wir begnügen uns damit, eine Objektmodellierung in OPAQUE-EM zu beschreiben und später die Umsetzung eines OMT-Objektmodells in ein OPAQUE-EM-Objektmodell zu erklären. OPAQUE-EM bietet keine Entwurfsunterstützung für dynamische Abläufe. Für die Modellierung der dynamischen Aspekte werden wir auf die im Rahmen von OMT vorgestellten Ausdrucksmittel (siehe Abschnitt 3.2) bzw. auf einfache, unmittelbar einsichtige Darstellungen zurückgreifen, die wir im Rahmen der dynamischen Modellierung vorstellen.

In den folgenden Abschnitten beschreiben wir, wie die statischen OPAQUE-Konzepte, die wir in Abschnitt 3.3 vorgestellt haben, mittels OPAQUE-EM dargestellt werden können.

3.4.2 Kontextschablonen

Kontextschablonen gliedern sich in drei Teile:

- den Namen der Kontextschablone,

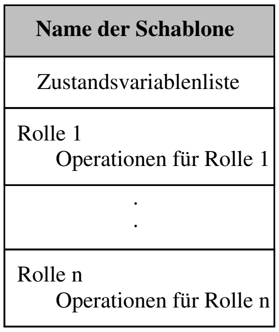


Abbildung 3.28: Eine Kontextschablone in OPAQUE-EM

Abbildung 3.29: Beispiel einer Kontextschablone `Richtlinie_Ktxt`

- die Zustandsvariable und
- die Rollen mit ihren zugehörigen Operationen.

Entsprechend gliedert sich auch die grafische Darstellung einer Kontextschablone in OPAQUE-EM, wie sie in Abb. 3.28 zu sehen ist:

- im oberen, grau unterlegten Teil steht der Name der Kontextschablone;
- anschließend folgt eine Liste der von der Kontextschablone verwendeten Zustandsvariable;
- danach schließt sich eine Liste der von der Schablone angebotenen Rollen mit den ihr zugehörigen Operationen an.

Abb. 3.29 zeigt ein Beispiel einer Kontextschablone in OPAQUE-EM. Im allgemeinen verzichten wir auf eine Auflistung der zu einer Rolle gehörenden Operationen, da diese sich erst im Zuge der dynamischen Modellierung ergeben werden.

3.4.3 Kristallschablonen und Klassen

Da wir keine Modellierung auf Instanzenebene durchführen wollen, Kristalle aber Instanzen sind, beschreiben wir mit OPAQUE-EM keine Kristalle, sondern Kristallschablonen. Sie werden durch Kreise oder Ovale dargestellt, die den Namen der Kristallschablone tragen (siehe Abb. 3.30).

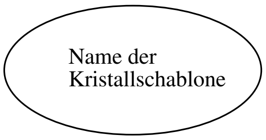


Abbildung 3.30: Eine Kristallschablone in OPAQUE-EM

Abbildung 3.31: Anknüpfung einer Kristallschablone an eine Kontextschablone über eine Rolle

Wie wir später bei der Transformation von OMT-Objektmodellen nach OPAQUE-EM sehen werden, repräsentieren die Kristallschablonen (in Verbindung mit den Rollen, an denen sie an Kontexten teilnehmen können) die Objektklassen in OPAQUE. Wie wir schon bei der Vorstellung OPAQUEs erläutert haben, besitzt OPAQUE ein dynamisches Klassenkonzept, d.h. ein Kristall kann problemlos von einer Klasse in eine andere überwechseln. Da wir hier mit OPAQUE-EM nur statische Strukturen beschreiben, sollte man bei der Betrachtung der Kristallschablonen als Repräsentanten der Objektklassen bedenken, daß es sich hierbei um eine Ausgangssituation handelt und daß es möglich sein kann, daß ein Kristall, der ursprünglich zu der Klasse A gehörte, im Laufe seiner 'Lebenszeit', also seiner Verweilzeit im System, in eine Klasse B überwechselt.

3.4.4 Rollen und Kardinalitäten

Kontexte bieten Rollen an, über die ein Kristall sich an den jeweiligen Kontext anhängen kann. Diese Anbindungen beschreiben wir, indem wir Verbindungen zwischen den Rollenslots der Kontextschablonen und den Kristallschablonen herstellen (siehe Abb. 3.31).

Es ist sinnvoll, diese Verbindungen zwischen Kristall- und Kontextschablonen noch weiter zu charakterisieren. Dazu dienen zwei verschiedene Arten von Kardinalitäten:

1. Die eine Kardinalität gibt aus Sicht des Kristalls bzw. der Kristallschablone an, an wievielen Kontexten der verbundenen Kontextschablone der Kristall (in der genannten Rolle) teilnehmen kann. Diese Kardinalität steht unmittelbar neben der Kristallschablone.
   In Abb. 3.32 bedeutet dies beispielsweise, daß ein Richtlinienkristall in der Kardinalität $< 1, * >$ an Kontexten der Schablone `hat_Richtlinienregelung_Ktxt` in der Rolle `Richtlinie` teilnehmen kann, d.h. ein Richtlinienkristall nimmt an *mindestens* einem solchen Kontext in dieser Rolle teil, möglicherweise sogar an mehreren. In diesem Beispiel modellieren wir damit die Tatsache, daß eine Richtlinie mindestens eine Richtlinienregelung enthält, theoretisch aber beliebig viele.

2. Die andere Kardinalität nimmt den Blickwinkel des Kontextes ein. Sie steht unmittelbar neben der Kontextschablone und gibt an, wieviele Kristalle der verbundenen Kristall-

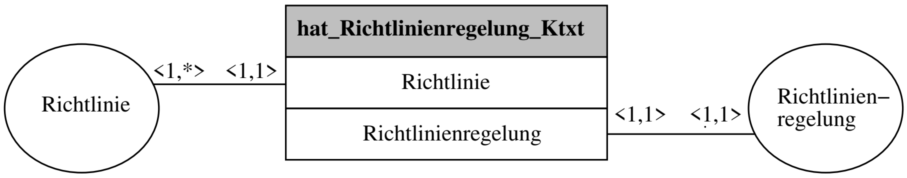


Abbildung 3.32: Beispiel einer Anknüpfung von Kristall- und Kontextschablonen mit Kardinalitäten

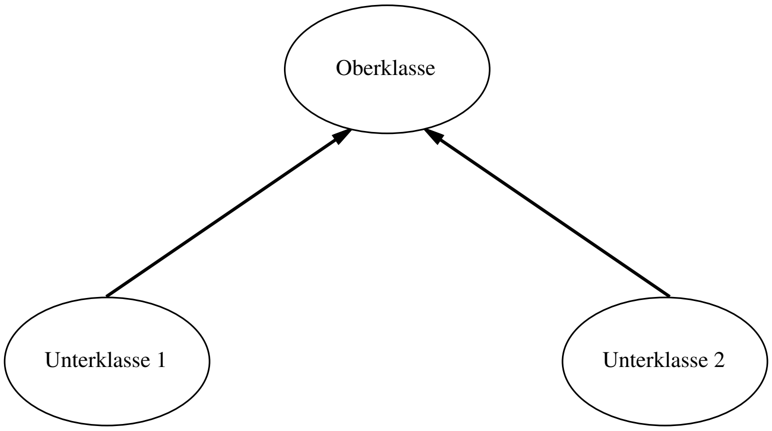


Abbildung 3.33: Vererbungsbeziehungen in OPAQUE-EM

schablone in der angebotenen Rolle teilnehmen können. Dabei erlaubt OPAQUE nur die Kardinalitäten $< 0, 1 >$ und $< 1, 1 >$.
In dem Beispiel aus Abb. 3.32 bedeutet dies beispielsweise für die Rolle `Richtlinie` eines Kontextes der Schablone `hat_Richtlinienregelung_Ktxt`, daß sie mit *genau* einem Kristall der Kristallschablone `Richtlinie` verbunden sein muß. Die Kardinalitäten zwischen der Kristallschablone `Richtlinienregelung` und der Kontextschablone `hat_Richtlinienregelung` modellieren das Faktum, daß eine Richtlinienregelung eindeutig einer Richtlinie zuzuordnen ist.

3.4.5 Vererbungsbeziehungen

Vererbungsbeziehungen beschreiben wir in OPAQUE-EM wie in Abb. 3.33 dargestellt. Wie wir oben schon erwähnt haben, repräsentieren die Kristallschablonen vom statischen Gesichtspunkt her die Objektklassen in OPAQUE. Daher werden Vererbungsbeziehungen durch Pfeile zwischen Kristallschablonen etabliert. Das bedeutet, daß die Kristallschablonen der Unterklassen alle Kontextschablonenanbindungen der Kristallschablone der Oberklasse übernehmen und sie um weitere, unterklassenspezifische ergänzen.

Darüberhinaus bietet OPAQUE und auch OPAQUE-EM die Möglichkeit, Mehrfachvererbung zu modellieren. Da wir dieses Element für den Entwurf innerhalb dieser Arbeit nicht benötigen, sei der interessierte Leser auf die Literatur verwiesen ([Kop93], [Lau93]).

3.5 Transformation von OMT nach OPAQUE-EM

3.5.1 Vorgehensweise

Wir beschreiben nun, wie man ein Objektmodell in OMT-Notation in die OPAQUE-EM-Notation übertragen kann. Die Übertragung erfolgt separat für jeden Baustein eines OMT-Objektmodells. Deshalb werden wir in den folgenden Abschnitten nur die Transformation der für uns wesentlichen Elemente von OMT beschreiben. Das sind:

- Objektklassen,
- Assoziationen,
- Aggregationen und
- Vererbungsbeziehungen.

3.5.2 Transformation von Objektklassen

In Abb. 3.34 sehen wir links die OMT-Objektklasse `Richtlinie`. Sie enthält neben ihrem Namen eine Reihe von Attributen, sowie zwei Operationen. Um nun Richtlinienobjekte in OPAQUE-EM darstellen zu können, benötigen wir Elemente, die einerseits die Identität der Objekte, andererseits die Attribute und Operationen aufnehmen können. Einzige mögliche Träger von Identitäten in OPAQUE sind Kristalle; Strukturbeschreibungen aus Attributen und Operationen können dagegen nur durch Kontexte repräsentiert werden. Daraus entsteht nun die folgende Vorschrift für die Transformation von OMT-Objektklassen nach OPAQUE-EM:

1. Für eine OMT-Objektklasse K schaffen wir eine Kristallschablone mit dem Namen K.
2. Weiterhin erzeugen wir eine Kontextschablone K`_Ktxt`[5], die Zustandsvariable mit den Namen der Attribute der OMT-Klasse K hat. Außerdem bietet K`_Ktxt` eine einzige Rolle mit dem Namen K an. Unter diese Rolle fallen auch die Operationen der Klasse K.
3. Die Kristallschablone K wird mit der Kontextschablone K verbunden und zwar mit den Kardinalitäten $< 1, 1 >$ auf beiden Seiten. Dadurch ist gewährleistet, daß es sich bei einem Kontext der Schablone K`_Ktxt` um einen 'privaten' Kontext für den jeweiligen Kristall handelt.

Abb. 3.34 zeigt die Transformation der Klasse `Richtlinie`.

3.5.3 Transformation von Assoziationen

Eine Assoziation beschreibt mögliche Beziehungen zwischen Objekten der an der Assoziation beteiligten Klassen. Auf den Gesprachgebrauch OPAQUEs umgesetzt bedeutet das, daß wir mögliche Beziehungen zwischen Kristallen beschreiben müssen. Wir können das mit Hilfe von

[5] O. Kopp bezeichnet Kontexte, die auf diese Art und Weise entstehen, als *Eigenkontexte* (siehe [Kop93]).

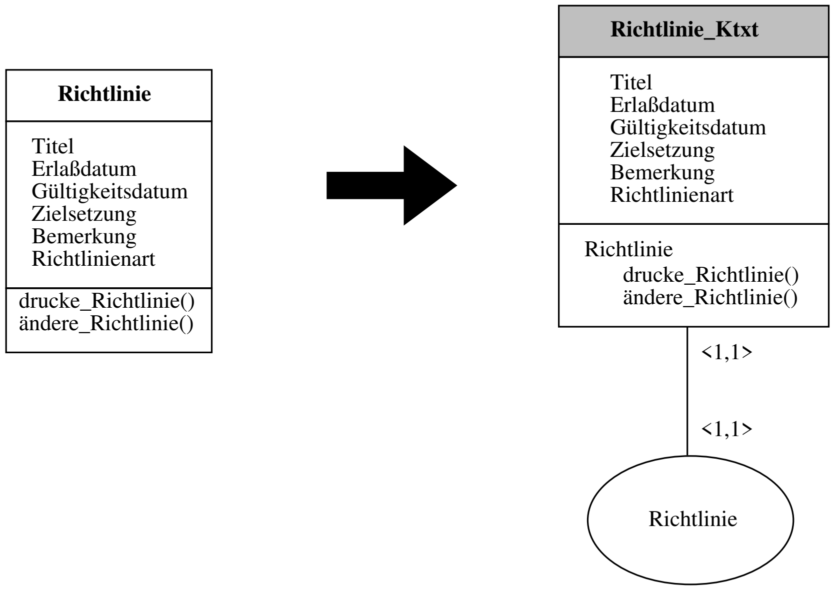


Abbildung 3.34: Transformation einer OMT-Klasse in eine OPAQUE-EM-Klasse

Kontexten tun, wobei ein Kontext genau eine Beziehung repräsentiert. Eine Assoziation wird deshalb als Kontextschablone dargestellt. Daraus ergibt sich die folgende Transformationsvorschrift:

1. Für eine Assoziation A erzeugen wir eine Kontextschablone A_`Ktxt`.
2. Falls die Assoziation Attribute trägt, so werden diese als Zustandsvariablen innerhalb der Schablone A_`Ktxt` dargestellt.
3. Für jede Klasse, die an der Assoziation A teilnimmt, bietet A_`Ktxt` eine Rolle mit dem Namen der betreffenden Klasse an.
4. Die Kristallschablonen, die die an der Assoziation beteiligten Klassen repräsentieren, werden mit der entsprechenden Rolle der Kontextschablone A_`Ktxt` verbunden.
5. Die Kardinalitäten der Assoziation werden entsprechend dem Schema übertragen, das sich aus Abb. 3.35 ergibt.

3.5.4 Transformation von Aggregationen

Man kann Aggregate als eine besondere Form einer Assoziation von mehreren Objektklassen betrachten. Dabei gibt es eine Klasse, die eine hervorgehobene Rolle spielt: es ist die Klasse, die das Aggregat repräsentiert. Die Darstellung einer Aggregation in OPAQUE-EM folgt der eben beschriebenen Betrachtung und so wird sie analog einer Assoziation beschrieben:

1. Für die Klasse K, die als Aggregat aus mehreren Klassen modelliert ist, erzeugen wir gemäß Abschnitt 3.5.2 eine Kristallschablone mit dem Namen K.

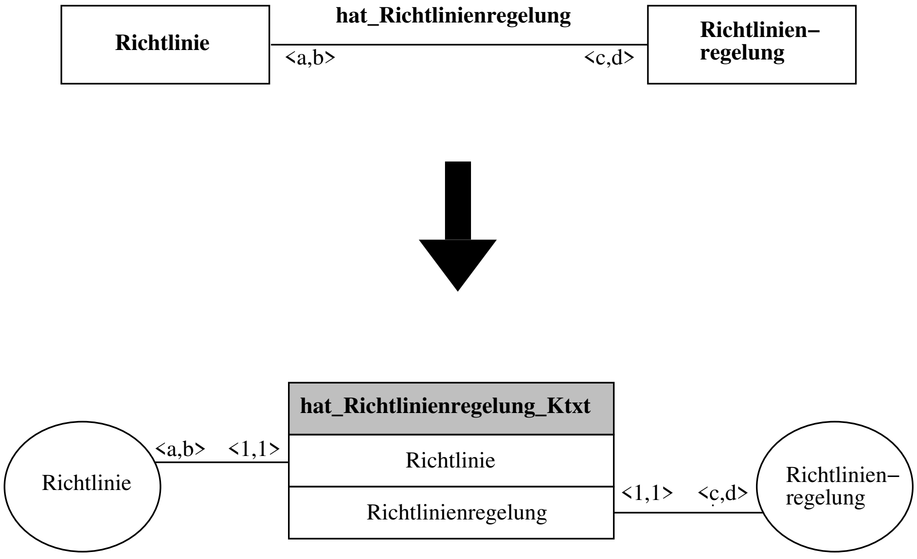


Abbildung 3.35: Transformation einer OMT-Assoziation nach OPAQUE-EM

2. Für das Aggregat wird eine Kontextschablone K`_Aggregat` erzeugt. Sie bietet für jede Klasse, die an der Aggregation beteiligt ist, eine Rolle an, die den Namen der betreffenden Klasse trägt. Außerdem gibt es eine Rolle K.

3. Die Kristallschablone K und die Kontextschablone K`_Ktxt` werden über $< 1, 1 >$ Kardinalitäten verbunden. K`_Aggregat` hat für K den Charakter eines Eigenkontextes (siehe Abschnitt 3.5.2). Deshalb können Attribute und Operationen der OMT-Klasse K in diesem Kontext aufgenommen werden.

4. Die übrigen Kardinalitäten werden entsprechend dem Schema übertragen, das sich aus Abb. 3.36 ergibt.

Man beachte vor allem, daß die im OMT-Objektmodell ausgezeichnete Position der Klasse K in der OPAQUE-EM-Modellierung untergeht. Wenn man beispielsweise die in Abb. 3.36 dargestellte Transformation betrachtet, so kann man feststellen, daß sich die Rolle `Richtlinienwert` innerhalb der Kontextschablone `Richtlinienwert_Aggregat` strukturell in keiner Weise von den übrigen Rollen unterscheidet; allenfalls durch die Benennung sind noch Rückschlüsse auf die ursprüngliche Beziehung der Klassen zueinander möglich.

3.5.5 Transformation von Vererbungsbeziehungen

Die Übertragung einer Vererbungsbeziehung von OMT nach OPAQUE-EM ist trivial und erfolgt wie im Beispiel aus Abb. 3.37 dargestellt.

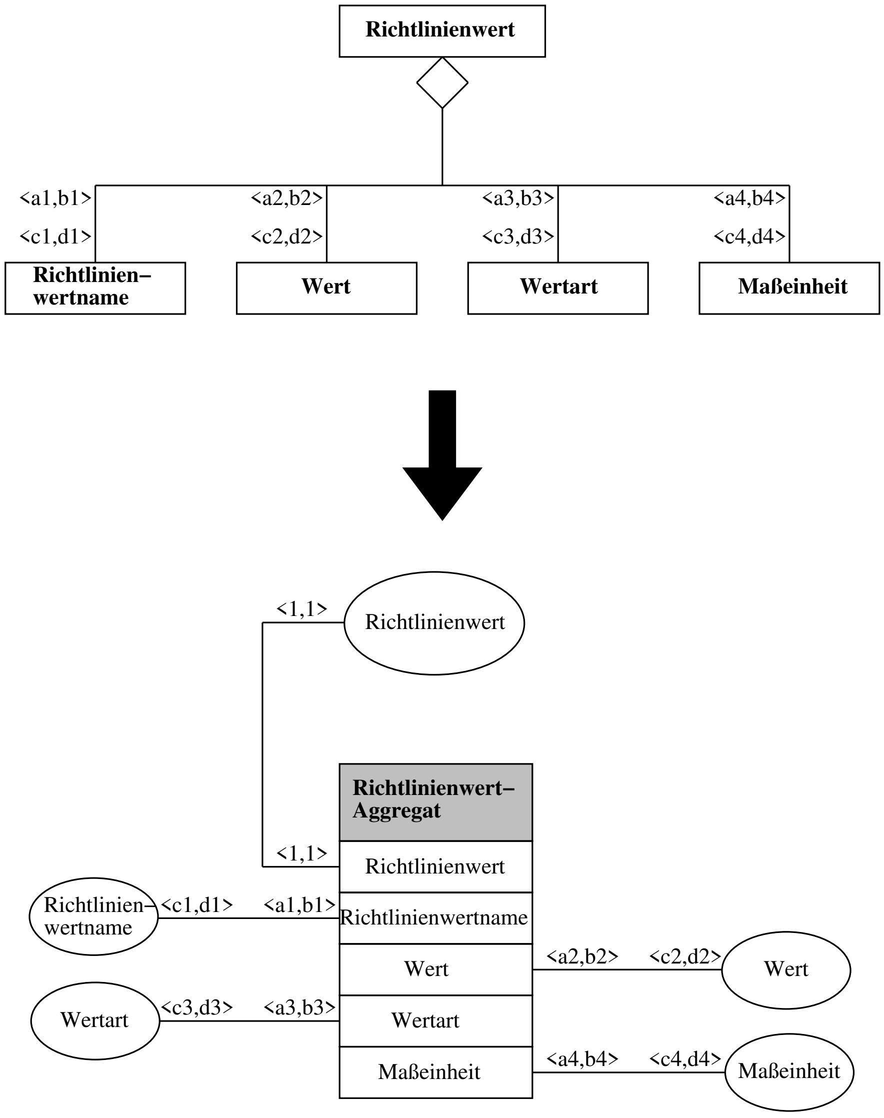


Abbildung 3.36: Transformation einer OMT-Aggregation nach OPAQUE-EM

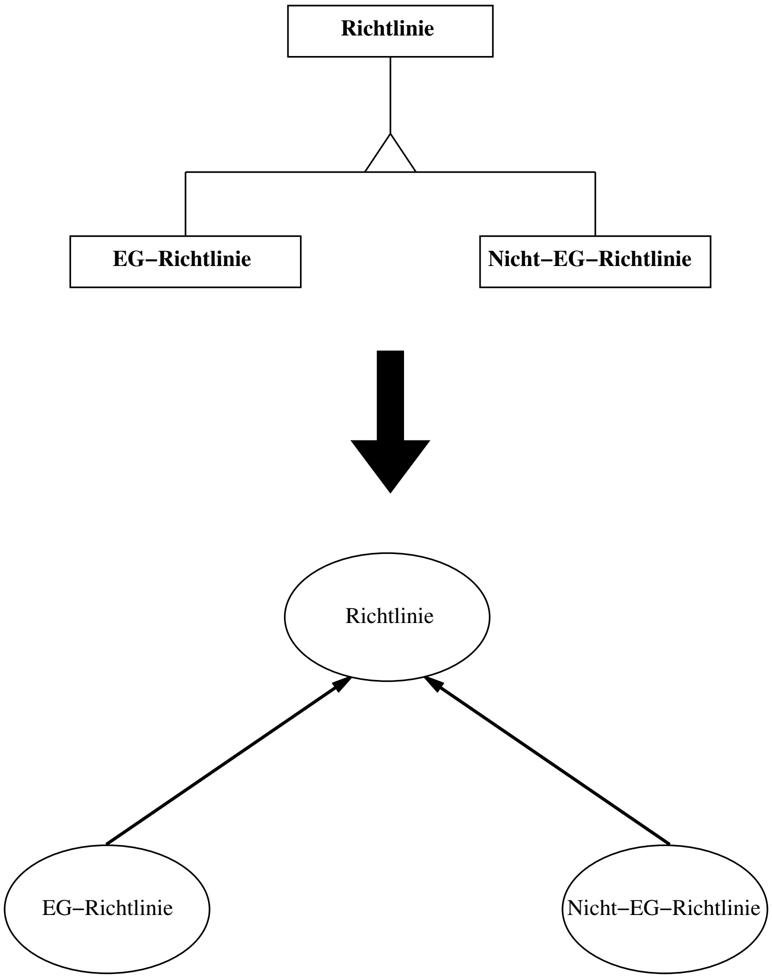


Abbildung 3.37: Transformation einer Vererbungsbeziehung nach OPAQUE-EM

Kapitel 4

Analyse der Diskurswelt

In diesem Kapitel wollen wir die in Kapitel 2 beschriebene Diskurswelt näher analysieren. Insbesondere wollen wir feststellen, auf welchen Grundlagen die im nächsten Kapitel folgende Modellierung aufbaut.

Dazu werden wir zunächst untersuchen, welche Arten von Daten unserer Modellierung zugrundeliegen. Dies ist notwendig, da in den Richtlinien immer wieder ähnliche Daten aufgeführt, aber teilweise mit ganz unterschiedlichen Begriffen bezeichnet werden. Daher werden wir in einem ersten Schritt eine Begriffsabgleichung durchführen.

Manche Richtlinien verwenden weiterhin sehr komplexe Datenstrukturen, wie z.B. eine *Grenzwertregelung*, während andere sehr einfache, weitgehend elementare Strukturen wie *Grenzwert*, *Einheit* u.ä. benutzen. Wir werden die komplexen Strukturen in elementare zerlegen und so einen Strukturabgleich der Richtlinien durchführen. Diese Festlegung geht natürlich einher mit dem Begriffsabgleich, da die dabei benutzten elementaren Felder in manchen Richtlinien bereits auftauchen und dort benannt werden.

Der angesprochenen Problematik eines kombinierten Begriffs- und Strukturabgleichs wollen wir uns im ersten Abschnitt dieses Kapitels, in der Datenanalyse zuwenden.

Im zweiten Abschnitt werden wir untersuchen, welche Benutzergruppen das Umweltinformationssystem verwenden. Das wird uns wichtige Hinweise geben, welche Anforderungen an das System gestellt werden. Anhand einiger typischer Benutzeranfragen wollen wir diese Anforderungen darstellen.

4.1 Datenanalyse

Grundlage der Analyse und der später folgenden Modellierung ist die Grenzwertbroschüre [LR92] der Landesanstalt für Umweltschutz, Baden-Württemberg. Darin sind in tabellarischer Form die Regelungen verschiedener Gesetze, Vorschriften, Leitlinien u.ä. aufgeführt. Wir werden diese einheitlich als *Richtlinien* bezeichnen. Wir beschränken uns auf die Richtlinien für die *Meßobjekte Luft, Wasser* und *Boden*. Dieser Ausschnitt umfaßt bereits die Komplexität der Diskurswelt. Eine Erweiterung des entstehenden Modells zur Erfassung der Richtlinien für Chemikalien, zum Lärmschutz oder für Störfälle sollte mit den hier beschriebenen Mitteln jederzeit möglich sein. Eine Liste der verwendeten Richtlinien findet sich in Anhang B.

Schadstoff/ Parameter	Bezugs- zeitraum	Regelung Grenzwert/ Leitwert	Einzuhalten ab	Analytik/ Referenzmethoden	Richtlinien- Nr., Titel
Blei (Pb)	Jahres- mittelwert	2 $\mu g/m^3$		Referenzmethode für die Analyse ist die Atomabsorptions- spektrometrie; der Analysenfehler bei der Bestimmung von Blei in den eingefan- genen Partikeln muß geringer als der Wert sein, der dem Bleige- halt der Luft von 0,1 $\mu g/m^3$ entspricht. [...]	82/884/EWG Richtlinie des Rates vom 3.12.82 be- treffend einem Grenzwert für den Blei- gehalt in der Luft
Schwebestaub	Winter 01.10.– 31.03.	den entsprechenden Grenzwerten für Schwefeldioxid zugeordneter Gren- zwert für Schwebes- taub in $\mu g/m^3$: $>$ 200 (für 130 $\mu g/m^3$ SO_2) (Median der im Winter gemessenen Tagesmittelwerte) $\leq$ 200 (für 180 $\mu g/m^3$ SO_2) (Median der im Winter gemessenen Tagesmittelwerte)	01.04.93	gravimetrische Methode	80/779/EWG [...]

Abbildung 4.1: Tabellenausschnitt aus den EG-Luftqualitätsnormen

4.1.1 Problematik nicht einheitlicher Richtlinienstrukturen

Die Richtlinien liegen uns durchgehend in Form von Tabellen vor (siehe z.B. Abb. 4.1). Jede Zeile einer solchen Tabelle enthält dabei eine Regelung für einen Schadstoff oder Parameter in Form eines Grenzwertes, eines Leitwertes, eines Prüfwertes, eines Qualitätszieles u.ä. Eine Zeile einer Tabelle wollen wir deshalb als eine *Richtlinienregelung* bezeichnen. Eine Tabelle und damit eine Richtlinie ist demzufolge eine Sammlung von Richtlinienregelungen.

Die Tabellen enthalten eine Strukturierung mittels ihrer Spalten. Jede Spalte enthält einen Titel, der die Art der Information beschreibt, die in dieser Spalte gegeben wird. Jede Zeile und damit jede Richtlinienregelung wird dadurch in verschiedene Informationsfelder unterteilt.

Wir wenden uns in diesem Abschnitt dieser Spaltenstruktur der Tabellen zu. Jede Richtlinie verwendet eine andere Gliederung, so daß sich sehr unterschiedliche Strukturen der Regelungen ergeben. Dabei ergeben sich die folgenden Probleme:

1. **Begriffsproblematik**
   Spalten, die dieselbe Information verwalten, werden in verschiedenen Richtlinien unterschiedlich bezeichnet. So sprechen manche Richtlinien von Regelungen für *Schadstoffe*, manche aber von Regelungen für *Parameter*. Wir wollen in Abschnitt 4.1.2 einheitliche

Begriffe einführen.

2. **Strukturproblematik**
   Manche Spalten enthalten nicht nur elementare Informationen, wie z.B. einen Zahlenwert oder den Namen eines Schadstoffes, sondern zusätzliche Daten, wie z.B. die Maßeinheit, in der der Stoff gemessen wird, oder ob es sich um einen Mittelwert, einen Median, einen 98%-Wert o.ä. handelt. Solche Spalten enthalten also Informationen mit einer komplexen, nicht-elementaren Datenstruktur. Ziel dieses Abschnittes ist es u.a., die komplexen Datenfelder in elementare zu zerlegen, um dadurch den Informationsgehalt einer Richtlinie klar beschreiben zu können. Vergleicht man die Richtlinientabellen mit den Relationen des relationalen Datenmodells, so entspricht dieses Vorgehen einer Überführung der Tabellen in 1. Normalform.

3. **Problematik unklarer Informationszuordnung**
   Einige Informationen sind manchmal in einer Spalte enthalten, ein anderes Mal in einer anderen. Beispielsweise ist in der Tabelle in Abb. 4.1 die Information über die Art des Richtlinienwerts (Mittelwert, Median, ...) bei der Regelung für Blei in der Spalte *Bezugszeitraum* ("Jahresmittelwert"), bei der Regelung für Schwebestaub dagegen in der Spalte *Regelung Grenzwert/Leitwert* ("Median der ...") aufgeführt.

   Die unklare Zuordnung von Daten zu Datenfeldern ist offenbar auch Ursache für die unter 2. diskutierte Strukturproblematik: Informationen, die in irgendeiner Weise zusammengehören, wurden in komplexen Datenfeldern abgelegt.

   Weiterhin existieren einige multifunktionale Datenfelder, wie beispielsweise das Feld *Bemerkung*, in dem Daten untergebracht werden, die man keiner anderen Rubrik der Richtlinie zuordnen will: *Bemerkung* enthält in manchen Richtlinien Daten über eine zur Regelung gehörende *Meßmethode*, während andere Richtlinien eigene Datenfelder für diese Art der Information vorsehen.

Als Ergebnis dieser Untersuchungen werden wir eine konsolidierte Struktur der Richtlinien erhalten, die nur noch einheitliche Begriffe für die Datenfelder verwendet, die stets atomaren Inhalt besitzen.

4.1.2 Vereinheitlichung und Definition der Begriffswelt

Grundlage unserer Untersuchungen sind die in Anhang B aufgeführten Richtlinienstrukturen. Wir werden im folgenden eine Semantik für die von uns verwendeten Grundbegriffe einführen. Sie bilden das Grundgerüst für die Analyse der Richtlinientabellen:

(a) Mit Ausnahme des Begriffs *Richtlinie* verweisen sie auf *elementare* Datenfelder, bilden also die kleinsten Informationseinheiten unserer Diskurswelt.

(b) Wenn wir die von uns beschriebenen elementaren Datenfelder als Bausteine voraussetzen, können wir komplexe, nicht-elementare Datenfelder in elementare zerlegen. Dadurch kann der Informationsgehalt jeder Richtlinientabelle beschrieben werden.

(c) Wir können die teilweise unklare Verteilung von Informationen auf mehrere Tabellenspalten (Datenfelder) auflösen.

Das Ergebnis der Analyse der Tabellenstrukturen mündet in eine konsolidierten Beschreibung der Tabellen, die sich im Anhang C befindet.

Richtlinie: Unter einer Richtlinie verstehen wir Gesetze, Vorschriften, Leitlinien, Normen, Verordnungen, Leitfäden u.ä., die Regelungen über den Wert von Schadstoffkonzentrationen, physikalisch-chemischen und mikrobiologischen Parametern enthalten. Diese Richtlinien werden von verschiedenen Körperschaften, wie beispielsweise der Europäischen Gemeinschaft (EG), den Gesetzesgebern in Deutschland, der Schweiz, in Österreich, Verbänden, wie dem VDI[1], oder internationalen Organisationen, wie der WHO[2], erlassen oder herausgegeben. Beispiele für solche Richtlinien sind

- die EG-Luftqualitätsnormen,
- die Technische Anleitung (TA) Luft der Bundesrepublik Deutschland,
- die VDI-Richtlinie 2310,
- die WHO-Luftqualitätsleitlinien.

Parameter: Ein Parameter ist die Bezeichnung für eine Beobachtungsgröße in Zusammenhang mit einem Meßobjekt (wie beispielsweise der Luft), die beschreibt, inwieweit ein Biosystem gefährdet ist. In unserem Szenario sind das

- die Konzentration eines Schadstoffes[3],
- einen physikalischen Parameter, wie *Temperatur* oder *Leitfähigkeit*,
- einen physikalisch-chemischen Parameter, wie den *pH-Wert*,
- mikrobiologische Indikatoren, wie die *Anzahl von Fäkal-Streptokokken in einer Probe von 100 ml Wasser*,
- andere Indikatoren, die direkt oder indirekt auf eine Gefährdung der Umwelt schließen lassen, wie *Geruch* oder *Geschmack*.

Unter dem Begriff *Parameter* fassen wir die Bezeichnungen, die in den Richtlinien in den Feldern

- Parameter/Schadstoff,
- Stoff,
- Parameter,
- geregelter Stoff.

zu finden sind.

Richtlinienwert: Wir wollen unter einem Richtlinienwert zunächst einmal die Wertinformationen verstehen, die in den Richtlinien unter den Begriffen, wie

- Grenzwert,

[1] Verein Deutscher Ingenieure

[2] World Health Organization

[3] Oftmals werden Schadstoffkonzentrationen einfach mit dem Namen des betreffenden Schadstoffes benannt. So spricht man nicht vom Parameter *Cadmiumkonzentration*, sondern kurz vom Parameter *Cadmium*.

- Leitwert,
- Hintergrundwert,
- Prüfwert,
- Richtwert,
- Qualitätsziel,
- Höchstkonzentration,
- Konzentration,
- ...

enthalten ist. Diese unterschiedlichen Bezeichnungen kommen u.a. daher, daß sich unterschiedliche Verwendungszwecke hinter dem betreffenden Richtlinienwert verbergen: ein Grenzwert hat beispielsweise bindenden Charakter, während ein Leitwert eher eine Orientierungshilfe darstellt. Deshalb wird es später wichtig sein, daß man trotz der vereinheitlichten Bezeichnung immer noch den ehemals verwendeten Namen abfragen kann, da dieser noch zusätzliche Informationen enthält.

Dies ist nicht immer nur die gesetzliche Bindungskraft. Beispielsweise geben die EG-Wasserqualitätsnormen Werte für die Membranfiltermethode und die Mehrfachröhrenmethode an, die der Einfachheit halber ebenfalls mit Membranfiltermethode und Mehrfachröhrenmethode bezeichnet werden. Exakt ausgedrückt handelt es sich aber um Richtlinienwerte, die mit Hilfe einer bestimmten Meßmethode (s.u.) überprüft werden müssen. Daneben gibt es weitere Beispiele für Richtlinienwertbezeichnungen, die eine Zusatzinformation tragen, wie

- Konzentration im Abgas,
- [Konzentration] bei einem Massenstrom,
- Salmonidengewässer-Leitwert,
- A1-Leitwert[4],
- Risikowert für krebserzeugende Verunreinigungen.

Wir werden unter einem Richtlinienwert nur einen konkreten Zahlenwert[5] oder ein, möglicherweise nach einer Seite offenes Intervall[6] betrachten. Ein Zahlenwert bzw. ein Intervall macht ohne *Maßeinheit* (s.u.) natürlich keinen Sinn. Deshalb steht ein Richtlinienwert immer in Beziehung zu einer Maßeinheit.

Maßeinheit: Wir wollen den Begriff *Maßeinheit* in dem Sinne verwenden, wie er in den Naturwissenschaften üblich ist. Beispiele für Maßeinheiten sind

- mg/m^3
- g/h

[4] Dieser Richtlinienwert stammt aus der EG-Wasserqualitätsnorm für Oberflächenwasser, das zur Trinkwassergewinnung verwendet wird. Dabei beschreibt A1 die Kategorie von Wasser, das mit einfachen physikalischen Mitteln (z.B. Schnellfilterung) aufbereitet und entkeimt wurde. Daneben existieren noch Kategorien A2 und A3.

[5] Beispielsweise 5 mg/m^3 für 1,2-Dibromethan im Abgas (TA Luft).

[6] Beispielsweise ≥ 25 g/h für 1,2-Dibromethan in einem Massenstrom (TA Luft).

- °C

Maßeinheitszusatz: Oftmals haben die Maßeinheiten einen Zusatz, der die Maßeinheit näher erläutert. Ein Beispiel ist der Zusatz *abgeleitetes Wasser* in

mg/l abgeleitetes Wasser

Richtlinienwertart: Unter *Richtlinienwertart* wollen wir die verschiedenen Arten zusammenfassen, einen Meßwert mathematisch zu ermitteln, um ihn mit dem Richtlinienwert zu vergleichen. Es gibt

- Mittelwerte über 24 Stunden, 1 Monat, 1 Jahr,
- Mediane,
- 98%-Wert,
- ...

Meßmethode: Mit einer *Meßmethode* bezeichnen wir die in den Richtlinien verwendeten Begriffe

- Analytik/Referenzmethode,
- Analytik/Referenzverfahren,
- Referenzmethode,
- Analysen- und Prüfungsverfahren,
- Analyse- oder Kontrollverfahren.

Beispiele für Meßmethoden sind die Black-Smoke-Methode, die gravimetrische Methode, chromatographische Verfahren etc.

Meßobjekt: Ein *Meßobjekt* ist ein Medium, für das Richtlinienregelungen definiert wurden. Beispielsweise gibt es das Meßobjekt Wasser, das sich noch weiter untergliedern läßt in Grundwasser, Oberflächenwasser, Badegewässer, Süßwasser usw. Wir beschränken uns auf die Betrachtung der Meßobjekte Luft, Wasser und Boden.

Emissionsquelle: Emissionsrichtlinien geben Regelungen für den Ausstoß von Schadstoffen in Zusammenhang mit einer *Emissionsquelle* an. Beispiele für Emissionsquellen sind

- Verbrennungsanlagen,
- Motoren,
- Produktionsstätten, die bestimmte Schadstoffe verwenden oder ausstoßen,
- Benzin und andere Brennstoffe.

Bezugszeitraum: Ein *Bezugszeitraum* ist die Zeitperiode, während der die Richtlinienregelung Gültigkeit besitzt. Dieses Feld existiert nur in den EG-Luftqualitätsnormen. Beispielsweise gilt eine besondere Regelung für SO_2 während des Winters (1.10. – 31.3.).

Einzuhalten ab: Dieses Datenfeld enthält ein Datum, ab dem die Regelung gelten soll. Gerade innerhalb der EG ist es so, daß eine Richtlinienregelung für einen noch weit in der Zukunft liegenden Termin erlassen wird und man bis dahin Übergangsregelungen schafft, die einen stufenweisen Übergang zu dieser Richtlinienregelung gewährleisten sollen. Dies gilt beispielsweise für die EG-Luft-Emissionsnormen für Dieselmotoren.

Probemenge: Mikrobiologische Parameter werden dadurch bestimmt, daß man eine Probe einer bestimmten Menge entnimmt und einen absoluten Wert angibt, der sich auf diese Probe bezieht. Beispielsweise schreibt die EG-Qualitätsnorm für Wasser für den menschlichen Gebrauch einen Leitwert von 100 für Fäkal-Streptokokken in einer Probe von 100 ml vor.

Probehäufigkeit: Manche Richtlinien schreiben die Häufigkeit der Probeentnahme vor. Typische Werte sind: wöchentlich, 14-tägig, monatlich.

Richtlinien ID: Die EG-Richtlinien zur Luft-Emission, die EG-Luftqualitätsnormen und die EG-Wasseremissionsnormen bestehen aus mehreren Teilrichtlinien auf die über eine eindeutige Nummer und den Titel der Teilrichtlinie verwiesen wird. Wir fassen dies unter dem Begriff einer Richtlinienidentifikation, kurz *Richtlinien ID*, zusammen.

Parameter Nr.: Die EG-Wasserqualitätsnormen verwenden eine Numerierung der Parameter.

Wir verweisen an dieser Stelle auf Anhang A, der die wichtigsten der hier erwähnten Begriffe in einer alphabetischen Übersicht enthält. Mit Hilfe dieser elementaren Datenfelder wurden die Richtlinienstrukturen aus Anhang B normalisiert. Das Ergebnis befindet sich in Anhang C.

4.2 Benutzerprofile

Wir werden zunächst anhand von Beispielen einige mögliche Benutzer des Umweltinformationssystems vorstellen, um diese später zu Gruppen zusammenzufassen, die jeweils ein eigenes, einheitliches Benutzerprofil aufweisen.

4.2.1 Beispiele von Benutzern

- **Mitarbeiter in Rathäusern, Justizbehörden oder technischen Überwachungsvereinen**
  Sie sind damit beschäftigt, die Einhaltung der Richtlinien in ihrem Wirkungsbereich zu überwachen. Dabei geht es einerseits um die Überprüfung von Verdachtsmomenten, wie z.B.

  - ob sich ein See weiterhin zum Baden eignet oder ob die Qualität des Wassers es notwendig macht, den See für die Öffentlichkeit zu sperren,
  - ob die Atemluft innerhalb einer Großstadt während der Verkehrsspitzenzeiten noch innerhalb der tolerierbaren Grenzen liegt oder ob Maßnahmen zum Schutz der gefährdeten Menschen einzuleiten sind,

  - ob das Sickerwasser einer Müllablagerungsstätte nicht Giftstoffe in einer derart hohen Konzentration enthält, daß der Eintrag ins Grundwasser die Trinkwasserversorgung in Gefahr bringt, oder
  - ob der Klärschlamm einer Kläranlage als Sondermüll oder als weitgehend ungefährlicher Müll zu betrachten ist.

  Andererseits müssen beispielsweise Industrieanlagen, wie

  - Müllverbrennungsanlagen,
  - Kohlekraftwerke,
  - chemische Produktionsanlagen,
  - Raffinerien,

  genehmigt werden. Dabei geht es vor allem darum, ob die von den Richtlinien vorgeschriebenen Auflagen eingehalten werden. Da diese Entscheidungen möglicherweise gerichtlich überprüft werden, müssen sich die Mitarbeiter über die exakten Regelungen der Richtlinien informieren können.

- **Politiker, politische Referenten, Kommissionsmitglieder**
  Sie möchten sich z.B. über die in Deutschland und den angrenzenden Staaten geltenden Richtlinien informieren. Dies kann notwendig sein, um diese Richtlinien beispielsweise innerhalb der Europäischen Gemeinschaft zu harmonisieren.

  Weiterhin können Vergleiche zwischen den existierenden Richtlinien Hinweise geben, welche Regelungen andere Institutionen für angemessen halten. Dies sind wichtige Argumente für den politischen Meinungsbildungsprozeß.

- **Entscheidungsträger und Führungskräfte in Industrieunternehmen**
  Sie müssen beispielsweise bei der Planung von Produktionsanlagen die für sie geltenden Richtlinienregelungen einbeziehen, da deren Einhaltung ja auch einem Kostenfaktor entspricht. Darüberhinaus kann es für sie interessant sein, sich z.B. über im Ausland geltende Regelungen zu informieren, die im Falle, daß dort schärfere Regelungen gelten, einen Hinweis geben können, daß sich die augenblicklich hierzulande geltende Regelung noch weiter verschärfen könnte und daß es langfristig wohl günstiger wäre, eine Produktionsanlage so zu konzipieren, daß sie den schärferen Bestimmungen genügt. Andererseits können rein wirtschaftlich kalkulierende Manager sich auch über den Standort informieren wollen, für den die schwächsten Richtlinienregelungen gelten und der dadurch kostengünstiger als andere sein kann.

  Diese Benutzer werden sich hauptsächlich für Emissionsrichtlinien interessieren, die für die Genehmigung von Anlagen relevant sind. Wichtig sind dabei vornehmlich alle Regelungen, die einen bestimmten Anlagentyp bzw. eine bestimmte Emissionsquelle betreffen und nicht nur die Regelungen für einzelne Schadstoffe.

- **Mitarbeiter von Umweltorganisationen**
  Als mögliche Kontrahenten der oben aufgeführten Benutzer in einem politischen oder juristischen Streit ist ihr Informationsinteresse sicher dem ihrer Mitstreiter ähnlich. Sie sind in der Regel Fachleute mit einem profunden Wissen im Bereich der Naturwissenschaften, die sich über den aktuellen Stand der Umweltgesetzgebung informieren möchten.

- **Naturwissenschaftler an Universitäten, Forschungslabors oder staatlichen Forschungseinrichtungen**
  Sie beschäftigen sich aus wissenschaftlichen Motiven mit den Regelungen der Richtlinien, um z.B.

  – Defizite der Regelungen zu entdecken, die eine Gefährdung der Umwelt mit sich ziehen können,
  – Verbesserungsvorschläge aufgrund neuester Forschungsergebnisse zu machen, oder
  – Neuregelungen zu erarbeiten und vorzuschlagen.

  Beispielsweise ist es möglich, daß ein bestimmter Stoff nach neuesten Erkenntnissen als besonders gesundheitsgefährdend eingeschätzt wird. Deshalb wird man überprüfen, inwieweit schon Richtlinienregelungen für diesen Stoff bestehen und ob diese ausreichenden Schutz für Mensch und Umwelt gewährleisten.

  Eine anderer Schwerpunkt dieser Benutzer kann darin liegen, die Verifikation der Regelungen zu verbessern. Beispielsweise können neue, genauere Meßmethoden entwickelt oder Nachweisverfahren geschaffen werden, die eine automatische Überwachung ermöglichen sollen. Solche Verfahren müssen natürlich den in den Richtlinienregelungen gegebenen Anforderungen entsprechen.

- **Gelegentliche Benutzer**
  Sie wollen sich über einige Regelungen z.B. aus privatem Interesse informieren, etwa als Mitglied einer Bürgerinitiative oder zur eigenen Meinungsbildung (vergleiche Ausführungen zu Politikern). Da sie mit der Diskurswelt in der Regel nicht besonders vertraut sind, werden sie eine allgemeinverständliche Präsentation der Informationen erwarten, die auch nicht alle Details der Regelung enthalten muß. Da die Motivation dieser gelegentlichen Benutzer, sich zu informieren, sehr manigfaltig sein kann, gibt es keine Richtlinie oder Richtlinienart, die vornehmlich angefragt werden wird. Beispielsweise wird sich eine Bürgerinitiative

  – vornehmlich mit allen Emissionsrichtlinien für Luft und Wasser befassen, falls sie den Bau eines Kraftwerkes verhindern möchten;
  – mit allen Richtlinien beschäftigen, die eine Regelung für Cadmiumkonzentrationen, wenn der z.B. Verdacht besteht, daß Parkanlagen, private Gärten, Spielplätze etc. mit Cadmium verseucht wurden.

4.2.2 Benutzergruppen

Anhand der in Abschnitt 4.2.1 aufgeführten Beispiele lassen sich die Benutzer einerseits nach ihrem Anfrageverhalten, andererseits nach Art und Form der Antworten, die sie erwarten, unterscheiden.

Auf der einen Seite existieren

- Benutzer mit exakten, zielgerichteten und
- Benutzer mit unvollständigen, eher vage formulierten

Anfragen. Auf der anderen Seite gibt es

- Benutzer, die exakte, detaillierte Antworten erwarten, und
- Benutzer, die abstrahierte Antworten wünschen, die frei von Details sind.

Das unterschiedliche Anfrageverhalten hat seine Ursache in einer mehr oder weniger ausreichenden Kenntnis über die Struktur der Diskurswelt, also den Aufbau der Richtlinien. Benutzer, die diesen Aufbau kennen, sind in der Lage, sehr zielgericht nach Information zu suchen, da sie bereits wissen, daß diese Information existiert, sie aber ihren Inhalt erfahren wollen. Typische Anfragen sind:

- Wie lautet die Regelung für Parameter P in der Richtlinie R ?
- Welches sind die Regelungen und/oder Richtlinien für Emissionsquellen vom Typ T ?
- Welche Richtlinien regeln den Parameter P ?
- Welche Richtlinien regeln das Meßobjekt M ?

Weniger sachkundige Anwender werden ihre Anfragen dagegen nicht so zielgerichtet stellen können, da sie möglicherweise von der Existenz einer Richtlinie wissen, dagegen nur sehr wenig oder überhaupt nichts vom Aufbau der Richtlinie. Daher können sie über die genauen Informationen, die die Richtlinien enthalten, nur mutmaßen und deshalb ihre Anfrage nur recht ungenau, unscharf oder gar unvollständig stellen. Typische Beispiele für solche Anfragen sind:

- Welche Regelungen existieren für Anlagen vom Typ T ?
- Welche Regelungen existieren für die Wasserqualität ?
- Wie wird die Einhaltung der Luftqualität überprüft ?

Die Antworterwartung der Benutzer dagegen divergiert aufgrund unterschiedlicher Kenntnis von (z.B. naturwissenschaftlichen) Zusammenhängen und Fachbegriffen. Anwender mit einem profunden Fachwissen werden sich für eine sehr detaillierte Beschreibung der Regelung interessieren, während fachlich weniger versierte Benutzer eine übersichtliche, Details übergehende, abstrahierende Darstellung der gewünschten Information erwarten. Beispielsweise wird sich ein Politiker nicht für eine detaillierte Beschreibung der Meßbedingungen beim Nachweis von Cadmium im Boden interessieren, dagegen schon eher für alle in Europa für Cadmium geltenden Richtlinienwerte, um diese vergleichen zu können.

Zusammenfassend haben wir also vier Benutzergruppen, die sich in ihrer Anfrage- und Antwortcharakteristik unterscheiden. Abb. 4.2 zeigt diese Gruppen in Form einer Tabelle. Sie zeigt die Zuordnung der in Abschnitt 4.2.1 genannten möglichen Benutzer zu den von uns identifizierten Benutzergruppen.

Anfrageverhalten ▷ ▽ Antworterwartung	zielgerichtet, exakt	ungenau, unvollständig
detailliert, exakt	Naturwissenschaftler	Industriemanager, Justizbehörden, Umweltorganisationen
abstrahiert, ohne Details	Politiker	gelegentlicher Benutzer

Abbildung 4.2: Zuordnung von Benutzern zu Benutzergruppen

Kapitel 5

Statischer Entwurf

5.1 Gliederung des Entwurfs

In diesem Kapitel präsentieren wir einen statischen Entwurf für unsere Diskurswelt. Er beschreibt deren Struktur, gewissermaßen die Umgebung, in der sich die Aktivitäten innerhalb des Systems oder der Diskurswelt bewegen können. Die wesentlichen Elemente dieses Gerüstes sind die Objekte und deren Beziehungen untereinander, so daß im Mittelpunkt des statischen Entwurfs die folgenden Fragen stehen:

- Welches sind die Objekte der Diskurswelt?
- Welches ist die Struktur dieser Objekte?
- In welchen Beziehungen können die Objekte untereinander stehen?

Um diese Fragen zu beantworten, entwickeln wir eine Modellierung, die die Objektstruktur wiedergibt. Wir nennen sie gemäß der Dreiteilung innerhalb von OMT (siehe Abschnitt 3.2) vereinfacht ein *Objektmodell*. Ein Objektmodell unserer Richtlinien-Diskurswelt in OMT-Notation stellen wir in Abschnitt 3.2.3 vor. Wir nennen es deshalb auch *das OMT-Objektmodell*.

In Abschnitt 5.3 transformieren wir das OMT-Objektmodell in eine Objektmodellierung in OPAQUE-EM-Notation. Wir nennen es das *OPAQUE-EM-Objektmodell*. Diese Transformation ist weitgehend mechanisch. Sie wurde in Abschnitt 3.5 beschrieben; das Result bedarf nur an einigen wenigen Stellen kleiner Korrekturen.

Die beiden genannten Objektmodelle sind das Grundgerüst aller weiteren Entwurfsentscheidungen, insbesondere für die dynamische Modellierung in den Kapiteln 6, 7 und 8.

5.2 Ein OMT-Objektmodell

5.2.1 Richtlinien und Richtlinienregelungen

Wie wir in Kapitel 4 gesehen haben, sind die zentralen Gegenstände unserer Diskurswelt die *Richtlinien*. Daher liegt es auf der Hand, sie als Objekte einer Klasse `Richtlinie` zu modellieren.

Wie wir weiterhin festgestellt haben, ist jede Richtlinie eine Sammlung von *Richtlinienregelungen*, die wir als Objekte einer Klasse `Richtlinienregelung` auffassen können. Jede Richtlinienregelung gehört eindeutig zu einer bestimmten Richtlinie.

Diese Zusammenhänge haben wir in Abb. 5.1 dargestellt, wobei wir zunächst auf Attribute für die beiden Objektklassen verzichtet haben: sie zeigt die beiden Klassen `Richtlinie` und `Richtlinienregelung`, die in einer Assoziation `hat_Richtlinienregelung` miteinander verbunden sind. Diese Assoziation besagt, daß Objekte der genannten Klassen in Beziehungen miteinander verbunden sein können und zwar im Rahmen der folgenden Kardinalitäten:

- Eine Richtlinie[1] enthält mindestens eine Richtlinienregelung; eine Obergrenze für die Anzahl enthaltener Regelungen gibt es nicht (Kardinalität $< 1, * >$ neben `Richtlinie`).
- Eine Richtlinienregelung gehört zu genau einer Richtlinie (Kardinalität $< 1, 1 >$ neben `Richtlinienregelung`).

In die Abbildung wurden außerdem schon Entwurfsentscheidungen angedeutet, die wir in folgenden Abschnitten erläutern werden:

- die Klasse `Richtlinie` steht in einer Vererbungsbeziehung zu zwei Unterklassen,
- Objekte der Klasse `Richtlinienregelung` sind Aggregate,
- sowohl Richtlinien als auch Richtlinienregelungen stehen in Beziehung zu anderen Objekten.

Diese Entscheidungen, die weiter unten näher beschrieben werden, sind angedeutet, um später den Zusammenhang zwischen den Abbildungen herstellen zu können, wodurch sich das gesamte OMT-Objektmodell ergibt. Weiterhin ist Abb. 5.1 der Ausgangspunkt für die Entwicklung des Objektmodells. Diese Entwicklung erfolgt entlang von zwei Strängen, deren Ausgangspunkte die beiden Klassen `Richtlinie` und `Richtlinienregelung` sind:

- Einerseits müssen Richtlinien ausreichend genau beschrieben werden können.
- Andererseits muß es gelingen, die Komplexität der unterschiedlichen Regelungen strukturell zu fassen. Dieses Problem wurde ja bereits in der Analyse in Kapitel 4 behandelt. Wir hatten dort einen Begriffsabgleich durchgeführt, um strukturelle Ähnlichkeiten der Regelungen herauszuarbeiten. Siehe dazu auch die Anhänge B und C.

[1] In Zusammenhang mit der Modellierung verstehen wir unter *Richtlinien* bzw. *Richtlinienregelungen* Objekte der Klassen `Richtlinie` bzw. `Richtlinienregelung`. Ähnliches gilt für die restlichen, noch vorzustellenden Klassen: *Institutionen* sind Objekte der Klasse `Institution`, *Gebietsbereiche* von `Gebietsbereich` etc.

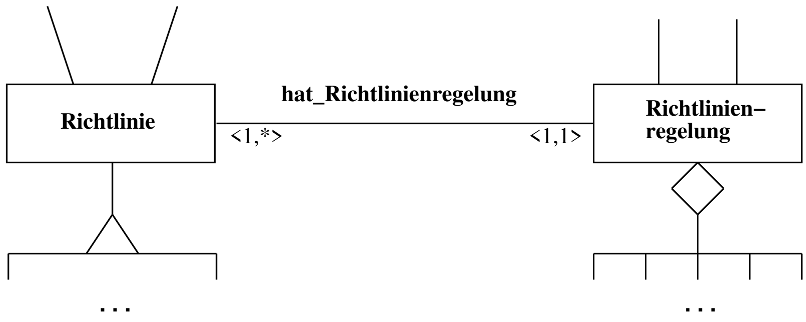


Abbildung 5.1: OMT-Objektmodell – Teil 1: Richtlinien und Richtlinienregelungen

5.2.2 Institutionen, Meßobjekte und Gebietsbereiche

Richtlinien werden von Ministerien, Gesetzesgebern, internationalen Organisationen, Berufsverbänden etc. erlassen, die wir zusammenfassend als *Institutionen* bezeichnen wollen. Beispiele für Institutionen sind die Europäische Gemeinschaft (EG), der Verband deutscher Ingenieure (VDI) oder die World Health Organization (WHO).

Wie aus den Beispielen bereits hervorgeht, hat jede Institution einen Namen, der meistens auch noch in einer Kurzfassung existiert. Deshalb beschreiben wir Institutionen mit den beiden Attributen `Kurzname` und `Langname`. Darüberhinaus gibt es möglicherweise weitere interessante Attribute zu einer Institution, wie beispielsweise den Sitz der Institution oder ihr Gründungsjahr. Wir möchten uns im Rahmen dieser Arbeit auf die wichtigsten Punkte beschränken und verzichten deshalb auf eine Diskussion, welche Attribute unter Umständen noch sinnvoll wären, da sich die hier entworfenen Klassen problemlos um zusätzliche Attribute erweitern lassen.

Jede Richtlinie wird von genau einer Institution erlassen. Andererseits kann eine Institution mehrere Richtlinien herausgeben. Sinnvollerweise werden wir in der Datenbasis nur Institutionen führen, die mindestens eine Richtlinie erlassen haben. Aus diesen Zusammenhängen heraus ergibt sich die Modellierung der Assoziation `erlassen_von` mit den zugehörigen Kardinalitäten in Abb. 5.2.

Jede Institution wirkt in einem gewissen *Gebietsbereich*. Für die EG ist der zugehörige Gebietsbereich beispielsweise EG-Europa, für die WHO die gesamte Erde. Es ist sinnvoll, Gebietsbereiche als eigenständige Objekte zu modellieren, anstatt sie mittels eines Attributs in `Institution` zu führen, da Gebietsbereiche andere Gebietsbereiche einschließen oder sich mit ihnen überschneiden. Wir wollen zumindest die Einschlußbeziehung zwischen Gebietsbereichen modellieren, um einen weniger informierten Benutzer unterstützen zu können, der beispielsweise nach Richtlinien sucht, die für die Bundesrepublik Deutschland gelten. Für ihn sind dann natürlich auch Richtlinien der EG oder der WHO interessant, da beide auf Gebietsbereichen wirken, die die Bundesrepublik Deutschland einschließen.

Die genannten Zusammenhänge sind in Abb. 5.2 mittels der Klassen `Institution` und `Gebietsbereich` und der Assoziationen `wirkt_in` und `schließt_ein` dargestellt. Die Kardinalitäten ergeben sich wie folgt:

- Eine Institution wirkt in *genau einem* Gebietsbereich (Kardinalität $< 1,1 >$ neben `Institution`).

- In einem Gebietsbereich wirkt *mindestens eine* Institution. Gebietsbereiche, die mit keiner Institution in Verbindung stehen, werden in der Datenbasis nicht geführt (Kardinalität $< 1, * >$ neben `Gebietsbereich`).
- Ein Gebietsbereich kann als Oberbereich mehrere andere Gebietsbereiche als Unterbereiche einschließen. Dies wird durch die Kardinalität $< 0, * >$ für Oberbereiche in der Assoziation `schließt_ein` ausgedrückt.
- Ein Gebietsbereich kann Unterbereich mehrerer Oberbereiche sein. Beispielsweise sind 'Erde' und 'Europa' Oberbereiche der 'Bundesrepublik Deutschland'. Deshalb gilt für Unterbereiche die Kardinalität $< 0, * >$.

Weiterhin gehört zu jeder Richtlinie ein *Meßobjekt*, wie Wasser, Luft oder Boden. Sie werden als eigenständige Objekte modelliert, da wir einerseits Beziehungen der Meßobjekte untereinander und andererseits Beziehungen zu anderen Objekten beschreiben wollen. Würden wir die Meßobjekte in Form eines Attributs beispielsweise der Klasse `Richtlinie` führen, so wäre dies nicht möglich. Deshalb werden die Meßobjekte in einer Klasse `Meßobjekt` zusammengefaßt.

Jede Richtlinie hat also genau ein Meßobjekt (Kardinalität $< 1, 1 >$ für Richtlinien in der Assoziation `hat_Meßobjekt`). Umgekehrt gibt es für jedes Meßobjekt eine oder mehrere Richtlinien (Kardinalität $< 1, * >$ in `hat_Meßobjekt`).

Weiterhin können sich Meßobjekte in ihrer Bedeutung überschneiden. Wie wir in Kapitel 4 bereits erwähnt haben, gibt es Richtlinien für Trinkwasser, Oberflächenwasser, Grundwasser, Fließwasser etc. Diese Begriffe sind nicht synonym, andererseits aber auch nicht disjunkt. Beispielsweise wird Grundwasser zur Trinkwassergewinnung verwendet, weshalb in diesen Fällen die Regelungen für Trinkwasser auch für Grundwasser anzuwenden sind. Derartige Überschneidungen beschreiben wir mit Beziehungen zwischen Meßobjekten (Assoziation `überschneidet_sich`). Die Kardinalitäten von jeweils $< 0, * >$ ergeben sich, da sich ein Meßobjekt nicht unbedingt mit einem anderen Meßobjekt in seiner Bedeutung überschneidet, u.U. aber mit mehreren.

Die beschriebene Modellierung ist in Abb. 5.2 zu sehen.

5.2.3 Vererbungshierarchie der Richtlinien

In Abschnitt 5.2.1 haben wir bereits eine Klasse `Richtlinie` geschaffen. Eine genauere Betrachtung der in der Umweltbroschüre [LR92] aufgeführten Richtlinien zeigt jedoch, daß eine derart grobe Modellierung nicht ausreicht.

Beispielsweise werden *EG-Richtlinien* durch eine mehrteiligen Richtlinienkennung markiert, die sich u.a. aus dem Festsetzungsjahr der Richtlinie und einer EG-eigenen Richtliniennummer zusammensetzt. EG-Richtlinien werden weiterhin in Amtsblättern der EG veröffentlicht. Jede EG-Richtlinie hat zu einem Zeitpunkt genau eine *Amtsblattreferenz*, die möglicherweise aufgrund von Änderungen in der Richtlinie irgendwann durch eine neue ersetzt wird. Daher können Amtsblattreferenzen in Vorgänger-Nachfolger-Beziehungen zueinander stehen. Diese Beziehungen machen es auch notwendig, die Amtsblattreferenzen als eigene Objekte und nicht einfach als Attribute einer Klasse `EG-Richtlinie` zu beschreiben.

Neben den EG-Richtlinien kann man eine Gruppe von *Nicht-EG-Richtlinien* bilden, also Richtlinien, die von anderen Institutionen als der EG erlassen wurden. Sie unterscheiden sich von

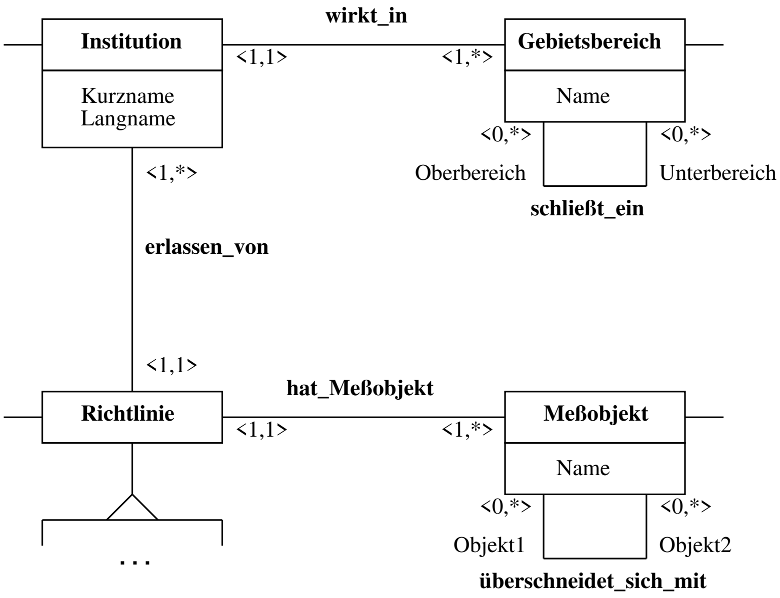


Abbildung 5.2: OMT-Objektmodell – Teil 2: Institutionen, Meßobjekte und Gebietsbereiche

den EG-Richtlinien dadurch, daß sie in irgendwelchen *Veröffentlichungen* herausgegeben werden. Eine solche Veröffentlichung ist z.B. ein Bundesgesetzblatt, herausgegeben vom Bundesumweltministerium und verlegt von der Bundesdruckerei. Natürlich kann eine solche Veröffentlichung mehr als eine Richtlinie enthalten.

Alle Richtlinien haben einige Attribute gemeinsam, wie etwa einen Titel, eine Zielsetzung, eine Richtlinienart[2], ein Erlaßdatum etc.

Wenn wir also die Richtlinien zusammenfassend betrachten, so kann man sagen, daß sie sich einerseits durch eine Reihe gemeinsamer Attribute beschreiben lassen, andererseits aber in zwei Gruppen mit spezifischen Eigenschaften (z.B. Attribute, Beziehungen) unterteilt werden können:

- in die der *EG-Richtlinien* und
- in die der *Nicht-EG-Richtlinien*.

Wir können dies mit einer Vererbungshierarchie mit der Oberklasse `Richtlinie` und den Unterklassen `EG-Richtlinie` und `Nicht-EG-Richtlinie` modellieren. Die Oberklasse `Richtlinie` erhält dabei die gemeinsamen Attribute, während die Unterklassen ihre spezifischen Attribute halten und in Assoziationen mit unterschiedlichen anderen Klassen stehen. Diese Hierarchie ist in Abb. 5.3 zu sehen.

Wir müssen das OMT-Objektmodell aber noch um zwei wichtige Fakten ergänzen, die sich in der grafischen Darstellung nicht modellieren lassen und die wir an dieser Stelle umgangssprachlich formuliert ergänzen:

[2]Unter der Richtlinienart verstehen wir die Kategorisierung einer Richtlinie als Emissions- oder Immissionsrichtlinie.

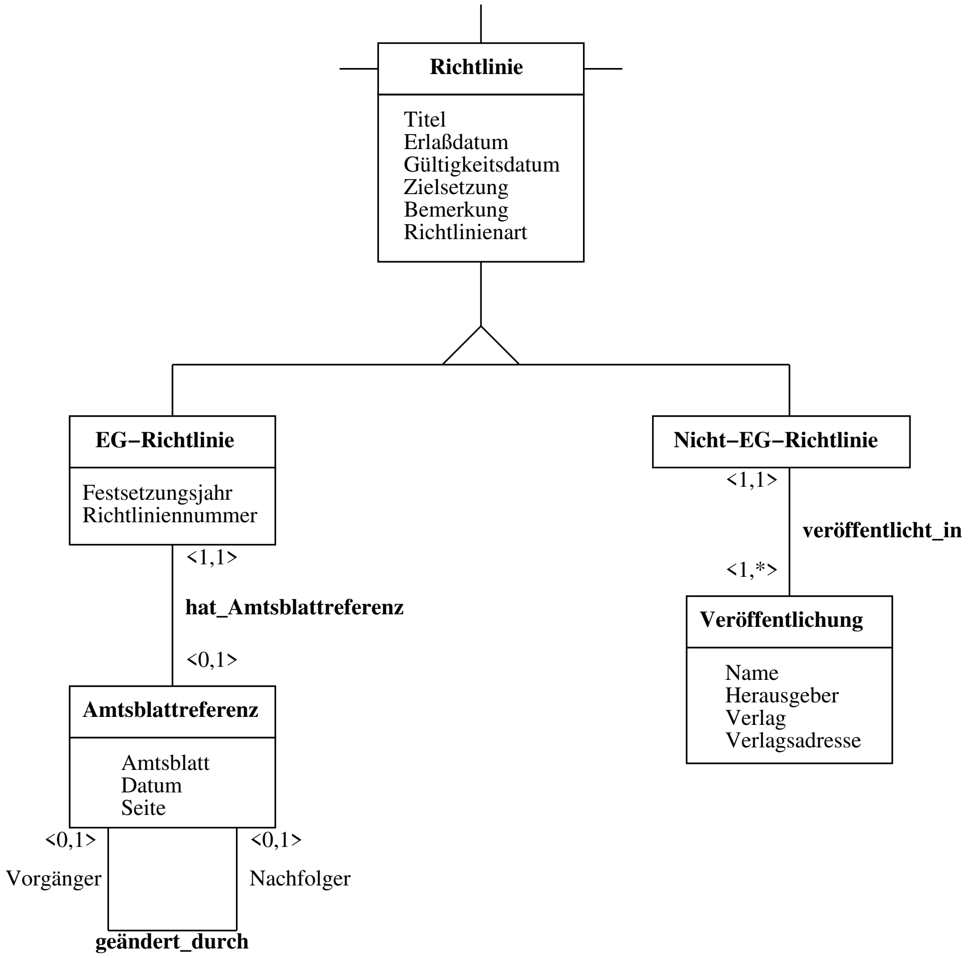


Abbildung 5.3: OMT-Objektmodell – Teil 3: Die Richtlinien-Vererbungshierarchie

- Aufgrund der Vererbungsbeziehung sind EG-Richtlinien auch Objekte der Klasse `Richtlinie`. Daher steht jede EG-Richtlinie in einer Beziehung zu einer Institution (siehe Abb. 5.2). Im Falle der EG-Richtlinien *muß* das natürlich die Institution 'EG' sein.
  Man kann sich an dieser Stelle natürlich fragen, ob es nicht besser wäre, nur Nicht-EG-Richtlinien in Beziehung zu Institutionen zu setzen, da für die EG-Richtlinien die Institution ja auf natürliche Weise feststeht. Dies hätte den Nachteil, daß man die Daten und Informationen für die EG nicht durch ein Institutionsobjekt 'EG' in der Datenbasis halten würde, sondern sie auf irgendeine andere Art und Weise einbringen müßte, beispielsweise über zusätzliche Attribute der EG-Richtlinien o.ä. Das würde aber möglicherweise zu einer redundanten Datenhaltung führen. Außerdem würde sich eine Modellierung des Gebietsbereichs der EG äußerst kompliziert gestalten.
  Aus den genannten Gründen ist es auf jeden Fall sinnvoll, die EG genauso wie alle anderen Institutionen als Objekt der Klasse `Institution` zu führen und die obengenannte Integritätsbedingung beispielsweise in den Code der Datenbankanwendungen einzuarbeiten.

- Eine weitere Bedingung, die in der Darstellung von Abb. 5.3 fehlt, ist, daß mittels der Assoziation `geändert_durch` eine Vorgänger-Nachfolger-Relation V zwischen zwei Amtsblattreferenzobjekten ausgedrückt wird. Diese Relation V hat die folgenden Eigenschaften:

  (V1) $(x,x) \notin V$, d.h. V ist nicht reflexiv.

  (V2) $(x,y) \in V \rightarrow (y,x) \notin V$, d.h. V ist nicht symmetrisch.

  (V3) $(x,y) \in V \wedge (y,z) \in V \rightarrow (x,z) \in V$, d.h. V ist transitiv.

  Die Eigenschaften (V1) und (V2) stellen Konsistenzbedingungen dar, die von den in der Assoziation `geändert_durch` Paaren (a,b) von Amtsblattreferenzen eingehalten werden müssen, d.h.

  (V1') Keine Amtsblattreferenz ist Vorgänger oder Nachfolger von sich selbst.

  (V2') Ist eine Amtsblattreferenz a Vorgänger einer Amtsblattreferenz b, so kann b auf keinen Fall Vorgänger von a sein. Dies entspricht den natürlichen Zusammenhängen.

  Eigenschaft (V3) läßt sich dagegen dazu verwenden, eine redundante Datenhaltung zu vermeiden, da wir mit Hilfe von Eigenschaft (V3) beispielsweise erschließen können, daß eine Amtsblattreferenz c Nachfolger von a ist, falls c Nachfolger von b und b Nachfolger von a ist. Das heißt, daß wir nur direkte Vorgänger-Nachfolger-Beziehungen in der Datenbasis halten müssen, da sich die übrigen aufgrund der Transitivitätseigenschaft ermitteln lassen. Die direkten Vorgänger-Nachfolger-Beziehungen werden einerseits durch die Konsistenzbedingungen (V1) und (V2) und andererseits durch die Kardinalitäten von jeweils $< 0,1 >$ erzwungen.

5.2.4 Richtlinienregelungs-Aggregat

Die Richtlinien enthalten eine Sammlung von Richtlinienregelungen. Dies haben wir in Abschnitt 5.2.1 und in Abb. 5.1 modelliert. Wie wir in Kapitel 4 gesehen haben, sind diese Regelungen sehr unterschiedlich strukturiert. Durch einen Begriffsabgleich (siehe Anhang C) konnten wir immerhin einige Ähnlichkeiten herausarbeiten. Eine weitere Betrachtung der konsolidierten Richtlinienstrukturen führt zu folgendem Ergebnis:

- Jede Regelung enthält einen *Parameter*.
- Eine Regelung enthält einen *Richtlinienwerte*[3].
- Manche Regelungen enthalten die Angabe
  - einer *Meßmethode*,
  - einer *Bemerkung*,
  - eines *Bezugszeitraums* und
  - einer *Emissionsquelle*.

Wir verzichten an dieser Stelle auf eine Diskussion weiterer möglicher Analyseergebnisse bezüglich der Richtlinienstrukturen, da man dazu tief in umwelt- und chemisch-technische Problemstellungen eindringen müßte, und bauen die nun folgende Modellierung einer Richtlinienregelung anhand der oben genannten Zusammensetzung auf. Es ist sinnvoll, die genannten Bestandteile einer Regelung selbst wieder als Objekte zu modellieren, da sie einerseits selbst komplexe Gegenstände darstellen (vgl. etwa Diskussion über Parameter oder Richtlinienwerte in Kapitel 4), andererseits eigenständige, abgeschlossene Informationseinheiten bilden, die u.U. in mehreren Regelungen auftreten können wie beispielsweise eine Meßmethode. Wir können dann die Objekte mit Hilfe einer Aggregation zusammenfassen, wobei das entstandene Aggregat der Modellierung einer Klasse `Richtlinienregelung` entspricht.

Die Struktur des Richtlinienregelungs-Aggregats ist in Abb. 5.4 zu sehen. Das Aggregat setzt sich aus den Klassen `Parameter`, `Richtlinienwert`, `Meßmethode`, `Bemerkung`, `Bezugszeitraum` und `Emissionsquelle` zusammen. Die Kardinalitäten ergeben sich aus den oben geschilderten Zusammenhängen. Wir gehen nun etwas genauer auf die dargestellten Klassen ein:

`Richtlinienregelung:` Wie oben beschrieben, ist sie ein Aggregat aus mehreren Klassen. Innerhalb der Richtlinien finden sich einige Regelungen, die nur in Zusammenhang mit anderen Regelungen gültig sind. Diese Beziehung wird durch eine Assoziation von `Richtlinienregelung` mit sich selbst (`gültig_zusammen_mit`) beschrieben, wobei jede Regelung mit beliebig vielen anderen verbunden sein kann (Kardinalitäten $< 0, * >$). Die beigefügten Rollen `Regelung1` und `Regelung2` völlig sind gleichwertig, d.h. falls Regelung a (in der Rolle `Regelung1`) in Beziehung steht zu Regelung b (in der Rolle `Regelung2`), dann würde der gleiche Sachverhalt genauso gut beschrieben, wenn b in der Rolle `Regelung1` in Beziehung zu a in `Regelung2` stünde. Um Redundanzen zu vermeiden, sollte nur eine der beiden Varianten in der Datenbasis gehalten werden.

`Parameter:` Jede Richtlinienregelung regelt genau einen Parameter. Man beschreibt ihn durch Attribute, wie `Kurzname` (wie beispielsweise 'Cd' als chemisches Symbol für Kadmium) und `Langname` (z.B. 'Kadmium').
Für einige Parameter gibt es synonyme Bezeichnungen: wie etwa 'pH-Wert', 'Wasserstoffionenkonzentration' oder 'H_3O^+-Ionen-Konzentration'. Zur Unterstützung der Benutzer wollen wir diese Synonym-Beziehungen zwischen Parametern mit einer Assoziation `ist_synonym` beschreiben.
Die Relation S aller Paare (x, y) von Synonymen hat die folgenden Eigenschaften:

[3] In manchen Richtlinien, wie beispielsweise den WHO-Luftqualitätsleitlinien werden mehrere Werte pro Regelung angegeben. Diese sind dann in Einzelregelungen zu unterteilen.

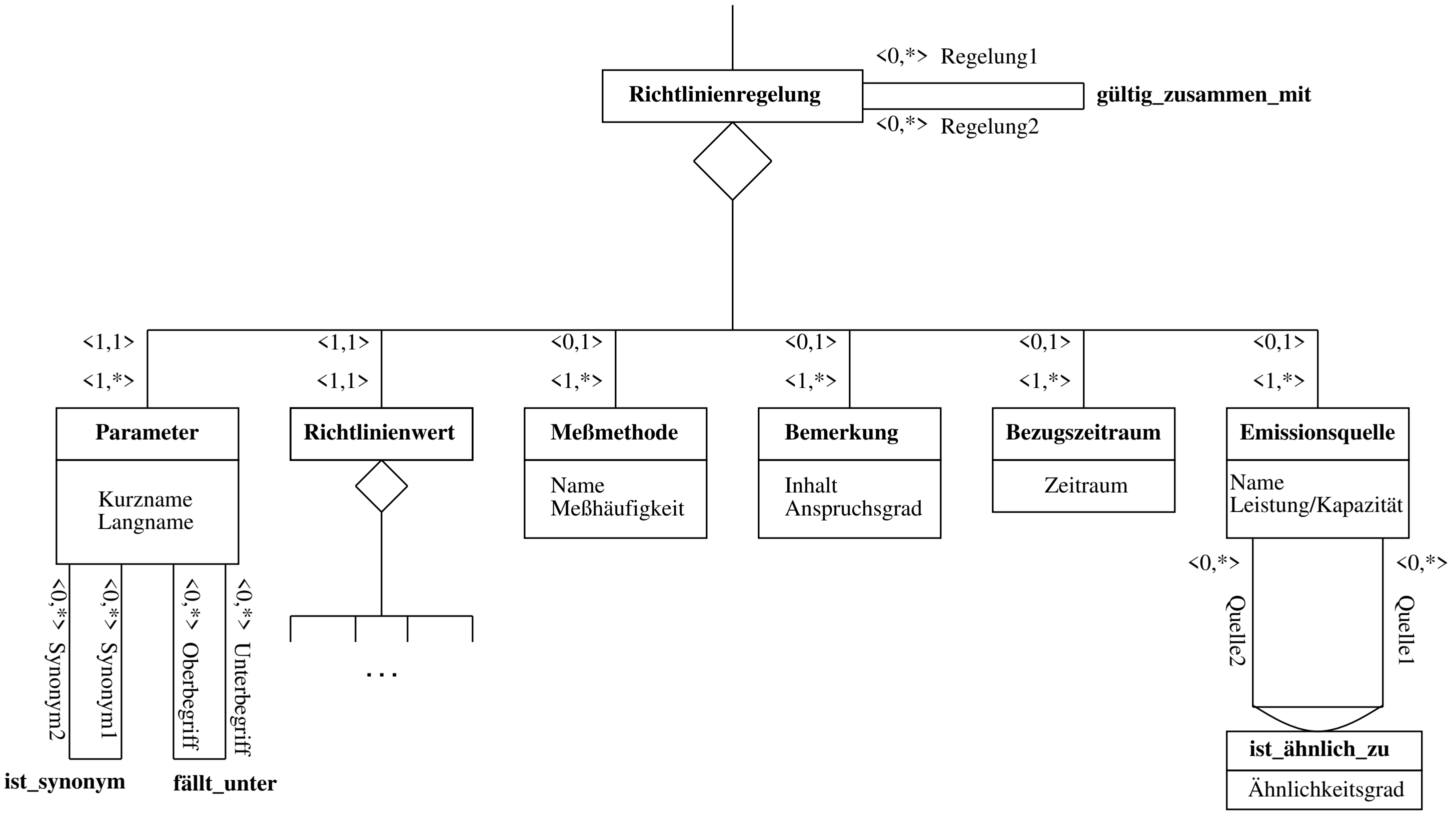


Abbildung 5.4: OMT-Objektmodell – Teil 4: Das Richtlinienregelungs-Aggregat

(S1) $(x,x) \in S$, d.h. S ist reflexiv.

(S2) $(x,y) \in S \rightarrow (y,x) \in S$, d.h. S ist symmetrisch.

(S3) $(x,y) \in S \wedge (y,z) \in S \rightarrow (x,z) \in S$, d.h. S ist transitiv.

Diese Eigenschaften (S1), (S2) und (S3) der Synonymrelation können dazu verwendet werden, um eine redundante Datenhaltung zu vermeiden.
Ein weiterer, insbesondere für einen weniger vertrauten Benutzer interessanter Aspekt sind die teilweise vorhandenen Unterbegriff-Oberbegriff-Beziehungen. So fallen unter den Parameter 'Fluorchlorkohlenwasserstoffe' beispielsweise die Parameter 'Tri' und 'Per', beides jeweils Fluorchlorkohlenwasserstoff-Verbindungen (FCKW). Um einem Benutzer, der nach Regelungen für FCKW sucht, auch Regelungen für 'Tri' oder 'Per' liefern zu können, gibt es eine Assoziation `fällt_unter` zwischen Parametern in den Rollen `Unterbegriff` und `Oberbegriff`.
Die Relation U aller Unterbegriff-Oberbegriff-Paare (x,y) hat die folgenden Eigenschaften:

(U1) $(x,x) \in U$, d.h. U ist reflexiv.

(U2) $(x,y) \in U \rightarrow (y,x) \notin U$, d.h. U ist nicht symmetrisch.

(U3) $(x,y) \in U \wedge (y,z) \in U \rightarrow (x,z) \in U$, d.h. U ist transitiv.

Die Eigenschaften (U1) und (U3) können zur Vermeidung von Redundanzen verwendet werden.

`Richtlinienwert`: Wie in Abb. 5.4 bereits angedeutet ist, ist die Klasse `Richtlinienwert` ein Aggregat, das sich aus mehreren Klassen zusammensetzt. Es wird in Abschnitt 5.2.5 behandelt.

`Meßmethode`: Wir beschreiben eine Meßmethode mittels der Attribute `Name` und `Meßhäufigkeit`.

`Bemerkung`: Neben dem Inhalt einer Bemerkung soll ein Attributwert für den Anspruchsgrad gehalten werden. Mit seiner Hilfe kann man einschätzen, ob eine Bemerkung auch für einen Nicht-Experten von Wert ist und diesem bei der Ausgabe mitgeteilt werden soll oder nicht.

`Bezugszeitraum`: Wir beschreiben einen Bezugszeitraum der Einfachheit halber mit einem Attribut `Zeitraum`. Es kann sinnvoll sein, weitere Attribute aufzunehmen, beispielsweise für Erläuterungen o.ä.

`Emissionsquelle`: Analog zu den Emissionsrichtlinien beschreiben wir Emissionsquellen mit Hilfe der Attribute `Name` und `Leistung/Kapazität`. Emissionsquellenobjekte nehmen nur an solchen Richtlinienregelungen teil, die mit einer Emissionsrichtlinie[4] verbunden sind. Diese Konsistenzbedingung kann im Objektmodell nicht beschrieben werden und soll in dieser Form hinzugefügt werden.
Für Emissionsquellen existieren zahlreiche unterschiedliche Bezeichnungen, die sich teilweise überschneiden. Um den Benutzer auch an dieser Stelle zu unterstützen, beschreiben wir die Begriffsähnlichkeiten mit einer Assoziation `ist_ähnlich_zu`. Diese Beziehungen sind ein wenig vergleichbar mit den Synonymbeziehungen zwischen zwei Parametern (s.o.). Der Unterschied liegt darin, daß Synonyme 100% identisch sind, während ähnliche Begriffe

[4] Das sind Richtlinien, die im Attribut `Richtlinienart` den Wert 'Emission' enthalten.

nur zu einem gewissen Grad deckungsgleich sind. Daher führen wir für die Assoziation ein Attribut `Ähnlichkeitsgrad`. Damit können wir beispielsweise abschätzen, inwieweit die Angabe einer Emissionsquelle durch den Benutzer auch auf eine verwandte oder ähnliche Quelle folgern läßt.

Diese Methodik, unsicheres Schließen durch die Angabe einer Wahrscheinlichkeit zu ermöglichen, ist dem Gebiet der Expertensysteme entnommen. Dort finden sich im Bereich der heuristischen Klassifikation *Evidenzen* oder *Wahrscheinlichkeiten* für sogenannte Symptom-Diagnose-Beziehungen. Der interessierte Leser sei auf [Pup90] und [Pup88] verwiesen.

Die Ähnlichkeitsbeziehungen unterliegen ähnlichen Eigenschaften wie die Synonymbeziehungen. Aufgrund der Gewichtung der Beziehung mit einem Ähnlichkeitsgrad sollte aber darauf verzichtet werden, die Transitivität der Ähnlichkeitsbeziehungen auszunützen, um Redundanzen zu vermeiden. Dagegen unterliegen die Reflexivität und die Symmetrie keinerlei Einschränkungen.

5.2.5 Richtlinienwert-Aggregat

Einen *Richtlinienwert* beschreiben wir ebenfalls mit Hilfe einer Aggregation. Das ist sinnvoll, weil die Richtlinienwerte in den Richtlinien eine sehr komplexe Struktur haben: sie bestehen aus

- einem *Richtlinienwertnamen*, wie etwa 'Grenzwert', 'Richtwert' oder 'Leitwert',
- einem *Wert*, wie etwa einem exakten Zahlenwert oder einem Intervall,
- einer *Wertart*[5], wie etwa 'Jahresmittelwert', und
- einer *Maßeinheit*.

Die Komponenten eines Richtlinienwert stellen in sich geschlossene Informationseinheiten dar, weshalb wir sie jeweils als Objekte modellieren. Daraus ergibt sich der Aufbau eines Richtlinienwert-Aggregats in Abb. 5.5.

Wir werden nun die einzelnen Klassen besprechen:

`Richtlinienwertname`: Wie oben bereits erwähnt, werden die Richtlinienwerte mit Namen versehen, die Auskunft über ihre gesetzliche Bindekraft geben. Ein solcher Name, wie etwa 'Grenzwert', wird in zahlreichen Richtlinienwerten verwendet (siehe Kardinalitäten in Abb. 5.5).

`Wert`: Ein Wert kann einerseits ein exakter Zahlenwert sein oder aber ein Intervall mit einer Unter- und einer Obergrenze. Dies läßt sich einfach mit einer Vererbungsbeziehung zwischen der Oberklasse `Wert` und den Unterklassen `Exaktwert` und `Intervallwert` modellieren.
Ein Wert ist ein individueller Bestandteil genau eines Richtlinienwerts (siehe Kardinalität $<1,1>$ für Werte in Beziehung zu einem Richtlinienwert-Aggregat).

[5] Wie aus den Richtlinienstrukturen in Anhang B ersichtlich ist, geben nicht alle Richtlinien eine Wertart für die geführten Richtlinienwerte an. Aus diesem Grund heraus ergibt sich die Kardinalität von $<0,1>$ in Abb. 5.5 für `Wertart`.

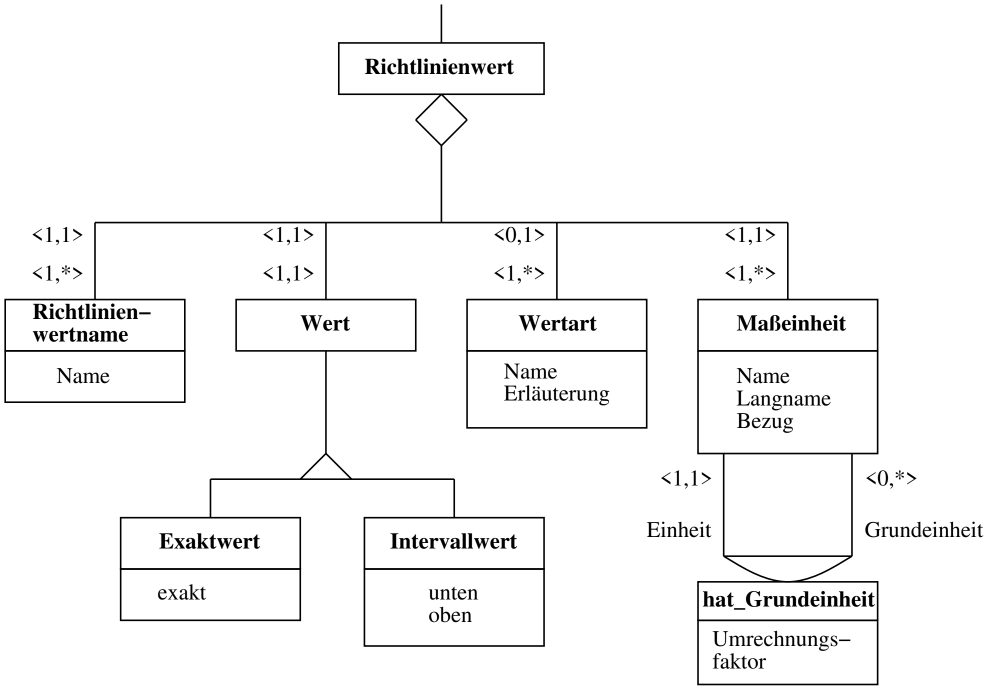


Abbildung 5.5: OMT-Objektmodell – Teil 5: Das Richtlinienwert-Aggregat

`Wertart`: Eine Wertart bedarf neben ihres Namens manchmal einer Erläuterung. Beispielsweise kann bei einem Monatsmittelwert vorgeschrieben sein, daß täglich zwei Proben zu entnehmen sind, wovon der jeweils höhere Wert in die Berechnung einfließen soll.
Eine Wertart ist nicht auf einen einzigen Richtlinienwert beschränkt, sondern kann in Beziehung zu mehreren stehen (siehe Kardinalität $< 1, * >$ für Wertarten in Beziehung zu einem Richtlinienwert-Aggregat).

`Maßeinheit`: Eine Maßeinheit hat neben einem Namen, wie etwa '°C', einen Langnamen, in diesem Fall 'Grad Celsius'. Darüberhinaus gibt es ein Attribut `Bezug`. Es dient dazu, eine Erläuterung oder Einordnung der Maßeinheit vorzunehmen. Beispielsweise werden manchmal die Anzahl der Mikroorganismen in einer Probe von 100 ml Wasser angegeben. In diesem Fall ist die Maßeinheit 'Stück' oder 'Anzahl Mikroorganismen'. Die Maßeinheit macht aber erst Sinn durch den Bezug 'in einer Probe von 100 ml Wasser'.
Um Maßeinheiten schnell ineinander umrechnen zu können, stellen wir jede Einheit in Verbindung zu einer Grundeinheit, in die wir sie mit dem entsprechenden Umrechnungsfaktor umrechnen können. Jede Maßeinheit hat genau eine Grundeinheit, während eine Maßeinheit Grundeinheit für beliebig viele Maßeinheiten sein kann (siehe entsprechende Kardinalitäten der Assoziation `hat_Grundeinheit`).

5.2.6 Benutzer und Anfragen

Wir haben die Diskurswelt in den vorangegangenen Abschnitten insoweit modelliert, als sich alle Daten, die in den Richtlinien vorhanden sind, im Objektmodell wiederfinden. Es fehlt nun noch die Modellierung des Benutzers und seines Zugangs zu den Informationen.

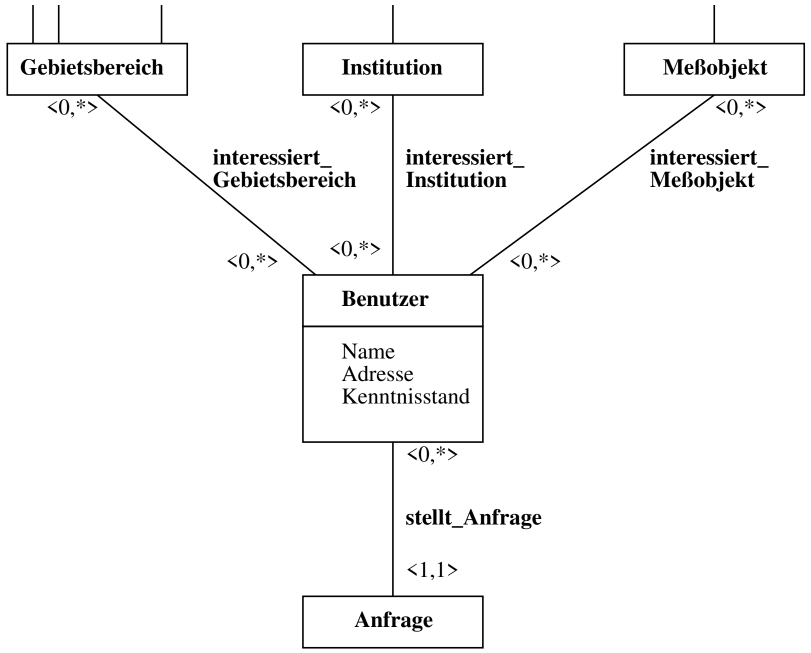


Abbildung 5.6: OMT-Objektmodell – Teil 6: Benutzer und Anfragen

Wir beschreiben einen *Benutzer* als Objekt einer Klasse `Benutzer`, das durch die Attribute `Name`, `Adresse` und `Kenntnisstand` charakterisiert wird. Darüberhinausgehend können noch weitere Attribute sinnvoll sein. Diese können je nach Bedarf der Klasse hinzugefügt werden.

Während die Attribute `Name` und `Adresse` eher einen administrativen Zweck haben, soll der Wert des Attributs `Kenntnisstand` dazu dienen, den Benutzer in seiner Vertrautheit mit der Struktur der Datenbasis und der Diskurswelt einzuschätzen. So kann beispielsweise in Verbindung mit dem Attribut `Anspruchsgrad` der Klasse `Bemerkung` entschieden werden, ob es sinnvoll ist, dem jeweiligen Benutzer die Bemerkung auszugeben oder nicht.

Ein Benutzer verschafft sich über eine *Anfrage* Zugang zur Datenbasis. Die Idee ist, eine Anfrage als Objekt zu modellieren, das zu einem Benutzer in Beziehung steht und das die notwendigen Operationen bereithält, um auf die Datenbasis zuzugreifen.

Um einen Benutzer weiter unterstützen zu können, setzen wir Benutzerobjekte in Beziehungen zu Gebietsbereichen, Institutionen und Meßobjekten. Dadurch drücken wir aus, daß sich ein Benutzer für einen Gebietsbereich oder eine Institution oder ein bestimmtes Meßobjekt interessiert. Das ist sinnvoll, da beispielsweise ein Angehöriger der EG-Verwaltung vornehmlich nach EG-Richtlinien suchen wird. Indem wir ihn in Beziehung zur Institution ‘EG’ setzen, drücken wir diese Präferenz aus.

Die genannten Zusammenhänge sind in Abb. 5.6 grafisch dargestellt.

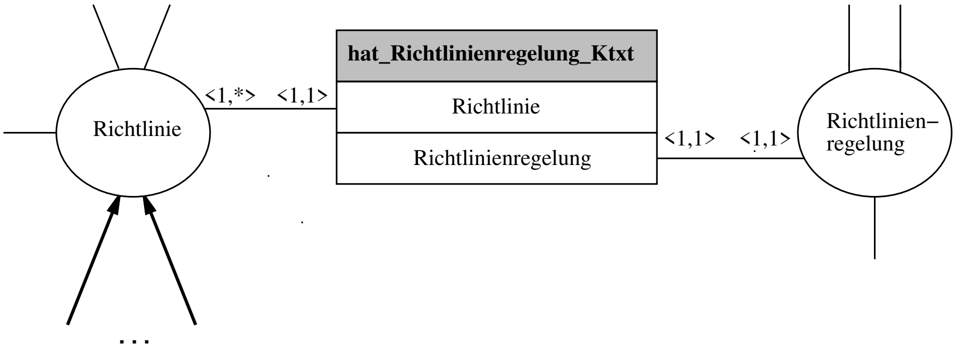


Abbildung 5.7: OPAQUE-EM-Objektmodell – Teil 1: Richtlinien und Richtlinienregelungen

5.3 Ein OPAQUE-EM-Objektmodell

In diesem Abschnitt setzen wir das oben entwickelte OMT-Objektmodell in die OPAQUE-EM-Notation (siehe Abschnitt 3.4 oder [Kop93]) um. Das entstehende Modell nennen wir das *OPAQUE-EM-Objektmodell*. Es hat die Eigenschaft, daß es zahlreiche OPAQUE-spezifische Aspekte – wie etwa die Strukturierung mit Hilfe von Kristallen und Kontexten – sichtbar macht. Wir betreten dadurch eine implementierungsnähere Entwurfsebene, auf der wir mit Hilfe der Konzepte modellieren können, die OPAQUE anbietet.

Die Umsetzung von der OMT- in die OPAQUE-EM-Notation erfolgt beinahe mechanisch; die dafür benötigten Techniken wurden in Abschnitt 3.5 beschrieben. Deshalb verzichten wir in den folgenden Abschnitten auf eine Beschreibung der Transformation.

Das OMT-Objektmodell wurde in den Abb. 5.1 – 5.6 grafisch dargestellt. Wir setzen nun jede einzelne dieser Abbildungen in die entsprechende OPAQUE-EM-Notation um. Um die Analogie zwischen OMT- und OPAQUE-EM-Objektmodell zu bewahren, haben wir die Gliederung aus Abschnitt 5.2 in diesem Abschnitt übernommen und analoge Abschnittsüberschriften und Bildunterschriften gewählt.

5.3.1 Richtlinien und Richtlinienregelungen

5.3.2 Institutionen, Meßobjekte und Gebietsbereiche

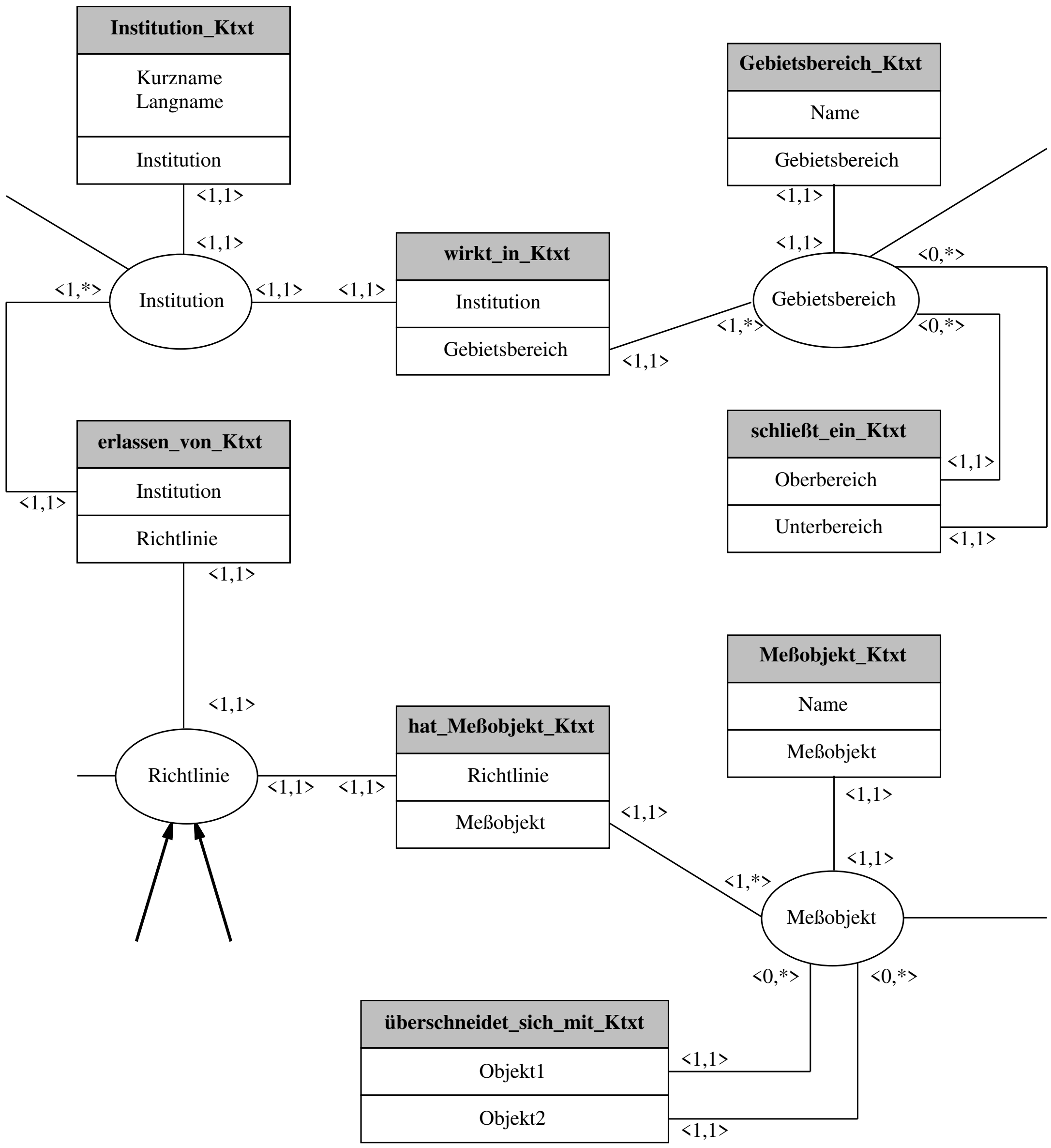


Abbildung 5.8: OPAQUE-EM-Objektmodell – Teil 2: Institutionen, Meßobjekte und Gebietsbereiche

5.3.3 Vererbungshierarchie der Richtlinien

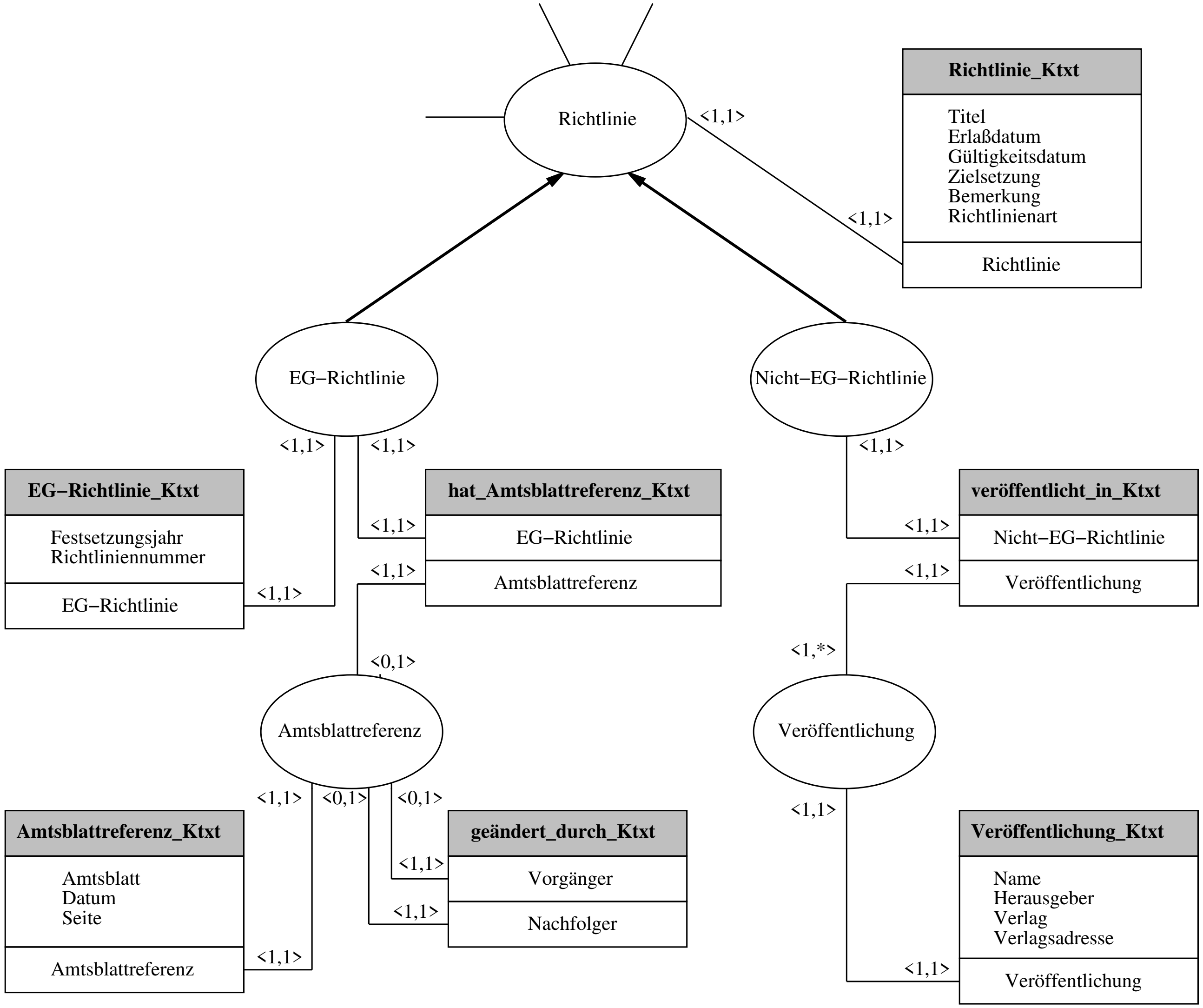


Abbildung 5.9: OPAQUE-EM-Objektmodell – Teil 3: Die Richtlinien-Vererbungshierarchie

5.3.4 Richtlinienregelungs-Aggregat

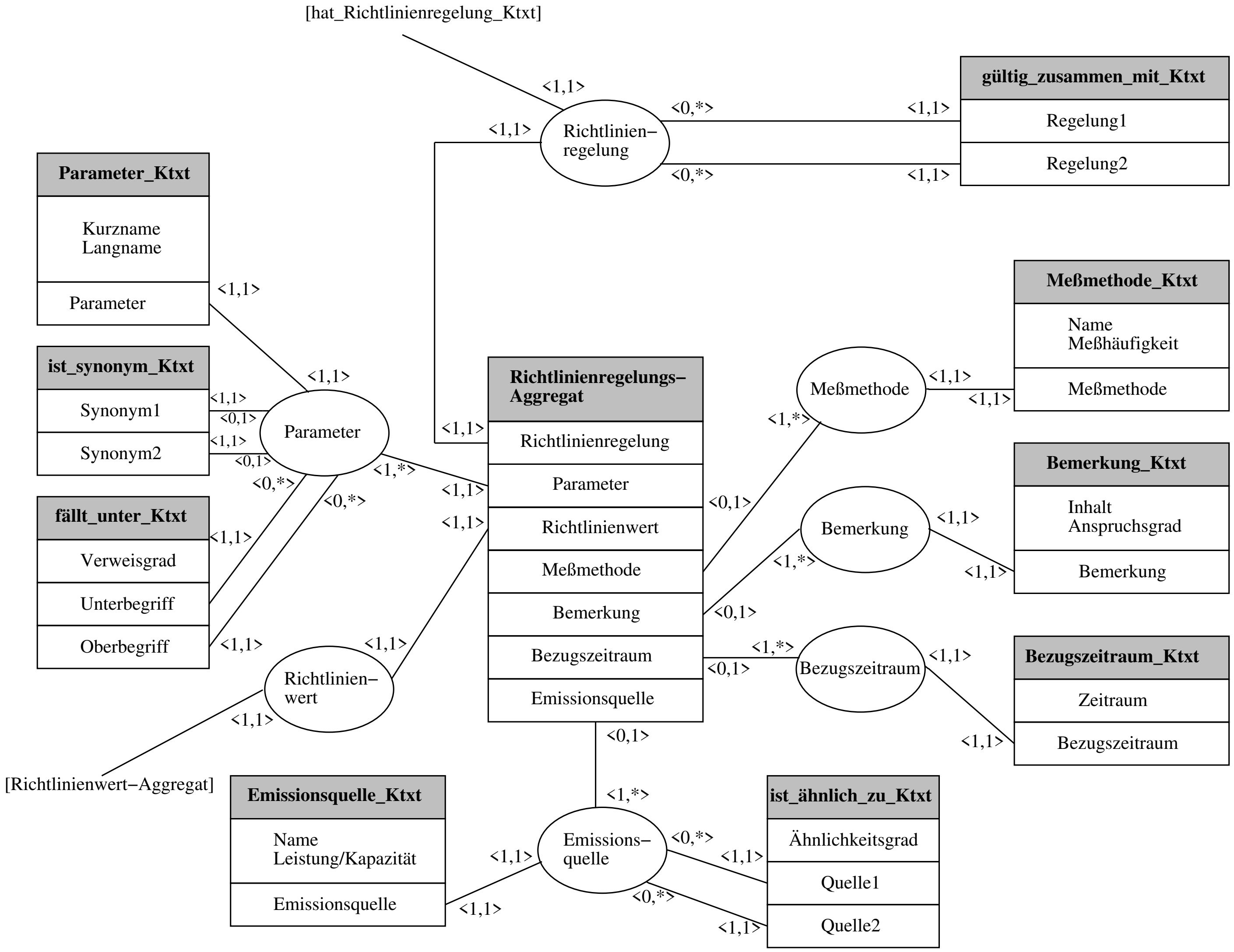


Abbildung 5.10: OPAQUE-EM-Objektmodell – Teil 4: Das Richtlinienregelungs-Aggregat

5.3.5 Richtlinienwert-Aggregat

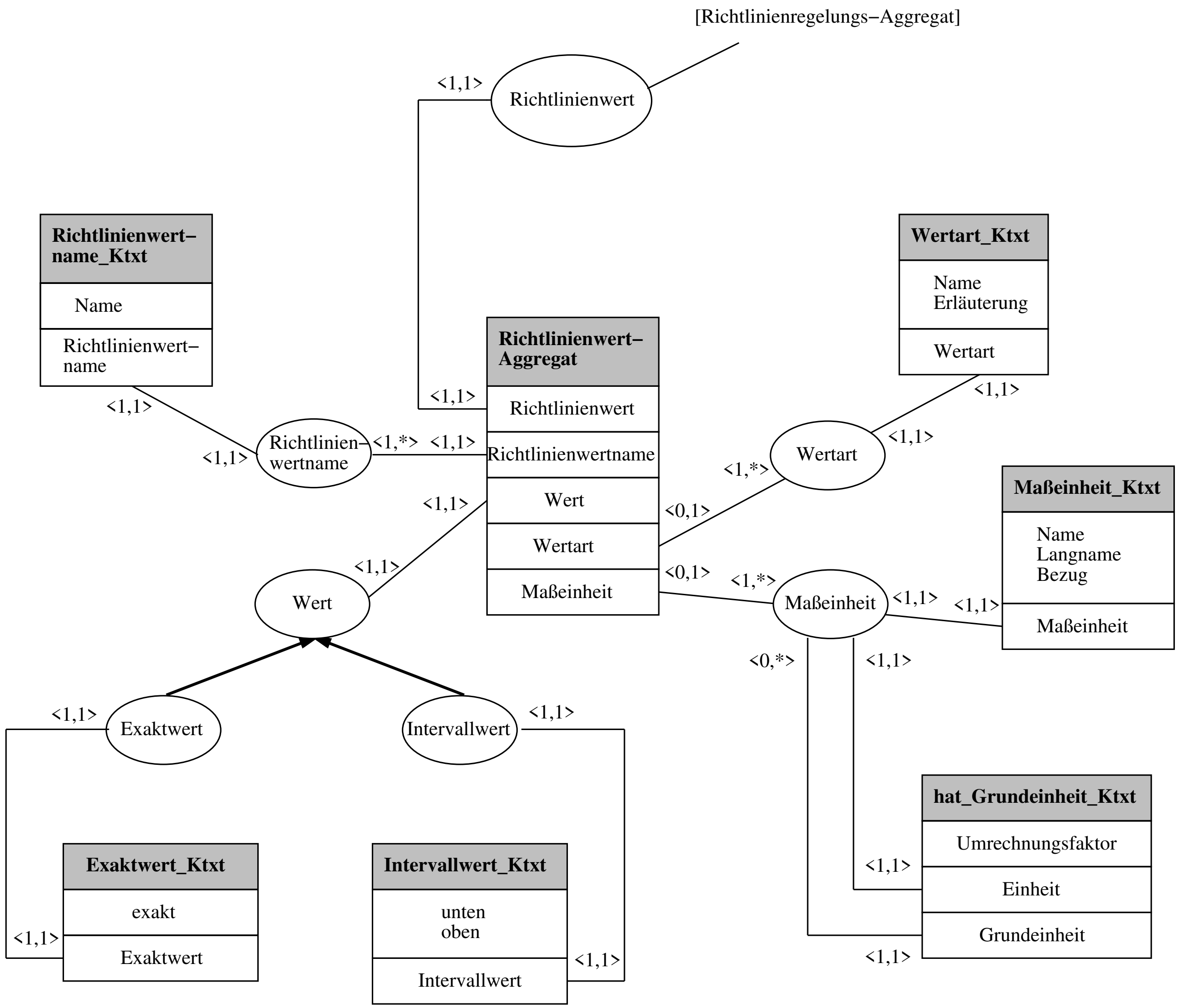


Abbildung 5.11: OPAQUE-EM-Objektmodell – Teil 5: Das Richtlinienwert-Aggregat

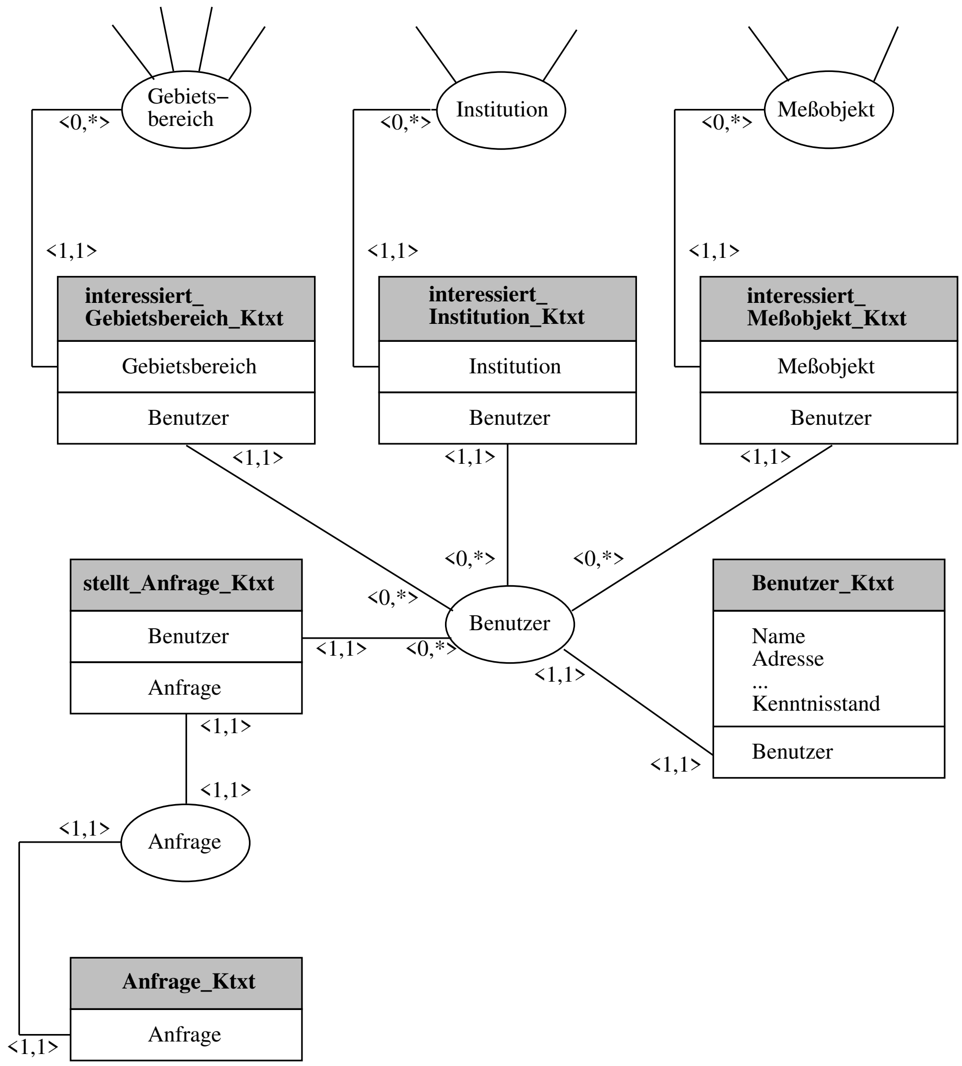


Abbildung 5.12: OPAQUE-EM-Objektmodell – Teil 6: Benutzer und Anfragen

5.3.6 Benutzer und Anfragen

5.4 Bemerkungen zum statischen Entwurf

Wir haben in diesem Kapitel zwei Objektmodelle entworfen:

- ein OMT-Objektmodell und
- ein OPAQUE-EM-Objektmodell.

Das OMT-Objektmodell zeichnet sich insbesondere durch seine Übersichtlichkeit gegenüber dem OPAQUE-EM-Objektmodell aus. Man vergleiche dazu beispielsweise Abb. 5.4 und das OPAQUE-EM-Pendant Abb. 5.10. Es eignet sich aufgrund der Übersichtlichkeit besser für globale Systembetrachtungen, auch weil es natürliche Konzepte, wie beispielsweise Aggregationen, auch noch als solche beschreibt.

Andererseits bietet erst das OPAQUE-EM-Objektmodell Ansatzpunkte, kontextsensitive Aspekte zu berücksichtigen. OMT bietet dafür keinerlei Möglichkeiten, wohingegen OPAQUE-EM die Aufteilung von Struktur- und Verhaltensbeschreibung auf mehrere Kontexte beschreiben kann. Das OPAQUE-EM-Objektmodell ist implementierungsnäher und eignet sich eher für spezifische Detailbetrachtungen.

Zahlreiche OMT-Konzepte, wie beispielsweise Objektklassenstruktur, Assoziationen und Aggregationen, werden allein durch ein Zusammenspiel von Kristall- und Kontextschablonen dargestellt, so daß durch die Transformation von OMT nach OPAQUE-EM aus einer manigfaltigen eine einfachere, implementierungsnähere Darstellung entsteht, die bestimmte Modellierungsentscheidungen nur noch bei genauer Betrachtung erkennen läßt: beispielsweise kann man bei Betrachtung von Abb. 5.10 nur noch an der Namensgebung erkennen, daß die Kontextschablone `Richtlinienregelungs-Aggregat` aus einer Aggregation entstanden ist (und nicht etwa aus einer Mehrfach-Assoziation). Es ist auch nicht mehr zu erkennen, daß die Rolle `Richtlinienregelung` eine den übrigen Rollen übergeordnete Position hat, da über sie konzeptuell die mit den übrigen Rollen verbundenen Objekte zu einem Aggregat zusammengefaßt werden sollen.

Das bedeutet, daß sich Entwurfsentscheidungen, die das OMT-Objektmodell betreffen, nahtlos in das OPAQUE-EM-Objektmodell übertragen lassen. Eine Durchgängigkeit in umgekehrter Richtung, also vom OPAQUE-EM- zum OMT-Objektmodell, ist dagegen nicht immer gewährleistet.

Kapitel 6

Vorüberlegungen zum dynamischen Entwurf

6.1 Erwartete Funktionalität

Bevor wir mit dem dynamischen Entwurf beginnen, wollen wir die Funktionalität beschreiben, die wir von der Datenbank erwarten.

Natürlich wollen wir jedes in der Datenbasis gespeicherte Objekt wiederauffinden können. Wie man an der statischen Modellierung sieht, gibt es zwei Möglichkeiten, ein Objekt zu identifizieren:

- durch seine Attributwerte (z.B. Richtlinien, die *1988* erlassen wurden) oder
- durch die mit ihm in Beziehung stehenden Objekte (z.B. alle Richtlinien, die von der *WHO* erlassen wurden)

Das zu entwerfende Anfragemodell sollte beide Suchmöglichkeiten unterstützen.

Als Ergebnis einer Anfrage soll das System eine Menge von Objekten zurückliefern, die die Kriterien der Anfrage erfüllen.

Ein Problem, das ganz besonders in der hier behandelten Diskurswelt der Umweltrichtlinien zum Tragen kommt, ist die Vielfalt der Fachbegriffe, Suchparameter etc. und die unterschiedliche Terminologie der Richtlinien, die allesamt dazu führen, daß der Benutzer in der Regel über Fachwissen verfügen muß, um eine *präzise* Anfrage stellen zu können. Das System muß insbesondere einen mit dem Fachvokabular nicht vertrauten Benutzer unterstützen, so daß dieser entweder in die Lage versetzt wird, präzise Anfragen stellen zu können, oder daß die von ihm formulierten, unpräzisen Anfragen dennoch zum gewünschten Ergebnis führen.

6.2 Möglichkeiten, die Funktionalität umzusetzen

Es stellt sich nun die Frage, wie die in Abschnitt 6.1 aufgestellten Anforderungen für die Funktionalität umgesetzt werden können. Insbesondere geht es darum, eine grundsätzliche Architektur für das zu entwerfende Anfrage- und das Suchmodell zu finden. In der Praxis finden im wesentlichen zwei Ansätze Anwendung:

- ein prozeduraler, d.h. auf elementaren Operationen basierender Ansatz, bei dem es dem Benutzer oder Anwendungsprogrammierer obliegt, die elementaren Operationen in geeigneter Form zu einer Suche zu kombinieren, oder
- ein deskriptiver Ansatz, bei dem der Benutzer nur beschreibt, *was* er in der Datenbasis sucht, und nicht, *wie* die gewünschte Lösung zu suchen ist.

Beide Ansätze lassen sich auch für unsere Zwecke verwenden:

Prozeduraler Ansatz: Dem Benutzer werden 'Anfrageprimitive', gewisse elementare Anfragemöglichkeiten, angeboten, mit deren Hilfe sich komplexe Anfragen aufbauen lassen. Die Anfrageprimitive können mit elementaren Suchoperationen, den Suchprimitiven, bearbeitet werden, so daß sich aus der Komposition einer Anfrage aus Anfrageprimitiven eine Komposition der Suche aus Suchprimitiven ergibt. Eine Suche setzt sich daher aus mit den Suchprimitiven korrespondierenden Teilsuchen zusammen.

Dabei liegt es in der Verantwortung des Benutzers, die Primitive zu einer komplexen Anfrage zusammenzufügen. Der Benutzer baut den Suchoperatorbaum also selbst auf.

Im Hinblick auf diesen Ansatz ergeben sich die folgenden Fragen:

- Welches sind die Anfrage- und Suchprimitive?
- Wie werden sie zu komplexen Anfragen zusammengesetzt?
- Wie werden die Teilsuchen einer komplexen Suche koordiniert?

Deskriptiver Ansatz: Dem Benutzer wird eine Art Formular bzw. eine Schablone vorgegeben, in der er seine Suchkriterien einträgt. Eine Möglichtkeit ist beispielsweise die Vorgabe einer Objektstruktur, die der Benutzer soweit wie möglich ausfüllt. Das Ergebnis einer Anfrage wäre damit ein rudimentäres Objektgerüst. Mit Hilfe eines Pattern Matching Verfahrens können wir in der Datenbasis nach Objekten suchen, die den in dem Gerüst dargestellten Suchkriterien entsprechen. Alle Objekte, die dieser Schablone entsprechen, qualifizieren sich für die Lösung.

Das führt im Gegensatz zum oben genannten prozeduralen Ansatz dazu, daß es keine Unterscheidung in der Bearbeitung einfacher und komplexer Anfragen gibt. Der Benutzer muß sich auch keine Gedanken um die prozeduralen Abläufe bei der Suche machen, da dies vom System selbst kontrolliert wird. Dieses kann dadurch korrigierend (also z.B. optimierend) eingreifen.

Für diesen Ansatz ergeben sich die folgenden Fragen:

- Wie ist eine Anfrage aufgebaut?
- Wie läuft die Suche ab?

Wir werden die beiden Ansätze in den Kapiteln 7 und 8 verfolgen.

Kapitel 7

Dynamischer Entwurf: Ein erster, prozeduraler Ansatz

7.1 Vorgehensweise

In diesem Kapitel entwickeln wir einen Entwurf, der das dynamische Verhalten des Datenbanksystems beim Wiederauffinden von Informationen modelliert. Wir verfolgen dazu die Idee, eine Anfrage aus *Anfrageprimitiven* zusammenzusetzen. Jedes dieser Anfrageprimitive kann mit einer einfachen Suchoperation, einem *Suchprimitiv*, beantwortet werden. Analog zur Zusammensetzung der Anfrage aus Anfrageprimitiven läßt sich die Suche nach der Antwort aus den korrespondierenden Suchprimitiven zusammensetzen. Die Komposition der Anfrage aus Primitiven entspricht also gleichzeitig einem Suchplan.

Der Vorteil dieses Ansatzes ist, daß sich komplexe und umfangreiche Anfragen des Benutzers auf einfache, leicht zu bearbeitende Anfragen zurückführen lassen. Daher sind einfache und unkomplizierte Suchoperationen zu erwarten.

Dieser unübersehbaren Vorteil wird allerdings durch den Nachteil erkauft, daß dem Benutzer durch die Komposition seiner Anfrage gleichzeitig die Komposition der Suche obliegt. Dies stellt einerseits höhere Ansprüche an den Benutzer, andererseits nimmt es dem Datenbanksystem die Möglichkeit, korrigierend in den Suchprozeß einzugreifen, um ihn beispielsweise zu optimieren.

Erst eine genaue Abwägung der Vor- und Nachteile wird Aufschluß über die Güte dieses Ansatzes geben. Wir wollen ihn deshalb innerhalb dieses Kapitels in hinreichender Ausführlichkeit behandeln, um aus einer Bewertung heraus wichtige Hinweise für den in Kapitel 8 verfolgten zweiten Ansatz zu erhalten.

Wir werden nun im Rahmen dieses Kapitels auf die folgenden Fragen eingehen:

- Welches sind die kleinsten Einheiten für eine Anfrage bzw. eine Suche? Welches sind also die Anfrage- und Suchprimitive?
- In welcher Weise werden die Primitive zu einer Anfrage kombiniert? Wie stellt der Benutzer eine Anfrage? Mit anderen Worten: wie gestaltet sich das Anfragemodell?
- Wie findet eine Suche in der Datenbasis statt? Welches sind die Suchoperationen? Wie

werden sie implementiert?

Aus diesen drei Fragestellungen heraus ergibt sich auch die Gliederung dieses Kapitels:

- Abschnitt 7.2 beschreibt, was wir unter Anfrage- und Suchprimitiven verstehen und wie sie zu komplexen Anfragen bzw. Suchen kombiniert werden können.
- Abschnitt 7.3 stellt ein Anfragemodell vor, das Anfrageprimitive zu komplexen Anfragen kombiniert und dadurch gleichzeitig die Suche steuert.
- In den Abschnitten 7.4 bis 7.6 entwerfen wir die Suchprimitive. Das sind die Operationen, mit deren Hilfe Objekte einer bestimmten Klasse in der Datenbasis wiederaufgefunden werden können. Der Entwurf der Operationen geschieht top-down und gliedert sich in drei Schritte, die wir in den genannten drei Abschnitten darstellen werden:
  - Abschnitt 7.4 geht von einer globalen Betrachtung der Objektmodelle und der Überlegung aus, wie sich die Objekte in der Datenbasis wiederauffinden lassen. Aus dieser Betrachtung heraus entsteht eine einfache, fast mechanische Herleitung der Spezifikationen der Suchoperationen.
  - In Abschnitt 7.5 betrachten wir nicht mehr das gesamte Objektmodell, sondern nur einzelne Klassen. Ihnen werden eine Reihe von Operationen zugeordnet, die die Suche unterstützen und deren Notwendigkeit sich während der Spezifikation der Suchoperationen ergeben hat.
  - In Abschnitt 7.6 stellen wir anhand von Beispielen dar, wie die bis dahin spezifizierten Operationen in OPAQUE implementiert werden können.
- Abschnitt 7.7 beschließt das Kapitel mit einer kurzen Bewertung dieses Entwurfs.

7.2 Anfrage- und Suchprimitive

Die Objekte sind die Träger der Informationen, die in der Datenbasis abgelegt sind. Der zentrale Punkt des Anfragemodells ist es, die durch die Anfrage gesuchten Objekte (und nur diese) in der Datenbasis wiederaufzufinden. Es gibt zwei Mittel, die Objekte zu finden:

1. durch die Angabe von Attributwerten und
2. durch die Angabe von Objekten, die zu den Suchobjekten in Beziehung stehen.

Punkt 2 enthält bereits einen Hinweis auf eine Iteration bzw. eine mögliche Zerlegung der Suche, da wir die Objekte, die zu den Suchobjekten in Beziehung stehen sollen, zunächst einmal auf irgendeine Art und Weise umschreiben müssen. Es liegt nahe, dies ebenfalls durch eine Anfrage zu tun. Dadurch entsteht ein Aufsetzen einer Anfrage auf einer anderen und insgesamt eine komplexe Anfrage, die aus der Iteration mehrerer Teilanfragen besteht. Diese Teilanfragen stellen die Einheiten einer Anfrage dar, die wir als *Anfrageprimitive* bezeichnen wollen.

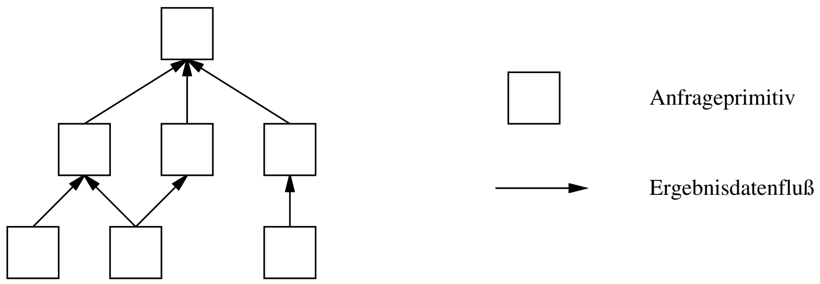


Abbildung 7.1: Zusammensetzung einer komplexen Anfrage aus Anfrageprimitiven

In Abb. 7.1 sehen wir eine schematische Darstellung der Komposition von Anfrageprimitiven zu einer komplexen Anfrage: die Ergebnisse eines Anfrageprimitivs werden von einem anderen weiterverarbeitet, das seine Ergebnisse wiederum weitergeben kann. Jedes Kästchen stellt ein Primitiv dar; die Pfeile beschreiben den Datenfluß der Ergebnisse. Dabei ist es durchaus möglich, daß das Ergebnis eines Primitivs mehrfach verwendet werden kann: z.B. könnte es sein, daß ein Anfrageprimitiv nach Institutionsobjekten sucht, die einerseits zum Auffinden von Richtlinien, andererseits zum Auffinden von Gebietsbereichen verwendet werden können (vgl. dazu beispielsweise Abb. 5.8).

Die Suche und damit die Beantwortung einer Anfrage verhält sich ganz analog, da wir die Suche nach einer Antwort für die Gesamtanfrage aus Suchen nach Teilantworten zusammensetzen können. Dies geschieht entsprechend Abb. 7.1: Die Anfrageprimitive werden durch *Suchprimitive* beantwortet. Die Ergebnisse der Teilsuchen fließen in die Beantwortung der nächsten Teilanfrage ein. So stellt Abb. 7.1 nicht nur die Zusammensetzung einer Anfrage, sondern gleichzeitig auch einen Operatorbaum zur Beantwortung der Anfrage dar.

Im folgenden Abschnitt stellen wir *eine* mögliche Realisierung einer Anfrageschnittstelle vor, die auf der Komposition von Teilanfragen zu Anfragen beruht.

7.3 Das Anfragemodell

7.3.1 Konzeption

Es gibt mehrere Möglichkeiten, die Kompositionen einer Anfrage aus Teilanfragen vorzunehmen. Man kann dem Benutzer beispielsweise eine graphische Schnittstelle anbieten, mit deren Hilfe er seine Anfrage, etwa in der Art und Weise wie in Abb 7.1 dargestellt, formulieren kann. Der dadurch gegebene Abhängigkeitsgraph zwischen den Anfrageprimitiven dient dem System dann dazu, einen Plan für die Abarbeitung der Suchprimitive zu erstellen.

Eine weitere Möglichkeit ist die folgende: Anstatt den Benutzer die Gesamtanfrage auf einmal formulieren zu lassen, kann man den Anfrage- und Suchprozeß miteinander verzahnen. Das bedeutet, daß der Benutzer eine Anfrage in Form eines Primitivs stellt. Das Ergebnis wird im Anschluß ermittelt. Danach kann der Benutzer eine neue Anfrage in Form eines Primitivs stellen und dabei auf das Ergebnis einer (nicht notwendigerweise unmittelbar) vorangegangenen Anfrage zurückgreifen.

Wir haben diesen zweiten Ansatz gewählt. Er unterscheidet sich prinzipiell nicht vom erst-genannten. Der Benutzer baut seine Anfrage sukzessiv aus Teilanfragen auf, deren Ergebnisse zunächst ermittelt werden, bevor sie in einer folgenden Teilanfrage verwendet werden können. Der Ablauf einer Anfrage ist im folgenden Algorithmus zusammengefaßt.

7.3.2 Der Anfragealgorithmus

1. Der Benutzer erzeugt ein Anfrageobjekt (vgl. Abb. 5.6).

2. Eine benutzerspezifische Anfrageschnittstelle wird aufgerufen.

3. Innerhalb der Anfrageschnittstelle wählt der Benutzer eine Objektklasse aus, die Ziel seiner Anfrage ist. Für diese Zielklasse wird eine leere (klassenspezifische) Ergebnismenge initialisiert.

4. Der Benutzer gibt die ihm bekannten Werte der Attribute ein, die ein Objekt seiner Zielklasse annehmen sollte. Falls in vorangegangenen Suchen bereits Objekte gefunden wurden, kann angegeben werden, ob die gesuchten Objekte mit diesen bereits gefundenen Objekten einer vorangegangenen Anfrage in Beziehung stehen sollen (siehe Schritt 6). Diese Suchkriterien werden mit einem logischen UND verknüpft.

5. Die Suche wird gestartet. Ergebnis ist eine möglicherweise leere Menge von Objekten der Zielklasse.

6. Der Benutzer hat die Möglichkeit,

   (a) sich die Ergebnismenge ausgeben zu lassen,

   (b) mit einer Suche nach weiteren Objekten *derselben* Zielklasse fortzufahren. Dazu erfolgt ein Sprung nach Schritt 4. Die Objekte, die bei dieser fortgesetzten Suche gefunden werden, werden zur bisherigen Ergebnismenge hinzugefügt, so daß eine logische ODER-Verknüpfung entsteht[1].

   (c) mit einer Suche nach Objekten einer *anderen* Zielklasse fortzufahren. Dazu erfolgt ein Sprung nach Schritt 3.

Die Iterationen durch die Schritte 6b und 6c macht es notwendig, daß das Anfrageobjekt, das die gesamte Anfrage bearbeitet, die Zwischenergebnisse der Teilsuchen speichert. Dazu dienen die in Schritt 3 erwähnten Ergebnismengen, von denen es eine für jede Objektklasse gibt und die die bei den Suchen aufgefundenen Objekte der jeweiligen Klasse speichern. Sie können dann dazu dienen, Objekte anderer Klassen aufzufinden, zu denen sie in Beziehung stehen (vergleiche dazu Schritt 4).

Die zentralen Abschnitte des Algorithmus' sind die Schritte 4 und 5, die jeweils ein Anfrageprimitv aufnehmen (Schritt 4) bzw. ein Suchprimitiv ausführen. Die übrigen Schritte dienen allein der Komposition mehrerer solcher Primitive zu einer komplexen Anfrage.

Wir erläutern die oben genannten Schritte des Anfragealgorithmus' nun anhand eines Beispiels.

[1]In Verbindung mit der UND-Verknüpfung in Schritt 4 entsteht daher eine Suche, die sich auf eine logische Formel in disjunktiver Normalform zurückführen läßt.

7.3.3 Eine Beispielanfrage

Als Anfrage wollen wir alle Veröffentlichungen suchen, in denen Emissionsrichtlinien des Bundesforschungsministeriums (BMFT) oder des Bundesumweltministeriums (BMUR) publiziert wurden.

Man betrachte dazu etwa die Abb. 5.8 und 5.9, die die für diese Anfrage relevanten Ausschnitte aus dem OPAQUE-EM-Objektmodell zeigen. Die Anfrage läuft dann wie folgt ab:

- Der Benutzer erzeugt ein Anfrageobjekt (Schritt 1).
- Die Anfrageschnittstelle wird aufgerufen (Schritt 2).
- `Institution` wird als Zielklasse ausgewählt. (Schritt 3).
- Als Wert für das Attribut `Kurzname` geben wir 'BMFT' an (Schritt 4).
- Die Suche wird gestartet und liefert eine Menge von Objekten, die 'BMFT' als Attributwert für `Kurzname` haben (Schritt 5).
- Da wir uns nicht nur für die Institution BMFT interessieren, sondern auch für das Bundesumweltministerium springen wir gemäß Schritt 6b wieder in Schritt 4.
- Als Wert für das Attribut `Kurzname` geben wir 'BMUR' an (Schritt 4).
- Die Suche wird gestartet und fügt zu der bisher gefundenen Menge von Institutionen die Objekte hinzu, die 'BMUR' als Attributwert für `Kurzname` haben (Schritt 5).
- Wir verfügen nun über alle für die Anfrage relevanten Institutionsobjekte, so daß wir nun die zu diesen Objekten in Beziehung stehenden Richtlinien suchen müssen. Dazu springen wir gemäß Schritt 6c zu Schritt 3.
- `Richtlinie` wird als Zielklasse ausgewählt[2]. Dazu wird die Ergebnismenge für Richtlinien mit der leeren Menge initialisiert (Schritt 3). Die Ergebnismenge für Institutionen ist davon nicht betroffen.
- Wir geben an, daß die oben gefundenen Institutionen zur Suche verwendet werden sollen und tragen unter dem Attribut `Richtlinienart` den Wert 'Emission' ein (Schritt 4).
- Die Suche nach den mit den Institutionen verbundenen Emissionsrichtlinien wird gestartet (Schritt 5).
- Wir haben nun alle Emissionsrichtlinien des BMFT und des BMUR in der Ergebnismenge. Wir müssen nun nur noch die mit diesen Objekten verbundenen Veröffentlichungen finden. Daher springen wir gemäß Schritt 6c zu Schritt 3.
- Wir wählen `Veröffentlichung` als Zielklasse aus. Es wird eine leere Ergebnismenge für Veröffentlichungen initialisiert (Schritt 3).
- Wir geben an, daß die gefundenen Richtlinien zur Suche verwendet werden sollen (Schritt 4).

[2] Alternativ hätte man ebenso die Klasse `Nicht-EG-Richtlinie` angeben können.

- Die Suche nach den mit den Richtlinien verbundenen Veröffentlichungen wird gestartet (Schritt 5).
- Das Ergebnis der Suche wird ausgegeben (Schritt 6a).

Der Ansatz für den in den Abschnitten 7.4 – 7.6 durchgeführten Entwurf ist Schritt 5 des oben genannten Algorithmus', der die Suche nach den Objekten initiiert. Den dafür benötigten Suchoperationen wollen wir uns nun zuwenden.

7.4 Globale Entwurfsebene

7.4.1 Motivation

Ausgangspunkt des Entwurfs ist die Frage, wie der Benutzer Objekte einer Klasse in der Datenbasis finden kann. Einen solchen Vorgang haben wir in Abschnitt 7.2 als Suchprimitiv bezeichnet. Wir beginnen dazu mit einer globalen Betrachtung des Schemas: die Daten sind gemäß des OPAQUE-EM-Objektmodells (Abb. 5.7 – Abb. 5.12) strukturiert; für Betrachtungen, die OPAQUE-spezifische Aspekte vernachlässigen, können wir auch das OMT-Objektmodell (Abb. 5.1 – Abb. 5.6) heranziehen. Diese beiden Modelle beschreiben für jede Klasse die Attribute und die Verbindungen (zu anderen Klassen), die notwendig sind, um Objekte dieser Klasse zu identifizieren.

Beispiel

Betrachten wir Abb. 7.2: sie zeigt die OPAQUE-Objektklasse `Gebietsbereich` mit ihren Kontextschablonen und den mit ihr verbundenen Objektklassen.

Das einzige Attribut von `Gebietsbereich` ist `Name` im Kontext `Gebietsbereich_Ktxt`. Suchen wir nun Objekte aus der Klasse `Gebietsbereich`, so können wir die gewünschten Objekte beispielsweise dadurch einschränken, indem wir einen Wert für das Attribut `Name` angeben und nach allen Gebietsbereichsobjekten[3] suchen, die diesen Attributwert enthalten. Als Ergebnis erhalten wir dann eine Menge von Gebietsbereichen.

Alternativ oder zusätzlich können wir auch den Kontext `wirkt_in_Ktxt` verwenden, um die gewünschten Gebietsbereiche zu suchen. Das bedeutet in diesem Fall, daß wir eine oder mehrere Institutionen angeben, die mit den gesuchten Gebietsbereichen über den Kontext `wirkt_in_Ktxt` in Verbindung stehen sollen.

Ähnlich kann man mit Hilfe von anderen Gebietsbereichsobjekten und den `schließt_ein_Ktxt` Kontexten oder mit Hilfe von Benutzern und `interessiert_Gebietsbereich_Ktxt` Kontexten suchen.

Zusammenfassend können wir also Objekte der Klasse `Gebietsbereich` suchen, indem wir

- einen Attributwert für `Name`,

[3] Wir werden im weiteren Verlauf manchmal auch einfach von *Gebietsbereichen* sprechen anstatt von Gebietsbereichsobjekten. Ähnliches gilt für Institutionen und Institutionsobjekte, Richtlinien und Richtlinienobjekte etc.

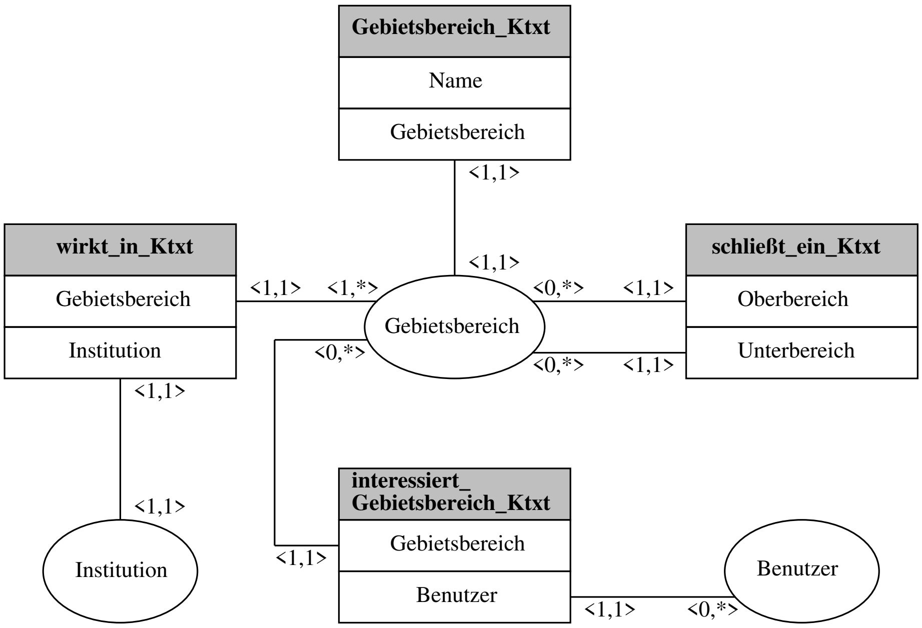


Abbildung 7.2: Die OPAQUE Objektklasse `Gebietsbereich`

- eine Menge von Institutionen,
- eine Menge von Oberbereichen,
- eine Menge von Unterbereichen und
- eine Menge von Benutzern

angeben. Jede einzelne Angabe stellt also ein Kriterium dar, das dazu dient, den Suchraum einzuschränken. Diese Kriterien können alternativ mit einer UND- oder mit einer ODER-Beziehung verknüpft werden.

Wie aus dem gezeigten Beispiel ersichtlich wird, lassen sich aus den Objektmodellen des vorangegangenen Kapitels beinahe mechanisch die Suchkriterien ableiten, mit denen man Objekte einer bestimmten Klasse auswählen kann. Es liegt nahe, für jede im Schema vorhandene Objektklasse eine Suchoperation anzugeben, die als Parameter die jeweils möglichen Suchkriterien erhält und sie mit UND verknüpft. Alternativ wäre an dieser Stelle auch eine ODER-Verknüpfung möglich.

Wir können diese Suchoperationen entweder auf die betreffenden Objektklassen verteilen oder an einer zentralen Stelle sammeln (Abb. 7.3 und 7.4) und sie dort dem Benutzer zur Verfügung stellen. Der Vorteil der dezentralen Lösung liegt darin, daß die Suchoperationen ja auf der Struktur einer bestimmten Objektklasse arbeiten. Würde sich diese Struktur einmal ändern, beispielsweise bei einer Erweiterung der Diskurswelt um neue Objektklassen und Beziehungen, so müßte sich die Implementierung der Suchoperation ebenfalls ändern. Aus softwaretechnischen Gründen ist es wünschenswert, notwendige Änderungen möglichst lokal zu halten, um damit den Aufwand und die Fehleranfälligkeit eines solchen Eingriffs möglichst klein zu halten. Ändert sich also die Struktur einer Objektklasse, so wäre es sinnvoll, die Suchoperation, die auf dieser

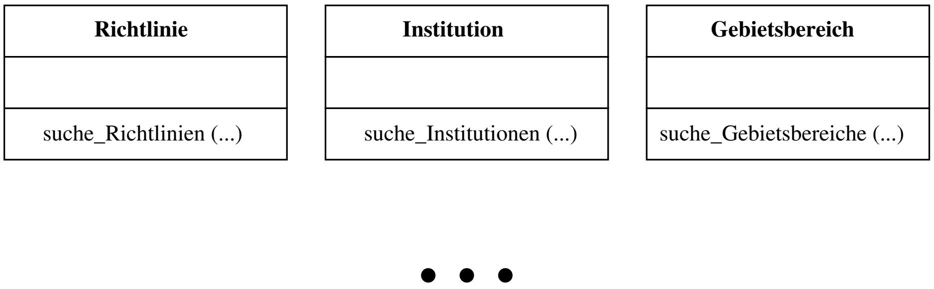


Abbildung 7.3: Dezentrale Lösung: Verteilung der Suchoperationen auf die einzelnen Objektklassen

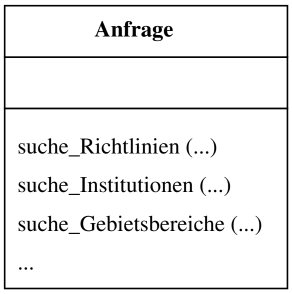


Abbildung 7.4: Zentrale Lösung: Sammlung aller Suchoperationen in `Anfrage`, also einer einzigen Objektklasse

Struktur aufbaut und deren Implementierung sich deshalb ebenfalls ändern müßte, direkt bei der Struktur, also der Klasse zu implementieren.

Andererseits würden wir dann immer ein Objekt der Objektklasse benötigen, um die Suchoperation aufrufen zu können. Nun steht die Suchoperation zu keinem Objekt der Klasse direkt in einer natürlichen Beziehung – im Gegenteil: sie sucht auf der Gesamtheit der vorhandenen Objekte der Klasse. Wir könnten uns dadurch behelfen, daß wir zunächst ein neues Objekt der Klasse erzeugen, dem wir keine weitere Bedeutung zumessen, sondern es einfach als das Objekt zu verwenden, das die Suche durchführt. Das wäre sehr unnatürlich und einem Anwendungsprogrammierer später möglicherweise nicht einsichtig.

Wir wollen deshalb die zentrale Lösung annehmen und alle Suchoperationen in der Klasse `Anfrage` anbieten. Dies ist softwaretechnisch vertretbar, da wir dadurch ein einziges Suchmodul erhalten, das auf der Struktur der Datenbasis aufsetzt. Außerdem wird die Suche natürlicherweise durch eine Anfrage, d.h. ein Anfrageobjekt, ausgelöst. Wir stellen uns also vor, daß bei einer Anfrage ein Anfrageobjekt erzeugt wird, über das die entsprechenden Suchoperationen aufgerufen werden können.

Weiterhin benötigen wir für jede Objektklasse eine Operation, die den Inhalt eines Objektes dieser Klasse, das sind im wesentlichen die Attributwerte, ausgibt. Der Aufruf einer solchen Operation ist eindeutig an ein bestimmtes Objekt gerichtet, schon weil es sich bei der Ausgabe um die Preisgabe des internen Zustands des Objekts handelt. Deshalb ist es sinnvoll und sogar notwendig, die Ausgabeoperationen jeweils bei den Objektklassen zu implementieren.

Wie wir bei der Betrachtung der Benutzerprofile gesehen haben, sollten wir mindestens zwei mögliche Formen der Ausgabe der gesuchten Daten anbieten:

- eine detaillierte, alle Informationen umfassende Ausgabe und
- eine von Details abstrahierende, Informationen zusammenfassende Ausgabe.

Um dies zu realisieren, gibt es auf Anhieb die folgenden Möglichkeiten:

1. Man implementiert zwei (oder mehr) Ausgabeoperationen je Klasse, die sich darin unterscheiden, in welcher Form und mit welchem Abstraktionsgrad die Informationen dargestellt werden.
2. Man bietet eine Ausgabeoperation an, der man als Parameter den gewünschten Abstraktionsgrad übergibt.

Wir werden der Einfachheit halber die Möglichkeit 2 wählen. Indem wir einen Parameter `Abstraktionsgrad` vorsehen, können wir im Datenflußdiagramm einer Ausgabeoperation in Abschnitt 7.4.3 deutlich machen, daß es mehrere unterschiedliche Ausgabevarianten geben wird. Würden wir einfach mehrere unterschiedliche Ausgabeoperationen pro Klasse vorsehen (Möglichkeit 1), dann müßten wir diesen Sachverhalt umgangssprachlich formuliert hinzufügen. Ansonsten sind beide Varianten natürlich völlig gleichwertig.

7.4.2 Suchoperationen

In diesem Abschnitt werden wir nun die Suchoperationen für die einzelnen Objektklassen spezifizieren. Wir bedienen uns dazu im wesentlichen der OMT-Notation für funktionale Modelle (Abschnitt 3.2.5). Dies ist ausreichend, da auf dieser Ebene OPAQUE-spezifische Aspekte noch nicht zum Tragen kommen.

Wie oben bereits erwähnt, sollen *alle* Suchoperationen von der Klasse `Anfrage` angeboten werden (zentrale Lösung, siehe Abb. 7.4). Man könnte dies schon als Übergang auf die Klassenebene bezeichnen. Andererseits stehen die Anfrageobjekte abseits der eigentlichen Kernstruktur unserer Diskurswelt. Sie sind es, die diese Struktur kennen und sie dem Benutzer von außen erschließen. Daher richtet sich an dieser Stelle der Blick von den Anfrageobjekten auf die gesamte Objektstruktur der Diskurswelt. Deshalb ist es gerechtfertigt, von einer Betrachtung auf globaler Ebene zu sprechen.

Wir wollen nun in den folgenden Abbildungen die graphische Spezifikation der Suchoperationen darstellen. Die Herleitungen der einzelnen Operationen sind sich sehr ähnlich. Deshalb verzichten wir darauf, jede Abbildung einzeln zu erläutern; wir werden dies anhand der Operation `suche_Gebietsbereiche` tun, um dann nur noch auf bestimmte Besonderheiten anderer Suchoperationen einzugehen.

Die Herleitung von `suche_Gebietsbereiche` wurde bereits im Beispiel auf Seite 108 angedeutet: die Objekte der Klasse `Gebietsbereich` haben einen Wert für das Attribut `Name` und stehen über Kontexte zu Objekten der Klassen `Institution`, `Benutzer` und Objekten der eigenen Klasse in Beziehungen (vgl. Abb. 7.2). Dies sind die Kriterien, über die sich Gebietsbereiche identifizieren lassen. Als zusätzliches Suchkriterium wollen wir ein *Stichwort* zulassen. Dies soll einem Benutzer, der sich in der Strukturierung der Daten nicht auskennt und deshalb keine präzisen Anfragen stellen kann, helfen, die von ihm gesuchten Informationen doch zu finden. Man stelle sich etwa vor, daß ein Objekt dann zur Ergebnismenge hinzugenommen werden soll, wenn einer seiner Attributwerte das genannte Stichwort enthält.

Betrachten wir nun Abb. 7.5: sie modelliert die Suchoperation `suche_Gebietsbereiche` als einen Prozeß, in den mehrere Datenflüsse einmünden und aus dem ein Datenfluß `Menge` entspringt. Letztgenannter Datenfluß endet in einem Datenbehälter `Gebietsbereichsmenge`. Diese Konstruktion stellt die Ausgabe der Operation in Form einer Ergebnismenge dar, die natürlicherweise aus Gebietsbereichsobjekten bestehen soll.

Unter den vielen Datenflüssen, die in `suche_Gebietsbereiche` einfließen, existieren zwei Gruppen:

- die Eingabeparameter der Operation (gestrichelter Datenfluß) und
- Nachrichten, die von Objektklassen an den Prozeß gesandt werden (durchgehend gezeichnete Datenflüsse). Diese Nachrichten werden sich später als Operationen entpuppen, die in einem oder mehreren Kontexten der Ausgangsklasse (des Datenflusses) vorhanden sind.

Da sich die Eingabeparameter und die beteiligten Objektklassen (deren Objekte die Datenquellen darstellen), wie wir oben gesehen haben, weitgehend mechanisch aus dem Objektmodell ableiten lassen, gilt unser Interesse den Nachrichten bzw. den Datenflüssen, die in den Objekten der Klassen `Gebietsbereich`, `Institution` und `Benutzer` entspringen: für jeden Eingabeparameter muß mindestens eine Nachricht existieren, die den jeweiligen Parameter überprüft und für die Suche auswertet. Handelt es sich bei dem Eingabeparameter um einen Attributwert wie beispielsweise bei `Name` in Abb. 7.5, so wird dieser Wert meist durch eine Operation mit dem Präfix `hat_` verarbeitet, in Abb. 7.5 also zum Beispiel durch `hat_Name`. Diese Zuordnung von Eingabeparametern und Nachrichten ist in den folgenden Abbildungen nicht ausdrücklich ersichtlich, sondern nur intuitiv durch die Benennung der Datenflüsse. Wir werden in Abschnitt 7.5 auf diese Zuordnung näher eingehen.

In Abb. 7.5 wollen wir zunächst einmal nur darstellen, daß wir offenbar Nachrichten benötigen, die uns besagen, ob ein Gebietsbereichsobjekt einen bestimmten Namen hat (`hat_Name`), ob es sich zu einem Stichwort zugehörig zählt (`passt_zu_Stichwort`), welches seine Oberbereiche bzw. Unterbereiche sind (`alle_Oberbereiche` bzw. `alle_Unterbereiche`), und wir wollen von Institutions- und Benutzerobjekten wissen, mit welchen Gebietsbereichen sie verbunden sind (`verbundene_Gebietsbereiche`). Dies sind die zentralen Konstruktionen, die sich durch all die funktionalen Modellierungen der Suchoperationen in den Abbildungen D.1 bis D.19 (siehe Anhang D) durchziehen.

Wir wollen nun noch auf einige Besonderheiten für bestimmte Suchoperationen hinweisen:

- Das OMT-Objektmodell modelliert Richtlinienregelungen und Richtlinienwerte als Aggregate (Abb. 5.4 und 5.5). Da wir die Suchoperation aus der Objektstruktur abgeleitet haben, ist es interessant, an dieser Stelle die Frage zu stellen, in welcher Form sich eine Aggregierung auf die Suchoperation niederschlägt. Dies ist schnell beantwortet, indem wir die Übertragung der Aggregierung nach OPAQUE bzw. OPAQUE-EM betrachten (Abb. 5.10 und 5.11). Daran sehen wir, daß eine Aggregierung durch einen Kontext modelliert wird, der Rollen für die an der Aggregierung beteiligten Klassen anbietet. Deshalb

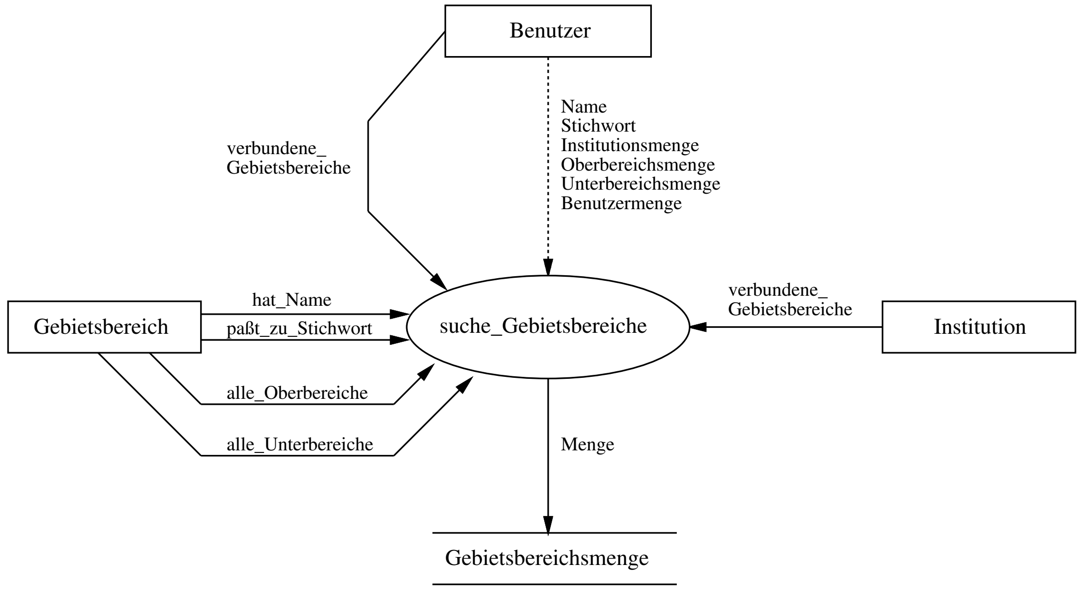


Abbildung 7.5: Operation `suche_Gebietsbereiche`

unterscheidet sich ein solcher Kontext nicht von einem, der eine Beziehung beispielsweise zwischen zwei Klassen herstellt. Wenn wir deshalb die Suchfunktionen der an den Aggregaten beteiligten Objektklassen ansehen (z.B. `suche_Richtlinienregelungen` in Abb. D.5), so unterscheiden sie sich dadurch, daß sie von Objekten einer größeren Anzahl von Klassen Nachrichten darüber zugesandt bekommen, welche Objekte jeweils verbunden sind.

- Aufgrund der strukturbedingten Herleitung der Suchoperationen ist es weiterhin interessant zu beobachten, welchen Niederschlag die Vererbungsbeziehungen zwischen `Richtlinie`, `EG-Richtlinie` und `Nicht-EG-Richtlinie`[4] bzw. zwischen `Wert`, `Exaktwert` und `Intervallwert`[5] auf die Suchoperationen finden.

  Dabei müssen wir die beiden genannten Vererbungskonstruktionen genauer betrachten, da sie sich unterscheiden:

  – Die Richtlinien-Vererbung ist notwendig, da sich die Klassen `EG-Richtlinie` und `Nicht-EG-Richtlinie` sowohl in ihren (zusätzlichen) Attributen, als auch in ihren Beziehungen unterscheiden. Deshalb sind die entsprechenden Suchoperationen als Erweiterungen der Suchoperation `suche_Richtlinien` (Abb. 7.6) dargestellt (siehe Abb. 7.7 und 7.8).

  – Die Wert-Vererbung kommt allein deshalb zustande, weil ein `Intervallwert` mit anderen Attributen beschrieben werden muß als ein `Exaktwert`. Die Unterschiede sind minimal, so daß man auf zwei unterschiedliche Suchoperationen verzichten kann und eine einzige für `Wert` anbietet (Abb. 7.9).

[4] Abb. 5.3, Seite 82 im OMT-Objektmodell bzw. Abb. 5.9, Seite 94 im OPAQUE-EM-Objektmodell
[5] Abb. 5.5, Seite 88 im OMT-Objektmodell bzw. Abb. 5.11, Seite 98 im OPAQUE-EM-Objektmodell

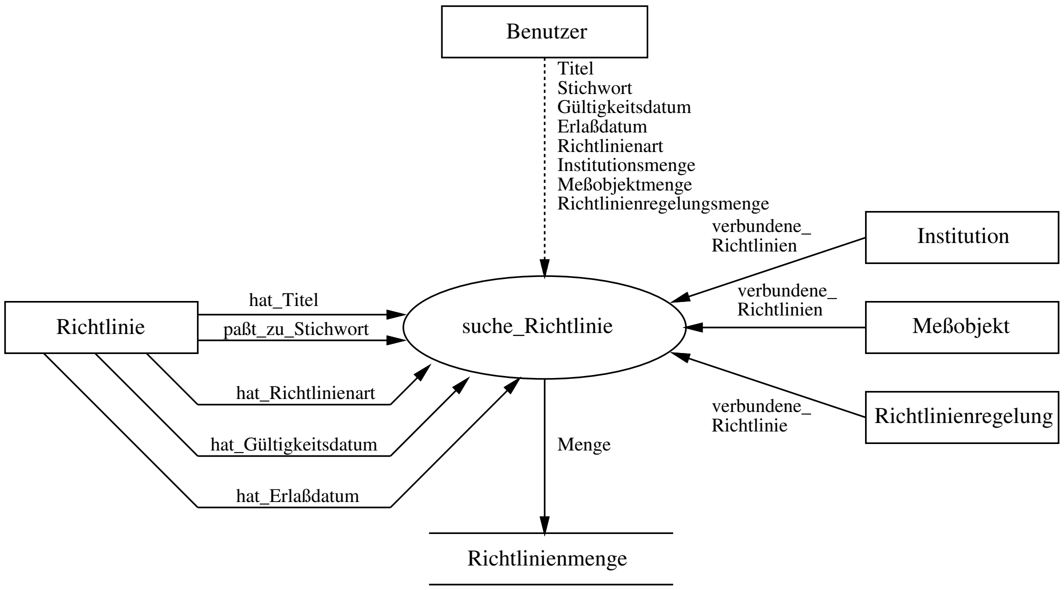


Abbildung 7.6: Operation `suche_Richtlinien`

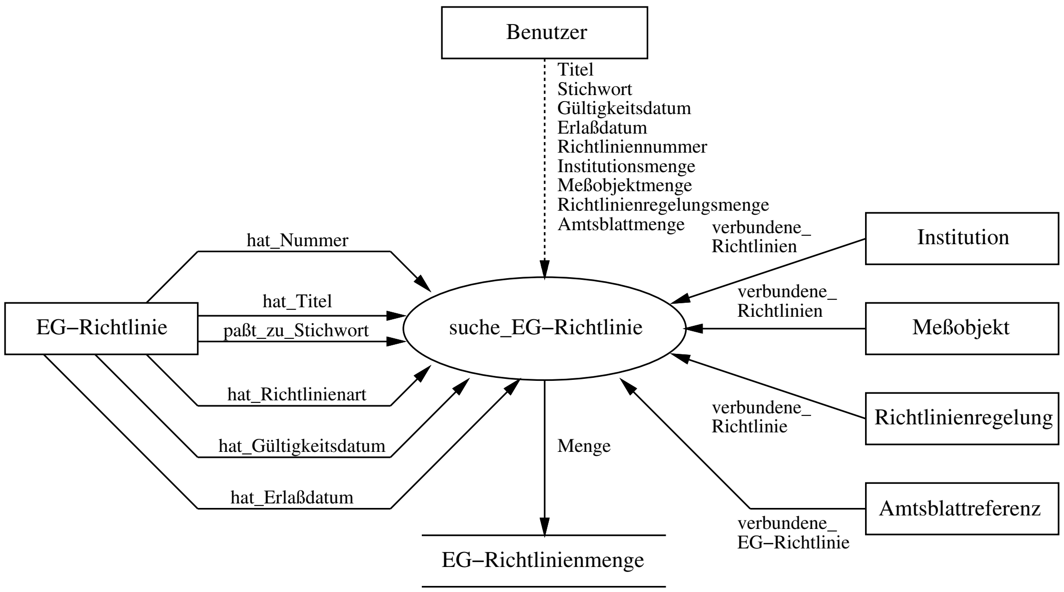


Abbildung 7.7: Operation `suche_EG-Richtlinien`

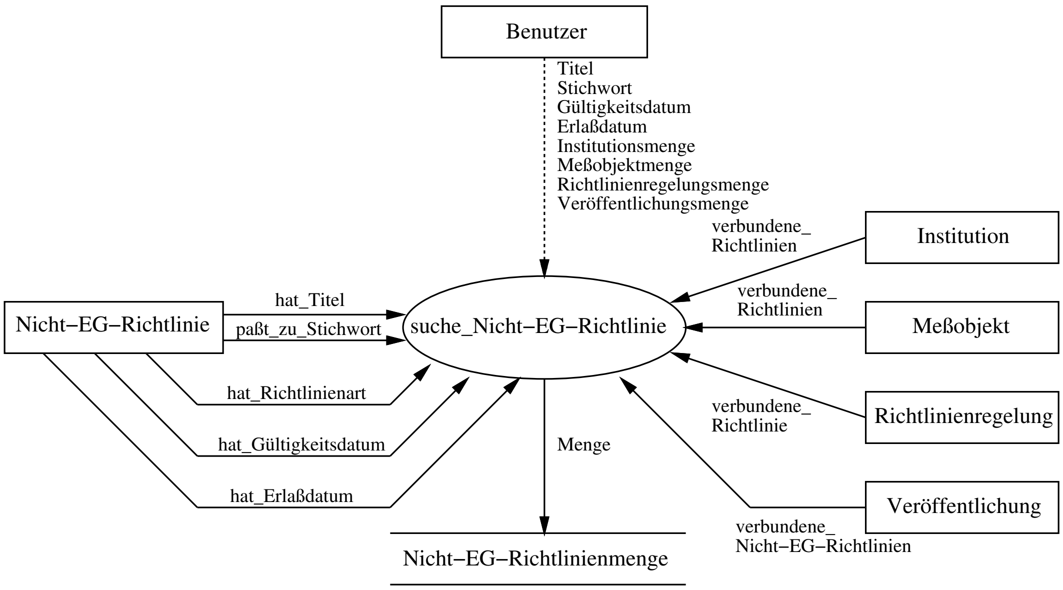


Abbildung 7.8: Operation `suche_Nicht-EG-Richtlinien`

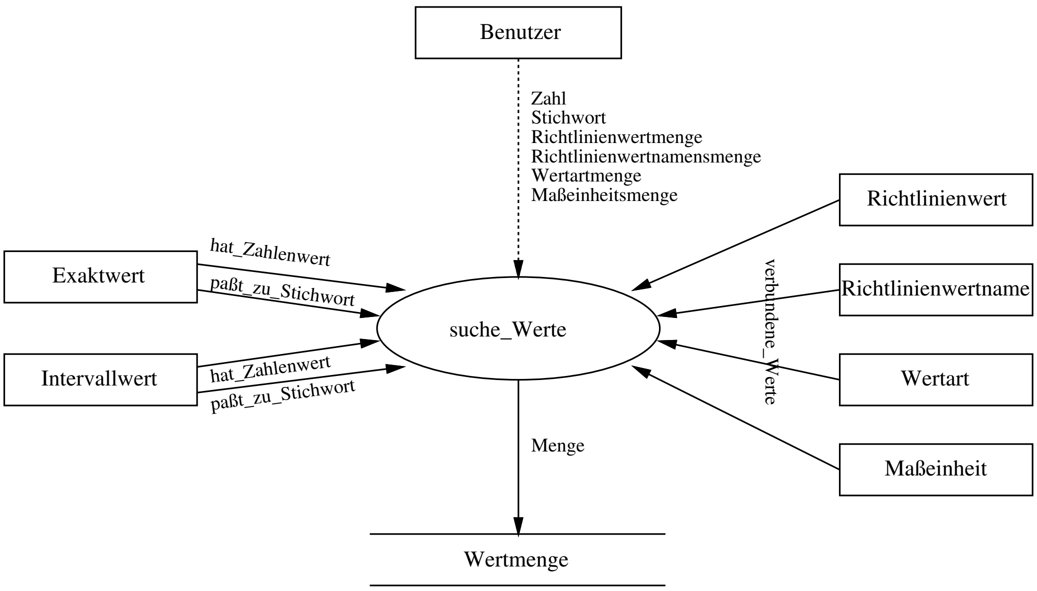


Abbildung 7.9: Operation `suche_Werte`

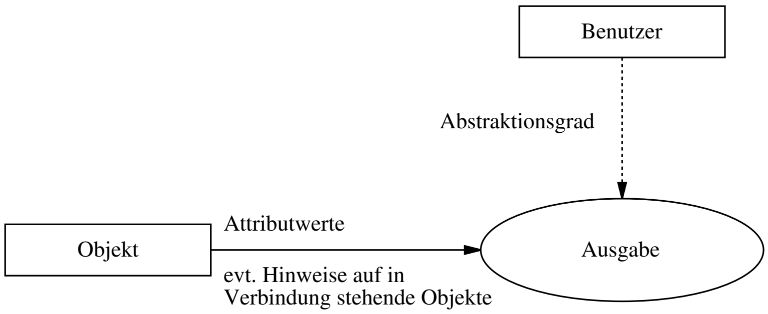


Abbildung 7.10: Operation `Ausgabe` allgemein für eine beliebige Objektklasse

7.4.3 Ausgabeoperationen

Wie wir oben bereits erwähnt haben, soll jede Objektklasse eine Ausgabeoperation `Ausgabe` erhalten, der als Parameter ein `Abstraktionsgrad` übergeben wird (Abb. 7.10). Abhängig davon, in welcher Weise das Objekt ausgegeben werden soll, wird die Ausgabeoperation Informationen vom betreffenden Objekt anfordern: das sind einerseits Attributwerte und andererseits Informationen über in Beziehung stehende Objekte.

Wir wollen diese Vorgehensweise anhand der Ausgabeoperation für die Objekte der Klassen `Institution` und `Richtlinienregelung` näher erläutern:

Beispiel

Wir nehmen einmal an, daß zwei Abstraktionsgrade für die Ausgabe vorgesehen seien: Grad 1 sei der Grad für Experten, Grad 2 der für mit der Diskurswelt weniger vertraute Benutzer. Dies *kann* nun für die Ausgabe eines Institutionsobjekts bedeuten, daß bei Grad 1 die Werte der Attribute `Kurzname` und `Langname` des Institutionsobjekts ausgegeben werden, während beim Grad 2 zusätzlich noch der Gebietsbereich ausgegeben wird, der mit dem Institutionsobjekt verbunden ist.

Beispiel

Wenn wir wieder von zwei Abstraktionsgraden ausgehen, so *kann* dies für die Ausgabe einer Richtlinienregelung bedeuten, daß bei Grad 1 alle mit dem Aggregat verbundenen Objekte ausgegeben werden und daß bei Grad 2 auf die Ausgabe der Meßmethode und möglicherweise auch der Bemerkung[6] verzichtet wird.

Es sei an dieser Stelle ausdrücklich darauf hingewiesen, daß es sich bei den Beispielen um Vorschläge handelt, wie der in diesem dynamischen Entwurf angebotene Mechanismus einer Ausgabe in Abhängigkeit von einem Abstraktionsgrad ausgenutzt werden kann. Auf welche Attributwerte verzichtet werden kann, welche verbundenen Objekte automatisch mit ausgegeben werden sollen, auf welche Teile eines Aggregats ein weniger informierter Benutzer verzichten

[6] Dies kann in Abhängigkeit vom Wert des Attributs `Anspruchsgrad` eines Bemerkungsobjekts geschehen.

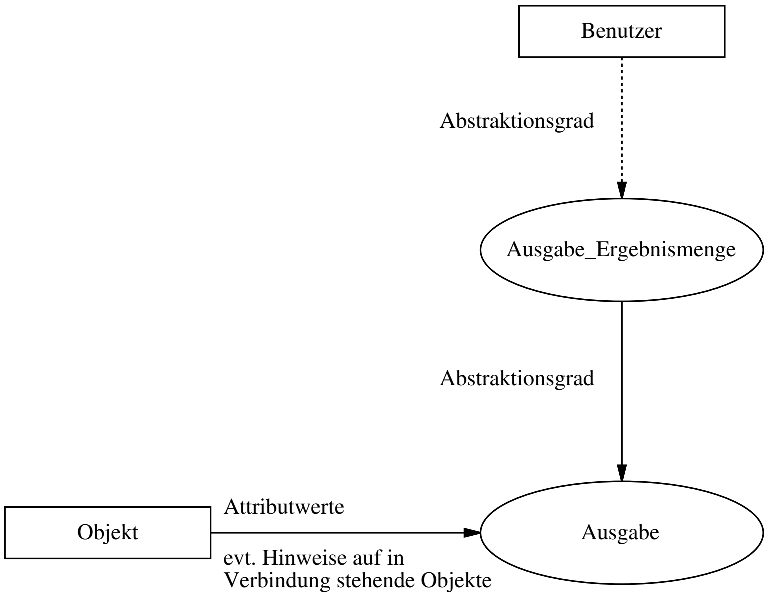


Abbildung 7.11: Operation `Ausgabe_Ergebnismenge`

kann, wieviele unterschiedliche Abstraktionsgrade sinnvoll sind etc., bedarf einer sehr genauen Benutzeranalyse, die über das hinausgeht, was wir im Rahmen dieser Arbeit vermitteln möchten. Der angebotene Mechanismus kann nach Belieben genutzt werden, in welcher Form das aber konkret geschieht, soll ganz bewußt nur angedeutet bleiben.

Wir wollen nun noch eine weitere nützliche Ausgabeoperation für die Klasse `Anfrage` anbieten, die ja bereits alle Suchoperationen enthält: jede dieser Operationen liefert als Ergebnis eine (möglicherweise leere) Menge von Objekten zurück. Deshalb ist es sinnvoll, für diese Ergebnismenge eine Ausgabeoperation in `Anfrage` zur Verfügung zu stellen. Wir nennen diese Operation `Ausgabe_Ergebnismenge`. Sie verwendet die oben angegebenen Ausgabeoperationen der einzelnen Objektklassen. Dieser Zusammenhang ist in Abb. 7.11 dargestellt.

7.5 Klassen- und Kontextebene

Im vorangegangenen Abschnitt haben wir eine globale, die gesamte Modellierung betrachtende Sichtweise eingenommen, wie sie etwa ein Benutzer oder Anwendungsprogrammierer haben könnte. Auf diese Art und Weise haben wir Operationen entworfen, die dazu dienen, Datenobjekte in der Datenbasis zu identifizieren und sie in einer Ergebnismenge zu sammeln. In diesem Zusammenhang ergab sich die Notwendigkeit von Operationen, die bei den Objektklassen implementiert sein sollten und die das Wiederauffinden ermöglichen.

Wir wollen in diesem Abschnitt nun von der globalen Ebene auf eine implementierungsnähere Ebene, die *Klassenebene*, übergehen. Das bedeutet, daß wir nun nicht das Gesamtmodell betrachten werden, sondern einzelne Klassen. Dazu werden wir in Abschnitt 7.5.1 die auftretenden Arten von Operationen beschreiben. In Abschnitt 7.5.2 erfolgt die Zuordnung der Operationen zu den Klassen. Weiterhin werden pro Operation die Eingabeparameter, der Ausgabetyp und die beteiligten Kontexte angegeben.

7.5.1 Beschreibung der Operationen

Bei der Entwicklung der funktionalen Modelle der Suchoperationen (Abb. D.1 bis D.19) haben wir gesehen, daß die einzelnen Klassen Nachrichten an die Suchoperationen senden. Es fällt auf, daß sich dabei bestimmte Konstruktionen wiederholen, die wir an dieser Stelle erläutern werden:

hat_X: Es gibt Operationen `hat_X`, wobei `X` der Name eines Attributs der betreffenden Klasse ist. Sie sollen angeben, ob ein Objekt der jeweiligen Klasse im Attribut `X` den Wert hat, der der Operation als Eingabeparameter übergeben wurde.

In der einfachsten Form einer solchen Operation kann man sich vorstellen, daß sie als Ergebnis einen Booleschen Wert zurückliefert. In einer alternativen Variante könnte man beispielsweise eine Ergebnismenge als zusätzlichen Eingabeparameter übergeben. Die Operation `hat_X` fügt dann das Objekt (über das sie aufgerufen wurde) hinzu, falls `X` den entsprechenden Attributwert hat. Als Ausgabe erhält man dann die Ergebnismenge.

paßt_zu_Stichwort: Jede Klasse hat eine Operation `paßt_zu_Stichwort`. Der Aufbau ist ähnlich wie der einer `hat_X` Operation: als Eingabeparameter wird ein Stichwort übergeben, als Ausgabe erhält man einen Booleschen Wert.

Auf welche Art und Weise entschieden werden soll, ob ein Objekt zu einem Stichwort paßt, kann sehr unterschiedlich sein: in einer sehr einfachen Form kann man sagen, daß es paßt, falls einer seiner Attributwerte das Stichwort enthält. In einer weiteren Stufe könnte und im Falle der Aggregierungen sollte man noch die in Beziehung stehenden Objekte in die Entscheidung miteinbeziehen: beispielsweise könnte eine Richtlinienregelung zu einem Stichwort passen, falls der mit ihr verbundene Parameter oder die verbundene Emissionsquelle zu dem Stichwort paßt. Weiterhin sollten die Ähnlichkeits- oder Synonym-Beziehungen eines Objekts zu anderen Objekten seiner Klasse einbezogen werden.

Als eine fortgeschrittene Variante ist denkbar, daß anstelle eines Booleschen ein numerischer Wert zurückgeliefert wird, der angibt, wie gut ein Objekt zu einem Stichwort paßt. Man könnte dann in Abhängigkeit vom Wissensstand des jeweiligen Benutzers entscheiden, ob das Objekt noch Relevanz hat oder nicht.

verbundene_X: Es gibt Operationen `verbundene_X`, wobei `X` der Name einer Klasse ist, deren Objekte in Beziehung stehen können (mit den Objekte der Klasse, die `verbundene_X` implementiert hat). Als Ergebnis erhalten wir entsprechend der für die Beziehung geltenden Kardinalitäten eines oder mehrere Objekte, die zum Ausgangsobjekt in Beziehung stehen. Als Ausgabe erhalten wir also entweder ein einzelnes Objekt oder eine Menge von Objekten.

Im Falle, daß eine Ergebnismenge berechnet werden muß, wird der zum jeweiligen Objektkristall gehörende private Kontext eine leere Menge initialisieren, die in der Ergebnispipeline abgelegt wird. Danach fügt jeder Kontext, der die durch `verbundene_X` angesprochene Beziehung modelliert und der mit dem Ausgangsobjekt verbunden ist, das verbundene Objekt aus der Klasse `X` zu der Menge hinzu, die in der Ergebnispipeline steht. Das Ergebnis wird also in einem Zusammenspiel zwischen dem privaten Objektkontext und den Beziehungskontexten berechnet.

'Mit-sich-selbst-Beziehungen' und transitive Hüllen: Im Objektmodell existieren einige Objektklassen, die mit sich selbst in in einer binären Beziehung stehen, wie beispielsweise `Parameter` oder `Richtlinienregelung`. Oft benötigt man alle zu einem Objekt in Beziehung stehende Objekte, ähnlich wie bei den `verbundene_X` Operationen.

Im Falle, daß es sich um transitive Relationen handelt, ist es meist notwendig, zusätzlich die transitive Hülle der von einem Objekt (direkt) ausgehenden Beziehungen zu bilden. Deutlich wird das z.B. für die `ist_synonym` Beziehung zwischen zwei Parametern: hier wollen wir nicht nur ein Synonym zu einem gegebenen Parameter, sondern alle durch die Transitivität der Relation implizierten Synonyme verwenden.

Wir haben an dieser Stelle die Implementierung der Operationen bewußt nur angedeutet, da diese in Abschnitt 7.6 anhand von Beispielen dargestellt wird.

7.5.2 Zuordnung der Operationen zu den Klassen

In diesem Abschnitt werden wir die Operationen den einzelnen Objektklassen zuordnen. Dies ist prinzipiell schon durch die funktionalen Modelle der Suchoperationen (Abb. D.1 bis D.19) geschehen: Abb. 7.12 zeigt einen Ausschnitt aus Abb. 7.5. Sie zeigt Datenflüsse, die von Objekten der Klasse `Gebietsbereich` ausgehen. Wohin diese Datenflüsse münden, ist an dieser Stelle unerheblich; ihr Ursprung in `Gebietsbereich` zeigt uns jedoch, daß Objekte dieser Klasse Nachrichten bzw. Operationen mit diesen Bezeichnungen zur Verfügung stellen müssen.

Die Tabellen D.1 bis D.21 (im Anhang D) enthalten diese Zuordnung der Operationen zu den einzelnen Objektklassen. Wir ergänzen dies um die in Abschnitt 7.5.1 erwähnten Eingabeparameter, Ausgabetyp und beteiligte Kontexte.

7.6 Implementierungsebene

Wir werden nun anhand von Beispielen die Implementierung der in den bisherigen Abschnitten genannten Operationen darstellen. Aus diesen Beispielimplementierungen läßt sich leicht der Code der übrigen Operationen ableiten.

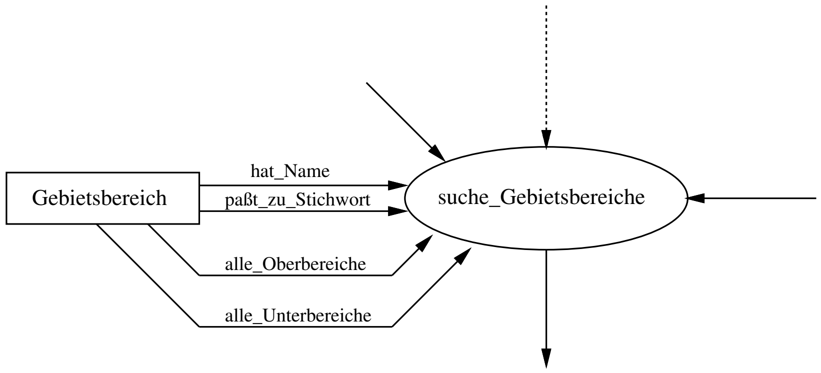


Abbildung 7.12: Datenflüsse, die in einem Gebietsbereich entspringen

7.6.1 Die Suchoperation `suche_Gebietsbereiche`

Wir werden hier beispielhaft die Implementierung der Suchoperation `suche_Gebietsbereiche` in Abb. 7.13 erläutern, deren Herleitung wir im Beispiel in Abschnitt 7.4.1 angedeutet haben und deren Spezifikation in Abb. 7.5 zu sehen ist. Es handelt sich dabei um die Implementierung der Version, die die Suchparameter mit einem logischen UND verknüpft. Das bedeutet, daß nur Suchobjekte in die Ergebnismenge übernommen werden, die *alle* Suchkriterien[7] erfüllen.

Die Implementierung gliedert sich wie folgt:

- Kopf der Operation:
  `Set suche_Gebietsbereiche ...`
- Variablendeklaration:
  `var {...}`
- Deklaration der Rollen des Kontextes `Anfrage_Ktxt`, über die die Operation aufgerufen werden darf:
  `roles {...}`
- Code der Operation:
  `code {...}`

  Der Code gliedert sich inhaltlich in vier Teile:

  - Initialisierung einer Menge, in der das Ergebnis im Verlaufe der Operation gesammelt wird,
  - der Behandlung der Eingabeparameter, die sich auf Attributwerte beziehen,
  - der Behandlung der Eingabeparameter, die sich auf Verbindungen zu anderen Objekten beziehen,
  - der Übergabe des Ergebnisses an die Ergebnispipeline.

[7] und nicht nur ein einziges! Vergleiche dazu Ausführungen auf Seite 109.

In der Kontextschablone `Anfrage_Ktxt`:

```
Set suche_Gebietsbereiche ( String  gebietsname,
                            String  stichwort,
                            Set     institutionsmenge,
                            Set     oberbereichsmenge,
                            Set     unterbereichsmenge,
                            Set     benutzermenge
                          )

   var   { Set             erg;
           Gebietsbereich  xg;
           Institution     xi;
           Benutzer        xb;
         }

   roles { anfrage }

   code  { /* Initialisieren einer leeren Ergebnismenge */

           erg := Set.create();


           /* Alle Objekte, die 'gebietsname' als Name haben */
           /* oder zu 'stichwort' passen, werden zur         */
           /* Ergebnismenge hinzugefuegt.                    */

           for all instances xg in Gebietsbereich
              if ( (xg.hat_Name(gebietsname) or gebietsname.empty()) and
                   (xg.passt_zu_Stichwort(stichwort) or stichwort.empty()) )
                 erg.add(xg);


           /* Schneide Ergebnismenge mit Gebietsbereichen, die mit   */
           /* Objekten aus 'institutionsmenge', 'oberbereichsmenge', */
           /* 'unterbereichsmenge' oder 'benutzermenge' in Beziehung */
           /* stehen.                                                */

           if ( not institutionsmenge.empty() )
              for all instances xi in institutionsmenge
                 erg.disjunction(xi.verbundene_Gebietsbereiche());

           if ( not benutzermenge.empty() )
              for all instances xb in benutzermenge
                 erg.disjunction(xb.verbundene_Gebietsbereiche());

           if ( not oberbereichsmenge.empty() )
              for all instances xg in oberbereichsmenge
                 erg.disjunction(xg.alle_Unterbereiche());

           if ( not unterbereichsmenge.empty() )
              for all instances xg in unterbereichsmenge
                 erg.disjunction(xg.alle_Oberbereiche());


           /* Uebergebe die Ergebnismenge an die Ergebnispipeline */

           return (erg);
         }
```

Abbildung 7.13: Code der Suchoperation `suche_Gebietsbereiche`

In der Kontextschablone Institution_Ktxt:

```
void Ausgabe ( Number abstraktionsgrad )

   var   { }

   roles { institution }

   code  { /* Ausgabe erfolgt in Abhaengigkeit des */
           /* Abstraktionsgrades                   */

           if ( Abstraktionsgrad = 1 ) then
              { print(Langname,"(",Kurzname,")"); }
           else
           if ( Abstraktionsgrad = 2 ) then
              { print(Langname,"(",Kurzname,")");
                gebietsbereich.Ausgabe(abstraktionsgrad);
              }

           return ();
         }
```

Abbildung 7.14: Code der Operation Ausgabe für eine Institution

7.6.2 Die Ausgabeoperation für Institution

Als Beispiel für eine Ausgabeoperation wollen wir die Implementierung von Ausgabe für die Klasse Institution erläutern. Sie ist in Abb. 7.14 zu sehen und enthält die zwei unterschiedlichen Ausgabevarianten, die wir im ersten Beispiel in Abschnitt 7.4.3 vorgeschlagen haben:

- Abstraktionsgrad 1 (für informierte Benutzer) gibt die Attributwerte von Kurzname und Langname aus.

- Abstraktionsgrad 2 (für weniger informierte Benutzer) gibt neben den Attributwerten von Kurzname und Langname den mit dem Institutionsobjekt (über wirkt_in_Ktxt) verbundenen Gebietsbereich aus.

7.6.3 Die Operation hat_Name aus Gebietsbereich

Die Operationen, die überprüfen, ob ein Attribut einen bestimmten Wert annimmt, werden in der Kontextschablone implementiert, die auch das Attribut enthält. Ein Beispiel für eine solche Operation ist hat_Name der Klasse Gebietsbereich. Sie wird in der Schablone Gebietsbereich_Ktxt implementiert und vergleicht einfach den Wert des Eingabeparameters gebietsname mit dem Attributwert name. Das Ergebnis wird in die Ergebnispipeline geschrieben (siehe Abb. 7.15).

In der Kontextschablone Gebietsbereich_Ktxt:

```
Boolean hat_Name ( String gebietsname )

   var   { }

   roles { gebietsbereich }

   code  { /* Ergebnis ist der Vergleich des gegebenen Namens */
           /* (gebietsname) mit dem im Kontext vorhandenen    */
           /* Namen (name).                                   */

           return ( name.equal(gebietsname) );
         }
```

Abbildung 7.15: Code der Operation hat_Name

In der Kontextschablone Gebietsbereich_Ktxt:

```
Boolean passt_zu_Stichwort ( String stichwort )

   var   { }

   roles { gebietsbereich }

   code  { /* Objekt soll zu Stichwort passen, falls     */
           /* das Stichwort im Namen des Gebietsbereichs */
           /* enthalten ist.                             */

           return ( name.contains(stichwort) );
         }
```

Abbildung 7.16: Code der Operation passt_zu_Stichwort

7.6.4 Die Operation passt_zu_Stichwort aus Gebietsbereich

Die Operation passt_zu_Stichwort wird in allen Kontextschablonen implementiert, die Attribute enthalten, deren Werte in Form eines Stichwortes gesucht werden können.

Im Falle der Klasse Gebietsbereich ist das sehr einfach, da es nur einen Kontext mit einem solchen Attribut gibt: Gebietsbereich_Ktxt. Hier wird einfach überprüft, ob das Stichwort im Wert des Attributs Name enthalten ist (siehe Abb. 7.16).

7.6.5 Die Operation verbundene_Institutionen der Klasse Gebietsbereich

Die Gruppe der Operationen, die eine Menge von Objekten liefern, die mit einem bestimmten Objekt in Beziehung stehen, und die wir mit dem Präfix verbundene_ versehen haben, bringt einen OPAQUE-spezifischen Aspekt in die Implementierung ein: eine Operation dieser Gruppe wird nicht nur in einem einzigen Kontext implementiert, sondern in mehreren, die über die Ergebnispipeline kooperieren und gemeinsam das Ergebnis berechnen.

Abb. 7.17 zeigt als Beispiel die Operation `verbundene_Institutionen`, die die mit einem Gebietsbereich in Beziehung stehenden Institutionen in einer Ergebnismenge ablegt. Dazu wird man an ein Gebietsbereichsobjekt `g` den Aufruf

```
g.verbundene_Institutionen()
```

schicken. Die Idee der Implementierung aus Abb. 7.17 ist nun die folgende:

1. Der Code von `verbundene_Institutionen` im Kontext `Gebietsbereich_Ktxt` von `g` initialisiert eine leere Ergebnismenge. Da `g` nur einen solchen Kontext besitzt, geschieht dies auch nur ein einziges Mal.

2. Jeder Kontext `wirkt_in_Ktxt`, der mit `g` verbunden ist, fügt das Objekt seiner Rolle `Institution` zur Ergebnismenge hinzu. Dazu muß gewährleistet sein, daß diese bereits in der Ergebnispipeline existiert. Durch die Bedingung

   $<$ `Gebietsbereich_Ktxt @ gebietsbereich` $>$

   stellen wir sicher, daß die Menge existiert und als `result.current` in der Ergebnispipeline liegt. Zu dieser Menge wird dann nur noch das Institutionsobjekt hinzugefügt. Dies macht jeder `wirkt_in_Ktxt`, der mit `g` verbunden ist, so daß wir am Ende das gewünschte Ergebnis in Form der Menge `result.current` vorliegen haben.

7.6.6 Die Operation `alle_Oberbereiche` aus `Gebietsbereich`

Die Idee der Implementierung einer Operation wie beispielsweise `alle_Oberbereiche` aus `Gebietsbereich` ist dieselbe wie die aus Abschnitt 7.6.5: es wird eine Ergebnismenge initialisiert, in die anschließend Objekte eingefügt werden. Der wesentliche Unterschied liegt darin, daß wir über die Kontexte `schliesst_ein_Ktxt` transitive Beziehungen zwischen Gebietsbereichsobjekten darstellen. Die Operation `alle_Oberbereiche` berechnet deshalb nicht nur die direkt mit einer Gebietsbereichsobjekt in Beziehung stehenden Objekte, sondern eine transitive Hülle der von einem Gebietsbereich ausgehenden Beziehungen. Dies geschieht durch den Aufruf

```
temp := oberbereich.alle_Oberbereiche();
```

der zu einer Rekursion führt.

7.7 Bewertung

Wir wollen an dieser Stelle ein kurzes Resümee der Vor- und Nachteile des in diesem Kapitel durchgeführten Entwurfs geben.

Einerseits besticht der Entwurf durch seine geradlinige, teilweise mechanische Herleitung. Dadurch können Änderungen auf einer Entwurfsebene problemlos auf eine andere übertragen bzw. zurückverfolgt werden. Dadurch lassen sich Änderungen sehr einfach einarbeiten.

In der Kontextschablone `Gebietsbereich_Ktxt`:

```
Set verbundene_Institutionen ()

   var   { }

   roles { gebietsbereich }

   code  { /* Initialisieren einer leeren Ergebnismenge */

           return (Set.create());
         }
```

In der Kontextschablone `wirkt_in_Ktxt`:

```
Set verbundene_Institutionen ()

   var   { }

   roles { gebietsbereich }

   code  { /* Initialisierung durch "verbundene_Institutionen" */
           /* im Kontext "Gebietsbereich_Ktxt" abwarten.       */

           <Gebietsbereich_Ktxt@gebietsbereich>


           /* Fuege zur Ergebnismenge die mit dem Kontext */
           /* verbundene Institution hinzu.               */

           return ( (result.current).add(institution) );
         }
```

Abbildung 7.17: Code der Operation `verbundene_Institutionen`

In der Kontextschablone `Gebietsbereich_Ktxt`:

```
Set alle_Oberbereiche ()

   var   { }

   roles { gebietsbereich }

   code  { /* Initialisieren einer (leeren) Ergebnismenge */

           return ( Set.create() );
         }
```

In der Kontextschablone `schließt_ein_Ktxt`:

```
Set alle_Oberbereiche ()

   var   { Set temp; }

   roles { unterbereich }

   code  { /* Initialisierung durch "Gebietsbereich_Ktxt" */
           /* abwarten.                                  */

           <Gebietsbereich_Ktxt@gebietsbereich>


           /* Berechne alle Oberbereiche des Oberbereichs */

           temp := oberbereich.alle_Oberbereiche();


           /* Fuege den Oberbereich hinzu */

           temp.add(oberbereich);


           /* Vereinige die errechnete Menge mit der Ergebnismenge */

           return ( (result.current).union(temp);
         }
```

Abbildung 7.18: Code der Operation `alle_Oberbereiche`

Außerdem ergab sich eine relativ unproblematische, fast schablonenhafte Implementierung der entworfenen Operationen.

Andererseits wirken sich Änderungen an einer dieser 'Schablonen' auf die Implementierung zahlreicher Operationen aus. Dies beeinträchtigt die oben angesprochene Änderungsfreundlichkeit des Systems: falls beispielsweise der Suchalgorithmus geändert werden soll, so wirkt sich das auf die Implementierungen von über 20 Suchoperationen aus. Diese Rückwirkung hängt ursächlich auch mit der hohen Anzahl notwendiger Operationen zusammen: so wird pro Klasse eine Suchoperation benötigt.

Weiterhin wird die statische Struktur einer Klasse doppelt implementiert: einerseits in der Klassendefinition, andererseits in der zur Klasse gehörenden Suchoperation, da diese wissen muß, über welche Attribute die Klasse verfügt und welche Beziehungen die Objekte einer Klasse eingehen können. Dies widerspricht einem Ziel der Softwaretechnik, da aus änderungstechnischen Gründen eine solche, mehrfache Implementierung ein und desselben Aspekts nicht wünschenswert ist.

Insgesamt ist der Entwurf sehr systemnah und prozedural. Das heißt, daß der Benutzer bzw. der Anwendungsprogrammierer sehr eng mit den Eigenschaften und Strukturen des Systems vertraut sein muß, da er den Suchprozeß selbst koordinieren muß oder kann. Für einen mit dem System weniger vertrauten Anwender ist dies ein Nachteil; für den eingearbeiten Benutzer kann das ein Vorteil sein.

Im folgenden Kapitel werden wir insbesondere die genannten Nachteile aufgreifen, um sie als Anstoß für einen zweiten dynamischen Entwurf verwenden, der ebensolche Nachteile vermeiden soll.

Kapitel 8

Dynamischer Entwurf: ein zweiter, deskriptiver Ansatz

In diesem Kapitel stellen wir einen alternativen dynamischen Entwurf vor, der auf dem zweiten, in Kapitel 6 vorgeschlagenen Ansatz beruht. Weiterhin fließen in ihn die Erfahrungen ein, die wir bei der Entwicklung des ersten dynamischen Entwurfs gewonnen und in Abschnitt 7.7 zusammengefaßt haben.

8.1 Zielrichtung und Grundideen des Entwurfs

Der erste Ansatz für einen dynamischen Entwurf war geprägt von seiner Geradlinigkeit und Einfachheit, in der er sich aus Betrachtungen des Objektmodells herleiten ließ. Wir sind diesem Weg zunächst einmal nachgegangen, um Erfahrungen über die dynamischen Möglichkeiten zu gewinnen, die sich in der gegebenen Objektmodellierung ergeben. Diese Erfahrungen wollen wir nun analysieren und gezielt einsetzen, um zu einer mit Hilfe von Zielvorgaben gestalteten Vorgehensweise zu gelangen:

- Das Anfragemodell des ersten Entwurfs gründet auf einem prozeduralen Ansatz: die Anfrage besteht in ihrem wesentlichen Punkt daraus, dem Datenbanksystem eine Kombination von Suchoperationen mitzuteilen, mit deren Hilfe die gewünschte Lösungsmenge aus der Datenbasis gewonnen werden kann. Die Gestaltung des Suchprozesses wird also vom Benutzer vorgegeben; das Datenbanksystem hat keinerlei Möglichkeiten, gestalterisch, zum Beispiel optimierend, in die Suche einzugreifen. Andererseits obliegt dem Benutzer die gesamte Verantwortung, dem Datenbanksystem einen geeigneten Weg zu beschreiben, wie es zur Lösung gelangen kann. Für einen erfahrenen Benutzer wird dies unproblematisch sein. Er verfügt in der Regel über die notwendigen Kenntnisse, den Suchprozeß in optimaler Weise zu gestalten. Der unerfahrene Benutzer könnte dieser Aufgabe jedoch hilflos gegenüberstehen.

  Wir wollen deshalb in diesem zweiten dynamischen Entwurf einen deskriptiven Ansatz für das Anfragemodell wählen. Das bedeutet, daß der Benutzer nur noch beschreiben soll, welche Eigenschaften die von ihm gesuchte Lösung hat, und nicht mehr, wie das System diese Lösung in der Datenbasis findet. Das ist einerseits eine Erleichterung für den

Benutzer, andererseits eröffnet dieser Ansatz dem Datenbanksystem die Möglichkeit, den Suchprozeß selbst zu gestalten und optimierend einzugreifen. Es wird also Verantwortung weg vom Benutzer hin zum Datenbanksystem verschoben.

Die Forderung, deskriptive Anfragemodelle für Datenbanksysteme anzubieten, ist nicht neu. In den 70er Jahren wurden zahlreiche Konzepte für deskriptive Anfragesprachen entwickelt. Bedeutende Vertreter davon sind SQL [ANS86], QBE [Zlo77] und Quel [SWKH76]. Wir können deshalb einige in diesen Ansätzen verfolgte Ideen aufgreifen und den Bedürfnissen OPAQUEs anpassen.

Dabei scheinen die Ideen und Konzepte, die QBE zugrunde liegen, besonders geeignet zu sein: ähnlich wie dort dem Benutzer eine Tupelstruktur vorgegeben wird, die dieser mit Suchparametern ausfüllt, kann das System dem Benutzer eine Objektstruktur vorgeben, die zu der Zielklasse seiner Anfrage gehört. Dadurch kann der Benutzer durch die Struktur der Datenbasis geleitet werden. Die vorgegebene Struktur wird von ihm dann mit Attributwerten aufgefüllt, so daß ein Muster der Objekte entsteht, die das Ergebnis der Anfrage darstellen. Dabei bleibt verborgen, wie das System mit Hilfe des Musters in der Datenbasis sucht.

- Ein Problem des Suchmodells des ersten Ansatzes war die Anzahl der notwendigen Suchoperationen, der damit verbundene Implementierungsaufwand und die darin enthaltene Redundanz (siehe Diskussion in Abschnitt 7.7). Wir erheben daraus die Forderung, daß es möglichst nur noch eine Suchoperation geben soll.
  Ursache für die vielen Suchoperationen innerhalb des ersten Ansatzes war, daß für die Suche Informationen über die Objektstruktur (Welches sind die Attribute des Objekts? Zu welchen Objekten können Beziehungen bestehen?) benötigt werden. In den Suchoperationen des ersten Ansatzes waren diese Informationen fest implementiert und in genau diesem Aspekt unterschieden sie sich untereinander. Daher muß es das Ziel sein, die Strukturinformation vom Suchalgorithmus zu trennen, sie aber in diesen einfließen zu lassen. Beispielsweise gäbe es die Möglichkeit, ein Pattern Matching Verfahren zu verwenden, das zunächst auf Strukturgleichheit prüft, um danach einen Vergleich der Attributwerte durchzuführen. Wie wir in Abschnitt 8.4 sehen werden, können dabei die Möglichkeiten von OPAQUE besonders gut ausgenutzt werden.

8.2 Vorgehensweise

Aufgrund der Ziele und Forderungen aus Abschnitt 8.1 ergeben sich die folgenden Aufgabenstellungen:

- Wie sieht ein deskriptives Anfragemodell für unsere Problemstellung aus?
- Wie wird eine im entworfenen Anfragemodell erstellte Anfrage in eine Suche umgesetzt?

Durch diese beiden Fragen ergibt sich auch gleich eine grobe Gliederung des Entwurfs:

- Abschnitt 8.3 stellt das Anfragemodell dieses dynamischen Entwurfs vor. Es handelt sich dabei um einen Ansatz, der Ideen der Anfragesprache QBE[1] aufnimmt, um den Benutzer

[1] Eine Einführung in QBE findet man beispielsweise in [KS91] oder [Dat91].

durch die Struktur der in der Datenbasis abgebildeten Diskurswelt zu führen.

- Abschnitt 8.4 beschreibt, wie eine deskriptive Anfrage in eine Suche umgesetzt wird.
- Abschnitt 8.5 stellt das Ausgabemodell dieses Entwurfs vor, das sich gegenüber dem des ersten dynamischen Entwurfs nicht ändert.

Wir verzichten darauf, innerhalb dieses Kapitels eine Bewertung des durchgeführten Entwurfs vorzunehmen. Diese erfolgt im Rahmen einer Gesamtbetrachtung des Entwurfs in Kapitel 9.

8.3 Das Anfragemodell

8.3.1 Konzeption des Anfragemodells

Wir stellen in diesem Abschnitt das Anfragemodell für diesen zweiten dynamischen Entwurf vor. Dem Benutzer soll die Möglichkeit gegeben werden, auf deskriptive Art und Weise eine Anfrage zu stellen. Das bedeutet, daß der Benutzer beschreibt, welche Eigenschaften die von ihm gesuchten Daten haben sollen; er beschreibt also, *was* er sucht, ohne zu sagen, *wie* gesucht werden soll.

Im Bereich relationaler Datenbanksysteme gibt es mehrere Ansätze, derartigen Anforderungen nachzukommen. Einer davon ist die Anfragesprache QBE. Der Benutzer wählt dort zunächst den Typ der von ihm gesuchten Daten aus. Über die Angabe des Datentyps ist das System in der Lage, dem Benutzer die Datenstruktur vorzugeben. Der Benutzer kann dann ein 'Anfrageskelett' mit dieser Struktur so weit mit Attributwerten ausfüllen, wie es ihm möglich ist. Im Falle von QBE, das auf dem relationalen Modell basiert, haben diese Anfrageskelette die Form von Tabellen, da sie sich in idealer Weise als grafisches Darstellungsmittel für Relationen eignen.

Als Ergebnis einer derartigen Anfrageschnittstelle erhalten wir einerseits eine Information über die Struktur der gewünschten Daten und andererseits die vom Benutzer vorgegebenen Informationen über die Attributwerte. Wie wir in Abschnitt 8.4 sehen werden, ist gerade auch das explizite Vorhandensein der Strukturinformation sehr wichtig, um eine von der Struktur unabhängige Suchprozedur anbieten zu können.

Wollen wir nun den Ansatz, den wir soeben für QBE beschrieben haben, auf OPAQUE übertragen, so ergibt sich die folgende Struktur des Anfragemodells:

- Der Benutzer muß den Typ der von ihm gesuchten Daten auswählen. Im Falle von OPAQUE ist dies eine Objektklasse, die wir als *Zielklasse* der Anfrage bezeichnen wollen.
- Da dem System die Struktur der Zielklasse bekannt ist, kann es diese dem Benutzer vorgeben. Idealerweise geschieht dies in Analogie zu QBE durch eine grafische Darstellung. Wir können uns dazu beispielsweise der Ausdrucksmittel von OPAQUE-EM (siehe Abschnitt 3.4) bedienen, da sie auf einfache Art und Weise auf OPAQUE-Strukturen abgebildet werden können. Es ist aber auch durchaus denkbar, eine andere Darstellung zu verwenden, wie beispielsweise eine, die an der OMT-Notation orientiert ist. Letztgenannte kann, wie wir in Abschnitt 3.5 gesehen haben, ebenfalls leicht auf OPAQUE-Strukturen übertragen werden kann.

- Der Benutzer füllt die vorgegebene Struktur aus, indem er bestimmte Attributwerte einträgt.

- Da Objekte einer Klasse nicht nur über ihre Attributwerte gesucht werden sollen, sondern auch über Objekte, zu denen sie in Beziehung stehen, müssen bei der Vorgabe einer Klassenstruktur auch mögliche Beziehungen der Objekte dieser Klasse angedeutet werden, damit der Anwender diese Beziehungen in seiner Anfrage mitberücksichtigen kann. Diese Andeutungen weisen ihn darauf hin, daß er seine bisher erstellte Anfragestruktur – wir nennen sie künftig ein *Anfrageskelett* – an diesen Stellen erweitern kann, um die Beziehungen der gesuchten Objekte miteinzubeziehen.

- Als Ergebnis der Anfragekonstruktion erhalten wir ein Skelett, das sich aus Strukturen zusammensetzt, die vom Benutzer teilweise mit Attributwerten oder -eigenschaften aufgefüllt wurden. Dieses Anfrageskelett wird später dem Suchalgorithmus übergeben, der einen Struktur- und Werte-Abgleich durchführt und dadurch nach Objekten in der Datenbasis sucht, die dem Muster entsprechen, das durch das Anfrageskelett vorgegeben wurde. Das bedeutet, daß die in einem Anfrageskelett vorgegebenen Attributwerte und Beziehungen (also die Objekteigenschaften) mit einem logischen UND verknüpft werden.

- Um dem Benutzer auch die Möglichkeit zu geben, alternative Muster, also mit einem logische ODER verknüpfte Objekteigenschaften vorzugeben, wollen wir in Anlehnung an QBE zulassen, daß in einer Anfrage nicht nur ein Anfrageskelett, sondern unter Umständen mehrere formuliert werden.

In den im nächsten Abschnitt beschriebenen Anfragealgorithmus sind die Forderungen und Hinweise eingearbeitet, die wir soeben aufgestellt haben.

8.3.2 Der Anfragealgorithmus

Aus der Konzeption des Anfragemodells heraus, die wir im vorangegangenen Abschnitt durchgeführt haben, ergibt sich der folgende Anfragealgorithmus. Für die grafischen Vorgaben greifen wir auf die OPAQUE-EM-Notation zurück. Wie wir oben bereits erwähnt haben, sind durchaus auch andere Darstellungsformen denkbar.

1. Der Benutzer erzeugt ein Anfrageobjekt.

2. Eine Anfrageschnittstelle wird aufgerufen.

3. Der Benutzer wählt eine Klasse aus, deren Objekte Ziel der Anfrage sind. Es erscheint die zu dieser Klasse gehörige Kristallschablone. Die mit der Kristallschablone verbundenen Kontextschablonen werden angedeutet (siehe Beispiel in Abb. 8.3). Diese Kristallschablone ist Fußpunkt eines Anfrageskeletts, das wir innerhalb dieses Algorithmus' aufbauen. Wir nennen diesen Fußpunkt das *Zielobjekt*. Da das Zielobjekt ebenfalls Einstiegspunkt für die später folgende Suche sein wird, markieren wir es (Schattierung in Abb. 8.3).

4. Der Benutzer wählt eine der Kontextschablonen aus. Es erscheint ein Kontext der gewählten Schablone (siehe Beispiel in Abb. 8.4). Klassen, deren Objekte Rollen in diesem Kontext annehmen können, werden ebenfalls angedeutet.

5. Der Benutzer hat nun folgende Schritte zur Auswahl:

   (a) Er wählt eine der angedeuteten Klassen aus (siehe Beispiel aus Abb. 8.5). Gehe zu Schritt 5.

   (b) Er wählt eine der angedeuteten Kontextschablonen aus (analog zu Schritt 4). Gehe zu Schritt 5.

   (c) Er trägt Eigenschaften oder Werte von Attributen in der aktuell ausgewählten Kontextschablone ein (siehe Beispiel aus Abb. 8.7). Bei einem numerischen Attribut könnte als Eigenschaft beispielsweise '> 100' o.ä., als Attributwert etwa '57.4' eingetragen werden. Weiter mit Schritt 5.

   (d) Der Benutzer baut ein weiteres, zusätzliches Anfrageskelett für *dieselbe* Zielklasse. Bei der später folgenden Suche werden alle vom Benutzer für eine Zielklasse formulierten Anfrageskelette in Einzelsuchen umgesetzt, deren Ergebnisse zum Gesamtergebnis vereinigt werden. Dadurch entsteht eine ODER-Verknüpfung. Der Aufbau eines neuen Anfrageskeletts erfolgt über Schritt 3.

   (e) Der Benutzer beendet die Formulierung seiner Anfrage und startet dadurch den Suchprozeß.

Der Anfragealgorithmus spricht in den Schritten 3, 4, 5a und 5b davon, daß weitere Strukturelemente, also Kontext- oder Kristallschablonen, angedeutet werden sollen, um den Benutzer durch die Struktur der Datenbasis zu führen. Bei der Erstellung des Anfrageskeletts ist dabei bezüglich der Kardinalitäten, die für die jeweiligen Verbindungen der Strukturelemente untereinander gelten, das Folgende zu beachten:

- Die Untergrenzen der Kardinalitäten haben für ein Anfrageskelett keinerlei Bedeutung, da es sich bei einem Anfrageskelett um ein Muster handelt, dem bestimmte Charakteristika beigefügt sind. Fehlt beispielsweise einem Anfrageskelett mit der Zielklasse `Richtlinie` ein Kontext der Schablone `Richtlinie_Ktxt`, so bedeutet dies, daß in der Anfrage keine Einschränkungen bezüglich der Variable und der Rollen dieses Kontextes vorgenommen werden. Daher ist ein Fehlen eines Kontextes der Schablone `Richtlinie_Ktxt` in einem Skelett für die Zielklasse `Richtlinie` trotz einer vorgeschriebenen Kardinalität von $< 1, 1 >$ erlaubt. Abb. 8.1 zeigt eine solche Situation.

- Dagegen sind die durch die Kardinalitäten gegebenen Obergrenzen auch für ein Anfrageskelett gültig. So dürfen in einem Richtlinien-Anfrageskelett niemals zwei Kontexte der Schablone `Richtlinie_Ktxt` auftreten. Abb. 8.2 zeigt ein solches ungültiges Anfrageskelett.

Es ist die Aufgabe der Benutzerschnittstelle, dem Anwender nur solche Vorgaben zu machen, die den beiden genannten Forderungen entsprechen.

8.3.3 Beispiel einer Anfrage

Wir wollen an dieser Stelle den oben beschriebenen Anfragealgorithmus an einem Beispiel demonstrieren. Wir verwendenen dazu dieselbe Anfrage wie in Abschnitt 7.3.3:

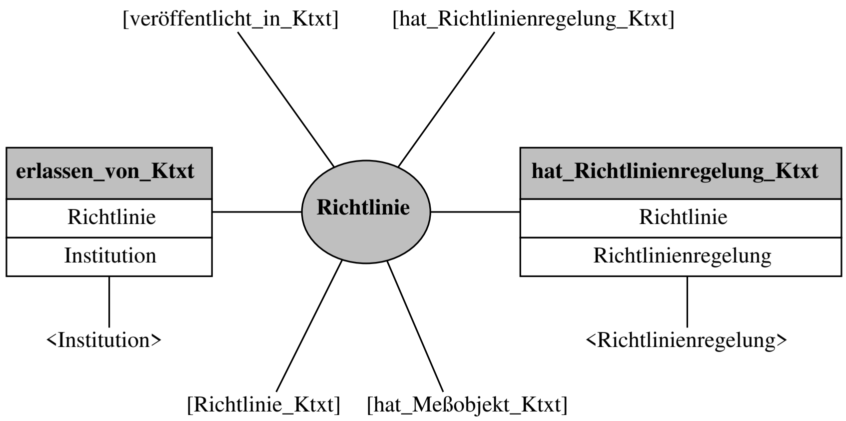


Abbildung 8.1: Ein gültiges Anfrageskelett

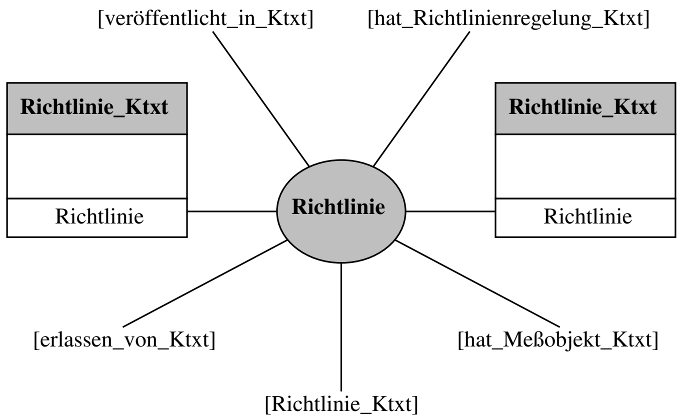


Abbildung 8.2: Ein ungültiges Anfrageskelett

Wir wollen nach allen Veröffentlichungen suchen, in denen Emissionsrichtlinien des Bundesforschungsministeriums (BMFT) oder des Bundesumweltministeriums (BMUR) publiziert wurden.

Dazu sei auf die Abb. 5.8 und 5.9 verwiesen, die die für die Anfrage relevanten Ausschnitte aus dem OPAQUE-EM-Objektmodell enthalten. Die Formulierung der Anfrage erfolgt in den folgenden Schritten; vergleiche dazu auch die entsprechenden Abbildungen:

- Der Benutzer erzeugt ein Anfrageobjekt (Schritt 1).
- Die Anfrageschnittstelle wird aufgerufen (Schritt 2).
- Wir wählen `Veröffentlichung` als Zielklasse aus (Schritt 3). Es erscheint die Struktur der Klasse `Veröffentlichung`, etwa wie in Abb. 8.3.
- Da wir Veröffentlichungen suchen, die Emissionsrichtlinien enthalten, müssen wir dies mit Hilfe von Kontexten der Schablone `veröffentlicht_in_Ktxt` tun (Schritt 4; vgl. Abb. 8.4). Für einen weniger vertrauten Benutzer mag dies an dieser Stelle angesichts einer Darstellung wie in Abb. 8.3 nicht offensichtlich sein. Man kann sich aber vorstellen, daß die Anfrageschnittstelle Hilfsfunktionen zu den abgebildeten Elementen anbietet oder den Benutzer auf andere Art und Weise unterstützt, sich in der Struktur des Objektmodells zurechtzufinden.
- Gemäß Schritt 5 müssen wir nun wählen, an welcher Stelle der in Abb. 8.4 dargestellten Struktur wir weitermachen. Da wir zu den Veröffentlichungen keine näheren Angaben mehr machen können, fahren wir gemäß Schritt 5a fort und wählen in der Darstellung aus Abb. 8.4 die Klasse `Nicht-EG-Richtlinie` aus, was zur Darstellung aus Abb. 8.5 führt.
- Da wir Emissionsrichtlinien suchen, müssen wir `Richtlinie_Ktxt` auswählen, um später den Attributwert 'Emission' für die Richtlinienart eintragen zu können (Schritt 5b; vgl. Abb. 8.6). Daß gerade `Richtlinie_Ktxt` dieses Attribut enthält, mag für den weniger vertrauten Benutzer wiederum nicht offensichtlich sein, was unsere Forderung nach einer Benutzerunterstützung und einer Hilfsfunktion für jeden Kontext bekräftigt.
- In Schritt 5c tragen wir im Attribut `Richtlinienart` aus `Richtlinie_Ktxt` den Wert 'Emission' ein (vgl. Abb. 8.7).
- Nun müssen wir noch sicher stellen, daß es sich um Richtlinien handelt, die vom BMFT bzw. vom BMUR erlassen wurden. Dazu wählen wir in Schritt 5b `erlassen_von_Ktxt` aus (siehe Abb. 8.8).
- In Schritt 5a wählen wir `Institution` (siehe Abb. 8.9).
- In Schritt 5b wählen wir `Institution_Ktxt` (siehe Abb. 8.10).
- Gemäß Schritt 5c tragen wir den Wert 'BMFT' in das Attribut `Kurzname` aus `Institution_Ktxt` ein. Dies führt zum Anfrageskelett aus Abb. 8.11, das die Spezifikation der Suche nach Veröffentlichungen von Emissionsrichtlinien des BMFT darstellt.
- Über Schritt 5d bauen wir analog zu den bisher genannten Schritten ein Anfrageskelett auf, das die Suche nach Veröffentlichungen von Emissionsrichtlinien des BMUR spezifiziert.

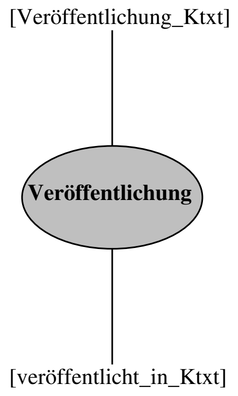


Abbildung 8.3: Aufbau einer Anfrage (1)

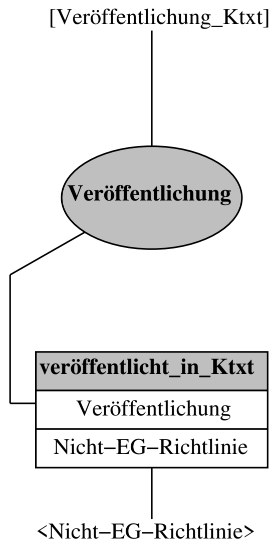


Abbildung 8.4: Aufbau einer Anfrage (2)

Wir verzichten an dieser Stelle auf eine Darstellung der Schrittfolge zur Erstellung dieses zweiten Skeletts, da sie der oben genannten entspricht. Das zweite Anfrageskelett ist in Abb. 8.12 abgebildet.

- Mit Schritt 5e beenden wir die Anfragebeschreibung und starten den Suchprozeß.

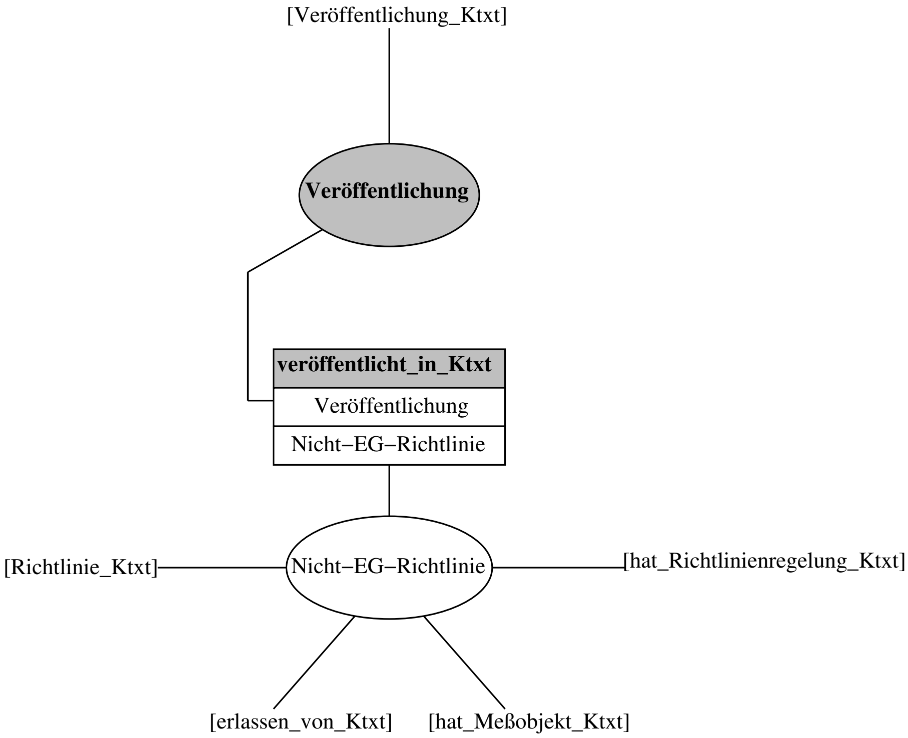


Abbildung 8.5: Aufbau einer Anfrage (3)

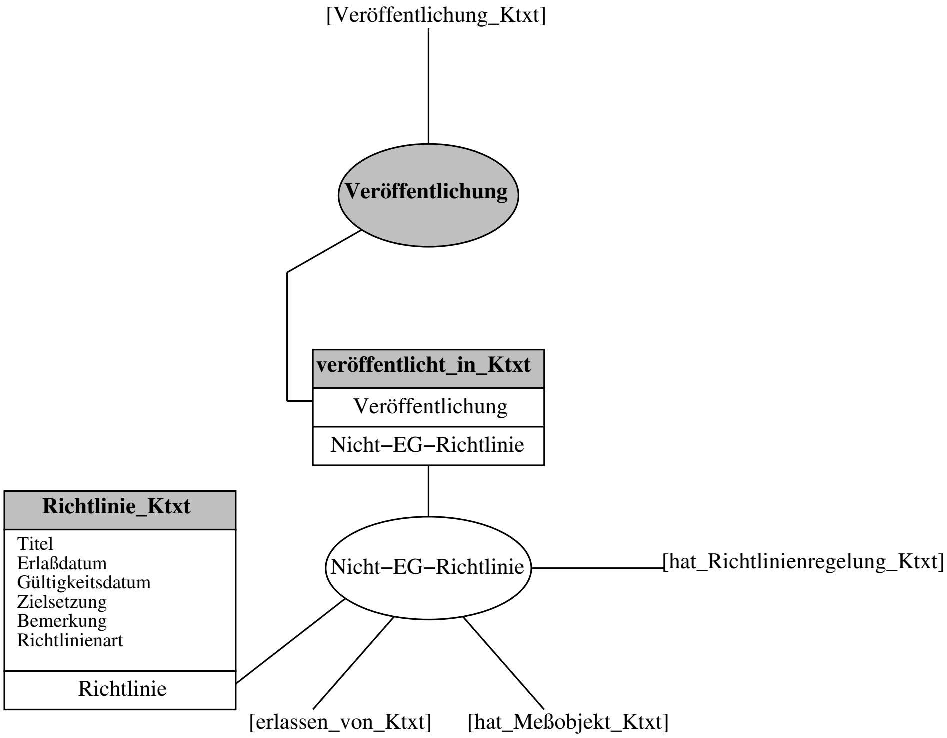


Abbildung 8.6: Aufbau einer Anfrage (4)

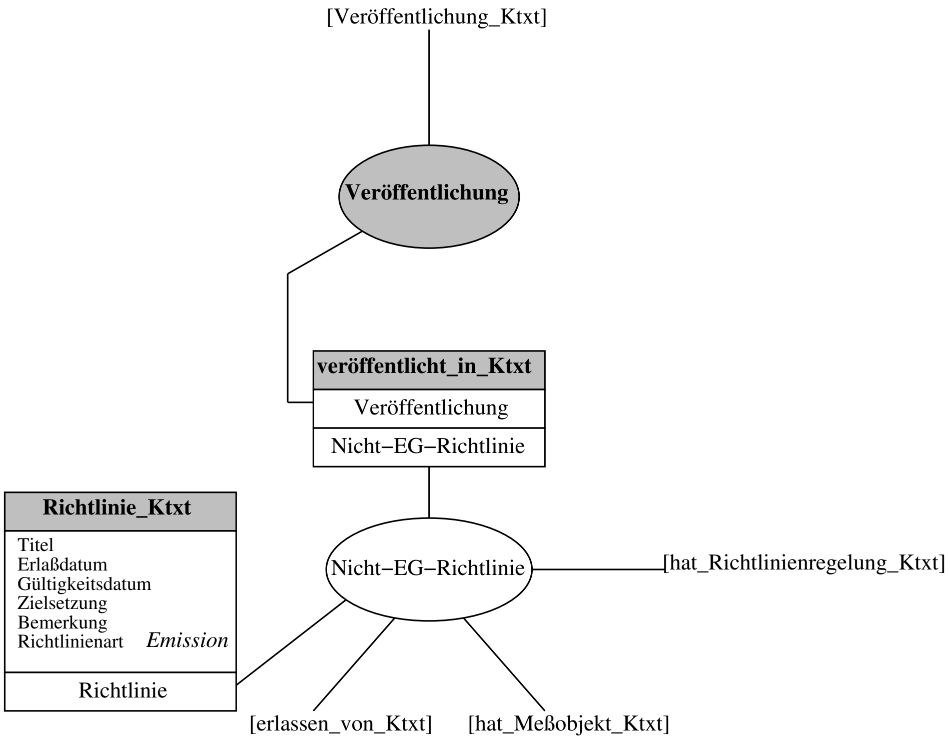


Abbildung 8.7: Aufbau einer Anfrage (5)

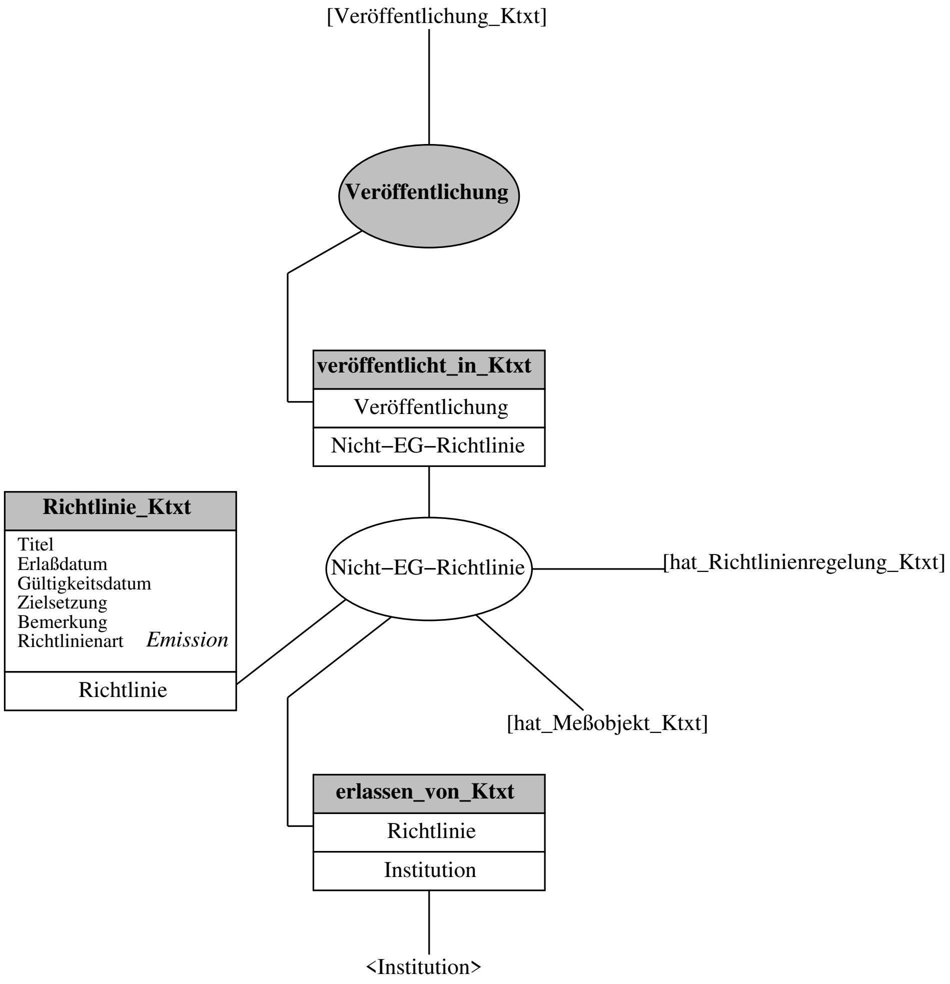


Abbildung 8.8: Aufbau einer Anfrage (6)

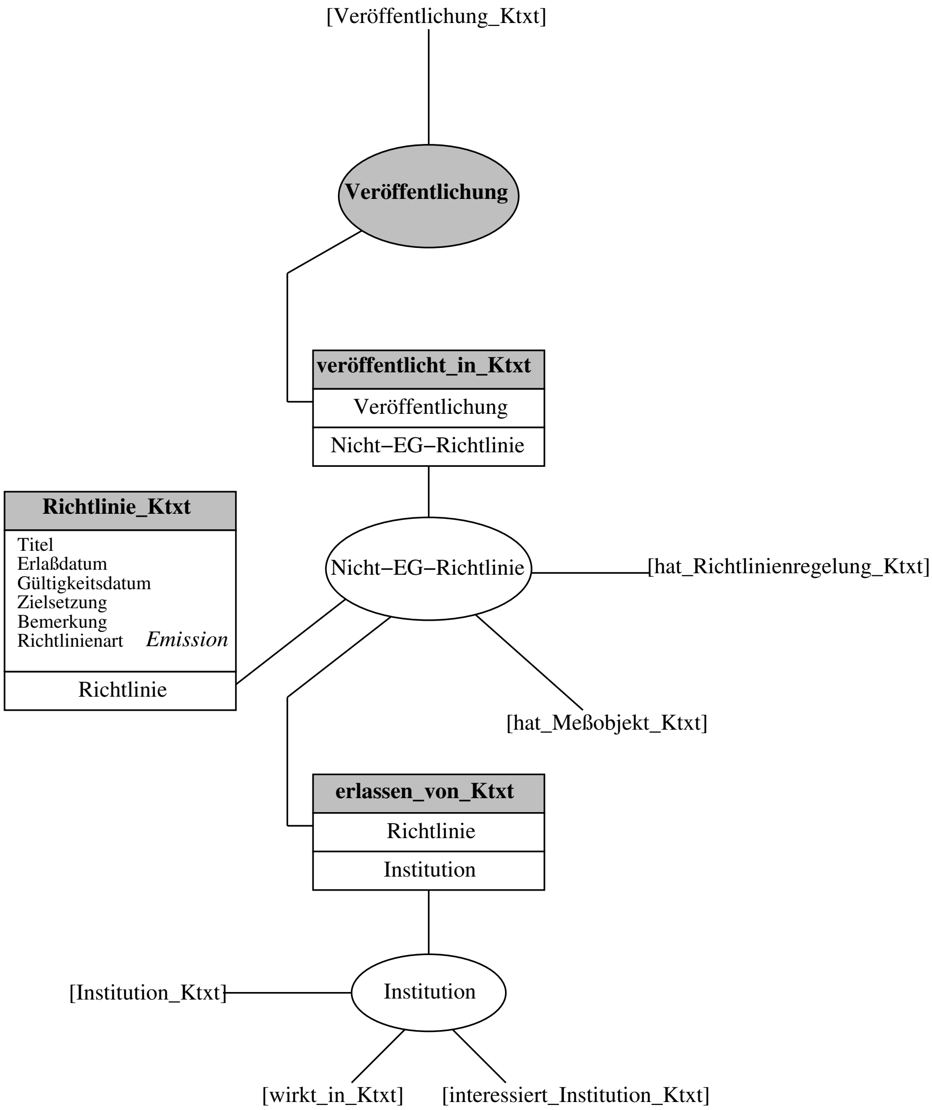


Abbildung 8.9: Aufbau einer Anfrage (7)

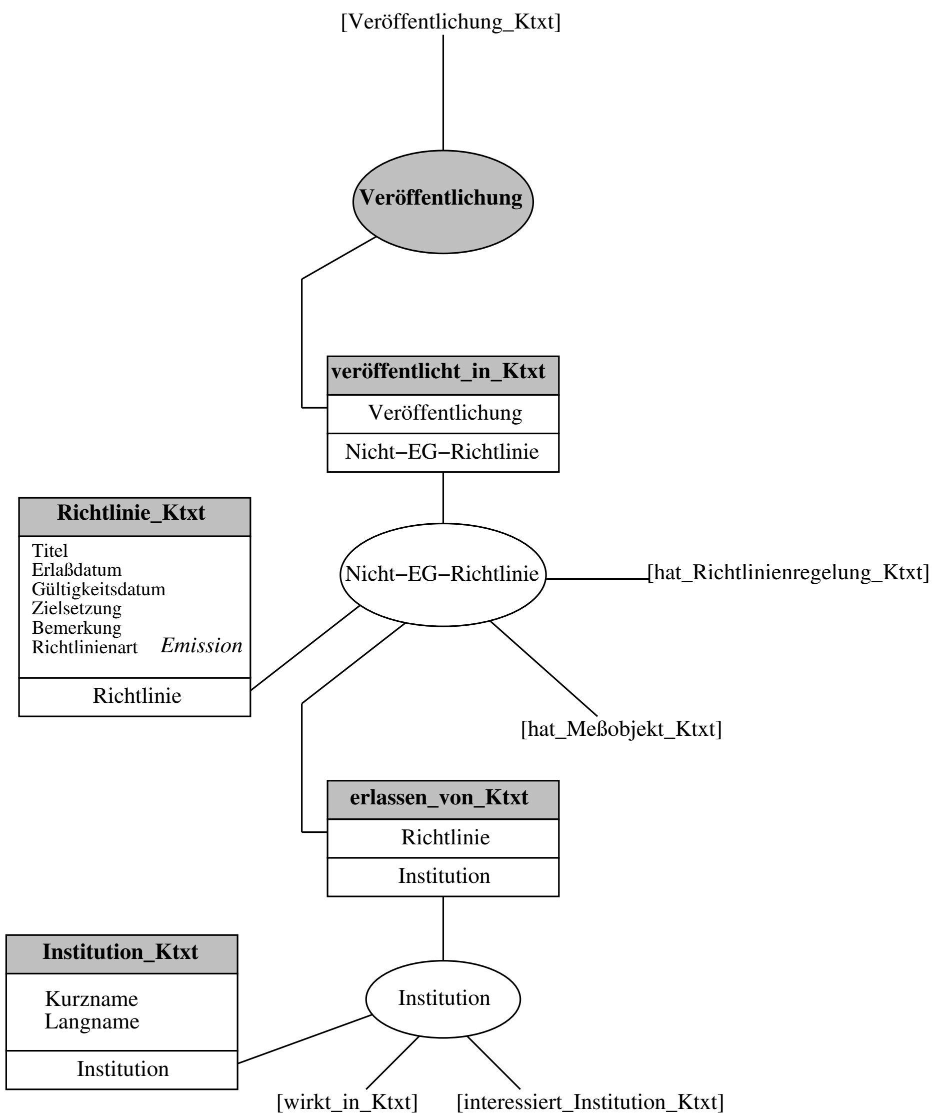


Abbildung 8.10: Aufbau einer Anfrage (8)

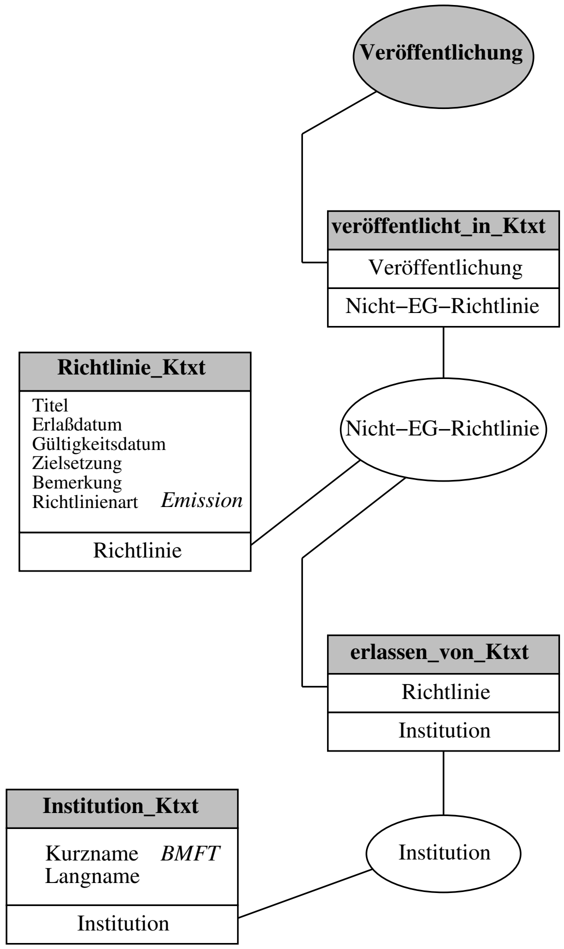


Abbildung 8.11: Erstes Anfrageskelett der Beispielanfrage

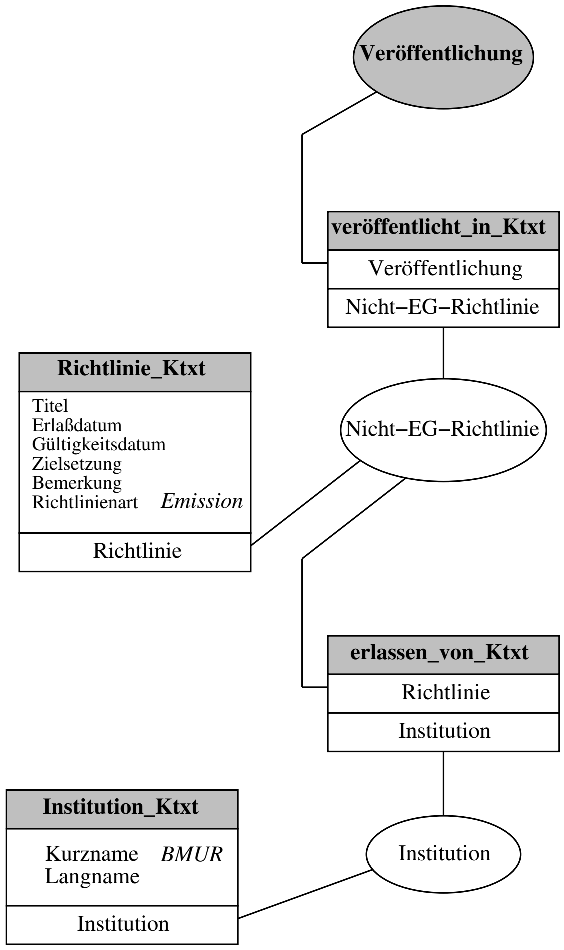


Abbildung 8.12: Zweites Anfrageskelett der Beispielanfrage

8.4 Das Suchmodell

8.4.1 Suchalgorithmus

In diesem Abschnitt wenden wir uns nun der Frage zu, wie wir eine Anfrage, die in Form eines oder mehrerer Anfrageskelette formuliert wurde, in eine Suche in der Datenbasis umsetzen können. Dazu betrachten wir noch einmal das Anfrageskelett aus Abb. 8.11: es zeigt uns ein Muster oder eine Schablone der vom Benutzer angefragten Objekte. Wir können die Anfrage also dadurch beantworten, indem wir in der Datenbasis nach Objekten der Zielklasse suchen, die strukturell und von den angegebenen Attributwerten her zum Muster des Anfrageskeletts passen. Was wir also benötigen, ist ein Pattern Matching Verfahren, das jedes Objekt der Zielklasse, das sich in der Datenbasis befindet, mit den in der Anfrage formulierten Anfrageskeletten vergleicht.

Unter Leistungskriterien betrachtet, erscheint diese Vorgehensweise als zu aufwendig, da es einen Vergleich der Anfrageskelette mit jedem Zielklassenobjekt der Datenbasis erfordert. Andererseits ist es jedoch so, daß die Menge der in einer Zielklasse vorhandenen Objekte in diesem Szenario durchaus überschaubar und daher der Einsatz dieses Verfahrens gerechtfertigt ist.

Zusammenfassend ergibt sich die folgende Struktur eines Suchalgorithmus':

- Eingabe: Eine Menge von Anfrageskeletten. Jedes Objekt der Datenbasis, das zu *einem* der Anfrageskelette paßt[2], soll in die Ergebnismenge aufgenommen werden. Jedes Anfrageskelett hat ein ausgezeichnetes Objekt, das Ziel der Suche ist. Wir nennen es das *Zielobjekt* und seine zugehörige Objektklasse die *Zielklasse* des Anfrageskeletts.
- Es wird eine leere Ergebnismenge initialisiert.
- Für jedes Anfrageskelett `a` der Eingabemenge werden folgende Schritte durchgeführt:
  - Für jedes Objekt `x` der Zielklasse, das sich in der Datenbasis befindet, wird mit dem Anfrageskelett `a` ein Struktur- und Wertevergleich durchgeführt. Dazu schicken wir `x` mit einer Nachricht an `a`, was dort einen solchen Vergleich zwischen Muster `a` und dem in der Datenbasis real existierenden Objekt `x` auslöst:

    `a.match(x)`
  - Objekte `x`, die als Ergebnis dieses Pattern Matching den Anforderungen des Anfrageskeletts `a` genügen, werden in die Ergebnismenge aufgenommen.
- Der Suchalgorithmus liefert als Resultat die Ergebnismenge.

Abb. 8.13 zeigt eine Implementierung dieses Suchalgorithmus' als OPAQUE-Operation in der Kontextschablone `Anfrage_Ktxt`[3].

[2] Dadurch ergibt sich eine ODER-Verknüpfung der Anfrageskelette. In Verbindung mit der UND-Verknüpfung der Struktur- und Attributanforderungen innerhalb eines Anfrageskeletts ergibt sich eine Anfrage, die sich auf eine logische Formel in disjunktiver Normalform zurückführen läßt.

[3] Der Idee, die Suchoperation in der Kontextschablone `Anfrage_Ktxt` zu implementieren, liegen dieselben Überlegungen zugrunde, wie sie bereits im ersten dynamischen Entwurf in Kapitel 7 angestellt wurden: eine Anfrage ist ein Objekt, das vom Benutzer erzeugt wird. Dieses Objekt bietet in Form einer Operation eine Möglichkeit der Anfrageformulierung, aber auch eine Operation zur Lösungssuche, also zur Beantwortung der Anfrage an.

```
Set suche ( Set skelette )

   var   { Set  ergebnis;
           Any  a, x;
         }

   roles { anfrage }

   code  { /* Initialisierung einer leeren Ergebnismenge */

           ergebnis := Set.create();


           /* Alle Anfrageskelette a aus skelette      */
           /* werden mit allen Objekten der jeweiligen */
           /* Zielklasse verglichen.                   */

           for all instances a of skelette
              for all instances x of a.class()
                 if a.match(x)
                    ergebnis.add(x)


           return(ergebnis);
         }
```

Abbildung 8.13: Eine Implementierung des Suchalgorithmus'.

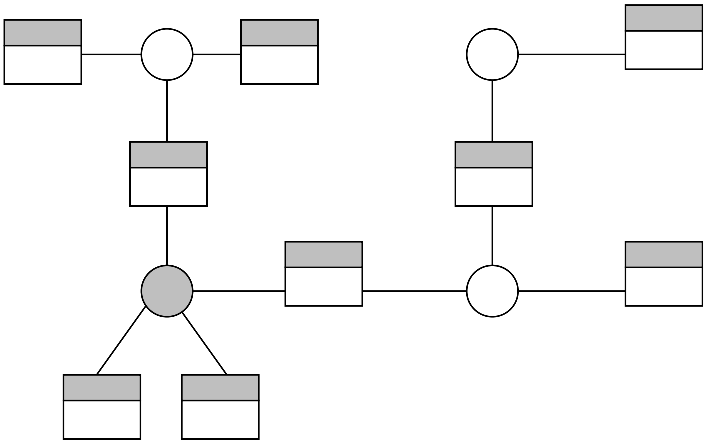


Abbildung 8.14: Objektstruktur eines Anfrageskeletts

Bei näherer Betrachtung der umgangssprachlichen Formulierung des Suchalgorithmus' wie auch seiner Implementierung in Abb. 8.13 zeigt, daß sein wesentlicher Punkt in der Durchführung des Struktur- und Wertevergleichs liegt, der durch die Nachricht `match` ausgelöst wird. Diesem Punkt wollen wir uns im folgenden Abschnitt zuwenden.

8.4.2 Der Struktur- und Wertevergleich

Die Eingabe für den Struktur- und Wertevergleich ist einerseits ein Anfrageskelett, andererseits sind es die Objekte innerhalb der Datenbasis. Abb. 8.14 und 8.15 zeigt diese Ausgangssituation in einer vereinfachten Darstellung.

Wir waren bisher von einem Anfrageskelett ausgegangen, das u.a. aus einem Zielobjekt besteht. Dies ist sinnvoll, da das Anfrageskelett die Formulierung der Anfrage darstellt, wobei das Zielobjekt der Ansatzpunkt der Anfrage ist. Betrachtet man nun ein solches Skelett aus einer implementierungsnäheren Sicht, so ist es gerechtfertigt, Zielobjekt und Anfrageskelett als ein und dasselbe zu betrachten: am Zielobjekt 'hängen' alle notwendigen Informationen der Anfrage – einerseits Attributwerte, andererseits Beziehungen zu anderen Objekten. Das Anfrageskelett ist also ein Musterobjekt der Zielklasse, das mit Attributwerten und Beziehungen zu Musterobjekten anderer Klassen charakterisiert wird. Die Spezifikation des durchzuführenden Struktur- und Wertevergleichs lautet demnach:

- Es existiert ein Zielobjekt, das ein Muster für die Objekte der Lösungsmenge darstellt.
- In der Datenbasis existieren Objekte der Zielklasse, die als Kandidaten für die Lösung in Frage kommen.
- Vergleiche die Objekte der Datenbasis, die Datenobjekte, mit dem Musterobjekt. Entspricht ein Datenobjekt dem Musterobjekt, dann beantworte den Vergleich mit `true`, ansonsten mit `false`.

Wie wir anhand der Abbildungen 8.14 und 8.15 sehen, sind die Objekte bzw. die Kristalle, als Repräsentanten der Objekte in OPAQUE, in eine Umgebung von Kontexten eingebettet. Diese

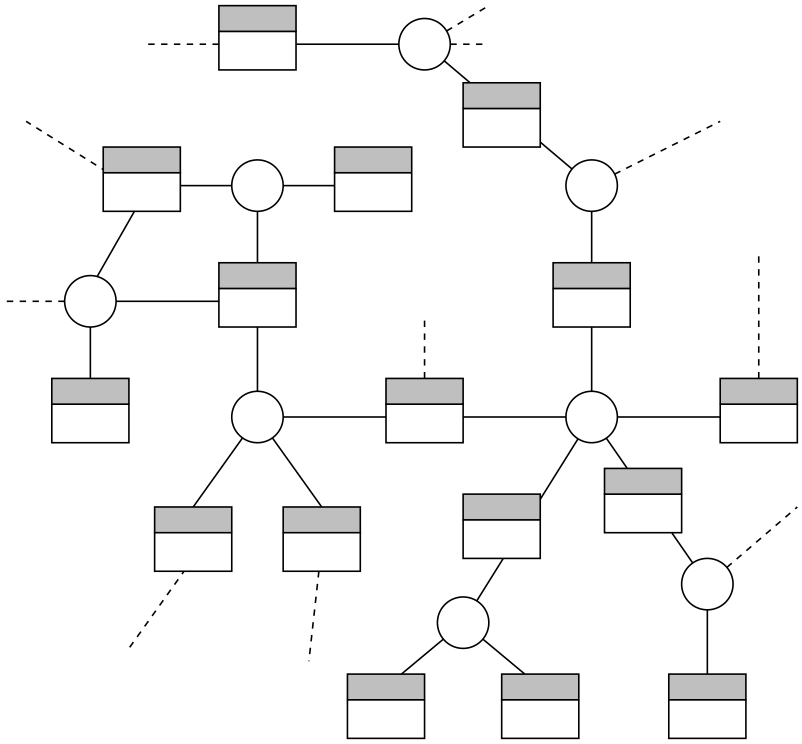


Abbildung 8.15: Objektstruktur einer Datenbasis

Umgebung ist genau der Teil der Objektstruktur, den wir untersuchen müssen, da die Kristalle ja keinerlei Struktur- oder Wertinformation tragen.

Beim Vergleich der Umgebungen zweier Kristalle können wir auf elegante Art und Weise die kontextsensitiven Eigenschaften OPAQUEs ausnützen: es ist nämlich nicht notwendig zu wissen, *wie* diese Umgebung eines Kristalls strukturell aufgebaut ist, d.h. aus welchen Strukturelementen, also Kontexten $\{K_1, K_2, \ldots, K_n\}$ sie besteht, um aufgrund dieses Wissens einen komponentenweisen Vergleich durchzuführen:

- Vergleiche K_1 in Umgebung 1 mit K_1 in Umgebung 2.
- Vergleiche K_2 in Umgebung 1 mit K_2 in Umgebung 2.
- ...
- Vergleiche K_n in Umgebung 1 mit K_n in Umgebung 2.

Dies wäre der Ansatz, den wir in den herkömmlichen Programmier- oder Datenmodellen verfolgen müßten, denen eine strenge Typisierung zugrundeliegt.

Wir können vielmehr auf die Eigendynamik der Umgebungskomponenten, also der Kontexte, zurückgreifen: man schickt einfach eine Nachricht an den Kristall, dessen Umgebung untersucht werden soll. Diese Nachricht wird von jedem an der Umgebung beteiligten Kontext empfangen. Jeder dieser Kontexte wird auf diese Nachricht reagieren entsprechend der Methode, die im Kontext für diese Nachricht implementiert ist[4] (siehe Abb. 8.16).

[4] Die Reaktion kann auch darin bestehen, daß die Nachricht ignoriert wird, da keine zu ihr passende Operation

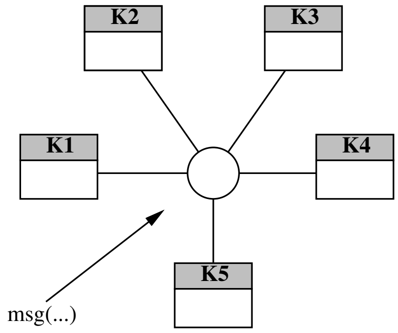


Abbildung 8.16: Eine Nachricht `msg(...)` an einen Kristall mit einer Umgebung von fünf Kontexten

Im Falle eines Struktur- und Wertevergleichs zwischen einem Musterobjekt und einem Datenobjekt kann eine solche, an die einzelnen Kontexte des Musterobjekts gerichtete Nachricht umgangssprachlich formuliert etwa so lauten:

> *Überprüfe, ob das Datenobjekt einen gleichartigen Kontext besitzt. Falls nicht, dann entspricht das Datenobjekt nicht dem Muster; falls doch, dann überprüfe, ob dieser Kontext inhaltlich (d.h. bezogen auf Attributwerte und die Besetzung der Rollenvariablen) gleichwertig ist. Besteht auch hierin eine inhaltliche Übereinstimmung, dann entspricht das Datenobjekt dem Muster, ansonsten eben nicht.*

Auf diese Art und Weise haben wir uns davon befreit, auf die genaue Zusammensetzung einer Umgebung eingehen zu müssen, da sich das Verhalten einer Umgebung eigenständig aus ihrer Zusammensetzung aus Kontexten konfiguriert. Diese Kontextsensitivität birgt entscheidende Vorteile, da wir nun nicht mehr wie im dynamischen Entwurf aus Kapitel 7 auf jede einzelne Umgebung (d.h. in unserem Falle entspräche eine Umgebung einer Objektklasse) eingehen müssen. Dies hatte, wie in jenem Entwurf beschrieben, eine Implementierung von über 20 Suchoperationen notwendig gemacht.

Algorithmus für den Struktur- und Wertevergleich

Wir wollen nun den Algorithmus für den Struktur- und Wertevergleich in einer halb-formalen Darstellung beschreiben. Dazu nehmen wir an, daß in jeder Kontextschablone `K` für jede dort vorhandene Rolle `r`, an der eine Kristallschablone/Objektklasse `O` hängt, eine Operation mit dem Prototyp

```
Boolean match ( O x )
   var   { Boolean  attr_match, role_match; }
```

vorhanden ist.

```
roles { r }
code  { ... }
```

implementiert ist. Wir werden nun den Code beschreiben und dabei zu großen Teilen auf umgangssprachliche Formulierungen zurückgreifen. Dies dient einer besseren Verständlichkeit der Ideen, die dem Algorithmus zugrundeliegen:

1. Sei

   `l := <k1, k2, ..., kn>`

   die Liste der Kontexte der Kontextschablone `K`, an denen `x` in der Rolle `r` hängt.

2. Führe für jedes `k` aus `l` die folgenden beiden Vergleiche durch:

   (a) Vergleiche die eigenen, nichtleeren Attributwerte mit den entsprechenden von `k`. Lege das Ergebnis dieses Vergleichs in der Variablen `attr_match` ab.

   (b) Vergleiche für jede Rolle `q` von `K`, ob die Objekte, die über `q` an `k` hängen, strukturell und von den vorgegebenen Werten den Objekten entsprechen, die über `q` hier an diesem Kontext hängen. Dies bedeutet einen rekursiven Aufruf von `match`, falls solche Objekte vorhanden sind. Lege das Ergebnis dieses Vergleichs in der Variablen `role_match` ab.

3. Die Suche kann abbrechen entweder

   (a) wenn ein `k` gefunden wurde, für das

   `attr_match AND role_match`

   gilt, also sowohl der Attribut- als auch der Rollenvergleich erfolgreich waren, oder

   (b) wenn alle `k` aus `l` überprüft wurden, ohne daß ein wie unter Fall 3a beschriebenes `k` gefunden wurde.

4. Erfolgte der Abbruch der Suche aufgrund von Fall 3a, dann ist das Ergebnis des Struktur- und Wertevergleichs `true`; ansonsten (Fall 3b) `false`.

Rekursion

Wie wir gesehen haben, beinhaltet der Algorithmus in Schritt 2b eine Rekursion. Wir wollen uns deshalb die Vorgehensweise des Algorithmus' anhand der schematischen Darstellung eines Anfrageskeletts aus Abb. 8.14 verdeutlichen:

- Ausgangspunkt ist das Anfrageskelett aus Abb. 8.14. Es entält vier Kristallschablonen, die jeweils von einer Reihe von Kontexten umgeben sind.

- In Phase 1 des Algorithmus erhalten die in Abb. 8.17 schwarz gekennzeichneten Kontexte die Nachricht, einen Struktur- und Wertevergleich mit einem Objekt derselben Klasse aus der Datenbasis durchzuführen. Dafür ist es aber auch notwendig, die über Kontexte angebundenen Objekte abzugleichen.

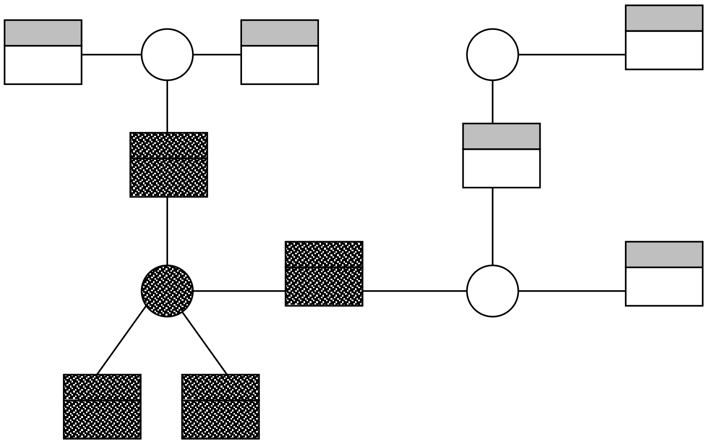


Abbildung 8.17: Phase 1 des Struktur- Wertevergleichs

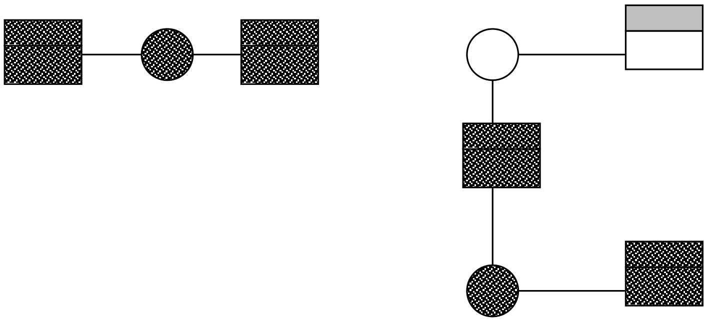


Abbildung 8.18: Phase 2 des Struktur- Wertevergleichs

- In Phase 2 stoßen wir also auf zwei Probleme, die dem Ausgangsproblem gleichgeartet sind und die allerdings eine im Vergleich zu diesem reduzierte Komplexität besitzen. Wir verwenden dasselbe Verfahren und schicken den in Abb. 8.18 schwarz hervorgehobenen Kristallen die Nachricht, jeweils einen Abgleich mit ihren Pendants auf Seiten des Ausgangsdatenobjekts durchzuführen.
- Phase 3 in Abb. 8.19 zeigt den letzten Schritt des rekursiven Abstiegs für unser Beispiel.

Wie wir an dem Beispiel leicht sehen, wird die Rekursion durch die Graphenstruktur des Anfrageskeletts bestimmt. Um den Abbruch der Rekursion zu gewährleisten, ergeht an sie die Forderung, daß sie keine Zyklen enthalten dürfen. Dies kann auf einfache Weise durch die Benutzerschnittstelle garantiert werden.

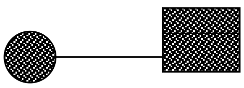


Abbildung 8.19: Phase 3 des Struktur- Wertevergleichs

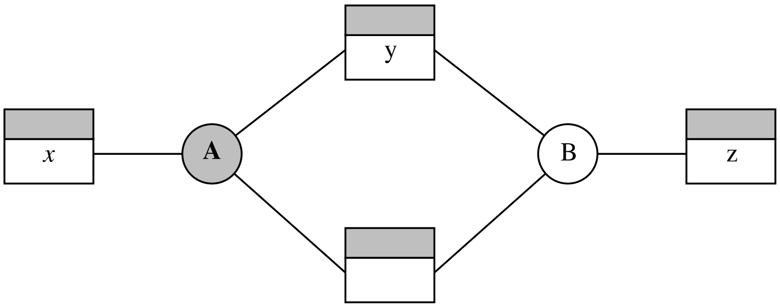


Abbildung 8.20: Ein Anfrageskelett mit Zyklus

Diese Forderung bedeutet keine wesentliche Einschränkung für das Anfragemodell:

- Einerseits können zyklisch formulierte Anfrageskelette in nicht-zyklische transformiert werden. Man betrachte dazu das Beispiel aus Abb. 8.20 und die mögliche Auflösung des Zyklus' gemäß Abb. 8.21. Die Idee ist die, daß man einen Knoten aus dem Zyklus herausbricht, diesen Knoten und die mit ihm verbundenen Teile dupliziert und jede der beiden entstandenen Kopien jeweils an ein offenes Ende des aufgebrochenen Zyklus' anhängt, wodurch der Zyklus beseitigt ist.

  Es ist nun so, daß diese Transformation kein äquivalentes Anfrageskelett erzeugt. So ist es möglich, daß das Anfrageskelett aus Abb. 8.21 mehr Lösungen zuläßt als das aus Abb. 8.20:

  Man stelle sich vor, daß in der Datenbasis ein Objekt a des Typs A existiere, das einen wie in der Abbildung charakterisierten Kontext mit dem Attributwert x habe. Gibt es nun zwei verschiedene Objekte b_1 und b_2 des Typs B, die beide im entsprechenden Kontext den Wert z halten und die über die entsprechenden Kontexte mit a in Beziehung stehen, so würde a dem Muster des Anfrageskeletts aus Abb. 8.21 entsprechen, nicht jedoch dem aus Abb. 8.20. Umgekehrt wird aber jedes Objekt, das dem Muster des Anfrageskeletts aus Abb. 8.20 entspricht, auch dem aus Abb. 8.21 entsprechen.

  Das bedeutet, daß das durch die Transformation gewonnene Anfrageskelett möglicherweise eine größere Lösungsmenge liefert als das ursprüngliche, daß diese Lösungsmenge aber auf jeden Fall die durch das ursprüngliche Anfrageskelett charakterisierten Lösungen enthält.

- Andererseits kann davon ausgegangen werden, daß derartige zyklische Anfrageskelette tatsächlich äußerst selten auftreten. Man betrachte dazu nur einmal das OPAQUE-EM-Objektmodell unserer Diskurswelt und versuche daraus ein Anfrageskelett zu entwickeln, das einen Zyklus enthält. Dies ist zwar möglich, man wird aber leicht sehen, daß für diese seltenen Fälle der Ausweg durch oben beschriebene Umformulierung der Anfrage ausreicht und keine wesentliche Einschränkung darstellt.

Die Forderung nach zyklenfreien Anfrageskeletten ist nicht prinzipieller Natur, sondern führt allein zu einer wesentlich einfacheren Implementierung des Algorithmus' zum Struktur- und Wertevergleich. Mit Hilfe des entsprechenden Aufwands ist es durchaus möglich, den Struktur- und Wertevergleich so zu implementieren, daß Anfrageskelette, die Zyklen enthalten, verarbeitet werden können.

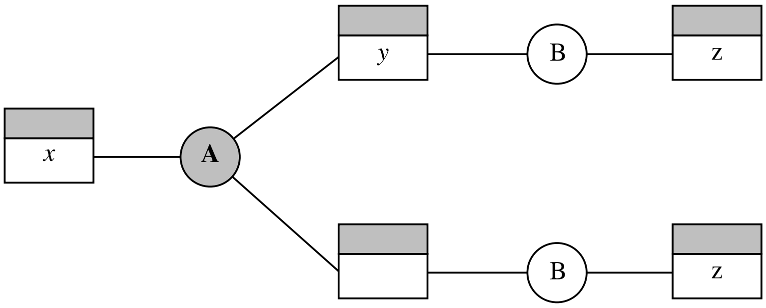


Abbildung 8.21: Auflösung des Zyklus'

Interaktion der Kontexte

Auf Seite 148 haben wir die Implementierung der Operation `match` innerhalb *eines* Kontextes beschrieben. Nun findet der Struktur- und Wertevergleich eines Anfrageskeletts mit einem Datenobjekt gerade in einem Zusammenspiel der Kontexte des Anfrageskeletts ab: sie überprüfen – jeder Kontext für sich – ob das Datenobjekt einen gleichgearteten Kontext besitzt. Als Resultat erhalten wir für jeden Kontext des Anfrageskeletts eine Aussage:

- ja, das Datenobjekt besitzt einen gleichgearteten Kontext – das ist der Fall, daß `match` das Ergebnis `true` zurückgibt – oder
- nein, das Datenobjekt besitzt keinen gleichgearteten Kontext – das ist der Fall, daß `match` das Ergebnis `false` zurückgibt.

Abb. 8.22 zeigt diese Situation für ein einfaches Anfrageskelett. Wir müssen nun diese vielen Einzelergebnisse zu einem Gesamtergebnis zusammenfügen. Vom logischen Gesichtspunkt ist es so, daß sich das Gesamtergebnis als logische UND-Verknüpfung der Einzelergebnisse ergibt; es stellt sich nur die Frage, wie wir die Kooperation der Kontexte realisieren, um dieses Gesamtergebnis zu berechnen.

Um eine Kooperation mehrerer Kontexte zu ermöglichen, bietet OPAQUE eine Ergebnispipeline an. Wir werden sie als den Datenbereich verwenden, an dem die Kontexte ihre Einzelergebnisse aus ihrer `match`-Operation ablegen und zum Gesamtergebnis kombinieren.

Die Ergebnispipeline wird mit dem Versenden der Nachricht `match` automatisch vom System mit dem Defaultwert `nil` initialisiert. Die Kontexte, die das Ergebnis ihrer `match`-Operation in der Pipeline ablegen möchten, überprüfen zunächst einmal das aktuell in der Pipeline anliegende Resultat `result.current`. Ist dies auf `nil` gesetzt, so hat noch kein anderer Kontext zuvor ein Ergebnis abgeliefert, so daß der Kontext, der auf diese Art und Weise auf `nil` stößt, sein `match`-Ergebnis in der Pipeline ablegt. Diejenigen Kontexte, die in `result.current` bereits einen booleschen Wert vorfinden, kombinieren diesen Wert mit dem Ergebnis ihrer `match`-Operation mit einem logischen UND und schreiben das Resultat in die Pipeline. So entsteht sukzessive eine UND-Verknüpfung aller Ergebnisse des Struktur- und Wertevergleichs der beteiligten Kontexte. Zusammengefaßt bedeutet das, daß eine `match`-Operation etwa mit folgendem Konstrukt enden muß[5]:

[5]Unter der Annahme, daß das Ergebnis des Struktur- und Wertevergleichs in einer booleschen Variablen `matchergebnis` abgelegt sei.

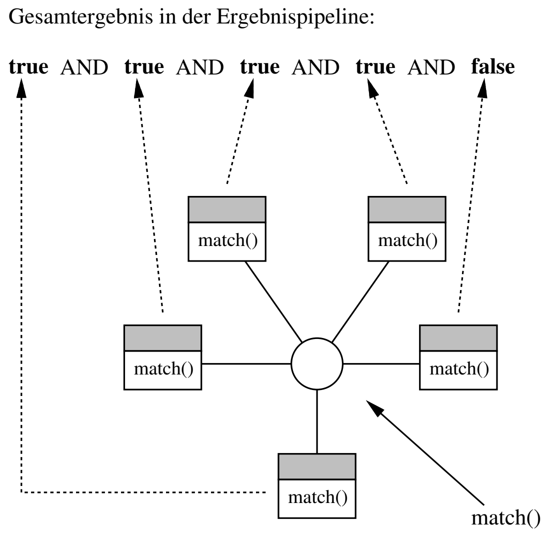


Abbildung 8.22: Einzelergebnisse eines Struktur- und Wertevergleichs für ein einfaches Anfrageskelett

```
if ( result.current = nil ) then
   { return ( matchergebnis ); }
else
   { return ( matchergebnis AND result.current ); }
```

8.5 Das Ausgabemodell

Für diesen dynamischen Entwurf übernehmen wir das in Kapitel 7 (Abschnitte 7.4.3 und 7.6.2) entworfene Ausgabemodell: jeder Kontext gibt die in ihm enthaltene Information eigenständig aus. Dies geschieht in Abhängigkeit von einem Abstraktionsgrad, der vom Benutzer vorgegeben wird. Ähnlich wie beim Struktur- und Wertevergleich ist diese Methodik hochgradig kontextsensitiv, da sich die Ausgabe der Information, die ein Objekt enthält, selbständig aus der Objektumgebung konfiguriert, ohne daß wir wissen, aus welchen Kontexten sich diese Umgebung im konkreten Fall zusammensetzt: befindet sich ein Kontext in der Umgebung eines Kristalls, so bringt er bei einem Aufruf zur Ausgabe automatisch die in ihm enthaltenen Informationen in die Gesamtausgabe mit ein.

Kapitel 9

Auswertung des Entwurfs

In diesem Abschnitt wollen wir den in Kapiteln 5 und 8 durchgeführten Entwurf auf Grundlage OPAQUEs auswerten. Insbesondere der in Kapitel 8 dargestellte dynamische Entwurf ist dabei von Interesse. Bei der Bewertung wollen wir besonders zwei Aspekte berücksichtigen:

- Wie flexibel sind die vorgestellten Modelle (Anfrage-, Such-, Ausgabemodell) gegenüber Änderungen in der Objektstruktur der Datenbasis, also gegenüber Änderungen am Objektmodell?
- Wie anpassungsfreundlich sind die Modelle? Können bestimmte Verfahren ausgetauscht werden, ohne daß dies grundlegende Änderungen nach sich zieht?

Diese beiden Aspekte werden in den Abschnitten 9.1 und 9.2 diskutiert.

9.1 Flexibilität gegenüber Änderungen am Objektmodell

Wir wollen die Flexibilität anhand einfacher Erweiterungen des Objektmodells untersuchen.

Dazu nehmen wir beispielsweise an, daß das in Kapitel 5 entworfene Objektmodell dahingehend erweitert werden soll, daß Daten und Informationen über Genehmigungsverfahren in der Datenbasis gehalten werden sollen, denen möglicherweise bestimmte Richtlinien zugrundeliegen. Dies bedeutet, daß eine neue Datenbasis geschaffen wird, die die Daten der Behörden, Genehmigungen etc. aufnimmt, und daß diese neue Datenbasis strukturell an die der Richtlinien und Richtlinienregelungen anzubinden ist. Abb. 9.1 zeigt eine Möglichkeit wie diese strukturelle Anbindung erfolgen kann.

Wir wollen nun nicht auf die Struktur oder Modellierung der neuen Datenbasis eingehen, sondern allein die Auswirkung auf unser bisheriges Objektmodell betrachten. Diese wird in Abb. 9.1 offensichtlich: die Klasse `Richtlinie` wurde einfach um die Kontextschablone `Genehmigungsverfahren_Ktxt` erweitert, die die Brücke zwischen der Richtlinien- und der Genehmigungsdatenbasis bildet. Aufgrund der beigefügten Kardinalitäten bedeutet dies, daß es in der neuen Datenbasis Richtlinienobjekte geben kann, die einen Kontext der genannten neuen Schablone an sich gebunden haben können. Es kann auch möglich sein, daß derartige Objekte einen solchen Kontext eines Tages wieder verlieren, weil beispielsweise ein Genehmigungsverfahren aufgrund

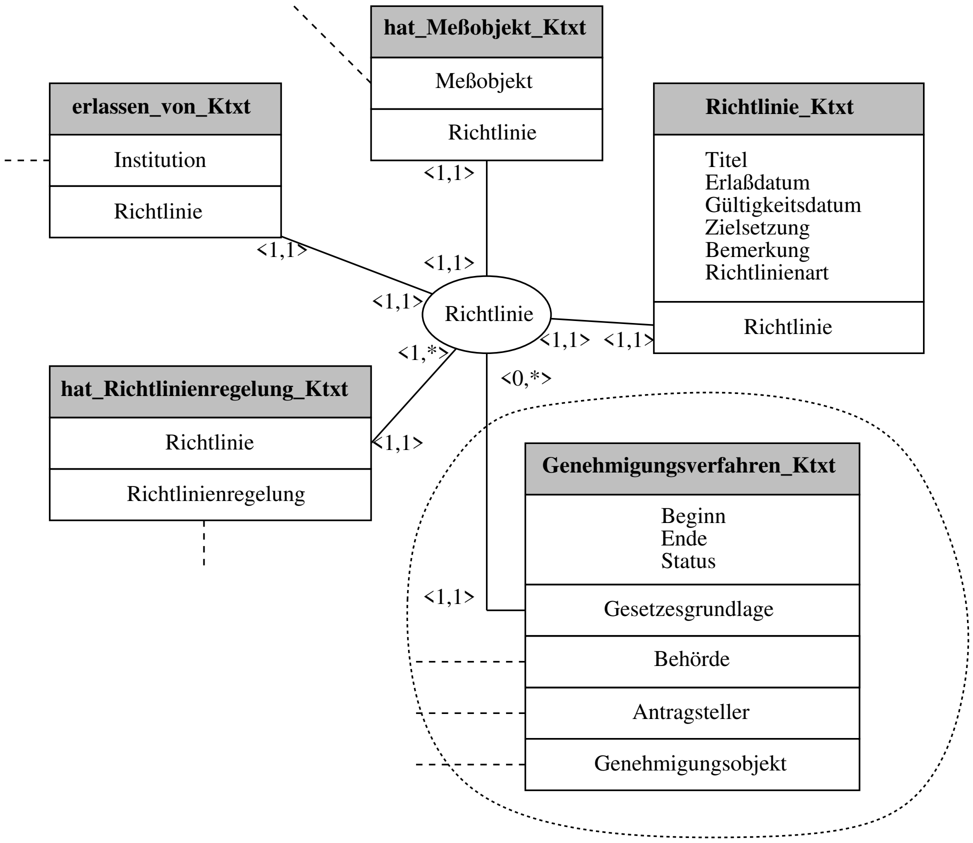


Abbildung 9.1: `Richtlinie` erweitert um die Kontextschablone `Genehmigungsverfahren_Ktxt`

veränderter Rechtslage (und damit veränderter Gesetzesgrundlage) neu aufgenommen werden könnte.

Ein weitere Änderung könnte darin bestehen, daß die Datenbasis erweitert wird, um Daten aufzunehmen, die bei Messungen gewonnen wurden. Abb. 9.2 zeigt eine mögliche Realisierung dieser Erweiterung im OPAQUE-EM-Objektmodell: sie besteht einfach darin, daß eine Kontextschablone `Meßung_Ktxt` eingeführt wird, die die bereits existierenden Klassen `Meßmethode`, `Parameter`, `Exaktwert` und `Maßeinheit` über die angebotenen Rollen einbezieht.

Es könnte auch möglich sein, daß die Richtliniendatenbasis beispielsweise an ein Chemie-Informationssystem angebunden werden soll, das zu einem chemischen Stoff u.a. auch (falls vorhanden) die entsprechende Richtlinieninformation liefert. Die Anbindung kann über eine Kontextschablone `hat_Parameter_Ktxt` erfolgen, die die Klasse `Parameter` mit einer Klasse `Stoff` der Chemie-Datenbasis verbindet. Abb. 9.3 zeigt diese Erweiterung.

Diese Beispiele zeigen, daß die Annahme durchaus realistisch ist, daß das entworfene Objektmodell langfristig zahlreichen Änderungen und Erweiterungen unterliegen wird. Daher ist ein wichtiger Aspekt hinsichtlich des durchgeführten Entwurfs, wie flexibel er sich diesen Änderungen anpassen läßt.

Wie wir anhand der Beispiele gesehen haben, ist das Objektmodell, also der statische Entwurf, leicht zu erweitern. In den Beispielen war es jeweils ausreichend, *eine* neue Kontextschablone zu definieren, die schon bestehende Klassen unter einem neuen Aspekt miteinander verbindet. Diese einfache Erweiterbarkeit des Objektmodells ist aber an sich kein hervorstechender Aspekt; beispielsweise könnte man vergleichbare relationale Datenbanken in ähnlicher Weise erweitern, indem man, anstatt *einer* Kontextschablone im Falle OPAQUEs, *eine* neue Relation einführt, die z.B. Tupel mit Schlüsselwerten für `Richtlinie`, `Behörde`, `Antragsteller` und `Genehmigungsobjekt` (siehe Beispiel aus Abb. 9.1) enthält, um die entsprechenden Objekte in Beziehung zu setzen. Es stellt sich vielmehr die Frage, wie das von uns entworfene Anfrage-, das Such- und das Ausgabemodell auf derartige Änderungen reagieren. Diese Frage werden wir in den folgenden Abschnitten diskutieren.

9.1.1 Anfragemodell

Da die Anfrageschnittstelle die Strukturvorgaben für die Anfrageskelette auf Basis des Objektmodells macht, fließen Änderungen am Objektmodell automatisch in die Anfrageschnittstelle mit ein. Wenn wir beispielsweise das erweiterte Szenario aus Abb. 9.1 aufgreifen und die Auswirkungen auf die in Abschnitt 8.3 in Abb. 8.1 dargestellte Situation betrachten, so ergibt sich das Skelett aus Abb. 9.4. Wie man leicht sieht, unterscheidet es sich von dem aus Abb. 8.1 nur dadurch, daß es zusätzlich die Schablone `Genehmigungsverfahren_Ktxt` andeutet, so daß Kontexte dieser Schablone wie jede andere für die Suche verwendet werden können.

Grundsätzlich unterliegt die Konzeption des Anfragemodells also keinen Auswirkungen aufgrund von Änderungen und Erweiterungen am Objektmodell, da das Anfrageverfahren unabhängig von der Struktur des Objektmodells ist und strukturelle Änderungen automatisch miteinfließen.

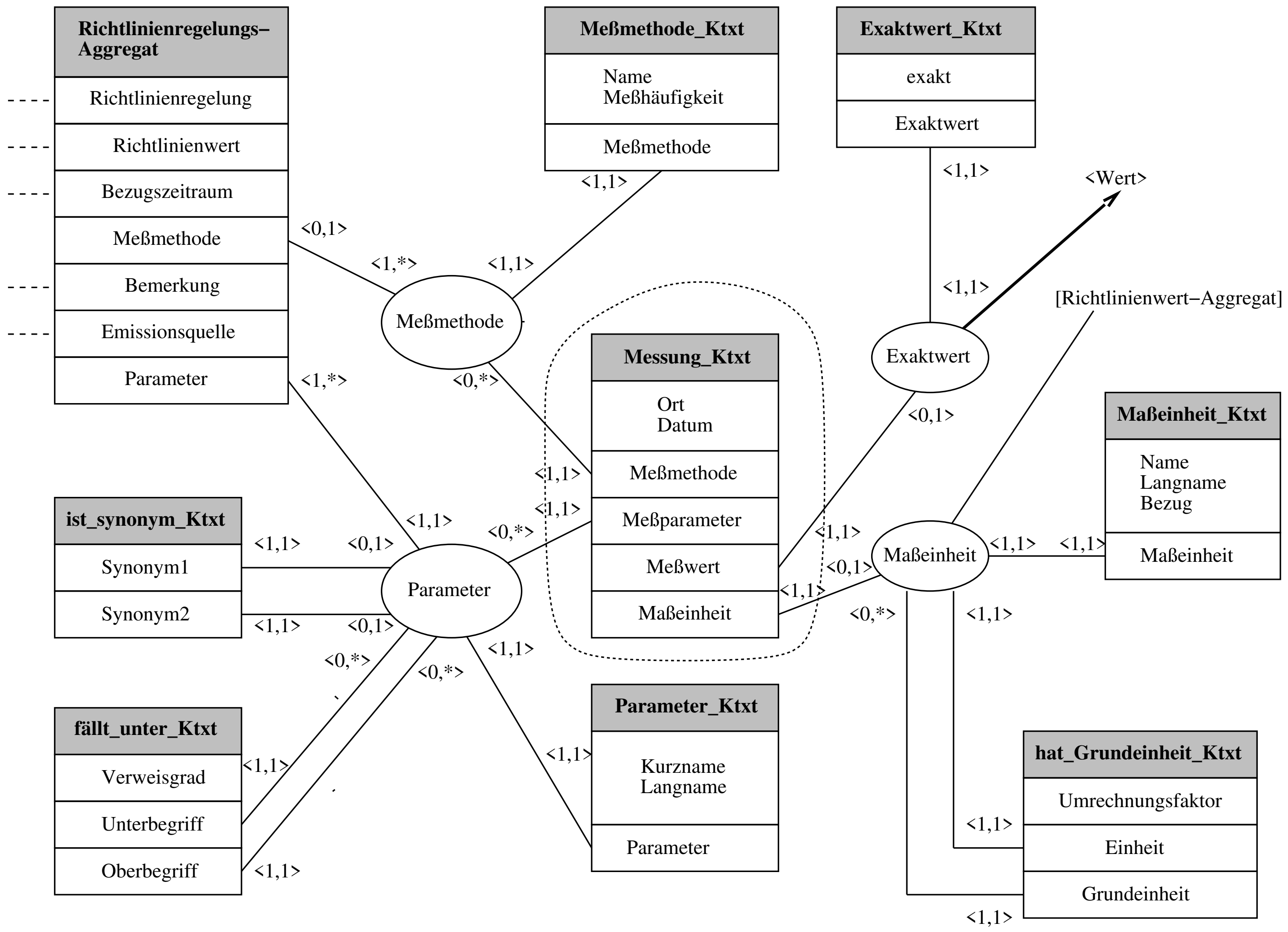


Abbildung 9.2: Erweiterung des Objektmodells zur Aufnahme von Meßdaten

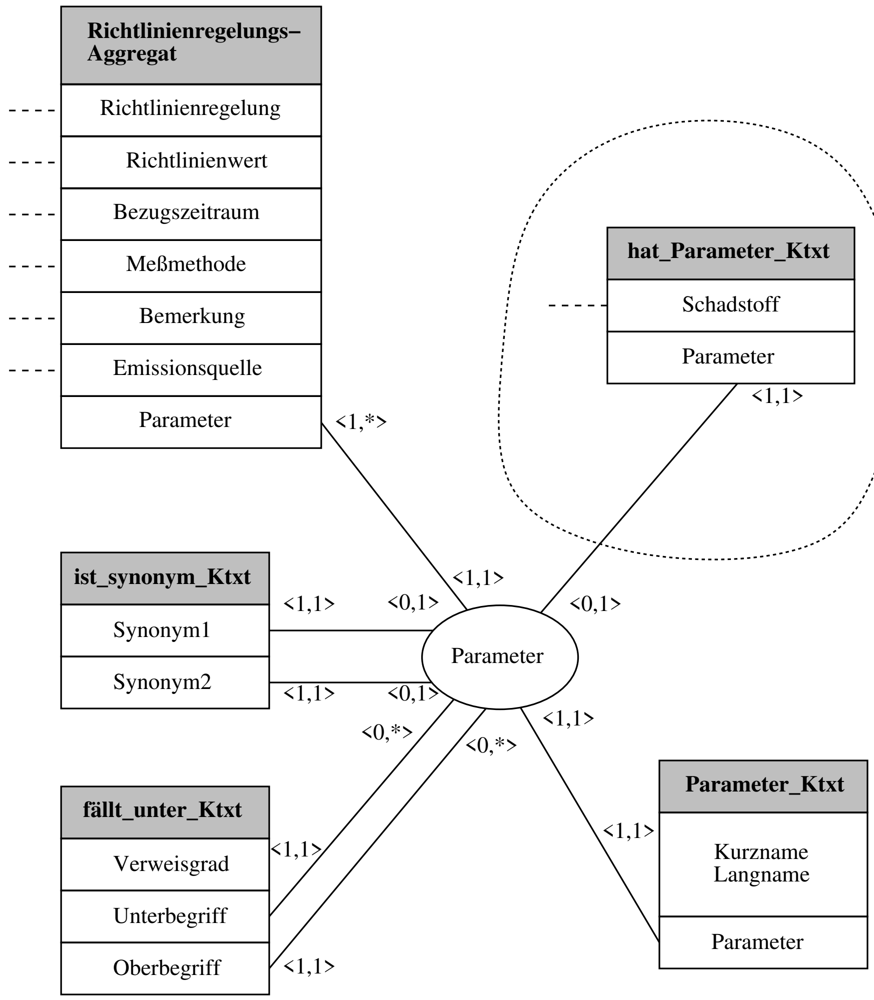


Abbildung 9.3: Erweiterung des Objektmodells zur Anbindung von Informationen über chemische Stoffe

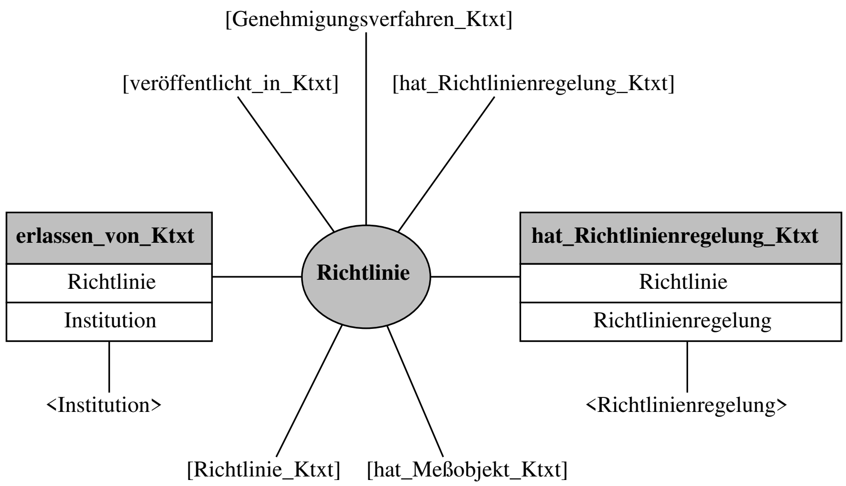


Abbildung 9.4: Auswirkung einer Erweiterung auf die Anfragesituation aus Abb. 7.1

9.1.2 Suchmodell

Der in Abschnitt 8.4.1 vorgestellte Suchalgorithmus ist von der Objektstruktur unabhängig. Erst im Struktur- und Wertevergleich der Operation `match` wird die Objektstruktur verarbeitet. Der Vorteil liegt aber darin, daß das Pattern Matching sich aufgrund der Strukturinformation konfiguriert, die durch das Anfrageskelett vorgegeben wird: die Gesamtreaktion setzt sich aus den Einzelreaktionen der beteiligten Kontexte zusammen. Die Auswirkungen einer Erweiterung des Objektmodells auf einen Struktur- und Werteabgleich ist in Abb. 9.5 dargestellt. Vergleiche dazu auch die Diskussion über die Strukturunabhängigkeit des Verfahrens auf Seite 147.

Hier zeigt sich ein echter Vorteil des OPAQUE-Systems. Diese Unabhängigkeit ist nämlich nicht durch einfache Kombination von Polymorphie und dynamischem Binden zu erreichen, wie sie von klassischen objektorientierten System angeboten werden. Der entscheidende Punkt liegt vielmehr darin, daß in OPAQUE die Gesamtreaktion auf eine Nachricht aus den Teilreaktionen der Kontexte erzeugt wird, das Verhalten sich also auch aus Umgebungsfacetten ergibt. Dabei muß die Zusammensetzung, also die Beschaffenheit der Umgebung *nicht* von vornherein bekannt sein.

9.1.3 Ausgabemodell

Ähnlich wie beim Struktur- und Wertevergleich setzt sich die Ausgabe aus den Einzelausgaben der an einem Objekt beteiligten Kontexte zusammen, so daß Änderungen an der Objektstruktur automatisch in die Ausgabe miteinfließen. Siehe dazu Abb. 9.6.

9.1.4 Zusammenfassung

Es zeigt sich also, daß die entworfenen Modelle ein Höchstmaß an Flexibilität gegenüber Änderungen und Erweiterungen am Objektmodell beinhalten. Man kann sogar sagen, daß die vorgestellten Modelle aufgrund der durch die Kontextsensitivität erreichten Flexibilität derart allgemein sind, daß sie auf *beliebige* Objektstrukturen anwendbar sind. Die einzigen Voraussetzungen sind allein die, daß jede Kontextschablone in jeder Rolle

- eine Operation `match` und
- eine Operation `ausgabe`

mit den in diesem Entwurf beschriebenen Eigenschaften anbieten muß, um sich nahtlos in das Such- und Ausgabemodell einzureihen. Fehlen eine oder gar beide dieser Operationen, dann bleibt die ursprüngliche Funktionalität des Systems erhalten – die neu hinzugefügte Schablone nimmt am Such- bzw. Ausgabeprozeß einfach nicht teil; das Hinzufügen hat also keine Seiteneffekte.

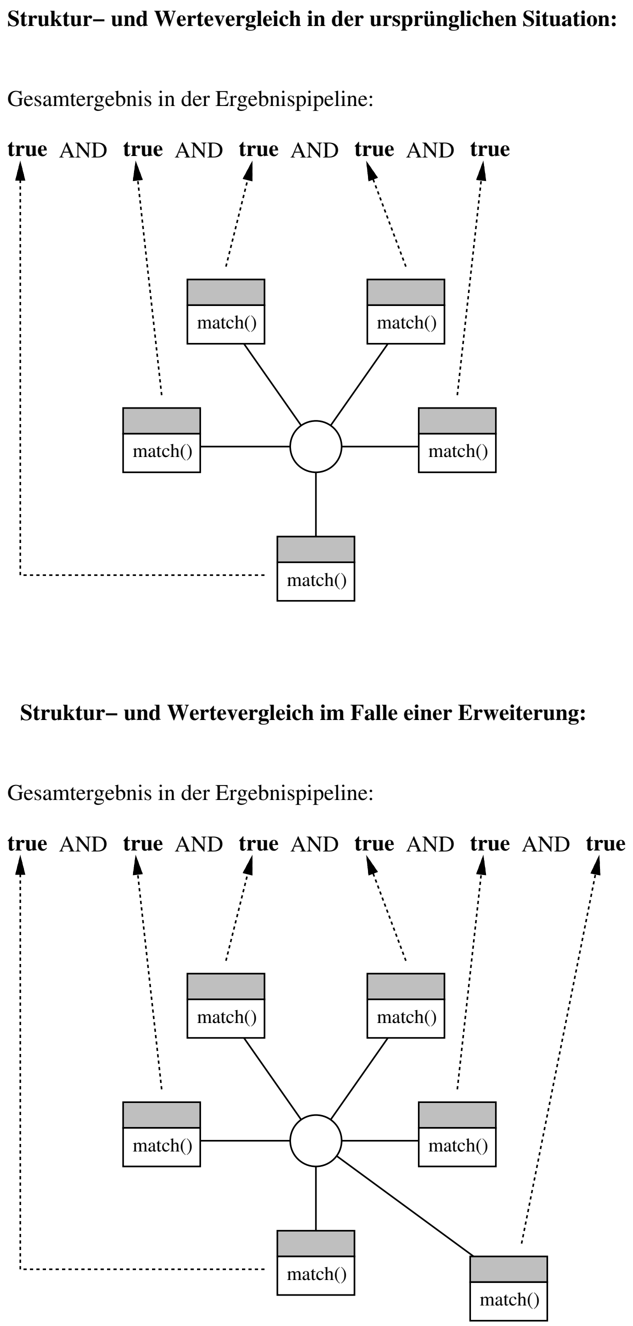


Abbildung 9.5: Auswirkungen einer Objektmodellerweiterung auf das Suchmodell

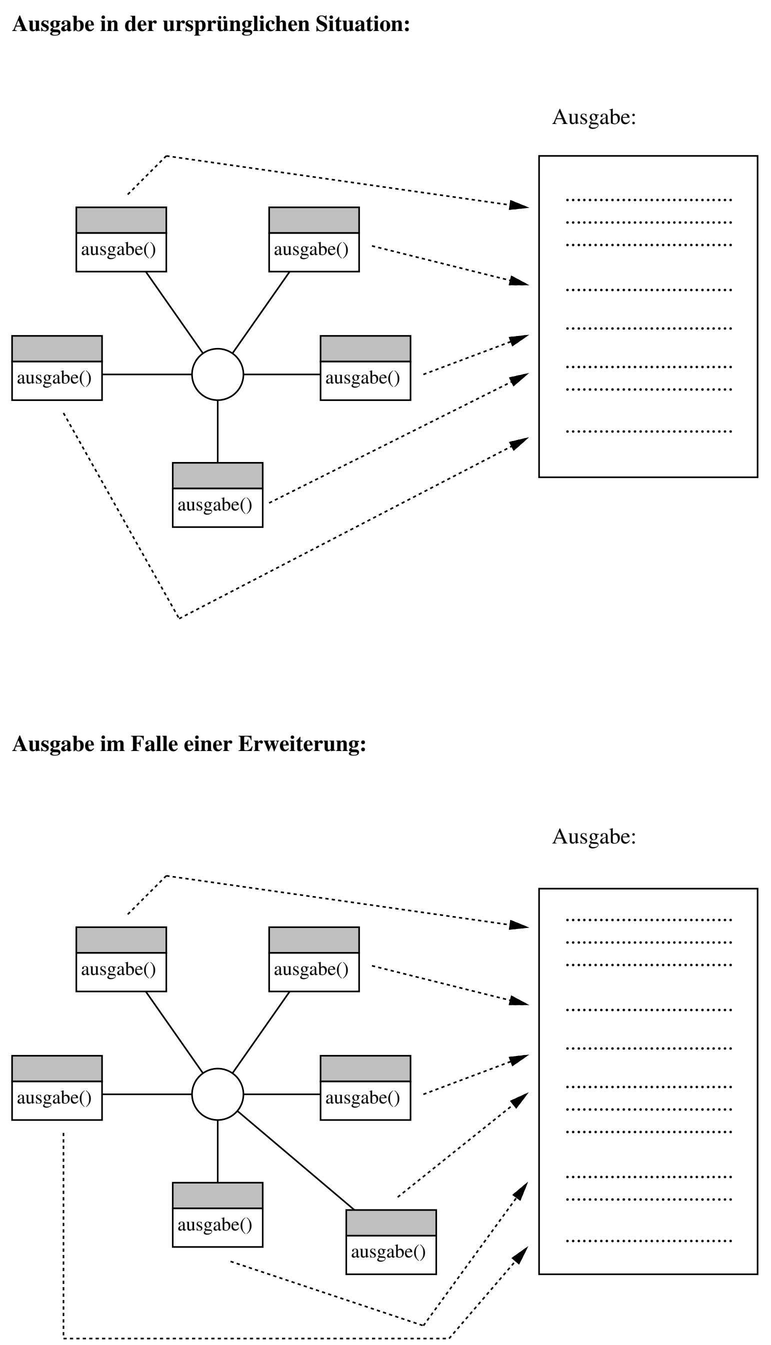


Abbildung 9.6: Auswirkungen einer Objektmodellerweiterung auf das Ausgabemodell

9.2 Anpaßbarkeit der Modelle

Während der Entwicklung des Anfrage- und des Suchmodells haben wir bereits auf mögliche Alternativen hingewiesen. Wir wollen u.a. diese Aspekte an dieser Stelle noch einmal aufnehmen.

9.2.1 Anfragemodell

Dem beschriebenen Anfragemodell liegt eine Benutzerschnittstelle zugrunde, die es zuläßt, Anfrageskelette aufbauend auf den grafischen Ausdrucksmitteln von OPAQUE-EM zu formulieren. Diese Anfrageskelette lassen sich dadurch auf einfache Art und Weise auf OPAQUE-Strukturen abbilden und als OPAQUE-Objekte halten. Diese OPAQUE-Objektstruktur stellt die Schnittstelle zwischen Anfrage- und Suchprozeß dar. Deshalb ist es unerheblich, wie ein Anfrageskelett als OPAQUE-Objekt entsteht, um an den Suchprozeß übergeben zu werden. Eine Reihe anderer geeigneter grafischer oder anderer Benutzerschnittstellen sind denkbar. In diesem Zusammenhang haben wir bereits darauf hingewiesen, daß durchaus auch eine an der OMT-Notation orientierte Benutzerschnittstelle denkbar ist, da sich OMT-Objektstrukturen automatisch, wie in Abschnitt 3.5 gezeigt, in OPAQUE-EM und von dort in OPAQUE übersetzen lassen. Abb. 9.7 zeigt ein Anfrageskelett in OMT-Notation, das mit Hilfe der in Abschnitt 3.5 vorgestellten Mechanismen in das OPAQUE-EM-Skelett aus Abb. 8.11 transformiert werden kann.

9.2.2 Suchmodell

Das beschriebene Suchmodell enthält einen Pattern-Matching-Algorithmus, der in den `match`-Operationen implementiert ist. Dieser Algorithmus ist natürlich austauschbar. Man kann beispielsweise auch daran denken, unterschiedliche Algorithmen für unterschiedliche Benutzergruppen anzubieten: so kann es für einen Benutzer, der mit der Diskurswelt wenig vertraut ist, sinnvoll sein, wenn die `match`-Operationen einen vom Benutzer vorgegebenen Attributwert nicht nur beim angegebenen Attribut, sondern auch in anderen zu suchen, um der Unsicherheit eines solchen Benutzers Rechnung zu tragen – er könnte aufgrund seiner Unsicherheit den Wert an falscher Stelle eingetragen haben.

In Abb. 9.8 ist ein solches Beispiel dargestellt. Dort gibt es zwei unterschiedliche Vergleichsoperationen `match1` und `match2`. Die Operation `match1` führt dabei einen strengen Vergleich der Attributwerte durch und kommt zum Ergebnis `false`; das in der Operation `match2` implementierte fehlertolerante Verfahren stellt fest, daß der Attributwert *Emission* zwar nicht im Attribut `Titel` dafür aber in `Richtlinienart` erscheint und liefert daher das Ergebnis `true`.

Eine weitere Möglichkeit besteht darin, daß ein Benutzer seine eigene `match`-Operation definiert, indem er einen Kontext mit dieser Operation erzeugt und diesen dem Anfrageskelett hinzufügt, das dadurch eine zusätzliche Funktionalität erhält. Abb. 9.9 zeigt diese Situation. Während der Anfrageformulierung kann der Benutzer dem System angeben, daß seine `match`-Operation bei der Suche zu berücksichtigen ist.

Prinzipiell ist auch ein Austausch des Vergleichsalgorithmus' unproblematisch; implementierungstechnisch kann ein solcher Austausch jedoch aufwendig werden, da wir für jede Kontextschablone im OPAQUE-EM-Objektmodell mindestens eine `match`-Operation vorsehen müs-

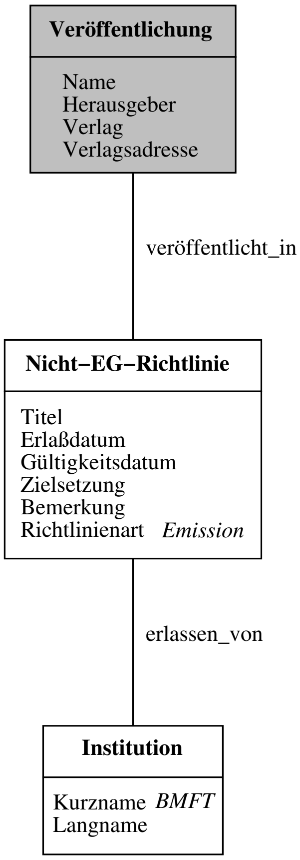


Abbildung 9.7: Ein Anfrageskelett in OMT-Notation

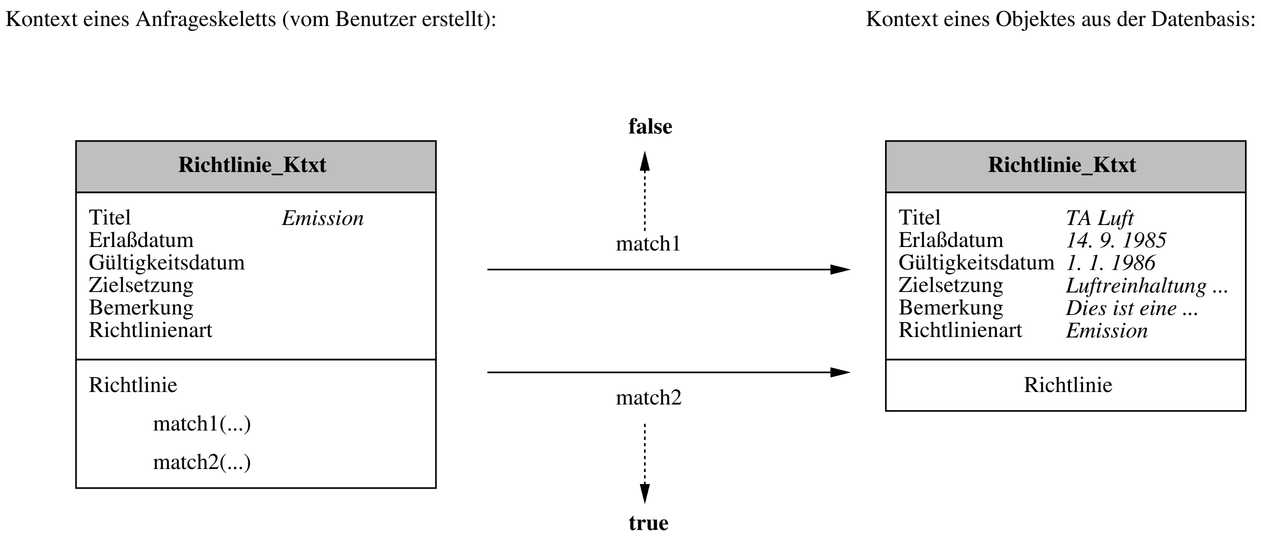


Abbildung 9.8: Ein strenger Attributvergleich (`match1`) und ein toleranter (`match2`)

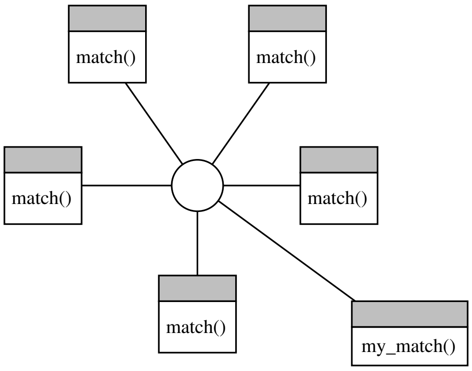


Abbildung 9.9: Benutzerdefinierte `match`-Operation an einem Anfrageskelett

sen, was zu über 30 Implementierungen führt. Dies entspricht dem Änderungsaufwand für die zahlreichen Suchoperationen des ersten dynamischen Entwurfs aus Kapitel 7. Dieser Nachteil ist der Preis für die oben beschriebene Flexibilität des Suchmodells. Wir hatten sie gerade dadurch gewonnen, daß wir Gesamtreaktionen aus den Einzelreaktionen der Kontexte erzeugt haben. Daher ist es unvermeidbar, daß mit einer steigenden Anzahl von Kontextschablonen der Implementierungs- und Änderungsaufwand wächst.

9.2.3 Ausgabe- und andere Auswertemodelle

Bei der Beschreibung des Ausgabemodells im Rahmen des ersten dynamischen Entwurfs (siehe Abschnitt 7.4.3) haben wir bereits darauf hingearbeitet, unterschiedlichen Benutzergruppen unterschiedliche Präsentationen der Informationen anzubieten. Die Art der Ausgabe hängt im wesentlichen von der Sichtweise und der Erwartungshaltung der Benutzer ab. Diese gruppenspezifischen Eigenschaften kann man als weitgehend konstant betrachten, so daß man das Ausgabemodell langfristig darauf anlegen kann, obwohl es natürlich möglich ist, daß ein Benutzer im Laufe der Zeit an Erfahrung und Wissen hinzu gewinnt und in eine Gruppe erfahrener Benutzer wechseln kann.

Andererseits können die Daten nicht nur der Information irgendwelcher Benutzer dienen, sondern auch in Auswertungen von Meßreihen, in wissenschaftliche Studien, in Analysen etc. einfließen. Ähnlich wie im Falle der Benutzergruppen werden hier Ausschnitte der Datenbasis unter einem bestimmten Aspekt betrachtet und die darin enthaltenen Informationen benutzt. Diese Betrachtungen sind im Gegensatz zu denen der Benutzergruppen eher kurzfristiger Natur: wissenschaftliche Studien laufen manchmal nur über ein paar Monate; es gibt Meßreihen, die sich nicht periodisch wiederholen, sondern einmalig durchgeführt werden, beispielsweise anläßlich eines Chemieunfalls, eines Verdachtsmoments o.ä. Wir wollen diese sehr individuellen, auf die konkrete Problemstellung zugeschnittenen, temporären Sichtweisen als *temporäre* Auswertemodelle bezeichnen, im Gegensatz zu den oben erwähnten gruppenspezifischen Ausgabemodellen, die wir als *langfristige* Auswertemodelle betrachten wollen. Der Auswertecharakter der Ausgabemodelle besteht einfach darin, die Information in einer benutzergerechten Form aufzubereiten und darzustellen.

Während des Entwurfs haben wir uns auf die langfristig angelegten Auswertemodelle konzentriert, da man über den Charakter und Zweck möglicher temporärer Auswertemodelle nur mutmaßen kann. Wir wollen deshalb untersuchen, welche Möglichkeiten der durchgeführte Entwurf bietet, temporäre Auswertemodelle zu installieren.

Relationale Datenbanksysteme bieten zur Definition individueller Sichtweisen das Konzept der *Views* an: Views sind virtuelle Relationen, deren Inhalt sich aus Relationen ergibt, die in der Datenbasis gehalten werden. Die Definition eines Views besteht aus einer Beschreibung, auf welche Art und Weise sich die virtuelle Relation aus den real existierenden berechnen läßt. Dieses View-Konzept relationaler Datenbanksysteme läßt sich in OPAQUE leicht nachmodellieren, indem man beispielsweise eine neue Ausgabeoperation definiert, die die gewünschte Sichtweise auf die Daten beinhaltet und z.B. bestimmte Attributwerte ausblendet, verändert ausgibt, statistisch auswertet etc.

Die Auswertemodelle können sich aber nicht nur darin unterscheiden, welche Daten sie verwenden, welche sie ausblenden und wie sie die Daten aufbereiten. Es ist auch durchaus möglich, daß die in der Datenbasis gehaltenen Objekte zum Zwecke einer individuellen Auswertung ein anderes oder erweitertes Verhaltensmuster annehmen sollen. Man denke beispielsweise daran, daß im Szenario aus Abb. 9.2 Meßwerte[1], die Konzentrationen der Art "Volumen pro Volumen" beschreiben[2], für eine spezielle Auswertung immer auf die Einheit ml/l (Milliliter pro Liter) umgerechnet werden sollen, um sie beispielsweise miteinander vergleichen zu können. Dazu hängen wir an diese Objekte der Klasse `Exaktwert` einfach einen zusätzlichen Kontext, der die entsprechende Funktionalität besitzt und einen Wert auf einen den Millilitern pro Liter entsprechenden Zahlenwert umrechnen kann. Abb. 9.10 zeigt einen Exaktwert, dessen Verhaltensmuster durch einen Kontext der Schablone `Umrechnung_Ktxt` erweitert wurde.

Wir haben in dieser Abbildung gezielt nur ein Exaktwert*objekt* dargestellt, um damit zu betonen, daß diese Erweiterung nicht grundsätzlich alle Exaktwerte betreffen muß. Der genannte Kontext kann dem Exaktwert durch ein Anwendungsprogramm vorrübergehend angehängt und später wieder abgenommen werden. Das bedeutet, daß eine solche temporäre Erweiterung auch nicht im Objektmodell modelliert werden muß. Das in Kapitel 5 entwickelte OPAQUE-Objektmodell bzw. die in diesem Abschnitt vorgestellten, möglichen Erweiterungen repräsentieren nur eine Ausgangskonstellation, die sich in der Dynamik der Abläufe ändern kann.

Gerade in dieser Flexibilität unterscheidet sich eine Modellierung in OPAQUE im allgemeinen und der vorliegende Entwurf im besonderen von Modellierungen auf Grundlage herkömmlicher objektorientierter oder gar relationaler Datenbanksysteme. Erstgenannte lassen aufgrund ihrer strengen Typisierung nur starre, einmal vorgegebene Verhaltensmuster zu: weder kann eine Klasse nachträglich um zusätzliche Operationen erweitert werden, noch kann ein Objekt einfach einer anderen Klasse zugeordnet werden. Um eine zu Abb. 9.10 vergleichbare Erweiterung einzuführen, müßte man eine Unterklasse `Exaktwert'` zur Oberklasse `Exaktwert` schaffen, die zusätzlich die Funktionalität der Kontextschablone `Umrechnung_Ktxt` enthielte. Um nun einem Objekt x der Klasse `Exaktwert` diese zusätzliche Funktionalität zu geben, müßte man ein Objekt x' der neuen Klasse `Exaktwert'` erzeugen, dessen Attribute mit den Werten von x instantiiert würden. Danach könnte man das Objekt x aus der Datenbasis entfernen und an seiner Stelle mit x' arbeiten. Derselbe Aufwand entstünde, wenn man x' diese zusätzliche Funktionalität wieder wegnehmen und x' nach `Exaktwert` übertragen wollte. Weitere Informationen zu dieser Problematik findet der interessierte Leser in [MZ92] und [RS91].

[1] siehe Rolle `Meßwert` der Kontextschablone `Messung_Ktxt`

[2] Das bedeutet, daß sie über einen Kontext der Schablone `Messung_Ktxt` mit Maßeinheiten verbunden sind, die Konzentrationen wie etwa l/m^3, cm^3/l u.ä. beschreiben.

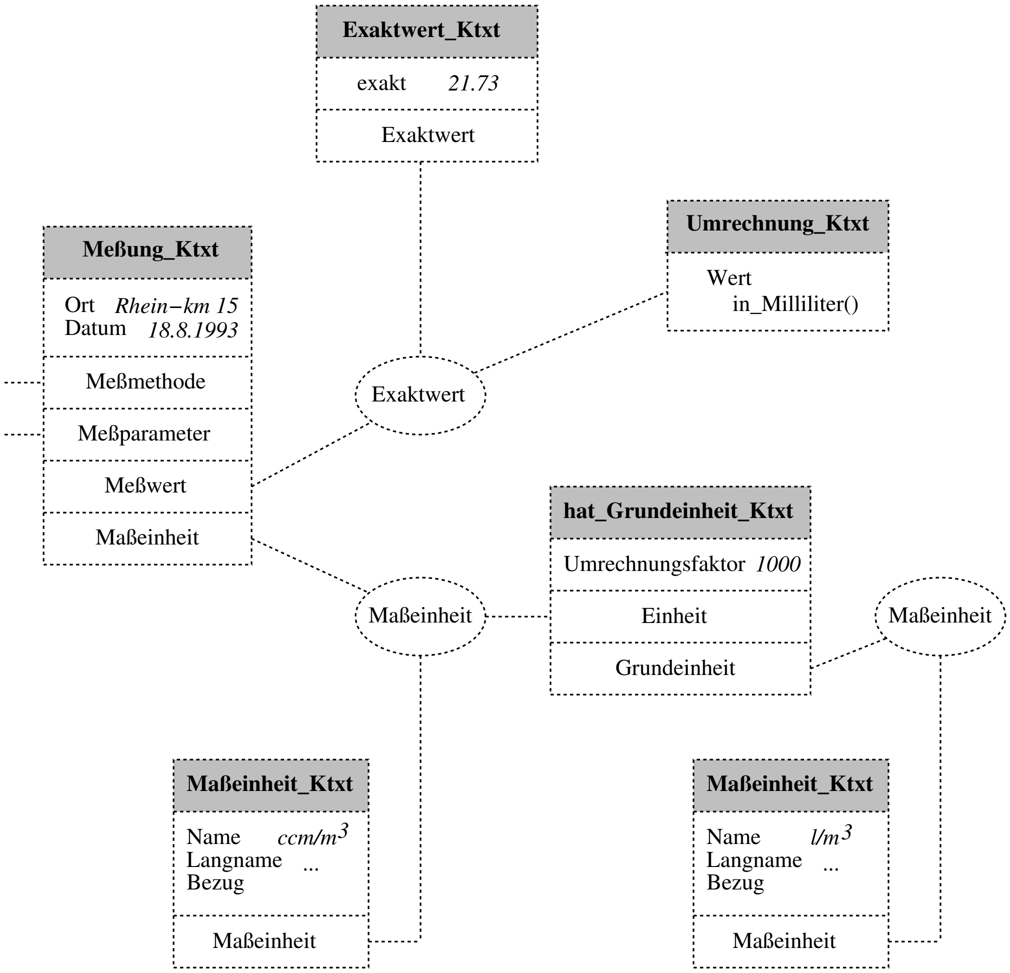


Abbildung 9.10: Ein Exaktwertobjekt erweitert um einen Kontext der Schablone `Umrechnung_Ktxt`

Wie wir sehen, sind kurzzeitige Änderungen des Verhaltensmusters eines oder mehrerer Objekte in herkömmlichen objektorientierten Systemen zwar möglich, aber in der Regel nur durch programmiertechnische Tricks zu realisieren, die mit einem beträchtlichen Aufwand einhergehen. Abgesehen davon, entspricht beispielsweise das Erzeugen, Kopieren und Löschen von Objekten keineswegs dem natürlichen Vorgang unseres Beispiels, nämlich daß ein Objekt sein Verhalten vorübergehend ändert oder erweitert.

9.3 Zusammenfassung

In den Abschnitten 9.1 und 9.2 haben wir die Flexibilität und Anpaßbarkeit des Entwurfs diskutiert. Es hat sich gezeigt, daß der Entwurf hinsichtlich Erweiterungen und Änderungen am Objektmodell, sowie Anpassungen, Veränderungen und Ausbaumöglichkeiten der Module des dynamischen Entwurfs in vielerlei Hinsicht Modellierungen auf Grundlage herkömmlicher objektorientierter Systeme überlegen ist. Das bedeutet *nicht*, daß die OPAQUE-Modellierung *mehr* kann als herkömmliche Modellierungen; die Schlußfolgerung lautet vielmehr, daß sie gerade auf Veränderungen flexibler und natürlicher reagieren kann als andere. Wie wir anhand der zahlreichen Beispiele gesehen haben, ist diese Eigenschaft im Bereich eines Umweltrichtlinienszenarios besonders wünschenswert.

Diese Flexibilität haben wir dadurch erreicht, daß wir letztendlich eine Modellierung entworfen haben, die nicht nur in sich selbst kontextsensitive Aspekte berücksichtigt, sondern auch insgesamt nach außen einen kontextsensitiven Charakter hat: wie wir gesehen haben, läßt sich der Entwurf in größere Szenarien beinahe mühelos einbetten, wie beispielsweise in eine Datenbank für Genehmigungsverfahren (Abb. 9.1), zur Erfassung von Meßdaten (Abb. 9.2), in ein Chemie-Informationssystem (Abb. 9.3) etc. Ähnlich wie ein Kristall in OPAQUE an verschiedenen Kontexten in unterschiedliche Rollen schlüpfen kann, kann das entworfene Umweltinformationssystem in unterschiedliche Szenarien, ja Zusammenhänge – also *Kontexte* im wörtlichen Sinn – integriert werden. Dies geschieht aufgrund des kontextsensitiven Charakters des Gesamtentwurfs weitgehend problemlos.

Dieses 'Von-innen-nach-außen-Tragen' eines Modellierungsparadigmas, wie in diesem Falle der Kontextsensitivität, kann man auch in vielen anderen Daten- und Programmiermodellen beobachten; man denke nur an die Forderungen des Software Engineering nach Modularisierung, Information Hiding etc., die zur Entwicklung von Programmiersprachen wie Pascal, Modula-2, Ada u.ä. geführt haben. Das Prinzip der Modularisierung beginnt ganz innen auf Prozedurebene und führt über die Modulebene bis nach außen zur Betrachtung des Gesamtsystems als Modul, das Bestandteil eines noch größeren modularen Systems sein kann.

Kapitel 10

Vergleich mit einem relationalen Entwurf

In diesem Kapitel präsentieren wir in einer Kurzfassung einen Entwurf für das Umweltrichtlinienszenario, der auf dem relationalen Datenmodell basiert und von einer Projektgruppe am Forschungszentrum Informatik (FZI) unter Leitung von Ralf Kramer erstellt wurde. In Abschnitt 10.1 stellen wir diesen relationalen Entwurf vor, um danach in Abschnitt 10.2 einen Vergleich zu den in den Kapiteln 5, 7 und 8 vorgestellten Entwürfen auf Grundlage des Kontextmodells OPAQUE zu ziehen. Ziel soll es dabei sein, die Unterschiede herauszuarbeiten, die sich aufgrund der unterschiedlichen logischen Datenmodelle ergeben.

10.1 Ein relationaler Entwurf

Dieser Abschnitt beschreibt in der notwendigen Kürze den am FZI Karlsruhe erstellten Entwurf auf Grundlage des relationalen Datenmodells. Dabei werden die in [Kra93b] und [HK93] vorgestellten Ergebnisse zusammengefaßt.

Um den in Abschnitt 10.2 folgenden Vergleich zum objektorientierten und kontextsensitiven Entwurf zu erleichtern, gliedert sich dieser Abschnitt in Anlehnung an vorangegangene Kapitel wie folgt:

- Abschnitt 10.1.1 beschreibt die Zielsetzung des Entwurfs.
- In Abschnitt 10.1.2 stellen wir die semantische Datenmodellierung in Form eines Entity-Relationship-Diagramms[1] vor. Dies entspricht der Modellierung der statischen Aspekte, die wir in Kapitel 5 in Form eines OMT- bzw. eines OPAQUE-EM-Objektmodells dargestellt haben.
- In Abschnitt 10.1.3 beschreiben wir das Anfragemodell.
- Abschnitt 10.1.4 faßt die Grundzüge des entworfenen Suchmodells zusammen.
- Abschnitt 10.1.5 stellt das Ausgabemodell des Entwurfs vor.

[1] Eine Einführung in die Symbolik einer Entity-Relationship-Modellierung findet der Leser in [KS91].

10.1.1 Zielsetzung

Der Entwurf, der am FZI Karlsruhe durchgeführt wurde, unterlag einer etwas anderen Zielsetzung als der von uns präsentierte. Es bestanden z.B. genaue Vorgaben bezüglich der Benutzergruppen und der angestrebten Einbindung der Implementierung in das UIS des Landes Baden-Württemberg. So war die logische Datenmodellierung 'nur' eine Zwischenstufe auf dem Weg zu einer Implementierung der Grenz-/Richtwertdatenbank. Viele Entwurfsentscheidungen orientierten sich an praktischen Gesichtspunkten, wie beispielsweise die Tatsache, daß in der täglichen Anwendung nicht alle in der Datenbasis gehaltenen Informationen für die Suche relevant sind und daher von einem Anfragemodell vernachlässigt werden können.

In der vorliegenden Arbeit war dagegen das Interesse auch auf den Entwurfsprozeß an sich, die verwendete Methodik und die Modellierungseigenschaften OPAQUEs gerichtet. Entscheidungen, die wie beim FZI-Entwurf aus dem Ziel einer Implementierung heraus entstanden, gab es nicht, da zum Zeitpunkt dieser Arbeit noch kein OPAQUE-Datenbanksystem zur Verfügung stand. Dies war ein zusätzlicher Freiheitsgrad, aus dem auch die Anforderungen nach Flexibilität, Anpaßbarkeit und Einbindung in *irgendein* UIS erwachsen waren.

Beim Vergleich der beiden Entwürfe müssen diese etwas auseinandergehenden Zielsetzungen und Randbedingungen beachtet werden, da sich daraus schon einige Unterschiede in den Entwürfen erklären lassen, die nicht unbedingt auf die verwendeten Datenmodelle zurückzuführen sind.

10.1.2 Die statische Modellierung

Abb. 10.1 zeigt das Entity-Relationship-Diagramm[2], das dem am FZI verfolgten Entwurf zugrunde lag [Kra93b]. Zum Vergleich zeigt Abb. 10.2 das von uns in Kapitel 5 entwickelte OMT-Objektmodell in einer vereinfachten Form: der Übersichtlichkeit halber wurden Kardinalitäten und die Namen der OMT-Assoziationen weggelassen. Wir werden im Verlaufe dieses Abschnitts oft einfach von *dem* ER-Diagramm, *der* ER-Modellierung oder *dem* ER-Modell sprechen, wenn wir das Diagramm aus Abb. 10.1 meinen, und von *dem* OMT-Objektmodell oder einfach nur von *dem* Objektmodell, um auf die Modellierung aus Abb. 10.2 zu verweisen[3].

In diesem Abschnitt geht es uns darum, den Leser mit der ER-Modellierung vertraut zu machen. Dies wollen wir tun, indem wir Gemeinsamkeiten und Unterschiede zwischen den Diagrammen identifizieren. Dieser Vergleich soll nicht dazu dienen, die unterschiedlichen Entwurfsentscheidungen zu diskutieren, sondern soll dem Leser helfen, sich in der vorgestellten ER-Modellierung besser zurechtzufinden, da er bereits mit der Modellierung aus Abb. 10.2 vertraut ist.

Generell lassen sich zwischen einer Entity-Relationship-Modellierung und einem OMT-Objektmodell die folgenden Analogien herstellen:

- Objektklassen in OMT entsprechen den Entity Sets (Gegenstandstypen) in einer ER-Modellierung.

[2] Anmerkung: die aktuelle Version des ER-Diagramms verzichtet auf den Beziehungstyp `gültig in Verbindung mit` und hält die ursprünglich dort vorhandenen Informationen mittels des Gegenstandstyps `Randbedingung`. Da dies aus praktischen und nicht aus Gründen der inhaltlichen Modellierung geschah, haben wir uns an dieser Stelle dafür entschieden, die ursprüngliche Version der ER-Modellierung zu verwenden.

[3] Bzw. auf die detaillierte Darstellung des OMT-Objektmodells in den Abb. 5.1 bis 5.6 in Kapitel 5.

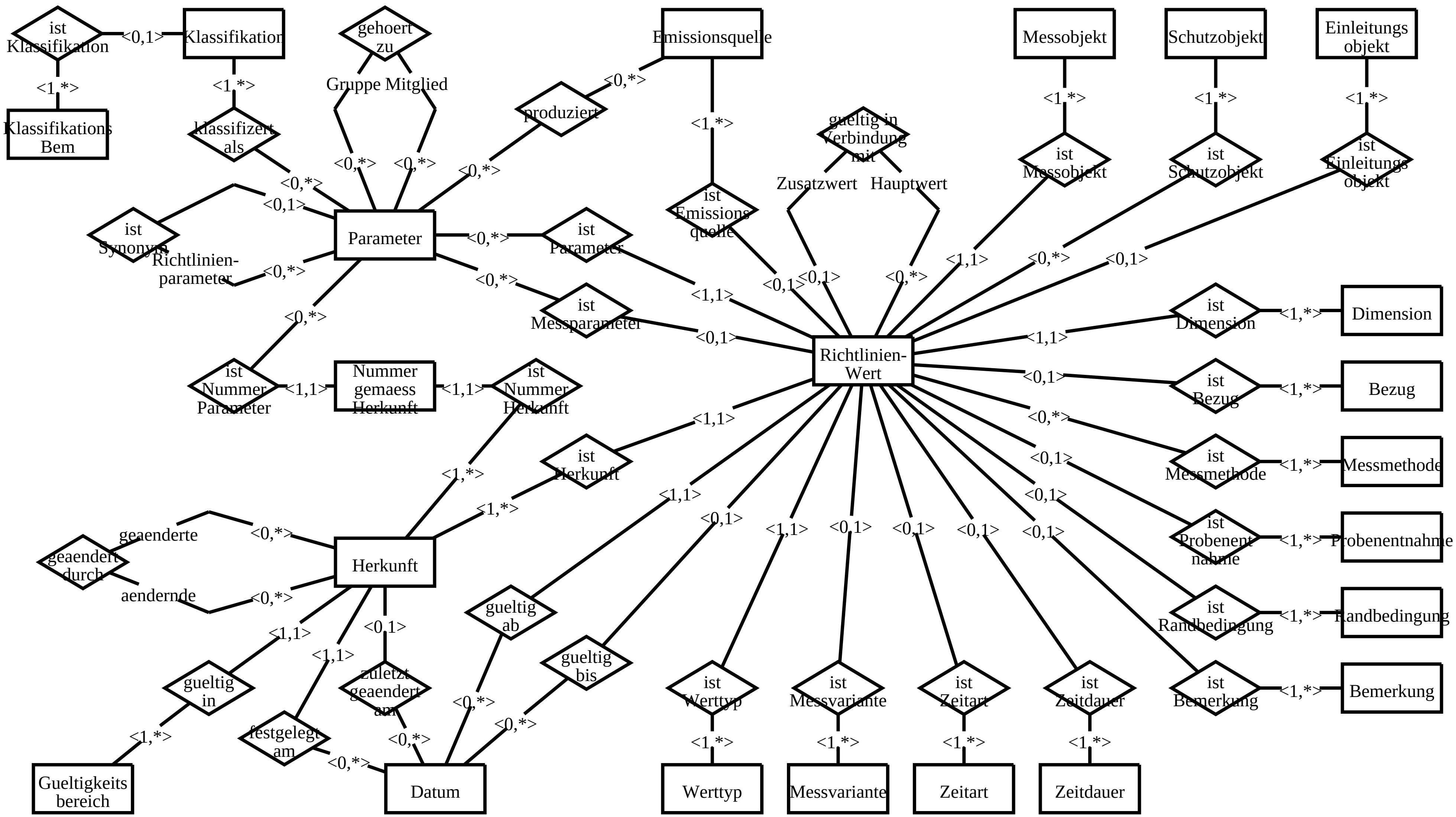


Abbildung 10.1: Eine Entity-Relationship-Modellierung

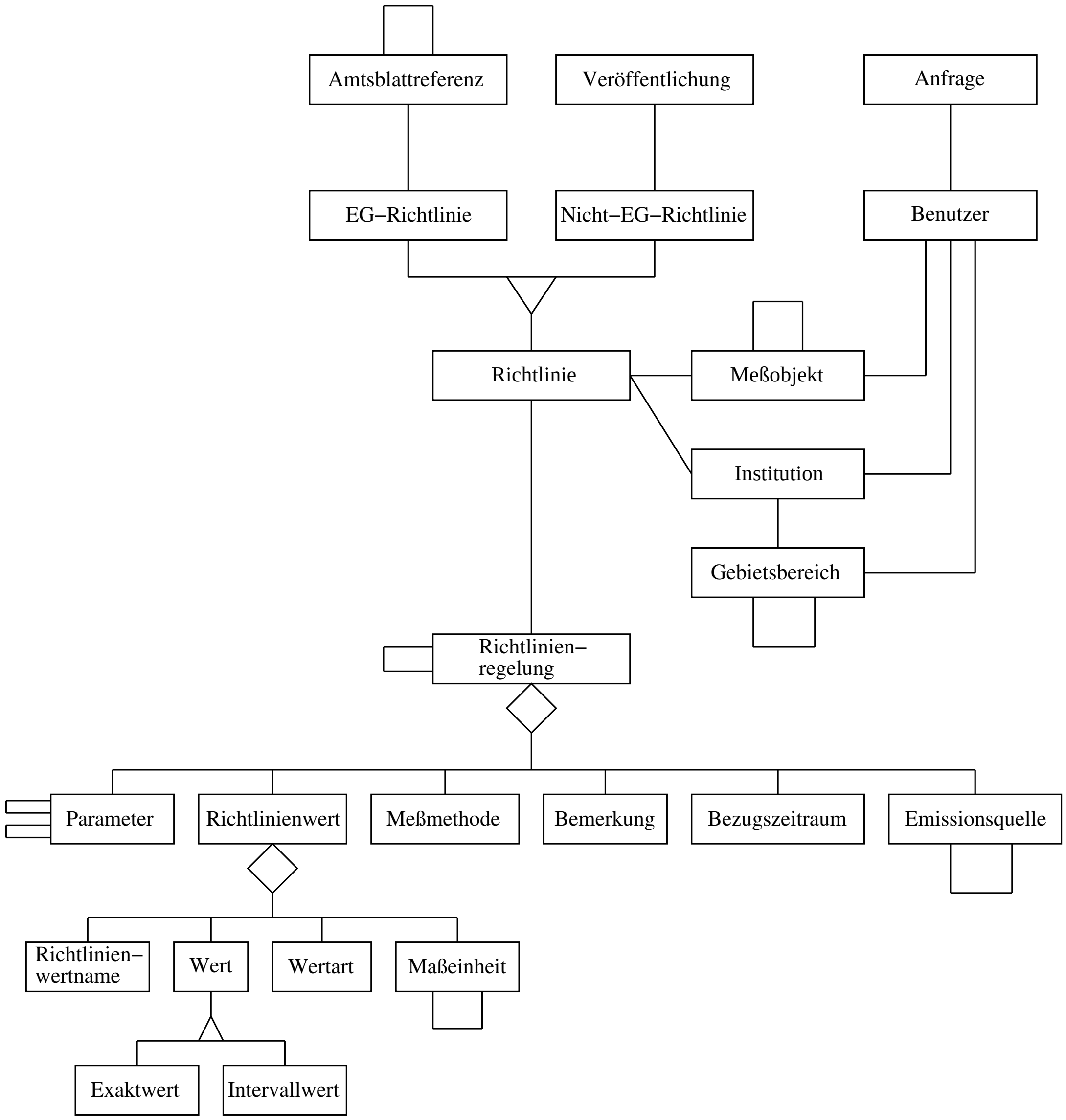


Abbildung 10.2: Das OMT-Objektmodell in vereinfachter Form

OMT-Objektmodell	**ER-Modellierung**
Richtlinie	Herkunft
Gebietsbereich	Gültigkeitsbereich
Richtlinienwertname	Werttyp
Wertart	Zeitart
Maßeinheit	Dimension
Richtliniennummer*	Nummer gemäß Herkunft
Richtlinienregelung, -wert	Richtlinienwert

* Nur im Sinne der Beziehung zwischen `Herkunft` und `Nummer gemäß Herkunft`.

Tabelle 10.1: Begriffsabgleich zwischen dem OMT-Objektmodell und der ER-Modellierung

- Assoziationen entsprechen den Relationships (Beziehungstypen).

Darüberhinaus gibt es Transformationen von Vererbungsbeziehungen und Aggregationen in Konstrukte des Entity-Relationship-Ansatzes. Für den unten folgenden Vergleich der beiden Diagramme sollen aber die beiden genannten Analogien genügen.

Ein Problem hinsichtlich eines Vergleiches liegt darin, daß beide Entwürfe teilweise unterschiedliche Begriffe für denselben Gegenstand verwenden. Tabelle 10.1 enthält eine Liste von Begriffen, die als gleichwertig betrachtet werden können. Eine etwas komplizierte Situation ergibt sich für den Begriff des *Richtlinienwerts*: sowohl im OMT-Objektmodell, als auch in der ER-Modellierung repräsentiert er den Zahlenwert einer Richtlinienregelung. Andererseits ist er im ER-Modell gleichzeitig der Vertreter einer Richtlinienregelung, wie wir sie im OMT-Objektmodell vorfinden.

Auf den ersten Blick konzentriert sich die ER-Modellierung sternförmig auf den Gegenstandstyp `Richtlinienwert`, der im Zentrum steht. Den Kern des OMT-Objektmodells bilden dagegen die Objektklassen `Richtlinie` und `Richtlinienregelung`, sowie die zwischen ihnen bestehende Assoziation. Dies teilt die OMT-Modellierung in zwei Teile:

1. in die Klassen und Assoziationen, die sich um die Klasse `Richtlinie` sammeln, und
2. in die Bestandteile des Aggregats `Richtlinienregelung`.

Diese Zweiteilung kann man auch im ER-Modell wiederfinden, wenn sie auch nicht so offensichtlich ist. Dazu betrachte man den Gegenstandstyp `Richtlinienwert` als Repräsentant einer Richtlinienregelung, den Gegenstandstyp `Herkunft` als Analogon zu `Richtlinie` und den Beziehungstyp `ist Herkunft` als Pendant zur Assoziation zwischen den OMT-Klassen `Richtlinie` und `Richtlinienregelung`, so daß sich folgende Zusammenhänge ergeben:

- Teil 1 findet sich in der linken unteren Ecke des ER-Diagramm wieder. Die ER-Modellierung geht nicht auf eine Unterscheidung zwischen EG- und Nicht-EG-Richtlinien ein, die ihre Ursache darin hatte, daß das OMT-Objektmodell die unterschiedlichen Publikationsweisen (siehe Klassen `Amtsblattreferenz` und `Veröffentlichung`) explizit modelliert. Kleine, aber wohl grundsätzlich unbedenkliche Abweichungen ergeben sich weiterhin hinsichtlich der Modellierung von Änderungsbeziehungen, Institutionen, Zeitdaten (im

OMT-Objektmodell als Attribute), sowie der Anbindung des Gültigkeits-/Gebietsbereichs an eine Richtlinie.

- Teil 2, der im OMT-Objektmodell aus dem Richtlinienregelungsaggregat besteht, wird – in einer groben Betrachtungweise – durch den Rest der ER-Modells repräsentiert. Die Gegenstandstypen `Klassifikation`, `Schutzobjekt` und `Einleitungsobjekt` haben kein Pendant im OMT-Objektmodell. Dies liegt u.a. daran, daß die ER-Modellierung einerseits unter Vorgaben der Landesanstalt für Umweltschutz (LfU) Baden-Württemberg stattfand und andererseits neben den Richtlinien für Luft, Wasser und Boden auch noch andere, wie beispielsweise für Abfall, Lärm u.ä., berücksichtigt hat, auf deren Modellierung wir aus schon genannten Gründen verzichtet haben. Die übrigen Gegenstandstypen finden sich entweder direkt oder indirekt im Aggregat `Richtlinienregelung` im Objektmodell wieder: `Parameter`, `Emissionsquelle`, `Dimension` (`Maßeinheit`), `Bezug` (als Attribut innerhalb der Klasse `Maßeinheit`), `Meßmethode`, `Werttyp` (`Richtlinienwertname`), `Zeitart` (`Wertart`), `Zeitdauer` (innerhalb von `Wertart` als Erläuterung), `Bemerkung`; die Gegenstandstypen `Probeentnahme`, `Randbedingung` und `Meßvariante` sind im Objektmodell in Anlehnung an die Grenzwertbroschüre [LR92] in ihrer Funktion als erläuternde Anmerkung in der Klasse `Bemerkung` zu halten. Die beiden Beziehungstypen `gültig ab` und `gültig bis` sind durch die Objektklasse `Bezugszeitraum` dargestellt.

Die ER-Modellierung wurde in das Relationenmodell nach folgendem Schema umgesetzt[4]:

- Alle Gegenstandstypen mit Ausnahme von `Datum` wurden in Relationen umgesetzt.
- `Datum` wurde als Feld den Relationen der Gegenstandstypen zugeschlagen, zu denen `Datum` in einer Beziehung steht (`Herkunft` und `Richtlinienwert`).
- Alle Beziehungstypen, die von `Richtlinienwert` mit einer $< 0,1 >$ oder $< 1,1 >$ Kardinalität ausgehen, wurden als Felder der Relation `Richtlinienwert` beschrieben.
- In gleicher Weise wurden andere Beziehungstypen mit $< 0,1 >$ oder $< 1,1 >$ Kardinalitäten als Felder innerhalb anderer Relationen umgesetzt.
- Alle anderen Beziehungstypen (`ist Schutzobjekt`, `ist Meßmethode`, `geändert durch`, `ist synonym`, `klassifiziert als`, `gehört zu` und `produziert`) wurden als eigenständige Relationen beschrieben.

Zusammenfassung

In beiden Modellierungen finden sich dieselben Strukturen wieder, auch wenn bei einer detaillierten Betrachtung durchaus unterschiedliche Entwurfsentscheidungen zu entdecken sind. Wir wollen auf eine derartige Diskussion aus folgendem Grund verzichten:

Aufgrund der Verwandtschaft der beiden Modellierungstechniken (OMT und ER) kommen auf der Ebene der statischen Modellierung nur Unterschiede zum Tragen, die auf unterschiedliche

[4] Diese Vorgehensweise entspricht (mit Ausnahme der Umsetzung des Gegenstandstyps `Datum`) der standardmäßigen Transformation eines ER-Diagramms in das relationale Modell. Sie wird beispielsweise in [DR90] beschrieben.

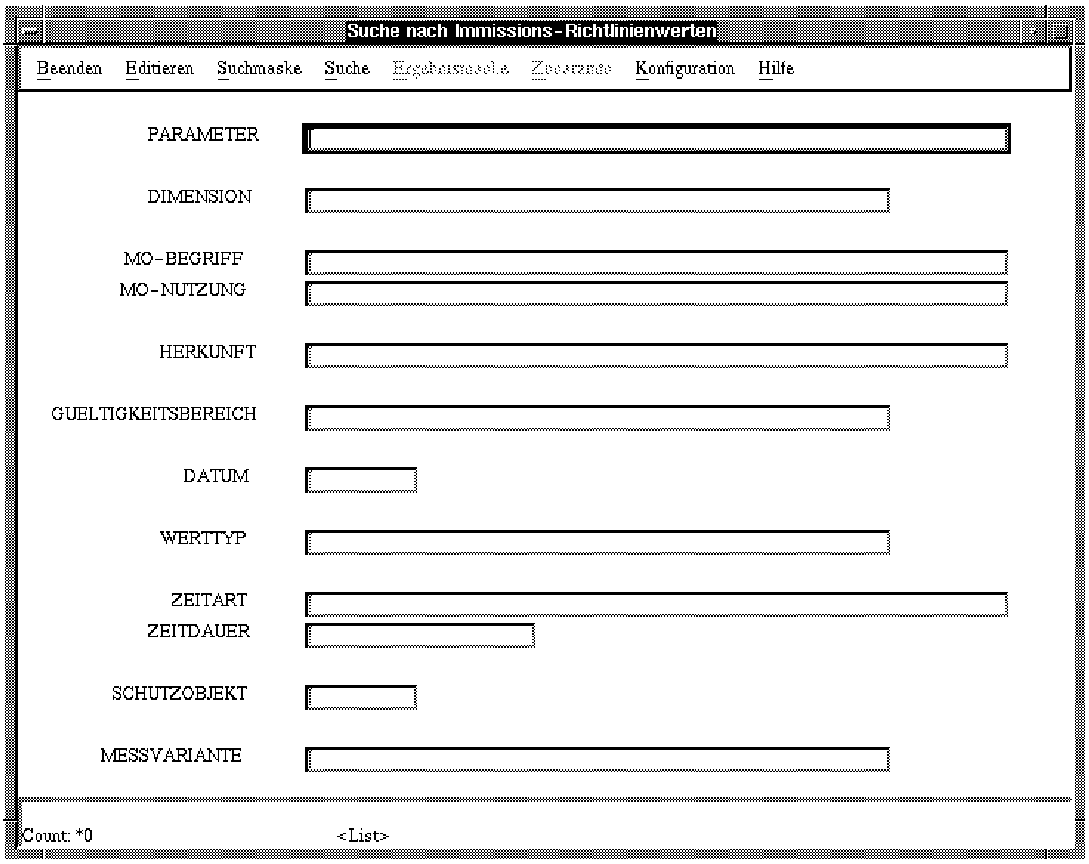


Abbildung 10.3: Maske zur Suche nach Immissions-Richtlinienwerten

Entwurfsentscheidungen und Sichtweisen auf die Diskurswelt zurückzuführen sind. Wir verzichten aber auf eine Diskussion dieser Unterschiede, da wir vielmehr die Charakteristika einer OPAQUE-Modellierung im Vergleich zu einer relationalen Modellierung ergründen möchten. Es soll also um einen Vergleich der Modellierungseigenschaften zweier logischer Datenmodelle (OPAQUE und Relationenmodell) gehen und nicht um das Für und Wider bestimmter Entwurfsentscheidungen. Um den genannten Vergleich ziehen zu können, müssen wir in den vorgelegten Entwürfen differenzieren können, welche Unterschiede auf die unterschiedlichen statischen Modellierungen und welche auf die unterschiedlichen logischen Datenmodelle zurückzuführen sind.

10.1.3 Das Anfragemodell

Das Anfragemodell des FZI-Entwurfs sieht vereinfacht dargestellt so aus [HK93]:

- Ziel einer Anfrage sind grundsätzlich Richtlinienwerte, also Richtlinienregelungen in unserem Sinne. Dazu kann der Benutzer zwischen einer Maske für Immissions-Richtlinienwerte (siehe Abb. 10.3) und einer für Emissions-Richtlinienwerte auswählen, die um die emissionsspezifischen Attribute erweitert ist.
- Der Benutzer füllt die ihm bekannten Suchattribute aus. Dabei wird er durch das System unterstützt: für jedes Attribut kann man über eine Hilfsfunktion die zulässigen Werte

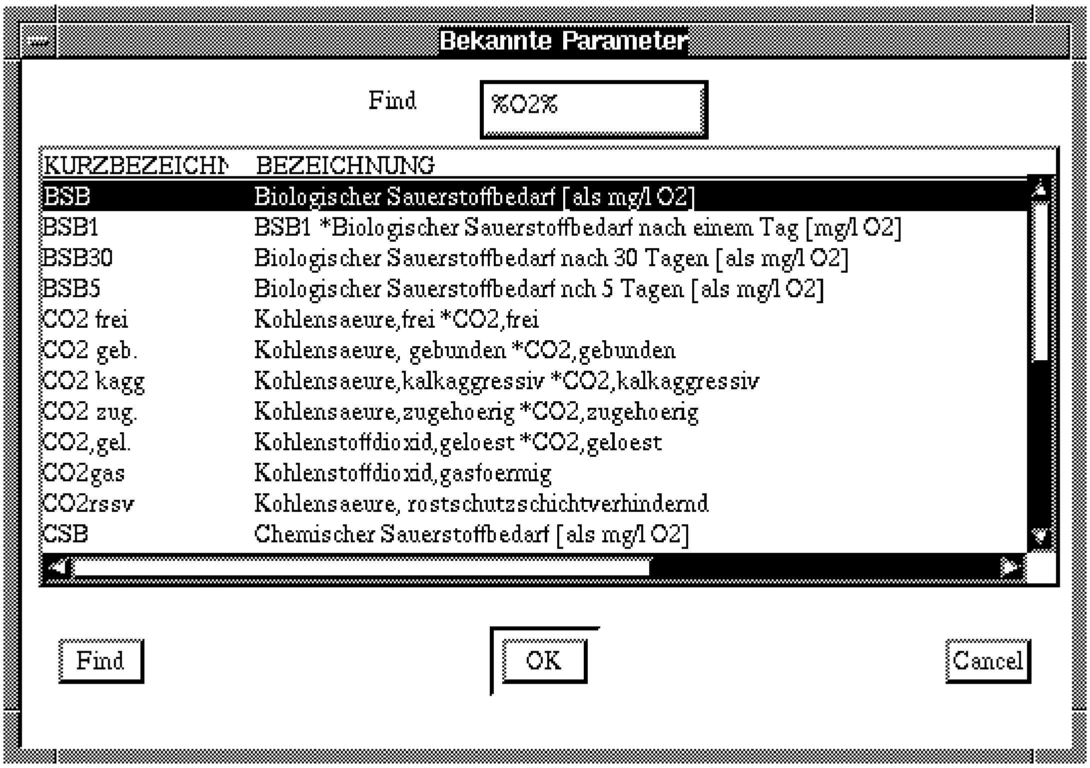


Abbildung 10.4: Eine Werteliste für das Suchattribut `Parameter`

abrufen. Abb. 10.4 zeigt eine Liste von Werten für `Parameter`, die in irgendeiner Form die Zeichenkette 'O2' enthalten.

- Die eingetragenen Suchattribute werden an eine Suchprozedur übergeben.

Die Suchattribute werden mit einem logischen UND verknüpft; ODER-Verknüpfungen sind nicht vorgesehen, da sie für die Umsetzung der Anforderungen der LfU nicht notwendig waren.

10.1.4 Das Suchmodell

Das Suchmodell besteht aus den folgenden Teilen [Kra93b]:

- einer Prozedur `gesamt_suche`, die als Eingabeparameter u.a. sämtliche in den Anfragemasken auftretenden Suchparameter erhält.

- Prozeduren `suche_parameter`, `suche_herkunft` etc., die die Teilsuchen nach Tupeln der jeweiligen Relationen übernehmen und als Eingabeparameter die jeweils relevanten Suchparameter erhält. Diese Suchparameter werden mit einem logischen UND verknüpft.

  Die einzelnen Werte der Suchattribute werden in vielen Fällen auf mehrere Attribute der Relationen angewandt. So wird beispielsweise der Wert des Suchattributs `Parameter` mit den Attributen `Kurzbezeichnung` und `Bezeichnung` der Relation `Parameter` verglichen. Umgekehrt werden nicht alle Attribute einer Relation für die Suche verwendet.

Diese Abbildung von Suchattributen auf Attribute der Relationen ist fest innerhalb der Suchprozeduren implementiert.

Es gibt nur etwa für die Hälfte der im ER-Modell auftretenden 22 Gegenstandstypen eine Suchprozedur. Dies hat seine Ursache darin, daß zahlreiche Gegenstandstypen, wie beispielsweise `Bemerkung`, `Dimension` oder `Probeentnahme`, theoretisch zwar zur Suche nach Richtlinienwerten (bzw. Richtlinienwertregelungen in unserem Sinne) herangezogen werden könnten, in der Praxis aber ausschließlich für die Ausgabe der in ihnen enthaltenen Information wichtig sind.

Ein Suchprozeß läuft wie folgt ab: die Prozedur `gesamt_suche` ruft die gegebenen Teilsuchprozeduren in einer festen Reihenfolge auf. Ergebnis einer Suche ist eine Menge von Richlinienwert-Identifikatoren, die sich aufgrund der gegebenen Suchattribute für die Beantwortung der Anfrage qualifiziert haben.

Es gibt mehrere Möglichkeiten, die Suche den benutzerspezifischen Erfordernissen anzupassen:

- Der Benutzer kann bei jeder Anfrage optional angeben, ob synonyme Parameter, abändernde Herkünfte (Richtlinien), Richtlinienwerte für ganze Stoffgruppen etc. bei der Suche automatisch mitberücksichtigt werden sollen.

- Benutzerspezifisch können Ähnlichkeitsbeziehungen zwischen Attributwerten definiert werden. Dies geschieht im wesentlichen einmal für einen Benutzer. Diese Ähnlichkeitsbeziehungen werden danach bei jeder Anfrage dieses Benutzers berücksichtigt.

  Die Ähnlichkeit zweier Werte wird durch einen Wert zwischen 0 und 1 beschrieben. Haben nun die Begriffe A und B eine Ähnlichkeit von 0.6 (für einen bestimmten Benutzer), so bedeutet dies, daß bei einer von diesem Benutzer gestellten Anfrage, die in einem Suchattribut den Wert A enthält, auch Tupel zur Lösung hinzugenommen werden, falls sie den Wert B im jeweiligen Suchattribut enthalten. Dadurch erhalten wir in der Ergebnismenge zwei Arten von Tupeln:

  1. Tupel, die alle in der Anfrage aufgestellten Charakteristika aufweisen und deshalb *jedem* Benutzer, der diese Anfrage stellt, als Antwort zurückgeliefert würden, und
  2. Tupel, die aufgrund von Ähnlichkeitsbeziehungen zur Ergebnismenge hinzugenommen wurden. Diese Tupel bilden den benutzerspezifischen Anteil der Antwort auf die Anfrage.

  Eine Suche liefert *immer* beide Arten von Tupeln. Den Tupeln der Art 1 wird eine Gesamtähnlichkeit von 1 zugemessen, während sich für die Tupel der Art 2 die Gesamtähnlichkeit als Produkt der Ähnlichkeiten ergibt, die bei der Suche dazu geführt haben, daß sie zur Lösung hinzugenommen wurden. Die Ergebnisrelation wird dann vor der Ausgabe sortiert (siehe unten).

  Die Definition von benutzerspezifischen Ähnlichkeitsbeziehungen soll helfen, die Probleme zu reduzieren, die aufgrund der von Richtlinie zu Richtlinie abweichenden Terminologie entstehen können (vgl. dazu Kapitel 4). Eine zusammenfassende Beschreibung dieses Suchmodells findet der interessierte Leser in [KS93].

Wir finden in diesem Suchmodell Ideen und Konzepte wieder, die wir im ersten dynamischen Entwurf (Kapitel 7) verwirklicht haben:

- Ein Suchprozeß gliedert sich in Teilsuchen.
- Pro Gegenstandstyp/Relation (entspricht einer Objektklasse in OMT bzw. OPAQUE) existiert prinzipiell eine eigene Suchoperation, es sei denn, man betrachtet eine Relation als für die Suche nicht relevant.
- Der grundlegende Algorithmus der Prozeduren für die Teilsuchen ist identisch.

Andererseits entfällt die in Kapitel 7 notwendige Koordination der Teilsuchen zu einer Gesamtsuche, da das Ziel einer Anfrage immer Richtlinienwerte (bzw. Richtlinienregelungen im Sinne unserer Modellierung) sind. Dadurch kann die Reihenfolge der Teilsuchen fest vorgegeben werden.

10.1.5 Das Ausgabemodell

Mit Hilfe der Prozedur `erzeuge_ergebnis` kann die Menge an Richtlinienwert-Identifikatoren, die der Suchprozeß zurückliefert, in eine Relation `Anfrageergebnis` umgewandelt werden. Man kann aus dieser Relation die Ergebnistupel abrufen und in Form einer Ergebnismaske ausgeben lassen (siehe Abb. 10.5 aus [HK93]). Die Reihenfolge der Tupel in der Ergebnisrelation unterliegt einer Sortierung nach den oben beschriebenen Gesamtähnlichkeiten. So sind Tupel, die nur aufgrund von Ähnlichkeitsbeziehungen gefunden wurden, weiter hinten angeordnet als solche, die sich direkt für die Lösung qualifiziert haben.

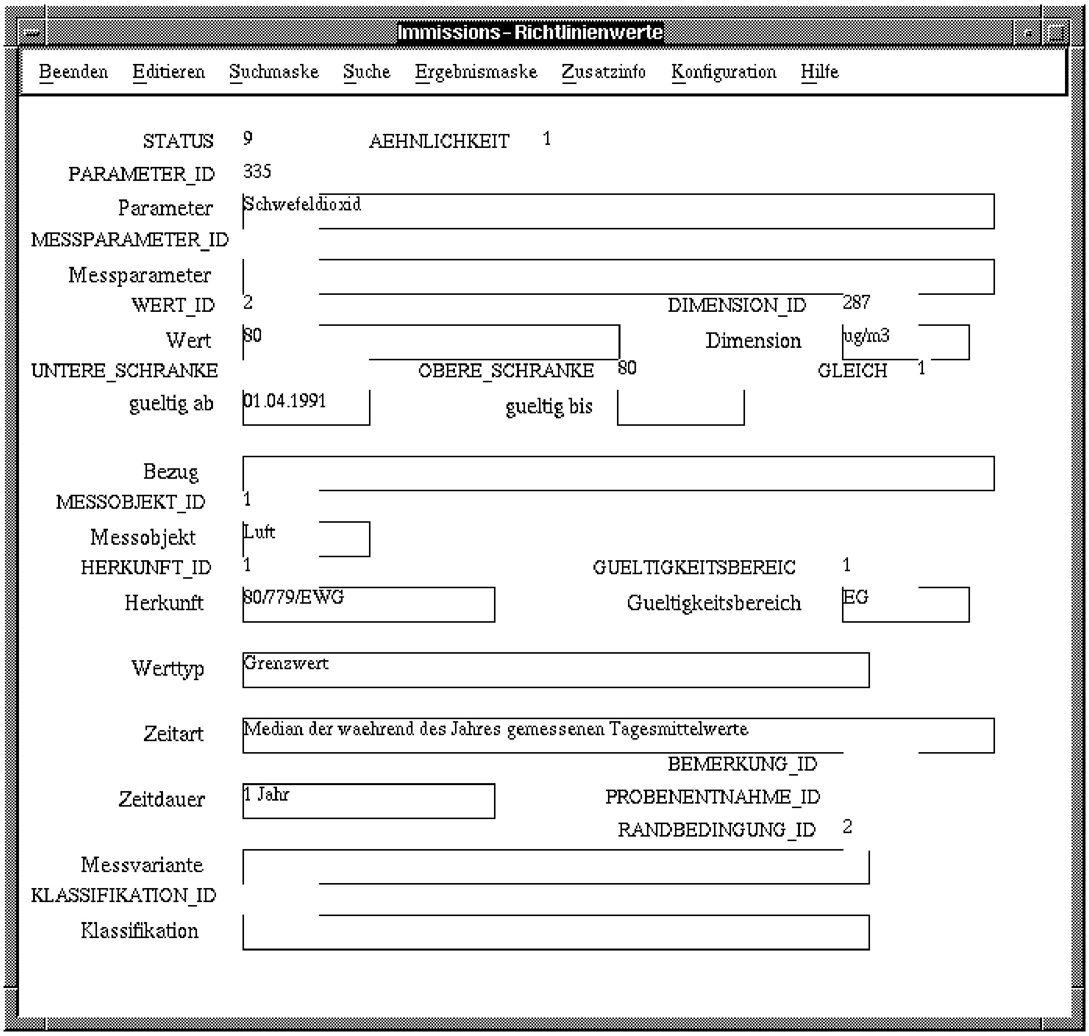


Abbildung 10.5: Ergebnismaske der Immissionssuche

10.2 Vergleichende Bewertung

In diesem Abschnitt geben wir eine kurze Bewertung des relationalen Entwurfs. Diese Bewertung soll in Anbetracht der Eigenschaften erfolgen, die eine objektorientierte und kontextsensitive Modellierung hat, und ist deshalb relativ zu dieser zu sehen. Die Vorgehensweise ist die folgende:

- In Abschnitt 10.2.1 ziehen wir Parallelen zum ersten dynamischen Entwurf aus Kapitel 7, da dessen Vor- und Nachteile bereits ausführlich diskutiert wurden und in einigen wesentlichen Gesichtpunkten auch für den vorgestellten relationalen Entwurf gelten.
- Abschnitt 10.2.2 bewertet den relationalen Entwurf anhand der von uns aufgestellten Anforderungskriterien.
- In Abschnitt 10.2.3 untersuchen wir die Flexibilität und Anpaßbarkeit des relationalen Entwurfs.

10.2.1 Parallelen zum ersten dynamischen Entwurf

Wie wir in Abschnitt 10.1 schon angedeutet haben, zeigt der vorgelegte relationale Entwurf in seinen dynamischen Aspekten viele Parallelen zu dem von uns in Kapitel 7 entwickelten ersten dynamischen Entwurf, die wir an dieser Stelle noch einmal anführen möchten:

- Eine Suche setzt sich aus Teilsuchen zusammen.
- Pro Gegenstandstyp existiert prinzipiell[5] eine eigene Suchoperation.
- Der grundlegende Algorithmus dieser Suchoperationen ist identisch.

Andererseits entfällt die in unserem ersten dynamischen Entwurf problematische Koordination der Teilsuchen zu einer Gesamtsuche. Diese hatte ihre Ursache darin, daß dem Benutzer die grundlegende Möglichkeit gegeben werden sollte, alle im Objektmodell vorhandenen Informationen abrufen zu können, wie beispielsweise alle für einen Gebiets-/Gültigkeitsbereich geltenden Richtlinien. Hierbei stellen die Richtlinien das Ziel der Anfrage dar, so daß als Ergebnis der Anfrage in beiden dynamischen Entwürfen eine Menge von Richtlinienobjekten zurückgeliefert wird.

Da das Ziel des Suchprozesses fixiert ist, kann die Reihenfolge der notwendigen Teilsuchen apriori festgelegt und fest implementiert werden. Es entfällt also die Notwendigkeit, die Reihenfolge der Teilsuchen anfrageabhängig vom Benutzer bestimmen zu lassen. Dies führt auch zu einer Vereinfachung des Anfragemodells und der Benutzerschnittstelle.

Andererseits bleiben die bereits in Zusammenhang mit dem ersten dynamischen Entwurf erwähnten Probleme einer umfangreichen, sich wiederholenden Implementierung der Suchoperation und die doppelte Implementierung der Datenstruktur in den Relationen und den Suchoperationen (vgl. Abschnitt 7.7).

[5] Wie bereits erwähnt, kann man einige Gegenstandstypen als für die Suche irrelevant betrachten und daher für sie keine eigene Suchoperation zu implementieren.

10.2.2 Eigenschaften des relationalen Entwurfs

In Abschnitt 4.2 haben wir die möglichen Benutzergruppen des Umweltinformationssystems identifiziert. Dies geschah einerseits aufgrund unterschiedlichen Anfrageverhaltens und andererseits aufgrund einer zu differenzierenden Erwartung bezüglich der Aufbereitung der gefundenen Informationen. Wir wollen nun erläutern, inwieweit der relationale Entwurf diesen Anforderungen gerecht wird.

Anfrageverhalten: Durch die Vereinfachung des Suchmodells (siehe Diskussion in Abschnitt 10.2.1) ist es gelungen, dessen prozeduralen Charakter hinter einem deklarativen Anfragemodell zu verbergen. Dadurch ist der Benutzer von der Verantwortung entbunden, den Suchprozeß selbst zu koordinieren.

Der unerfahrene Benutzer wird unterstützt, indem seinem persönlichem Wissensstand entsprechende Ähnlichkeiten zwischen Begriffen bzw. allgemein zwischen Attributwerten definiert werden können. Das bewirkt, daß nach Werten gesucht wird, aus der Sicht des Benutzers als ähnlich betrachtet werden können. Dadurch erhält der weniger erfahrene Benutzer möglicherweise trotzdem die von ihm gesuchten Informationen. Auf diese Art und Weise kann dem unsicheren Anfrageverhalten der Benutzer Rechnung getragen werden.

Ein Problem stellt aber die Aufstellung der Ähnlichkeitsbeziehungen dar:

- Gerade der unerfahrene Benutzer kann die Vielfalt der verwendeten Begriffswelt nicht übersehen. Daher ist es fraglich, ob er überhaupt eine hinreichend vollständige Liste von Ähnlichkeitsbeziehungen erstellen kann, da er beispielsweise von der Existenz bestimmter Begriffe gar nichts weiß.
- Ein dafür eventuell hinzugezogener Fachmann hat es wiederum schwer, eine eher 'naive' Sichtweise auf sein Fachgebiet einzunehmen. Für ihn stellen sich die Fragen, welche Ähnlichkeitsbeziehungen für den speziellen Benutzer relevant sein und mit welchem Zahlenwert die eher subjektiv zu betrachtende Ähnlichkeit zwischen zwei Attributwerten zu bemessen ist.

Problematisch ist auch, daß die Ähnlichkeiten grundsätzlich bei jeder Anfrage des Benutzers berücksichtigt werden, unabhängig davon, ob die Anfrage so präzise formuliert ist, daß die vom Benutzer tatsächlich vorgegebenen Suchwerte bereits ausreichen, um Ergebnistupel zu finden. Das bedeutet, daß im Falle einer präzisen Anfrageformulierung mit einer großen Menge an 'exakt' gefundenen Ergebnistupeln noch eine große Menge von Tupeln hinzukommen kann, die aufgrund von Ähnlichkeitsbeziehungen gefunden werden. Dies führt neben der Überhäufung des Benutzers mit eventuell irrelevanten[6] Daten zu einer Verlangsamung der Bearbeitung durch das Datenbanksystem.

Informationsaufbereitung: Das Ausgabemodell differenziert nicht zwischen unterschiedlichen Benutzergruppen und gibt alle, einen Richtlinienwert betreffenden Informationen aus. Durch die Sortierung der Ergebnistupel anhand der errechneten Ähnlichkeiten wird erreicht, daß potentiell wichtigere Tupel zu Beginn der Ergebnisrelation stehen.

[6] Diese Irrelevanz muß *vom Benutzer* nachträglich festgestellt werden, indem er die Ergebnismenge dahingehend untersucht.

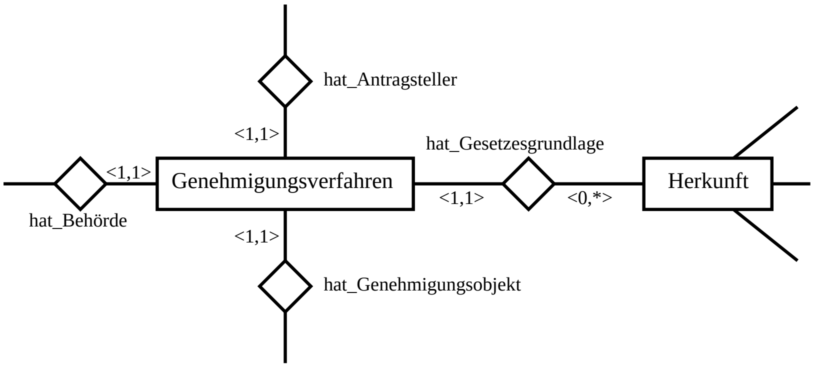


Abbildung 10.6: Erweiterung des Entity-Relationship-Diagramms

```
CREATE TABLE Genehmigungsverfahren (
   Verfahren_ID            NUMBER(6)     ...
   Beginn                  DATE          ...
   Ende                    DATE          ...
   Status                  CHAR(10)      ...
   Gesetzesgrundlage_ID    NUMBER(6)     ...
   Behoerde_ID             NUMBER(6)     ...
   Antragsteller_ID        NUMBER(6)     ...
   Genehmigungsobjekt_ID   NUMBER(6)     ...
)
```

Abbildung 10.7: Die Relation `Genehmigungsverfahren`

10.2.3 Erweiterbarkeit

Wir haben in Abschnitt 9.1 die Flexibilität des OPAQUE-Entwurfs gegenüber Änderungen und Erweiterungen in der Datenbasis untersucht. Wir greifen hier noch einmal eines der Beispiele auf um zu sehen, welche Auswirkungen sich für das Anfrage-, das Such- und das Ausgabemodell des relationalen Entwurfs ergeben.

Wir verwenden dazu das bereits in Abschnitt 9.1 erwähnte Beispiel der Anbindung einer Datenbasis, die die Daten über Genehmigungsverfahren enthält. Im OPAQUE-EM-Objektmodell war es dazu notwendig, eine neue, zusätzliche Kontextschablone einzuführen. Ebenso einfach läßt sich das ER-Diagramm aus Abb. 10.1 für diese Zwecke erweitern: dazu führt man einen neuen Gegenstandstyp `Genehmigungsverfahren` ein, der in Beziehung zu `Herkunft` steht. Abb. 10.6 zeigt diese Erweiterung. Die Erweiterung des ER-Modells schlägt sich in einer neuen Relation nieder (siehe Abb. 10.7).

Dies hat die folgenden Auswirkungen:

Anfrage- und Suchmodell: Eine Schemaänderung hätte zur Folge, daß zur Aufnahme neuer Suchattribute die Anfragemasken (siehe z.B. Abb. 10.3) und der Programmcode der Prozedur `gesamt_suche` angepaßt und neu übersetzt, sowie eine Suchoperation für die neue Relation `Genehmigungsverfahren` implementiert werden müßte.

Da das Anfrage- und das Suchmodell aber allein auf die Anfrage und Suche nach Richtlinienwerten ausgelegt sind, ist eine Erweiterung der Datenbasis insofern problematisch, als daß sich neue Anfrage- und damit auch Suchkonstellationen ergeben, die von den fest implementierten Anfragen erfaßt werden. Man denke an das oben genannte Beispiel der Genehmigungsverfahren: würde man nach allen Genehmigungsverfahren einer bestimmten Richtlinie fragen, so würde der Suchprozeß für jedes mit dieser Richtlinie in Beziehung stehende Genehmigungsverfahren *alle* Richtlinienwerte der Richtlinie zurückliefern.

Die Ursache für dieses Verhalten liegt darin, daß der relationale Datenbankentwurf gezielt auf den Abruf von Richtlinienwerten ausgerichtet war und dadurch sowohl das Anfrage- als auch das Suchmodell auf diesen Zweck ausgerichtet wurden. Dies zog viele, insbesondere für den späteren Benutzer angenehme Vereinfachungen mit sich (siehe Abschnitt 10.2.2), führt aber im Falle einer Erweiterung der Datenbasis und einer damit verbundenen Erweiterung der gewünschten Funktionalität (also z.B. der Anfrage nach Genehmigungsverfahren zu einer Richtlinie) zur Notwendigkeit, das Suchmodell und damit vermutlich auch das Anfragemodell neu konzipieren zu müssen.

Ausgabemodell: Ebenso wie beim Anfragemodell kann vordergründig eine Erweiterung der Ausgabemaske (siehe z.B. Abb. 10.5) genügen. Aber auch das Ausgabemodell ist allein auf die Ausgabe von Richtlinienwerten ausgelegt, was beispielsweise zur Vereinfachung geführt hat, daß nur Ausgabemasken für Immissions- und Emissionsrichtlinienwerte notwendig waren. Auch hier wäre im Falle einer Erweiterung der Datenbasis und der Funktionalität des Systems eine deutliche Erweiterung des Ausgabemodells notwendig.

Wie man sieht, schlägt eine Erweiterung des Schemas auf beinahe alle Komponenten des Datenbanksystems durch. Größere Schemaerweiterungen könnten sogar dazu führen, daß einige Komponenten neu entworfen werden müßten. Dies widerspricht der Flexibilität und Anpaßbarkeit, die wir aufgrund der sich ständig ändernden Bedingungen und Anforderungen angestrebt haben.

Kapitel 11

Zusammenfassung und Ausblick

11.1 Zusammenfassung

Wir haben in dieser Arbeit einen Entwurf einer Richtlinien-Datenbank für ein Umweltinformationssystem auf Grundlage des Kontextmodells OPAQUE entwickelt. Die besondere Herausforderung bestand

- in der Komplexität des Anwendungsszenarios,
- in der Diversität der Benutzer, die zu berücksichtigen waren, und
- in der Forderung nach Flexibilität des Entwurfs hinsichtlich Änderungen, Erweiterungen und Einbindung in ein UIS.

Zunächst haben wir einen statischen Entwurf angefertigt. Dabei hat sich gezeigt, daß in der Modellierung der statischen Strukturen der Diskurswelt keine besonderen Unterschiede zwischen einer Modellierung auf Basis von OPAQUE und der auf Grundlage anderer Datenmodellen ergaben. Herkömmliche Entwicklungsmethoden, wie OMT oder das ER-Modell, waren ausreichend, um die statischen Aspekte zu beschreiben.

Die dynamischen Aspekte der Anwendung haben wir in zwei Entwürfen modelliert: einerseits in einem konventionellen, prozeduralen Entwurf, und andererseits in einem deskriptiven Entwurf.

Die Vorzüge des ersten Ansatzes liegen in seiner einfachen, geradlinigen Entwicklung. Andererseits ergaben sich auch erhebliche Redundanzen: so war beispielsweise die statische Struktur nicht nur in der Schemadefinition, sondern auch innerhalb der Suchoperationen implementiert, da diese Operationen diese Struktur kennen mußten. Dies führte zum einen zu einer umfangreichen Implementierung und zum anderen zu mangelnder Flexibilität hinsichtlich von Änderungen und Erweiterungen des Schemas. Weiterhin wurden in diesem Entwurf nur sehr wenige der besonderen Konzepte OPAQUEs ausgenutzt, so daß dieser Entwurf ohne große Änderungen beispielsweise auch in einem herkömmlichen objektorientierten oder gar relationalen Datenbanksystem implementiert werden könnte. Dies unterstreicht auch der später durchgeführte Vergleich zum relationalen Entwurf des FZI, wo sich zahlreiche Parallelen ergaben.

Der zweite dynamische Entwurf wirkte den Defiziten des ersten gezielt entgegen. Dies geschah einerseits dadurch, daß ein deskriptiver Ansatz gewählt wurde, und andererseits dadurch, daß die dynamischen Konzepte OPAQUEs ausgenutzt wurden. So wurde ein deskriptives Anfragemodell, ein Such- und ein Ausgabemodell entworfen, die allesamt strukturunabhängig waren. Erweiterungen und Änderungen am Schema fügen sich in diese Konzeption ein, ohne daß diese Änderungen der dynamischen Komponenten nach sich ziehen. Darüberhinaus bot das Suchmodell die Möglichkeit, gruppen- und sogar benutzerspezifische, individuelle Suchprozeduren zu definieren. Damit konnte den Anforderungen der unterschiedlichen Anwender Rechnung getragen werden.

Schließlich haben wir den Entwurf anhand mehrerer Beispiele hinsichtlich Anpaßbarkeit und Flexibilität untersucht und ihn mit einem Entwurf verglichen, der auf Grundlage des relationalen Datenmodells am FZI Karlsruhe erstellt wurde. Hier zeigten sich insbesondere die Vorteile des OPAQUE-Entwurfs in der Modellierung der benutzerspezifischen Aspekte und in den Möglichkeiten, Änderungen und Anpassungen einzuarbeiten, wohingegen die statischen Aspekte des Szenarios in beiden Entwürfen gleichermaßen erfaßt wurden.

Der Entwurf hat gezeigt, daß die Vorteile OPAQUEs im Bereich der dynamischen Modellierung liegen. Dies ist insbesondere auf die Konzeption zurückzuführen, daß es möglich ist, die Zustands- und Verhaltensbeschreibung auf mehrere Kontexte zu verteilen und danach eine Gesamtreaktion aus den Einzelreaktionen der Kontexte zu generieren. Erweiterungen oder Änderungen an der Funktionalität eines Objektes können in den meisten Fällen dadurch realisiert werden, daß ein oder mehrere Kontexte, die die gewünschte zusätzliche oder veränderte Funktionalität beinhalten, dem Objekt einfach hinzugefügt werden. In herkömmlichen imperativen oder objektorientierten Modellen ist ein solches Hinzufügen prinzipiell natürlich auch möglich. In den meisten Fällen betrifft dies jedoch den Code oder die Schnittstellen bereits implementierter Prozeduren, was zahlreiche Folgeänderungen in anderen Prozeduren zur Folge haben kann[1]. Kann der Entwurf so gestaltet werden, daß die Interaktion verschiedener Kontexte, also das Generieren einer Gesamtreaktion eines Objektes (auf eine Nachricht) aus den Teilreaktionen der am Objekt beteiligten Kontexte, bereits von Anfang an berücksichtigt wird, so beschränken sich die meisten Änderungen darauf, zusätzliche Kontexte dem System hinzuzufügen, die die zusätzliche Funktionalität enthalten.

11.2 Ausblick

Die Vorteile OPAQUEs liegen, wie beschrieben, in seinen dynamischen Konzepten. Wie man anhand der beiden in dieser Arbeit vorgestellten dynamischen Entwürfe sieht, kommt es darauf an, den Entwurf in einer Art und Weise zu gestalten, so daß die besonderen Konzepte OPAQUEs ausgenützt werden können. Der erste dynamische Entwurf konnte das nicht. Das bedeutet, daß die Güte des Entwurfs in besonderer Weise von der Phase der dynamischen Modellierung abhängt. Es wäre wünschenswert, an dieser Stelle eine Entwurfsmethodik zu besitzen, die diese Entwurfsphase unterstützt.

Weiterhin wäre eine Implementierung des Entwurfs mit Hilfe des existierenden Prototyps des OPAQUE-Systems anzustreben. Damit könnte der Entwurf anhand realer Daten und von Benutzern getestet werden, um ganzheitliche praktische Auswertungsergebnisse zu erhalten und

[1] Ein Beispiel dafür ist der in dieser Arbeit vorgestellte erste dynamische Entwurf oder auch der relationale Entwurf des FZI mit den zahlreichen Suchprozeduren, auf die sich Änderungen am Schema unmittelbar auswirken. Beispielsweise können sich die Prozedurschnittstellen ändern, da plötzlich neue Suchparameter relevant werden.

die Güte des Entwurfs in der praktischen Anwendung beurteilen zu können.

Sollte sich der Entwurf in der Praxis bewähren, so wäre es denkbar, insbesondere den deskriptiven dynamischen Entwurf zu verallgemeinern und für andere Retrievalsysteme einzusetzen, die komplexen Benutzeranforderungen gerecht werden müssen. Dies sollte keine großen Probleme bereiten, da dieser dynamische Entwurf so konzipiert wurde, daß Anfrage-, Such- und Ausgabemodell strukturunabhängig sind.

Anhang A

Glossar

Diese zusammenfassende Darstellung der zahlreichen Begriffe, die in den Richtlinien und in dieser Arbeit verwendet werden, soll dem Leser einen Überblick geben und ihm die Lektüre erleichtern.

Analytik: siehe Meßmethode

Analyseverfahren: siehe Meßmethode

Bezugszeitraum: Ein Bezugszeitraum ist die Zeitperiode, während der die Richtlinienregelung Gültigkeit besitzt. Dieses Feld existiert nur in den EG-Luftqualitätsnormen. Beispielsweise gilt eine besondere Regelung für SO_2 während des Winters (1.10. – 31.3.).

Eingreifrichtwert: Richtwert, der Maßnahmen z.B. für eine Sanierung oder Genehmigung erfordert.

Emission: Ausstoß von Substanzen aus einer Emissionsquelle wie z.B. Auto oder Verbrennungsanlagen.

Emissionsquelle: Emissionsrichtlinien geben Regelungen für den Ausstoß von Schadstoffen in Zusammenhang mit einer Emissionsquelle an. Beispiele für Emissionsquellen sind

- Verbrennungsanlagen,
- Motoren,
- Produktionsstätten, die bestimmte Schadstoffe verwenden oder ausstoßen,
- Benzin und andere Brennstoffe.

Genehmigungswert: Grenzwert, der für die Genehmigung z.B. einer Abwassereinleitung erforderlich ist.

Grenzwert: Gesetzlich festgelegter Höchstwert für Schadstoffe, Strahlung und sonstige Emissionen

Immission: Einwirkung von Schadstoffen in der Luft, Strahlung, Lärm u.a. auf die Umwelt.

Leitwert: Qualitätsziel; kein gesetzlicher Wert.

Kontrollverfahren: siehe Meßmethode

Maßeinheit: Wir wollen den Begriff Maßeinheit in dem Sinne verwenden, wie er in den Naturwissenschaften üblich ist. Beispiele für Maßeinheiten sind

- mg/m^3
- g/h
- °C

Maßeinheitszusatz: Oftmals haben die Maßeinheiten einen Zusatz, der die Maßeinheit näher erläutert. Ein Beispiel ist der Zusatz *abgeleitetes Wasser* in

mg/l abgeleitetes Wasser

Meßmethode: Methode, um den Wert für einen Parameter zu messen. Mit einer Meßmethode bezeichnen wir die in den Richtlinien verwendeten Begriffe

- Analytik/Referenzmethode,
- Analytik/Referenzverfahren,
- Referenzmethode,
- Analysen- und Prüfungsverfahren,
- Analyse- oder Kontrollverfahren.

Beispiele für Meßmethoden sind die Black-Smoke-Methode, die gravimetrische Methode, chromatographische Verfahren etc.

Meßobjekt: Ein Meßobjekt ist ein Medium, für das Richtlinienregelungen definiert wurden. Beispielsweise gibt es das Meßobjekt Wasser,das sich noch weiter untergliedern läßt in Grundwasser, Oberflächenwasser, Badegewässer, Süßwasser usw. Wir beschränken uns auf die Betrachtung der Meßobjekte Luft, Wasser und Boden.

Parameter: Ein Parameter ist die Bezeichnung für eine Beobachtungsgröße in Zusammenhang mit einem Meßobjekt (wie beispielsweise der Luft), die beschreibt, inwieweit ein Biosystem gefährdet ist. Das ist z.B. eine Schadstoffkonzentration oder -menge oder eine physikalische Größe, wie pH-Wert, Temperatur, Leitfähigkeit, ..., oder ein anderer Indikator, der auf eine Gefährdung schließen läßt, wie Konzentration an Darmviren, Salmonellen etc. in Badegewässern.

Prüfwert: Höchstkonzentration eines Stoffes in einem Umweltmedium, die eine Prüfung für die Festlegung eines Grenzwertes erfordert.

Referenzmethode: siehe Meßmethode

Richtlinie: Unter einer Richtlinie verstehen wir Gesetze, Vorschriften, Leitlinien, Normen, Verordnungen, Leitfäden u.ä., die Regelungen über den Wert von Schadstoffkonzentrationen, physikalisch-chemischen und mikrobiologischen Parametern enthalten. Diese Richtlinien werden von verschiedenen Körperschaften, wie beispielsweise der Europäischen Gemeinschaft (EG), den Gesetzesgebern in Deutschland, der Schweiz, in Österreich, Verbänden, wie dem VDI[1], oder internationalen Organisationen, wie der WHO[2], erlassen oder herausgegeben.

Richtlinienwert: Wir wollen unter einem Richtlinienwert zunächst einmal die Wertinformationen verstehen, die in den Richtlinien unter den Begriffen, wie

- Grenzwert,
- Leitwert,
- Hintergrundwert,
- Prüfwert,
- Richtwert,
- Qualitätsziel,
- Höchstkonzentration,
- Konzentration,
- ...

enthalten ist. Diese unterschiedlichen Bezeichnungen kommen u.a. daher, daß sich unterschiedliche Verwendungszwecke hinter dem betreffenden Richtlinienwert verbergen: ein Grenzwert hat beispielsweise bindenden Charakter, während ein Leitwert eher eine Orientierungshilfe darstellt. Deshalb wird es später wichtig sein, daß man trotz der vereinheitlichten Bezeichnung immer noch den ehemals verwendeten Namen abfragen kann, da dieser noch zusätzliche Informationen enthält.

Dies ist nicht immer nur die gesetzliche Bindungskraft. Beispielsweise geben die EG-Wasserqualitätsnormen Werte für die Membranfiltermethode und die Mehrfachröhrenmethode an, die der Einfachheit halber ebenfalls mit Membranfiltermethode und Mehrfachröhrenmethode bezeichnet werden. Exakt ausgedrückt handelt es sich aber um Richtlinienwerte, die mit Hilfe einer bestimmten Meßmethode (s.u.) überprüft werden müssen. Daneben gibt es weitere Beispiele für Richtlinienwertbezeichnungen, die eine Zusatzinformation tragen, wie

- Konzentration im Abgas,
- [Konzentration] bei einem Massenstrom,
- Salmonidengewässer-Leitwert,
- A1-Leitwert[3],
- Risikowert für krebserzeugende Verunreinigungen.

Wir werden unter einem Richtlinienwert nur einen konkreten Zahlenwert[4] oder ein, möglicherweise nach einer Seite offenes Intervall[5] betrachten. Ein Zahlenwert bzw. ein Intervall

[1] Verein Deutscher Ingenieure

[2] World Health Organization

[3] Dieser Richtlinienwert stammt aus der EG-Wasserqualitätsnorm für Oberflächenwasser, das zur Trinkwassergewinnung verwendet wird. Dabei beschreibt A1 die Kategorie von Wasser, das mit einfachen physikalischen Mitteln (z.B. Schnellfilterung) aufbereitet und entkeimt wurde. Daneben existieren noch Kategorien A2 und A3.

[4] Beispielsweise 5 mg/m^3 für 1,2-Dibromethan im Abgas (TA Luft).

[5] Beispielsweise ≥ 25 g/h für 1,2-Dibromethan in einem Massenstrom (TA Luft).

macht ohne Maßeinheit natürlich keinen Sinn. Deshalb steht ein Richtlinienwert immer in Beziehung zu einer Maßeinheit.

Richtlinienwertart: Unter Richtlinienwertart wollen wir die verschiedenen Arten zusammenfassen, einen Meßwert mathematisch zu ermitteln, um ihn mit dem Richtlinienwert zu vergleichen. Es gibt

- Mittelwerte über 24 Stunden, 1 Monat, 1 Jahr,
- Mediane,
- 98%-Wert,
- ...

Richtwert: Im strengen Sinn gesetzlich nicht bindender Wert, der aber in der aber in der Praxis eingehalten werden soll bzw. der bei Überschreiten der Grenze Maßnahmen erfordert.

Schadstoff: Chemische Stoffe, die für Mensch, Tier und Pflanze schädlich oder auf ein Ökosystem ungünstig wirken.

Stoff: Schadstoff

Anhang B

Struktur der Richtlinien

Die untenstehenden Listen führen die Bezeichnungen der Tabellenköpfe der in der Grenzwert-Broschüre [LR92] enthaltenen Tabellen, die die Bestimmungen der verschiedenen Umweltrichtlinien enthalten. Die Listen zeigen also die Gliederung der Richtlinieneintragungen. Diese Gliederung ist ein wichtige Grundlage für die Modellierung der Zusammenhänge im Rahmen eines Datenbankentwurfs.

B.1 Richtlinien zum Meßobjekt Luft

EG-Richtlinien zur Luft-Emission

- Parameter/Schadstoff
- Emissionsquelle
- Grenzwertregelung
- Einzuhalten ab
- Analytik/Referenzmethode
- Richtlinien Nr., Titel

EG-Luftqualitätsnormen

- Parameter/Schadstoff
- Bezugszeitraum
- Regelung Grenzwert/Leitwert
- Einzuhalten ab
- Analytik/Referenzverfahren
- Richtlinien Nr., Titel

TA Luft (Luft-Immission)

- Stoff
- Konzentration im Abgas
- bei einem Massenstrom

Die genannten Eintragungen gibt es für die (disjunkten) Stoffklassen

- Krebserzeugende Stoffe
- Staubförmige anorganische Stoffe
- Dampf- oder gasförmige anorganische Stoffe
- Organische Stoffe

TA Luft (Luft-Emission)

- Art der Anlage
- geregelter Stoff
- Grenzwertregelung
- Bemerkung

WHO-Luftqualitätsleitlinien

- Parameter
- Leitwert für toxische Verunreinigungen
- Mittelwert über[1]
- Leitwert für geruchsintensive Verunreinigungen
- Wahrnehmungsschwelle
- Erkennungsschwelle
- Leitwert zum Schutz der Vegetation
- Mittelwert über[2]
- Risikowerte für krebserzeugende Verunreinigungen
- Tumorlokalisation

VDI-Richtlinie 2310 (MIK-BRD)

- Parameter
- Grenzwert
- Mittelwert über

[1] bezieht sich auf den Leitwert für toxische Verunreinigungen
[2] bezieht sich auf den Leitwert zum Schutz der Vegetation

Luftreinhalteverordnung Schweiz

- Parameter
- Grenzwert
- Mittelwert über

IUFRO Österreich

- Parameter
- Grenzwert
- Mittelwert für

B.2 Richtlinien zum Meßobjekt Wasser

EG-Wasseremissionsnormen

- Parameter
- Typ des Industriebetriebs
- Grenzwertregelung
  - Typ des Mittelwerts
  - Grenzwerte in Fracht
  - Konzentration
- Einzuhalten ab
- Referenzmethode
- EG-Richtlinie Nr., Titel

EG-Wasserqualitätsnormen

1. Wasser für den menschlichen Gebrauch
   (a) Organoleptische Parameter, physikalisch-chemische Parameter, Parameter für unerwünschte Stoffe, Parameter für toxische Stoffe
      - Nr. gemäß EG-Richtlinie
      - Parameter
      - Art der Darstellung der Ergebnisse
      - Leitwert
      - Höchstkonzentration (ZHK)
      - Bemerkungen

   (b) Mikrobiologische Parameter
      - Nr. gemäß EG-Richtlinie
      - Parameter
      - Ergebnisse: Probemenge in ml
      - Leitwert

    - Membranfiltermethode
    - Mehrfachröhrenmethode (MPN)

   (c) Erforderliche Mindestkonzentrationen für Wasser, das enthärtet worden ist und zum menschlichen Gebrauch geliefert wird.
    - Nr. gemäß EG-Richtlinie
    - Parameter
    - Darstellungsweise der Ergebnisse
    - Erforderliche Mindestkonzentration
    - Bemerkungen

2. Oberflächenwasser zur Trinkwassergewinnung
   - Nr. gemäß EG-Richtlinie
   - Parameter
   - Aufbereitungsmaßnahmen-Grenzwerte
     - A1-Leitwert
     - A1-Höchstkonzentration
     - A2-Leitwert
     - A2-Höchstkonzentration
     - A3-Leitwert
     - A3-Höchstkonzentration

3. Badegewässer
   - Nr. gemäß EG-Richtlinie
   - Parameter
   - Leitwert
   - Höchstkonzentration
   - Mindesthäufigkeit der Probeentnahme
   - Analysen- und Prüfungsverfahren

4. Süßwasser
   - Parameter
   - Salmonidengewässer-Leitwert
   - Salmonidengewässer-Höchstwert
   - Cyprinidengewässer-Leitwert
   - Cyprinidengewässer-Höchstwert
   - Analyse- oder Kontrollverfahren
   - Regelhäufigkeit der Probeentnahme und Messungen
   - Bemerkungen

Trinkwasserverordnung der BR Deutschland

- Parameter
- Grenzwert
- Bemerkungen

WHO-Leitlinien für Trinkwasser

- Parameter
- Richtwert

DVGW-Arbeitsblatt W 151 BRD (für Oberflächenwasser)

- Parameter
- Richt- und Vergleichswert A
- Richt- und Vergleichswert B

Qualitätsziele für Fließgewässer und Flußstaue Schweiz

- Parameter
- Qualitätsziel

Leitfaden Bodensanierung Niederlande (Grundwasser)

- Parameter
- Hintergrundwerte
- Prüfwerte
- Grenzwerte

B.3 Richtlinien zum Meßobjekt Boden

EG-Bodenqualitätsnormen

- Geregelte Stoffe
- Grenzwertregelung
  - Wert
  - Maßeinheit
  - exaktes Meßobjekt (Schlamm, Boden, ...)
- Bemerkungen

Anhang C

Konsolidierte Struktur der Richtlinien

Die untenstehenden Listen zeigen die konsolidierte Struktur der Richtlinien als Ergebnis der Datenanalyse in Abschnitt 4.1. Insbesondere wurden dabei Begriffe vereinheitlicht[1] und einige Tabelleneintragungen aufgebrochen, um eine gleichartige und damit vergleichbare Struktur der Tabellen zu erhalten, ohne dabei die Aussagefähigkeit zu ändern oder gar zu vermindern[2].

C.1 Richtlinien zum Meßobjekt Luft

EG-Richtlinien zur Luft-Emission

- Parameter
- Emissionsquelle
- Richtlinienwert
- Maßeinheit
- Bemerkung zum Richtlinienwert
- Einzuhalten ab
- Meßmethode
- Richtlinien ID

EG-Luftqualitätsnormen

- Parameter
- Bezugszeitraum

[1]Beispielsweise umfaßt der hier verwendete Begriff *Parameter* die unter B aufgeführten Begriffe *Schadstoff*, *geregelter Stoff*, *Stoff* und *Parameter*.

[2]Beispielsweise gibt es zu jedem Zahlenwert innerhalb einer Grenzwertregelung auch eine Maßeinheit. Teilweise wurde die Maßeinheit beim Grenzwert selbst aufgeführt, manchmal beim Parameter, manchmal gab es eine eigene Spalte, um die Maßeinheit aufzunehmen.

- Richtlinienwert
- Maßeinheit
- Richtlinienwertart
- Verweis[3]
- Einzuhalten ab
- Meßmethode
- Richtlinien ID

TA Luft (Luft-Immission)

- Parameter
- Richtlinienwert 1
- Maßeinheit 1
- Richtlinienwert 2
- Maßeinheit 2

TA Luft (Luft-Emission)

- Emissionsquelle
- Parameter
- Richtlinienwert
- Maßeinheit
- Bemerkung zum Richtlinienwert und/oder zur Emissionsquelle
- Bemerkung

WHO-Luftqualitätsleitlinien

- Parameter
- Richtlinienwert 1
- Maßeinheit 1
- Bemerkung zum Richtlinienwert 1 (selten)
- Richtlinienwertart 1
- Richtlinienwert 2
- Maßeinheit 2
- Richtlinienwert 3
- Maßeinheit 3
- Richtlinienwert 4
- Maßeinheit 4

[3] auf einen anderen Richtlinienwert, der ebenfalls eingehalten werden muß.

- Richtlinienwert 5
- Maßeinheit 5
- Bemerkung zum Richtlinienwert 5 (selten)
- Richtlinienwertart 5
- Richtlinienwert 6
- Maßeinheit 6
- Bemerkung zum Richtlinienwert 6 (war: Tumorlokalisation)

VDI-Richtlinie 2310 (MIK-BRD)

- Parameter
- Richtlinienwert
- Maßeinheit
- Richtlinienwertart

Luftreinhalteverordnung Schweiz

- Parameter
- Richtlinienwert
- Maßeinheit
- Richtlinienwertart

IUFRO Österreich

- Parameter
- Richtlinienwert
- Maßeinheit
- Richtlinienwertart

C.2 Richtlinien zum Meßobjekt Wasser

EG-Wasseremissionsnormen

- Parameter
- Emissionsquelle
- Richtlinienwertart 1 & 2
- Richtlinienwert 1
- Maßeinheit 1
- Maßeinheitszusatz 1
- Richtlinienwert 2

- Maßeinheit 2
- Maßeinheitszusatz 2
- Einzuhalten ab
- Meßmethode
- Richtlinien ID

EG-Wasserqualitätsnormen

1. Wasser für den menschlichen Gebrauch
   (a) Organoleptische Parameter, physikalisch-chemische Parameter, Parameter für unerwünschte Stoffe, Parameter für toxische Stoffe
      - Parameter Nr.
      - Parameter
      - Maßeinheit 1 & 2
      - Maßeinheitszusatz
      - Richtlinienwert 1
      - Richtlinienwert 2
      - Bemerkungen

   (b) Mikrobiologische Parameter
      - Parameter Nr.
      - Parameter
      - Probemenge
      - Richtlinienwert 1
      - Richtlinienwert 2
      - Richtlinienwert 3

   (c) Erforderliche Mindestkonzentrationen für Wasser, das enthärtet worden ist und zum menschlichen Gebrauch geliefert wird.
      - Parameter Nr.
      - Parameter
      - Maßeinheit
      - Maßeinheitszusatz
      - Richtlinienwert
      - Bemerkungen

2. Oberflächenwasser zur Trinkwassergewinnung
   - Parameter Nr.
   - Parameter
   - Maßeinheit
   - Maßeinheitszusatz
   - Richtlinienwert 1
   - Bemerkung zum Richtlinienwert 1 (selten)
   - Richtlinienwert 2
   - Bemerkung zum Richtlinienwert 2 (selten)

- Richtlinienwert 3
- Bemerkung zum Richtlinienwert 3 (selten)
- Richtlinienwert 4
- Bemerkung zum Richtlinienwert 4 (selten)
- Richtlinienwert 5
- Bemerkung zum Richtlinienwert 5 (selten)
- Richtlinienwert 6
- Bemerkung zum Richtlinienwert 6 (selten)

3. Badegewässer

   - Parameter Nr.
   - Parameter
   - Probemenge (manchmal)
   - Maßeinheit
   - Richtlinienwert 1
   - Richtlinienwert 2
   - Bemerkung zum Richtlinienwert 2
   - Probehäufigkeit
   - Meßmethode

4. Süßwasser

   - Parameter
   - Maßeinheit
   - Maßeinheitszusatz
   - Richtlinienwert 1
   - Richtlinienwertart 1
   - Bemerkung zum Richtlinienwert 1
   - Richtlinienwert 2
   - Richtlinienwertart 2
   - Bemerkung zum Richtlinienwert 2
   - Richtlinienwert 3
   - Richtlinienwertart 3
   - Bemerkung zum Richtlinienwert 3
   - Richtlinienwert 4
   - Richtlinienwertart 4
   - Bemerkung zum Richtlinienwert 4
   - Meßmethode
   - Probehäufigkeit
   - Bemerkungen

Trinkwasserverordnung der BR Deutschland

- Parameter
- Bemerkung zum Richtlinienwert
- Richtlinienwert 1
- Maßeinheit 1
- Richtlinienwert 2[4]
- Maßeinheit 2
- Bemerkungen[5]

WHO-Leitlinien für Trinkwasser

- Parameter
- Maßeinheit
- Richtlinienwert

DVGW-Arbeitsblatt W 151 BRD (für Oberflächenwasser)

- Parameter
- Richtlinienwert 1
- Maßeinheit 1
- Richtlinienwert 2
- Maßeinheit 2

Qualitätsziele für Fließgewässer und Flußstaue Schweiz

- Parameter
- Richtlinienwert
- Maßeinheit

Leitfaden Bodensanierung Niederlande (Grundwasser)

- Parameter
- Richtlinienwert 1
- Maßeinheit 1
- Richtlinienwert 2
- Maßeinheit 2
- Richtlinienwert 3
- Maßeinheit 3

[4] oft; entspricht Richtlinienwert 1 umgerechnet auf Maßeinheit 2.
[5] enthält meist einen Verweis auf eine Meßmethode.

C.3 Richtlinien zum Meßobjekt Boden

EG-Bodenqualitätsnormen

- Parameter
- Richtlinienwert
- Maßeinheit
- Bemerkung zum Richtlinienwert[6]
- Bemerkungen

[6]exaktes Meßobjekt (Schlamm, Boden, ...).

Anhang D

Spezifikationen zum ersten dynamischen Entwurf

D.1 Spezifikationen der Suchoperationen

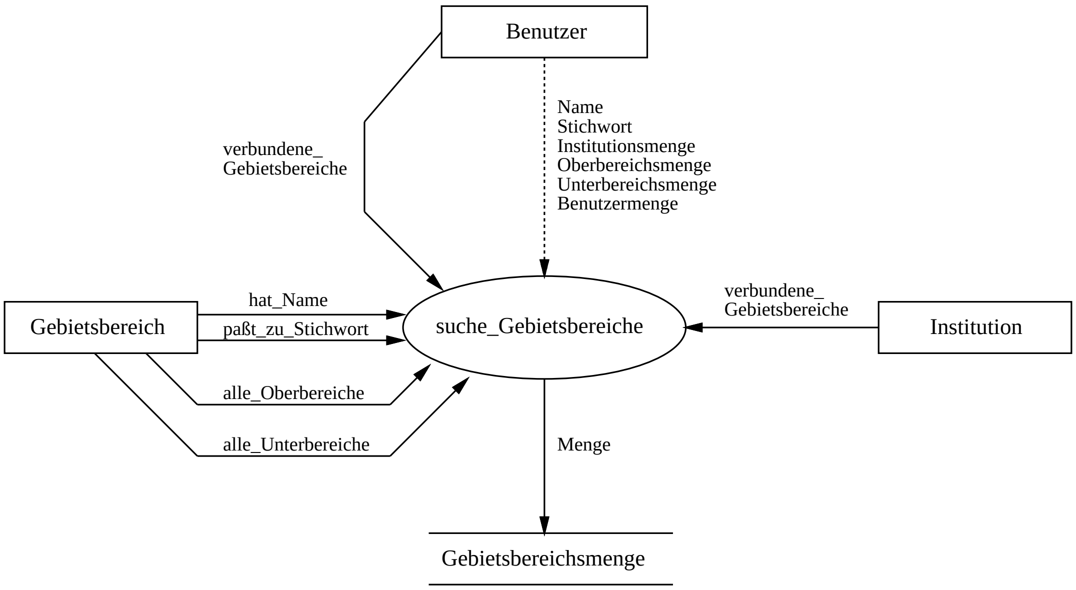


Abbildung D.1: Operation `suche_Gebietsbereiche`

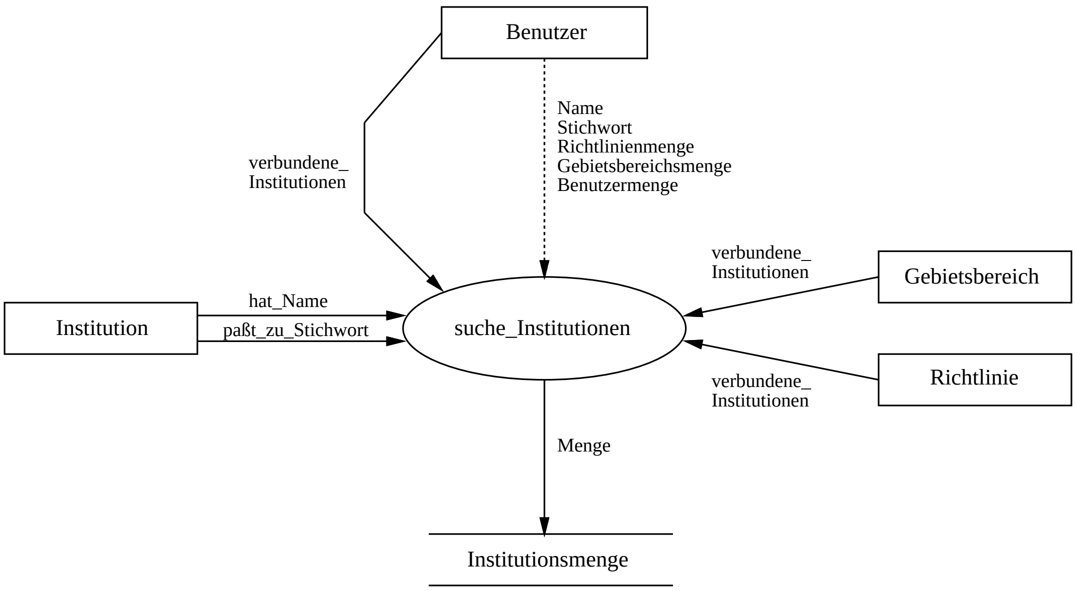


Abbildung D.2: Operation `suche_Institutionen`

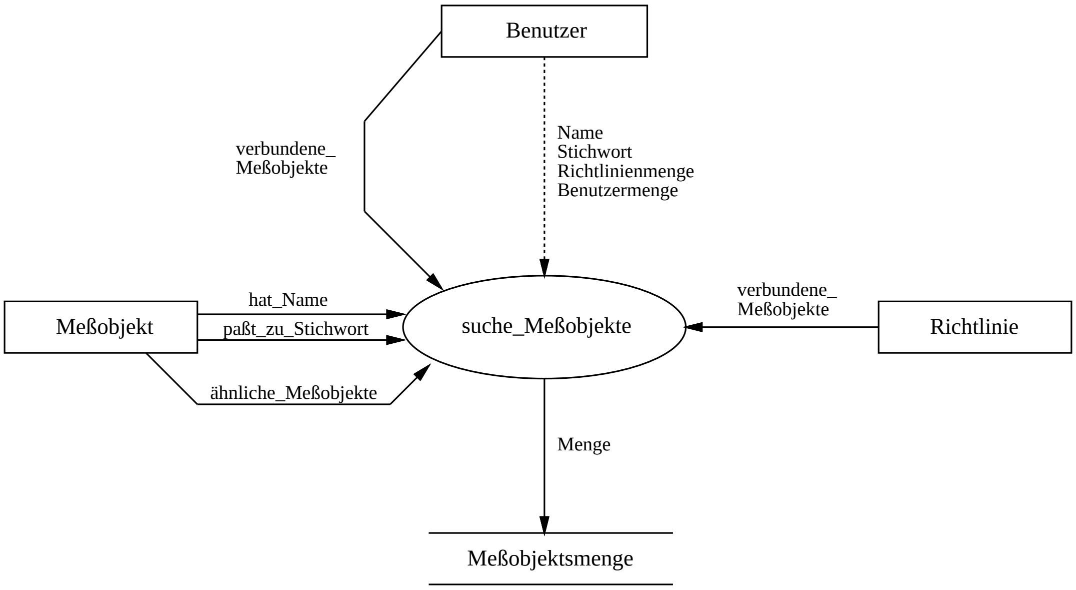


Abbildung D.3: Operation `suche_Meßobjekte`

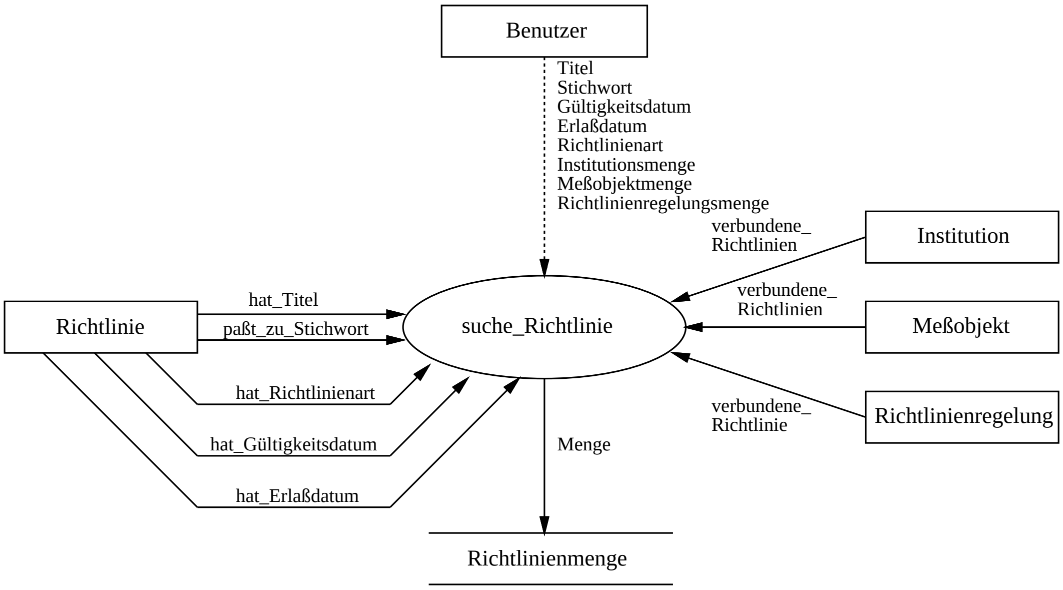


Abbildung D.4: Operation `suche_Richtlinien`

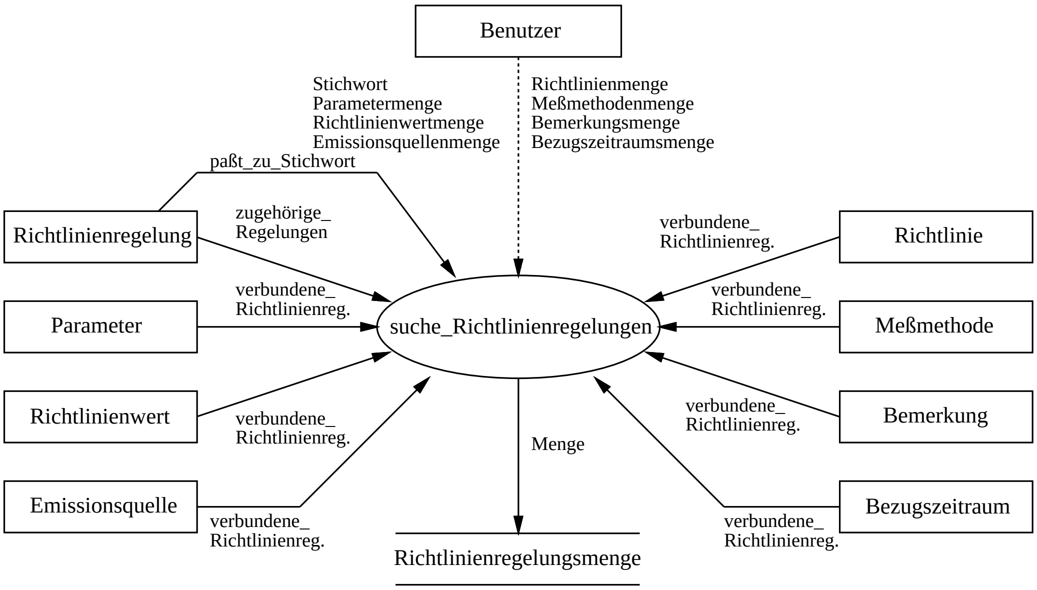


Abbildung D.5: Operation `suche_Richtlinienregelungen`

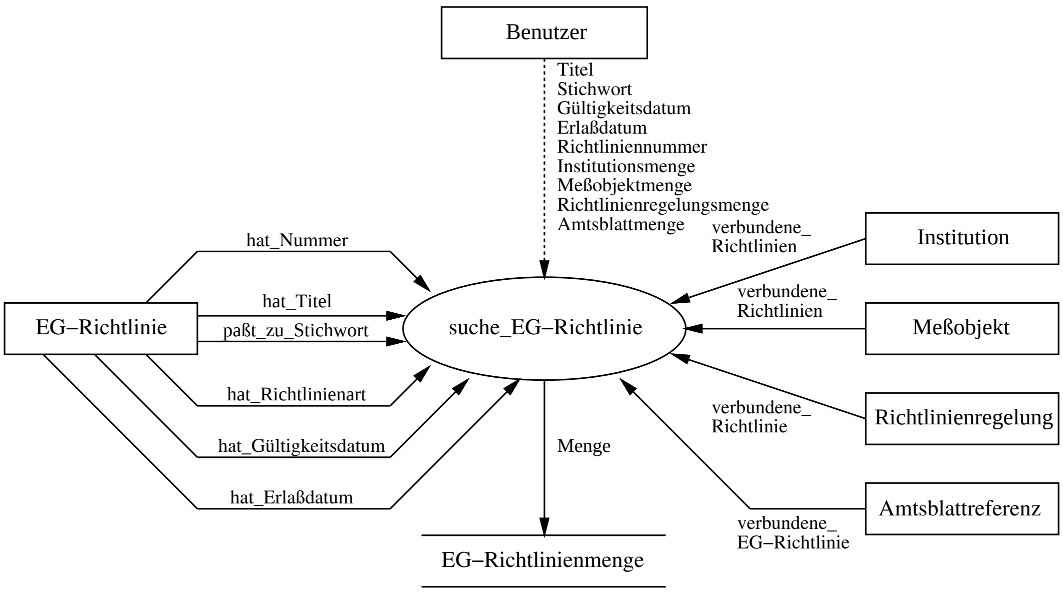


Abbildung D.6: Operation `suche_EG-Richtlinien`

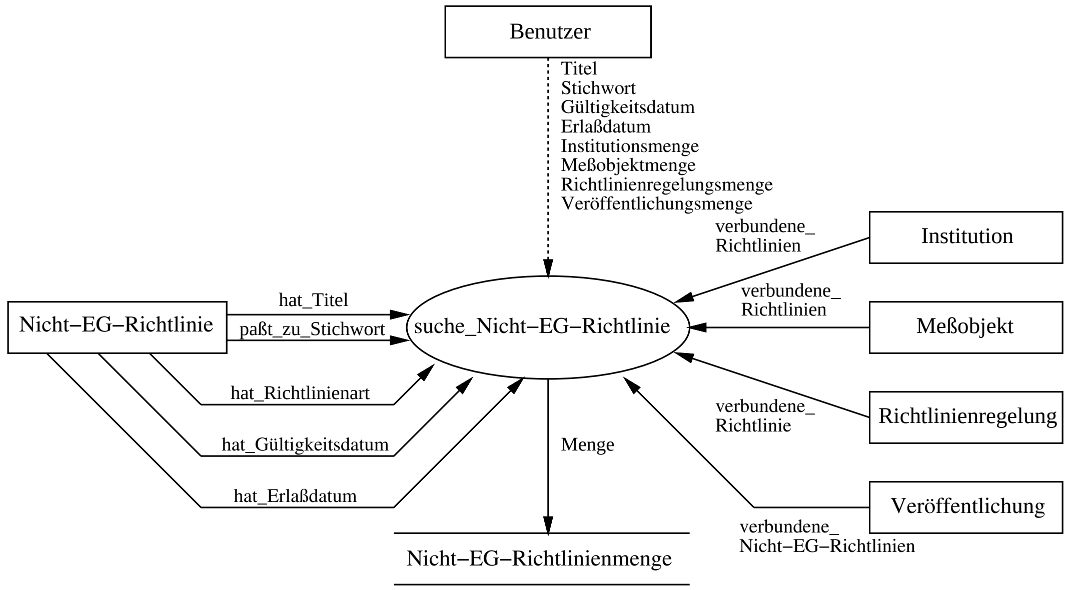


Abbildung D.7: Operation `suche_Nicht-EG-Richtlinien`

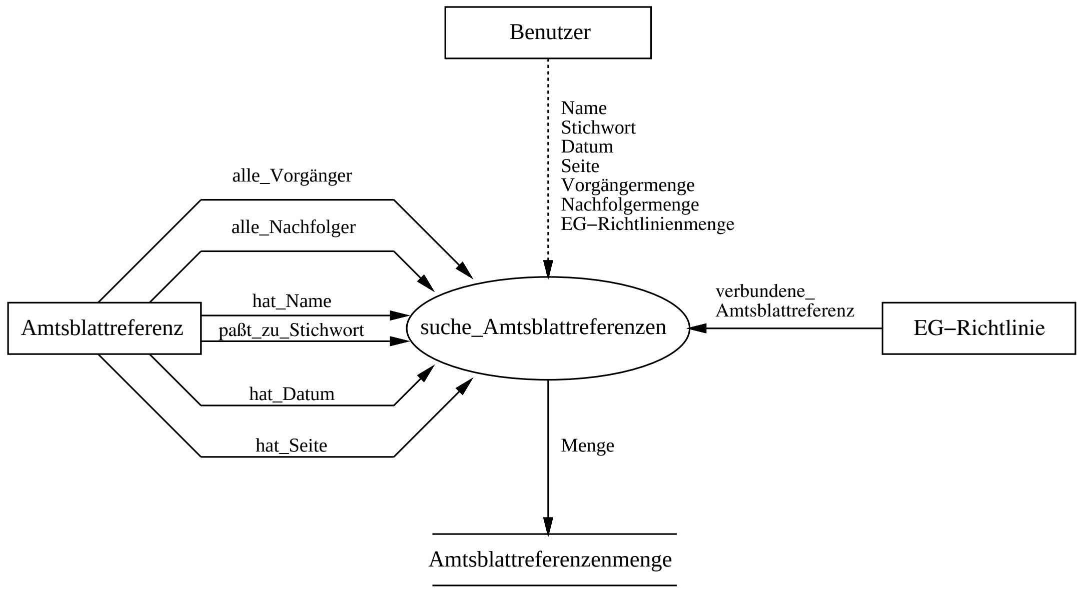


Abbildung D.8: Operation `suche_Amtsblattreferenzen`

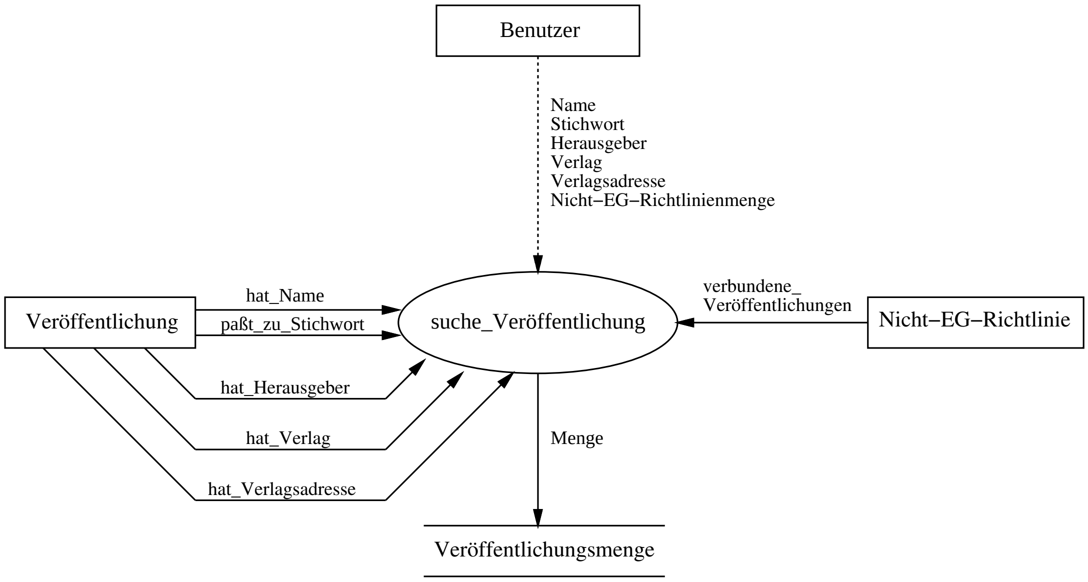


Abbildung D.9: Operation `suche_Veröffentlichungen`

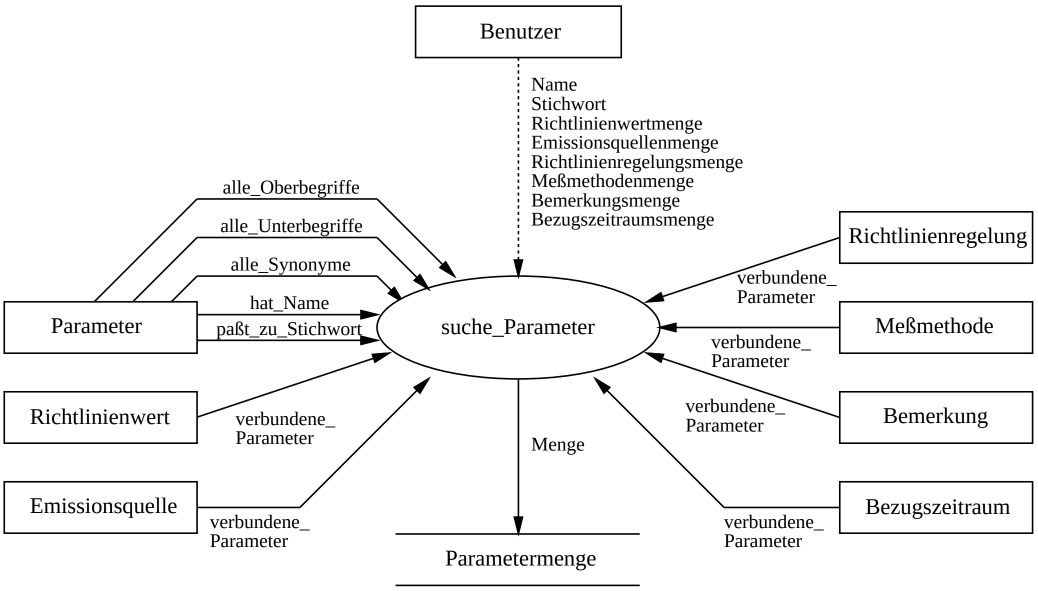


Abbildung D.10: Operation `suche_Parameter`

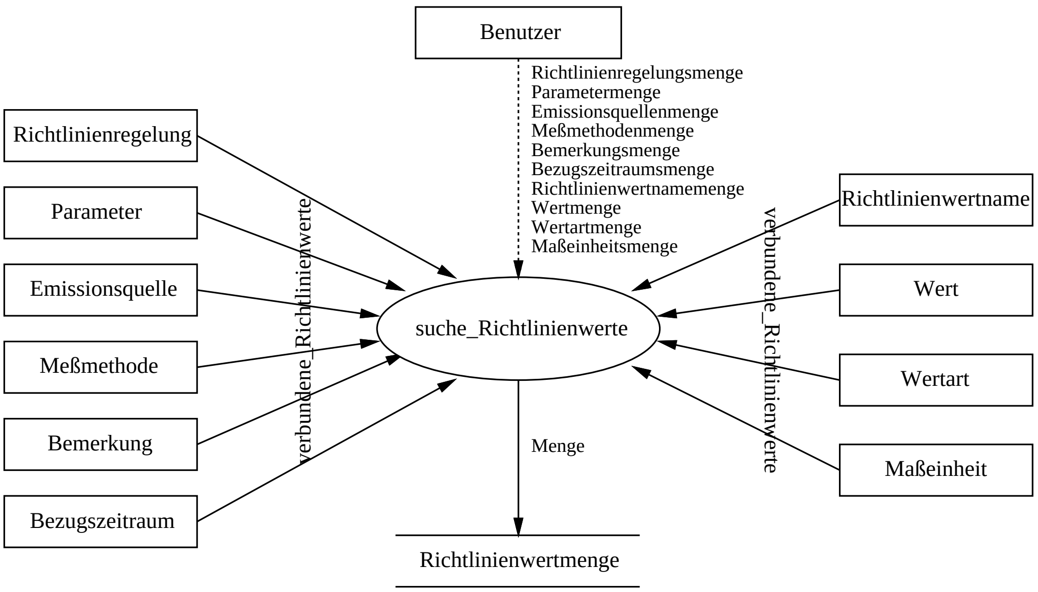


Abbildung D.11: Operation `suche_Richtlinienwerte`

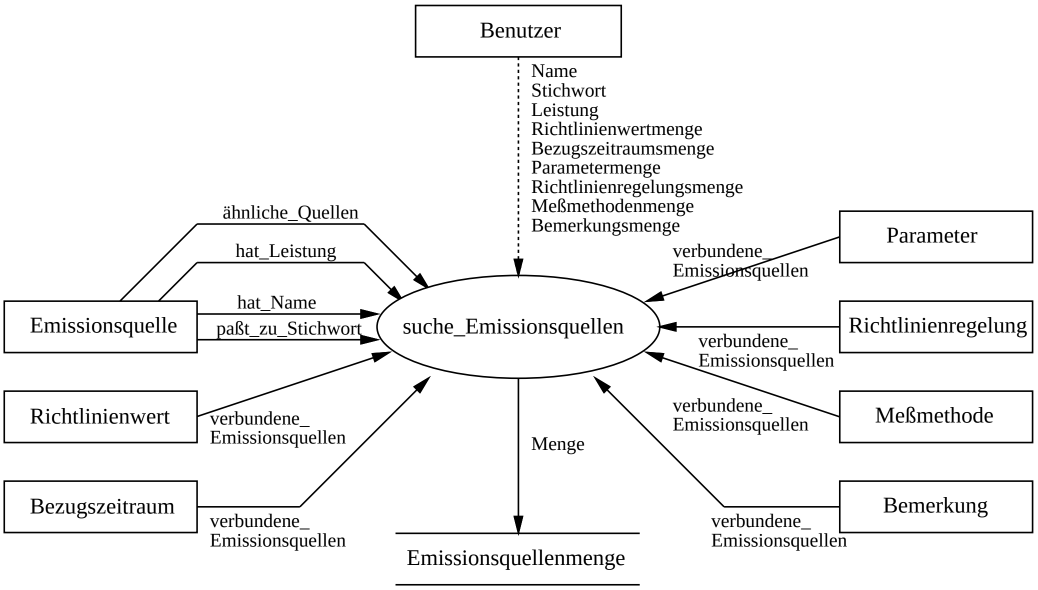


Abbildung D.12: Operation `suche_Emissionsquellen`

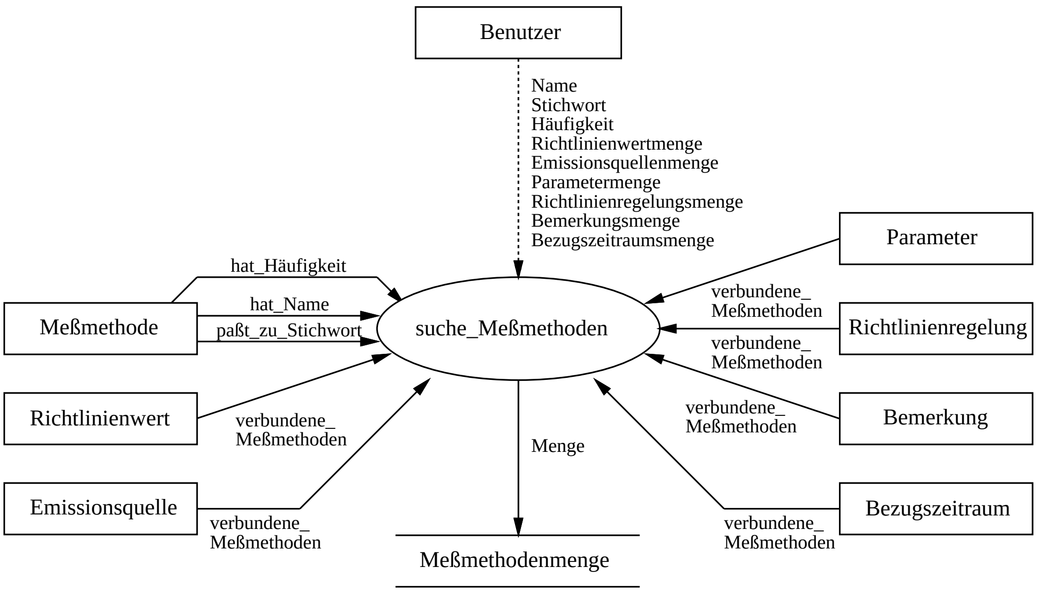


Abbildung D.13: Operation `suche_Meßmethoden`

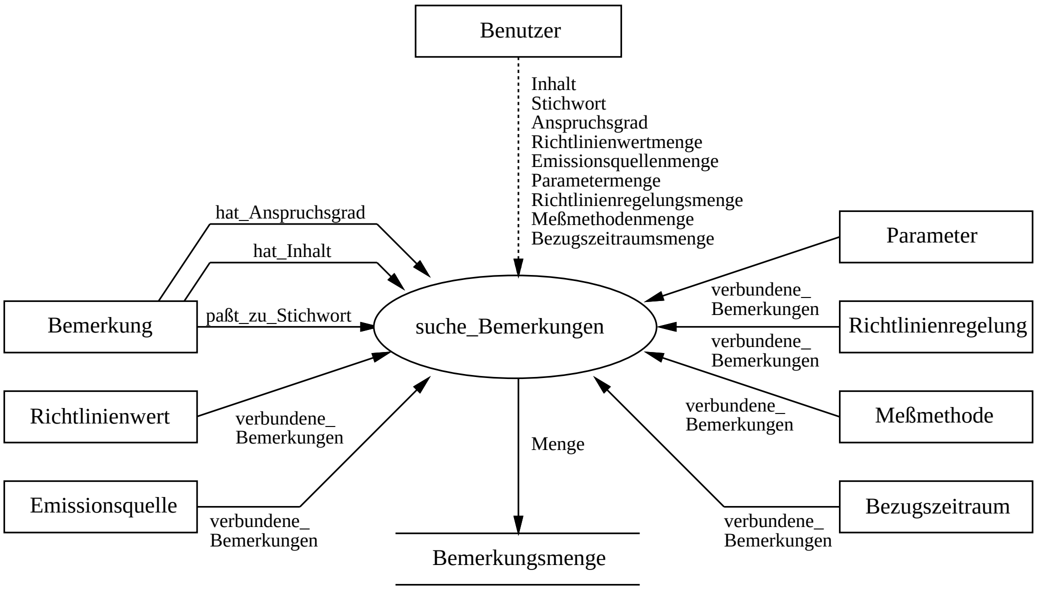


Abbildung D.14: Operation `suche_Bemerkungen`

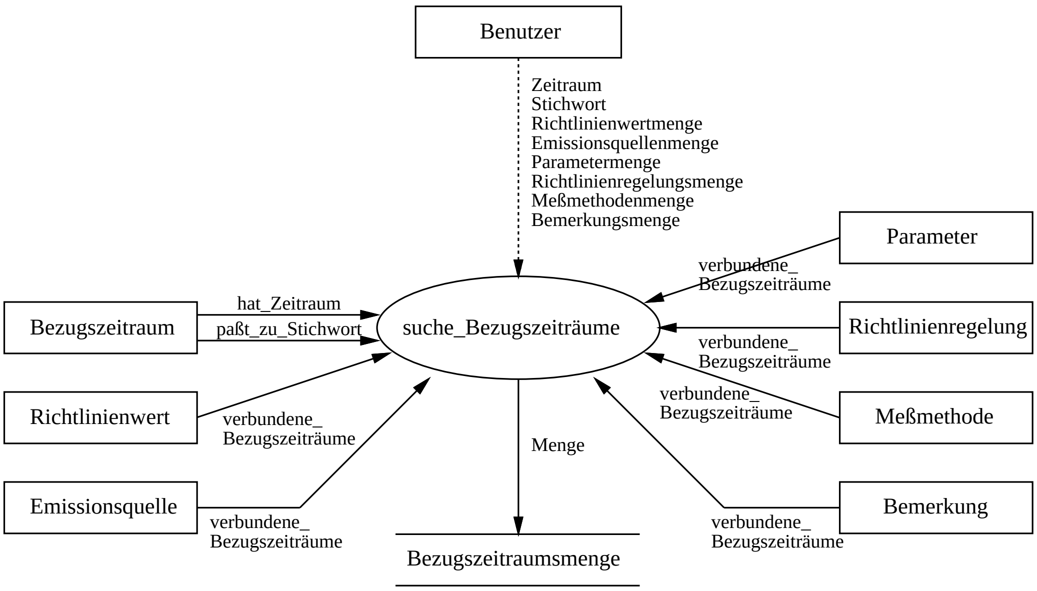


Abbildung D.15: Operation `suche_Bezugszeiträume`

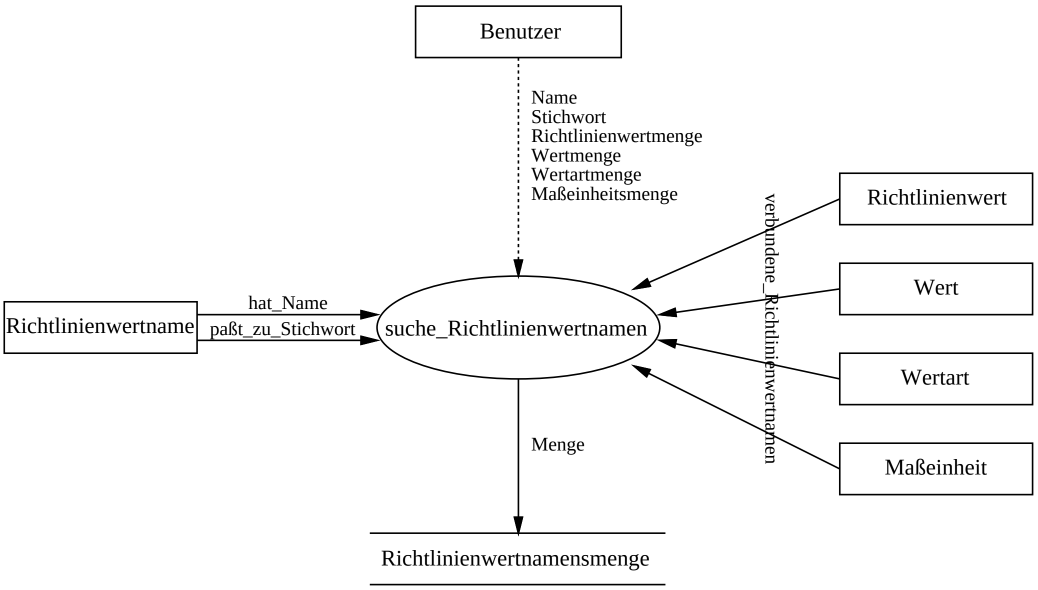


Abbildung D.16: Operation `suche_Richtlinienwertnamen`

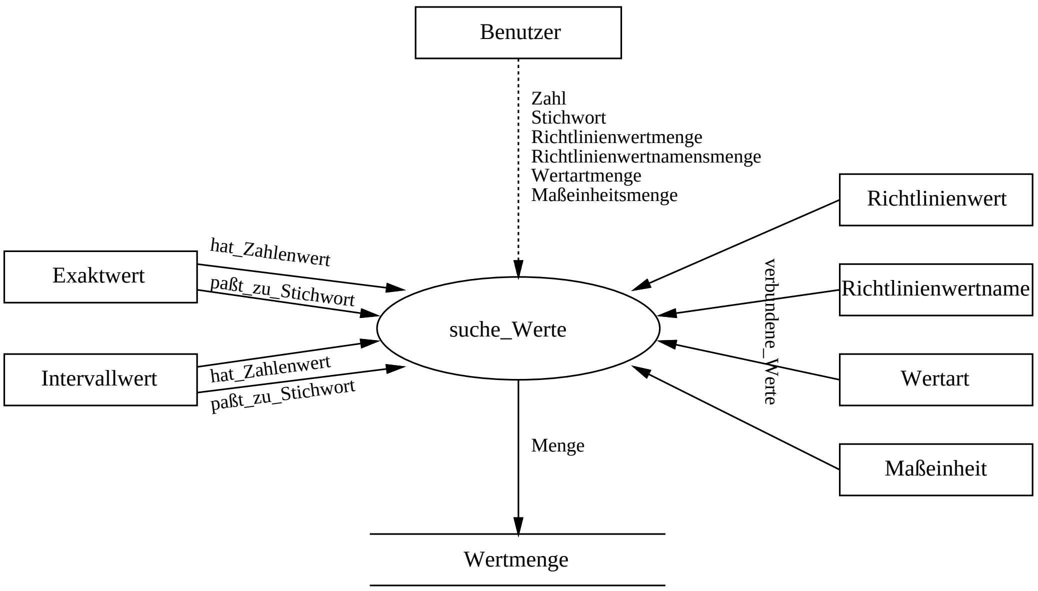


Abbildung D.17: Operation `suche_Werte`

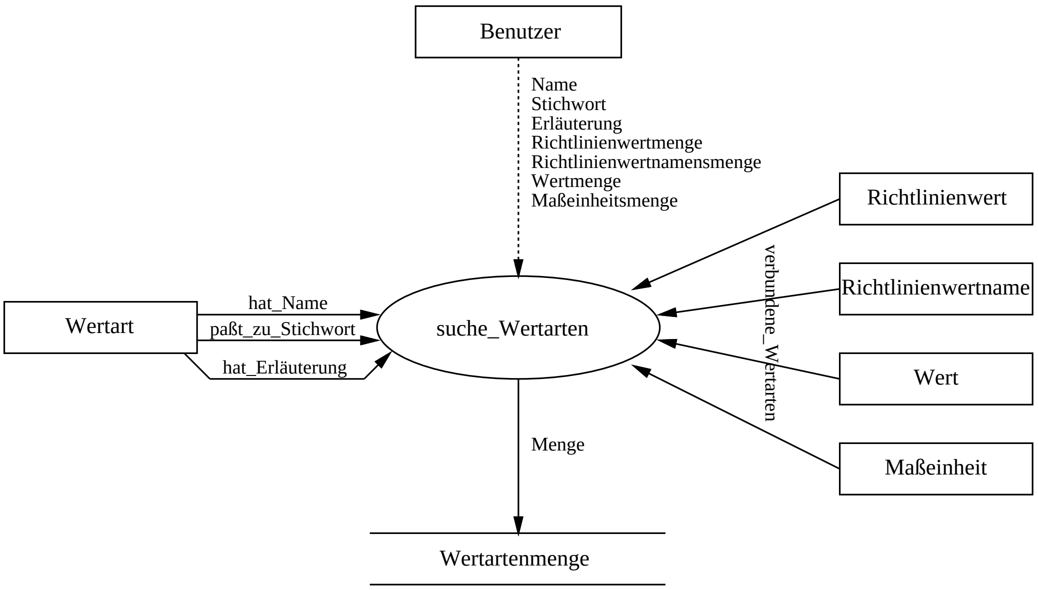


Abbildung D.18: Operation `suche_Wertarten`

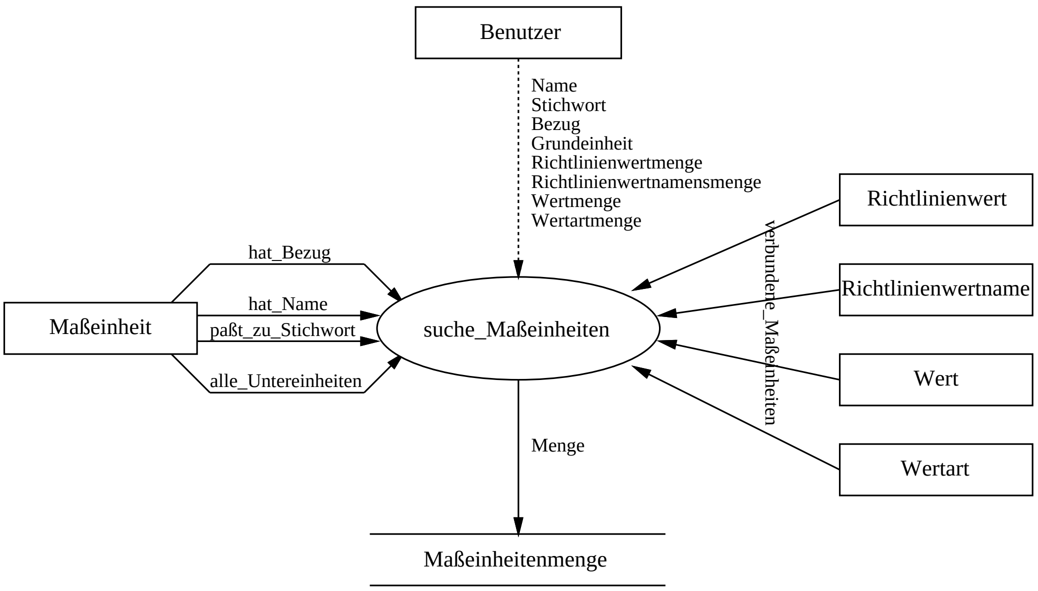


Abbildung D.19: Operation `suche_Maßeinheiten`

D.2 Zuordnung der Operationen

Operationsname	*Eingabeparameter*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Name	Gebietname	boolean	Gebietsbereich_Ktxt
passt_zu_Stichwort	Stichwort	boolean	Gebietsbereich_Ktxt
alle_Oberbereiche		Gebietsbereichs-menge	Gebietsbereich_Ktxt schließt_ein_Ktxt
alle_Unterbereiche		Gebietsbereichs-menge	Gebietsbereich_Ktxt schließt_ein_Ktxt
verbundene_Institutionen		Institutionsmenge	Gebietsbereich_Ktxt wirkt_in_Ktxt
verbundene_Benutzer		Benutzermenge	Gebietsbereich_Ktxt interessiert_Gebietsbereich_Ktxt
Ausgabe	Abstr.grad		Gebietsbereich_Ktxt

Tabelle D.1: Operationen der Klasse `Gebietsbereich`

Operationsname	*Eingabeparameter*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Name	Institutionsname	boolean	Institution_Ktxt
passt_zu_Stichwort	Stichwort	boolean	Institution_Ktxt
verbundene_Richtlinien		Richtlinienmenge	Institution_Ktxt erlassen_von_Ktxt
verbundene_Gebietsbereiche		Gebietsbereichs-menge	Institution_Ktxt wirkt_in_Ktxt
verbundene_Benutzer		Benutzermenge	Institution_Ktxt interessiert_Institution_Ktxt
Ausgabe	Abstr.grad		Institution_Ktxt

Tabelle D.2: Operationen der Klasse `Institution`

Operationsname	*Eingabeparameter*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Name	Meßobjektsname	boolean	Meßobjekt_Ktxt
passt_zu_Stichwort	Stichwort	boolean	Meßobjekt_Ktxt
ähnliche_Meßobjekte		Meßobjektmenge	Meßobjekt_Ktxt überschneidet_sich_mit_Ktxt
verbundene_Richtlinien		Richtlinienmenge	Meßobjekt_Ktxt erlassen_von_Ktxt
verbundene_Benutzer		Benutzermenge	Meßobjekt_Ktxt interessiert_Meßobjekt_Ktxt
Ausgabe	Abstr.grad		Meßobjekt_Ktxt

Tabelle D.3: Operationen der Klasse `Meßobjekt`

Operationsname	*Eingabeparameter*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Titel	Richtlinientitel	boolean	Richtlinie_Ktxt
passt_zu_Stichwort	Stichwort	boolean	Richtlinie_Ktxt
hat_Richtlinienart	Richtlinienart	boolean	Richtlinie_Ktxt
ist_gültig_am	Gültigkeitsdatum	boolean	Richtlinie_Ktxt
wurde_erlassen_am	Erlaßdatum	boolean	Richtlinie_Ktxt
verbundene_Richtlinien-regelungen		Richtlinien-regelungsmenge	Richtlinie_Ktxt hat_Richtlinienregelung_Ktxt
verbundene_Institutionen		Institutionsmenge	Richtlinie_Ktxt erlassen_von_Ktxt
verbundene_Meßobjekte		Meßobjektsmenge	Richtlinie_Ktxt hat_Meßobjekt_Ktxt
Ausgabe	Abstr.grad		Richtlinie_Ktxt

Tabelle D.4: Operationen der Klasse `Richtlinie`

Operationsname	*Eingabeparameter*	*Ausgabetyp*	*beteiligte Kontexte*
passt_zu_Stichwort	Stichwort	boolean	Richtlinie_Ktxt EG-Richtlinie_Ktxt
hat_Nummer	Richtliniennummer	boolean	EG-Richtlinie_Ktxt
verbundene_Amtsblatt-referenz		Amtsblattreferenz	hat_Amtsblattreferenz_Ktxt
Ausgabe	Abstr.grad		EG-Richtlinie_Ktxt

Bemerkung: Die Klasse `EG-Richtlinie` erbt die Operationen der Klasse `Richtlinie`. Die Operation `passt_zu_Stichwort` wird *zusätzlich* (zum Kontext `Richtlinie_Ktxt`, der durch Vererbung einbezogen wird) noch im Kontext `EG-Richtlinie_Ktxt` angeboten.

Tabelle D.5: Operationen der Klasse `EG-Richtlinie`

Operationsname	*Eingabeparameter*	*Ausgabetyp*	*beteiligte Kontexte*
verbundene_Veröffentlichung		Veröffentlichung	veröffentlicht_in_Ktxt
Ausgabe	Abstr.grad		Nicht-EG-Richtlinie_Ktxt

Bemerkung: Die Klasse `Nicht-EG-Richtlinie` erbt die Operationen der Klasse `Richtlinie`.

Tabelle D.6: Operationen der Klasse `Nicht-EG-Richtlinie`

Operationsname	*Eingabeparameter*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Name	Amtsblattreferenz	boolean	Amtsblattreferenz_Ktxt
passt_zu_Stichwort	Stichwort	boolean	Amtsblattreferenz_Ktxt
hat_Datum	Datum	boolean	Amtsblattreferenz_Ktxt
hat_Seite	Seite	boolean	Amtsblattreferenz_Ktxt
alle_Vorgänger		Amtsblattre-ferenzmenge	Amtsblattreferenz_Ktxt geändert_durch_Ktxt
alle_Nachfolger		Amtsblattre-ferenzmenge	Amtsblattreferenz_Ktxt geändert_durch_Ktxt
verbundene_EG-Richtlinie		EG-Richtlinie	hat_Amtsblattreferenz_Ktxt
Ausgabe	Abstr.grad		Amtsblattreferenz_Ktxt

Tabelle D.7: Operationen der Klasse `Amtsblattreferenz`

Operationsname	*Eingabeparameter*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Name	Veröffentlichung	boolean	Veröffentlichung_Ktxt
passt_zu_Stichwort	Stichwort	boolean	Veröffentlichung_Ktxt
hat_Herausgeber	Herausgeber	boolean	Veröffentlichung_Ktxt
hat_Verlag	Verlag	boolean	Veröffentlichung_Ktxt
hat_Verlagsadresse	Verlagsadresse	boolean	Veröffentlichung_Ktxt
verbundene_Nicht-EG-Richtlinie		Nicht-EG-Richtlinie	veröffentlicht_in_Ktxt
Ausgabe	Abstr.grad		Veröffentlichung_Ktxt

Tabelle D.8: Operationen der Klasse `Veröffentlichung`

Operationsname	*Eingabepar.*	*Ausgabetyp*	*beteiligte Kontexte*
verbundene_Richtlinie		Richtlinie	hat_Richtlinienregelung_Ktxt
verbundene_Regelungen		Richtlinien-regelungsmenge	Richtlinienregelungs-Aggregat gültig_zusammen_mit_Ktxt
verbundener_Parameter		Parameter	Richtlinienregelungs-Aggregat
verbundene_Richtlinienwerte		Richtlinienwert-menge	Richtlinienregelungs-Aggregat
verbundene_Meßmethode		Meßmethode	Richtlinienregelungs-Aggregat
verbundene_Bemerkung		Bemerkung	Richtlinienregelungs-Aggregat
verbundener_Bezugszeitraum		Bezugszeitraum	Richtlinienregelungs-Aggregat
verbundene_Emissionsquelle		Emissionsquelle	Richtlinienregelungs-Aggregat
Ausgabe	Abstr.grad		Richtlinienregelungs-Aggregat

Tabelle D.9: Operationen der Klasse `Richtlinienregelung`

Operationsname	*Eingabepar.*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Name	Parameter	boolean	Parameter_Ktxt
passt_zu_Stichwort	Stichwort	boolean	Parameter_Ktxt
alle_Unterbegriffe		Parametermenge	Parameter_Ktxt fällt_unter_Ktxt
alle_Oberbegriffe		Parametermenge	Parameter_Ktxt fällt_unter_Ktxt
alle_Synonyme		Parametermenge	Parameter_Ktxt ist_synonym_Ktxt
verbundene_Regelungen		Richtlinien-regelungsmenge	Parameter_Ktxt Richtlinienregelungs-Aggregat
verbundene_Richtlinienwerte		Richtlinien-wertmenge	Parameter_Ktxt Richtlinienregelungs-Aggregat
verbundene_Meßmethoden		Meßmethoden-menge	Parameter_Ktxt Richtlinienregelungs-Aggregat
verbundene_Bemerkungen		Bemerkungs-menge	Parameter_Ktxt Richtlinienregelungs-Aggregat
verbundene_Bezugszeiträume		Bezugszeitraums-menge	Parameter_Ktxt Richtlinienregelungs-Aggregat
verbundene_Emissionsquellen		Emissionsquellen-menge	Parameter_Ktxt Richtlinienregelungs-Aggregat
Ausgabe	Abstr.grad		Parameter_Ktxt

Tabelle D.10: Operationen der Klasse `Parameter`

Operationsname	*Eingabepar.*	*Ausgabetyp*	*beteiligte Kontexte*
passt_zu_Stichwort	Stichwort	boolean	Richtlinienwert-Aggregat
verbundene_Regelung		Richtlinienregelung	Richtlinienregelungs-Aggregat
verbundener_Parameter		Parameter	Richtlinienregelungs-Aggregat
verbundene_Meßmethode		Meßmethode	Richtlinienregelungs-Aggregat
verbundene_Bemerkung		Bemerkung	Richtlinienregelungs-Aggregat
verbundener_Bezugszeitraum		Bezugszeitraum	Richtlinienregelungs-Aggregat
verbundene_Emissionsquelle		Emissionsquelle	Richtlinienregelungs-Aggregat
verbundener_Richtlinienwertname		Richtlinienwertname	Richtlinienwert-Aggregat
verbundener_Wert		Wert	Richtlinienwert-Aggregat
verbundene_Wertart		Wertart	Richtlinienwert-Aggregat
verbundene_Maßeinheit		Maßeinheit	Richtlinienwert-Aggregat
Ausgabe	Abstr.grad		Richtlinienwert-Aggregat

Tabelle D.11: Operationen der Klasse `Richtlinienwert`

Operationsname	*Eingabepar.*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Name	Name	boolean	Emissionsquelle-Aggregat
passt_zu_Stichwort	Stichwort	boolean	Emissionsquelle-Aggregat
hat_Leistung	Leistung	boolean	Emissionsquelle-Aggregat
ähnliche_Quellen		Emissionsquellenmenge	Emissionsquelle_Ktxt ist_ähnlich_zu_Ktxt
verbundene_Parameter		Parametermenge	Emissionsquelle_Ktxt Richtlinienregelungs-Aggregat
verbundene_Regelungen		Richtlinienregelungsmenge	Emissionsquelle_Ktxt Richtlinienregelungs-Aggregat
verbundene_Richtlinienwerte		Richtlinienwertmenge	Emissionsquelle_Ktxt Richtlinienregelungs-Aggregat
verbundene_Meßmethoden		Meßmethodenmenge	Emissionsquelle_Ktxt Richtlinienregelungs-Aggregat
verbundene_Bemerkungen		Bemerkungsmenge	Emissionsquelle_Ktxt Richtlinienregelungs-Aggregat
verbundene_Bezugszeiträume		Bezugszeitraumsmenge	Emissionsquelle_Ktxt Richtlinienregelungs-Aggregat
Ausgabe	Abstr.grad		Emissionsquelle_Ktxt

Tabelle D.12: Operationen der Klasse `Emissionsquelle`

Operationsname	*Eingabepar.*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Name	Name	boolean	Meßmethode-Aggregat
passt_zu_Stichwort	Stichwort	boolean	Meßmethode-Aggregat
hat_Meßhäufigkeit	Häufigkeit	boolean	Meßmethode-Aggregat
verbundene_Parameter		Parameter-menge	Meßmethode_Ktxt Richtlinienregelungs-Aggregat
verbundene_Regelungen		Richtlinien-regelungsmenge	Meßmethode_Ktxt Richtlinienregelungs-Aggregat
verbundene_Richtlinienwerte		Richtlinien-wertmenge	Meßmethode_Ktxt Richtlinienregelungs-Aggregat
verbundene_Bemerkungen		Bemerkungs-menge	Meßmethode_Ktxt Richtlinienregelungs-Aggregat
verbundene_Bezugszeiträume		Bezugszeitraums-menge	Meßmethode_Ktxt Richtlinienregelungs-Aggregat
verbundene_Emissionsquellen		Emissionsquellen-menge	Meßmethode_Ktxt Richtlinienregelungs-Aggregat
Ausgabe	Abstr.grad		Meßmethode_Ktxt

Tabelle D.13: Operationen der Klasse `Meßmethode`

Operationsname	*Eingabepar.*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Inhalt	Inhalt	boolean	Bemerkung-Aggregat
passt_zu_Stichwort	Stichwort	boolean	Bemerkung-Aggregat
hat_Anspruchsgrad	Anspruchsgrad	boolean	Bemerkung-Aggregat
verbundene_Parameter		Parameter-menge	Bemerkung_Ktxt Richtlinienregelungs-Aggregat
verbundene_Regelungen		Richtlinien-regelungsmenge	Bemerkung_Ktxt Richtlinienregelungs-Aggregat
verbundene_Richtlinienwerte		Richtlinien-wertmenge	Bemerkung_Ktxt Richtlinienregelungs-Aggregat
verbundene_Meßmethoden		Meßmethoden-menge	Bemerkung_Ktxt Richtlinienregelungs-Aggregat
verbundene_Bezugszeiträume		Bezugszeitraums-menge	Bemerkung_Ktxt Richtlinienregelungs-Aggregat
verbundene_Emissionsquellen		Emissionsquellen-menge	Bemerkung_Ktxt Richtlinienregelungs-Aggregat
Ausgabe	Abstr.grad		Bemerkung_Ktxt

Tabelle D.14: Operationen der Klasse `Bemerkung`

Operationsname	*Eingabepar.*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Zeitraum	Zeitraum	boolean	Bezugszeitraum-Aggregat
passt_zu_Stichwort	Stichwort	boolean	Bezugszeitraum-Aggregat
verbundene_Parameter		Parameter-menge	Bezugszeitraum_Ktxt Richtlinienregelungs-Aggregat
verbundene_Regelungen		Richtlinien-regelungsmenge	Bezugszeitraum_Ktxt Richtlinienregelungs-Aggregat
verbundene_Richtlinienwerte		Richtlinien-wertmenge	Bezugszeitraum_Ktxt Richtlinienregelungs-Aggregat
verbundene_Meßmethoden		Meßmethoden-menge	Bezugszeitraum_Ktxt Richtlinienregelungs-Aggregat
verbundene_Bemerkungen		Bemerkungs-menge	Bezugszeitraum_Ktxt Richtlinienregelungs-Aggregat
verbundene_Emissionsquellen		Emissionsquellen-menge	Bezugszeitraum_Ktxt Richtlinienregelungs-Aggregat
Ausgabe	Abstr.grad		Bezugszeitraum_Ktxt

Tabelle D.15: Operationen der Klasse `Bezugszeitraum`

Operationsname	*Eingabepar.*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Name	Name	boolean	Richtlinienwertname-Ktxt
passt_zu_Stichwort	Stichwort	boolean	Richtlinienwertname-Ktxt
verbundene_Richtlinienwerte		Richtlinien-wertmenge	Richtlinienwertname_Ktxt Richtlinienwert-Aggregat
verbundene_Werte		Wertmenge	Richtlinienwertname_Ktxt Richtlinienwert-Aggregat
verbundene_Wertarten		Wertartmenge	Richtlinienwertname_Ktxt Richtlinienwert-Aggregat
verbundene_Maßeinheiten		Maßeinheitsmenge	Richtlinienwertname_Ktxt Richtlinienwert-Aggregat
Ausgabe	Abstr.grad		Richtlinienwertname_Ktxt

Tabelle D.16: Operationen der Klasse `Richtlinienwertname`

Operationsname	*Eingabepar.*	*Ausgabetyp*	*beteiligte Kontexte*
passt_zu_Stichwort	Stichwort	boolean	Richtlinienwert-Aggregat
verbundener_Richtlinienwert		Richtlinienwert	Richtlinienwert-Aggregat
verbundener_Richtlinienwertname		Richtlinienwert-name	Richtlinienwert-Aggregat
verbundene_Wertart		Wertart	Richtlinienwert-Aggregat
verbundene_Maßeinheit		Maßeinheit	Richtlinienwert-Aggregat
Ausgabe	Abstr.grad		Richtlinienwert_Aggregat

Tabelle D.17: Operationen der Klasse `Wert`

Operationsname	*Eingabepar.*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Zahlenwert	Zahl	boolean	Exaktwert_Ktxt
passt_zu_Stichwort	Stichwort	boolean	Exaktwert_Ktxt
Ausgabe	Abstr.grad		Exaktwert_Ktxt

Bemerkung: Die Klasse `Exaktwert` erbt die Operationen der Klasse `Wert`. Die Operation `passt_zu_Stichwort` wird *zusätzlich* (zum Kontext `Richtlinienwert-Aggregat`, der durch Vererbung einbezogen wird) noch im Kontext `Exaktwert_Ktxt` angeboten.

Tabelle D.18: Operationen der Klasse `Exaktwert`

Operationsname	*Eingabepar.*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Zahlenwert	Zahl	boolean	Intervallwert_Ktxt
passt_zu_Stichwort	Stichwort	boolean	Intervallwert_Ktxt
Ausgabe	Abstr.grad		Intervallwert_Ktxt

Bemerkung: Die Klasse `Intervallwert` erbt die Operationen der Klasse `Wert`. Die Operation `passt_zu_Stichwort` wird *zusätzlich* (zum Kontext `Richtlinienwert-Aggregat`, der durch Vererbung einbezogen wird) noch im Kontext `Intervallwert_Ktxt` angeboten.

Tabelle D.19: Operationen der Klasse `Intervallwert`

Operationsname	*Eingabepar.*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Name	Name	boolean	Wertart-Ktxt
passt_zu_Stichwort	Stichwort	boolean	Wertart-Ktxt
hat_Erläuterung	Erläuterung	boolean	Wertart-Ktxt
verbundene_Richtlinienwerte		Richtlinien-wertmenge	Wertart_Ktxt Richtlinienwert-Aggregat
verbundene_Richtlinienwertnamen		Richtlinien-wertnamensmenge	Wertart_Ktxt Richtlinienwert-Aggregat
verbundene_Werte		Wertmenge	Wertart_Ktxt Richtlinienwert-Aggregat
verbundene_Maßeinheiten		Maßeinheitsmenge	Wertart_Ktxt Richtlinienwert-Aggregat
Ausgabe	Abstr.grad		Wertart_Ktxt

Tabelle D.20: Operationen der Klasse `Wertart`

Operationsname	*Eingabepar.*	*Ausgabetyp*	*beteiligte Kontexte*
hat_Name	Name	boolean	Maßeinheit-Ktxt
passt_zu_Stichwort	Stichwort	boolean	Maßeinheit-Ktxt
hat_Bezug	Bezug	boolean	Maßeinheit-Ktxt
verbundene_Richtlinienwerte		Richtlinien-wertmenge	Maßeinheit_Ktxt Richtlinienwert-Aggregat
verbundene_Richtlinienwertnamen		Richtlinien-wertnamensmenge	Wertart_Ktxt Richtlinienwert-Aggregat
verbundene_Werte		Wertmenge	Maßeinheit_Ktxt Richtlinienwert-Aggregat
verbundene_Wertarten		Wertartenmenge	Maßeinheit_Ktxt Richtlinienwert-Aggregat
Ausgabe	Abstr.grad		Maßeinheit_Ktxt

Tabelle D.21: Operationen der Klasse `Maßeinheit`

Literaturverzeichnis

[ANS86] American National Standard for Information Systems: Database Language SQL. American National Standards Institute, 1986. FDT, ANSI X3, 135–1986.

[Boo91] G. Booch. *Object-Oriented Design With Applications*. Benjamin/Cummings, Redwood City CA, USA, 1991.

[Che76] P. P. Chen. The Entity-Relationship Model: Toward A Unified View Of Data. *ACM Trans.Database Systems*, 1(1):9–36, Januar 1976.

[CY91] P. Coad und E. Yourdon. *Object-Oriented Design*. Prentice Hall, 2nd edition, 1991.

[Dat91] C. J. Date. *An Introduction to Database Systems*, volume I. Addison-Wesley, 5th edition, 1991.

[DR90] M. Dürr und K. Radermacher. *Einsatz von Datenbanksystemen*. Springer, 1990.

[Heu93] A. Heuer. Objektorientierter Datenbankentwurf. *Informatik-Spektrum*, 16(2):96–97, April 1993.

[HK93] St. Hemmer und R. Kramer. Grenzwert-/Richtwert-Datenbank für UIS – Benutzerhandbuch Suchmodul, November 1993. FZI Bericht 20/93.

[IBM78] IBM Corporation. *Query-by Example Terminal Users Guide*, 1978. IBM Form Number SH20–20780.

[KM90] T. Korson und J. D. McGregor. Understanding Object-Oriented: A Unifying Paradigm. *Communications of the ACM*, 33(9):40–60, September 1990.

[Kop93] O. Kopp. Konzeption und Realisierung von Datenbank-Entwurfstechniken im Hinblick auf eine Abbildung auf OPAQUE-Schemata. Diplomarbeit, Fakultät für Informatik, Universität Karlsruhe, August 1993.

[Kra93a] R. Kramer. Grenzwert-Datenbank für UIS. Arbeitspapier, Juni 1993.

[Kra93b] R. Kramer. Grenzwert-/Richtwert-Datenbank für UIS – Gesamtdokumentation. unveröffentlicht, November 1993.

[KS91] H. F. Korth und A. Silberschatz. *Database System Concepts*. Mc-Graw Hill, 2nd edition, 1991.

[KS93] R. Kramer und H. Spandl. Limits Database for an Environmental Information System – A Fuzzy Set based Querying Approach. Konferenzpapier in Vorbereitung, November 1993.

[Lan93] S.M. Lang. Kontextsensitivität als Modellierungsparadigma für objektorientierte Systeme, 1993. Arbeitspapier.

[Lau93] S.-E. Lautemann. OPAQUE: Definition eines Objektmodells auf der Basis von Kontexten. Diplomarbeit, Fakultät für Informatik, Universität Karlsruhe, Januar 1993.

[LR92] Landesanstalt für Umweltschutz und Dr. L. Roth, Hrsg. *Grenzwerte*. ecomed, 1992.

[LS87] P. C. Lockemann und J. W. Schmidt, Hrsg. *Datenbank-Handbuch*. Springer, 1987.

[May93] R. Mayer-Föll. Das Umweltinformationssystem Baden-Württemberg; Zielsetzung und Stand der Realisierung. In A. Jaeschke, T. Kämpke, J. Page, und F. J. Radermacher, Hrsg., *Informatik für den Umweltschutz*, Seite 313–337. Springer, 1993. 7. Symposium "Informatik für den Umweltschutz" Ulm 1993.

[MZ92] G. Moerkotte und A. Zachmann. Multiple Substitutability Without Affecting the Taxonomy. In *Proc. Intl. Conf. on Extending Database Technology (EDBT)*, Seite 1–15, Vienna, Austria, März 1992. Lecture Notes in Computer Science, Springer.

[Nau93] C. M. Nau. Ein Übersetzer für OPAQUE. Studienarbeit, Fakultät für Informatik, Universität Karlsruhe, 1993.

[Pup88] F. Puppe. *Einführung in Expertensysteme*. Springer, 1988.

[Pup90] F. Puppe. *Problemlösungsmethoden in Expertensystemen*. Springer, 1990.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, und W. Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall, 1991.

[RS91] J. Richardson und P. Schwarz. Aspects: Extending objects to support multiple, independent roles. In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, Seite 298–307, Denver CO, USA, Juni 1991.

[SWKH76] M. Stonebraker, E. Wong, P. Kreps, und G. D. Held. The Design and Implementation of INGRES. *ACM Trans. Database Systems*, 1(3):189–222, September 1976.

[Zlo77] M. M. Zloof. Query-by-Example: A Data Base Language. *IBM Systems Journal*, 16(4):324–343, 1977.

Index

`Anfrage`, 88, 99
`Bemerkung`, 86
`Benutzer`, 88, 99
`Bezugszeitraum`, 86
`Bezug`, 88
`Dimension`, 175
`EG-Richtlinie`, 81
`Emissionsquelle`, 86
`Exaktwert`, 87
`Gültigkeitsbereich`, 175
`Gebietsbereich`, 108, 175
`Genehmigungsverfahren`, 184
`Herkunft`, 175
`Institution`, 91
`Intervallwert`, 87
`Masseinheit`, 88
`Maßeinheit`, 175
`Meßmethode`, 86
`Meßobjekt`, 80
`Nicht-EG-Richtlinie`, 81
`Nummer gemäß Herkunft`, 175
`Parameter`, 84
`Richtliniennummer`, 175
`Richtlinienregelung`, 77, 84, 90, 95, 175
`Richtlinienwertname`, 87, 175
`Richtlinienwert`, 86, 87, 97, 175
`Richtlinie`, 77, 90, 93, 175
`Umrechnung_Ktxt`, 167
`Wertart`, 88, 175
`Werttyp`, 175
`Wert`, 87
`Zeitart`, 175

Aggregat, 84
Aggregation, 33, 60, 84
Aktor, 38
Amtsblattreferenz, 80
Analyse, 65
Analyseverfahren, 20, 191
Analytik, 20, 191
Anforderungen, 101
Anfrage, 103
Anfragealgorithmus, 106, 132
Anfragemodell, 101, 105, 131, 157, 163, 177, 184
Anfrageprimitiv, 102–104
Anfrageskelett, 132
Anpaßbarkeit, 163, 184
Assoziation, 31, 59, 175
Ausgabemodell, 153, 160, 166, 180, 185
Auswertemodell, 166

Beispielanfrage, 107, 133
Bemerkung, 84
Benutzer, 71
Benutzerprofile, 71
Bewertung, 124, 182
Beziehungstyp, 175
Bezugszeitraum, 70, 84, 191
BMFT, 107, 135
BMUR, 107, 135
Boden, 199, 207

Datenanalyse, 65
Datenbankentwurf, 23
Datenbehälter, 38
Datenfluß, 38
Datenflußdiagramm, 37
Datenobjekt, 146
DDL, 25
deskriptiv, 131
deskriptive Anfrage, 129
deskriptiver Ansatz, 102, 129
Diskriminator, 33
Diskurswelt, 19
dynamisches Binden, 160
dynamisches Modell, 31, 35

EG, 68, 192

EG-Richtlinie, 80
Eigenkontext, 59
Eingreifrichtwert, 191
Einzuhalten ab, 71
Emission, 191
Emissionsquelle, 21, 70, 84, 191
Emissionsrichtlinienwert, 21
Entity Set, 172
Entity-Relationship, 26
Entity-Relationship-Diagramm, 171
Entity-Relationship-Modellierung, 172
Entwurfsbewertung, 124, 182
ER, 24, 26
Ereignis, 35
Ergebnispipeline, 53, 152
Erweiterbarkeit, 155, 184
Evidenz, 87
Evolution, 44
Expertensystem, 87

FCKW, 86
Flexibilität, 184
Flexibiltät, 155
Funktionales Modell, 111
funktionales Modell, 31, 37
Funktionalität, 101
FZI, 171

Gebietsbereich, 79, 91
Gegenstandstyp, 172
Geheimnisprinzip, 40
Genehmigungswert, 191
Generalisierung, 29
Globale Entwurfsebene, 108
Grenzwert, 19, 191
Grenzwertbroschüre, 16

heuristische Klassifikation, 87

Identität, 29
Immission, 191
Immissionsrichtlinienwert, 21
Implementierungsebene, 119
Information Hiding, 169
information hiding, 40
Informationsbedarfsanalyse, 24, 25
Institution, 79
Interaktion, 52, 152

Kardinalität, 57, 78
Kardinalität (OMT), 31
Klasse, 50, 59
Klasse (in OPAQUE-EM), 56
Klassenebene, 118
Kontext, 42, 48
Kontextebene, 118
Kontextschablone, 42, 48, 55
Kontextsensitivität, 169
Kontrollverfahren, 20, 192
konzeptueller Entwurf, 24, 25, 28
konzeptuelles Schema, 24, 26
Kooperation, 152
Kristall, 42, 45, 56
Kristallschablone, 56

langfristiges Auswertemodell, 166
Leitwert, 19, 191
LfU, 16
Link, 31
logischer Entwurf, 24–26, 28
logisches Datenmodell, 24, 26
logisches Schema, 24, 26
Luft, 195, 201

Maßeinheit, 69, 87, 192
Maßeinheitszusatz, 70, 192
Messobjekt, 91
Methode, 30
Meßmethode, 20, 70, 84, 192
Meßobjekt, 65, 70, 80, 192
Miniwelt, 19
Modularisierung, 40, 169

Nachricht, 52
Nachrichtenmechanismus, 52
Nicht-EG-Richtlinie, 80

Oberklasse, 30
Object-Modeling-Technique, 26
Object-Modeling-Technique s. OMT, 28
Objekt, 29, 40
Objekt (OPAQUE), 42, 45
objektbasiert, 40
Objektklasse, 29, 59, 172
Objektmodell, 31, 59, 77, 90, 167, 172
Objektschablone s. Klasse, 50
OMT, 26–28, 172

OMT-Objektmodell, 77, 99
OPAQUE, 39, 54
OPAQUE-EM, 26, 27, 54
OPAQUE-EM-Objektmodell, 77, 90, 99
OPAQUE-Programm, 51
Operatorbaum, 105

Parameter, 20, 68, 84, 192
Parameter Nr., 71
Pattern Matching, 102, 144, 146
physischer Entwurf, 25
Polymorphie, 160
Prüfwert, 19, 192
Primitiv, 103
Primitive, 104
Probehäufigkeit, 71
Probemenge, 71
prozedurale Anfrage, 129
prozeduraler Ansatz, 102, 103
Prozeß, 38

QBE, 130
Qualitätsnorm, 21
Quel, 130

Referenzmethode, 20, 192
reflexiv, 83, 86
rekursive Anfrageskelette, 149
Relation, 83
Relationaler Entwurf, 171
Relationenmodell, 171
Relationship, 175
result.after, 53
result.before, 53
result.current, 53
Richtlinie, 19, 65, 68, 77, 80, 192
Richtlinien ID, 71
Richtlinienart, 81
Richtlinienregelung, 66, 78
Richtlinienregelungs-Aggregat, 83, 95
Richtlinienstruktur, 195, 201
Richtlinienwert, 19, 68, 84, 87, 193
Richtlinienwertart, 70, 194
Richtlinienwertname, 87
Richtwert, 19, 194
Rolle, 42, 48, 57
Rolle (OMT), 31, 84
Rollenslot, 48

Schadstoff, 19, 194
semantisches Datenmodell, 24, 25
semantisches Schema, 26
Spezialisierung, 29
SQL, 130
Stoff, 194
Struktur- und Wertevergleich, 146
Strukturvariable, 48
Suchalgorithmus, 144
Suchmodell, 144, 160, 163, 178, 184
Suchoperation, 209
Suchprimitiv, 102, 103, 105
symmetrisch, 83, 86
Systembegriff, 26, 29

TA, 68
Technosphäre, 15
temporäres Auswertemodell, 166
Transformation OMT $\rightarrow$ OPAQUE-EM, 59
transitiv, 83, 86

UIS, 15
Umweltinformationssystem, 15
unsicheres Schließen, 87
Unterklasse, 30

VDI, 68, 193
Veröffentlichung, 81
Vererbung, 29, 61
Vererbung (OMT), 33
Vererbung (OPAQUE), 51
Vererbung (OPAQUE-EM), 58
Verhalten (eines Objekts), 29
Verhaltensmuster, 167
View, 167

Wahrscheinlichkeit, 87
Wasser, 197, 203
Wert, 87
Wertart, 87
WHO, 68, 193

Zielklasse, 131, 144
Zielobjekt, 132, 144
Zustand, 35
Zustand (eines Objekts), 29, 35
Zustandsübergang, 35
Zustandsübergangsdiagramm, 36

Zustandsvariable, 48
zyklische Anfrageskelette, 149
Zyklusauflösung, 151