Von Multiagentensystemen zu Multiorganisationssystemen

Modellierung auf Basis von Petrinetzen

Dissertation

zur Erlangung des akademischen Grades
Dr. rer. nat
an der Fakultät für Mathematik, Informatik und Naturwissenschaften
der Universität Hamburg

eingereicht beim Fachbereich Informatik von

Matthias Wester-Ebbinghaus

aus Bonn

Hamburg, 2010

Genehmigt von der MIN-Fakultät, Fachbereich Informatik der Universität Hamburg auf Antrag von

Dr. Daniel Moldt,
Prof. Dr. Rüdiger Valk und
Prof. Dr. Bernd Page

Hamburg, den 17. Dezember 2010 (Tag der Disputation)

Prof. Dr. Horst Oberquelle
Leiter des Fachbereichs Informatik

Meiner Mutter gewidmet

Danksagung

Die vorliegende Arbeit ist während meiner Zeit als Stipendiat der Abteilung Forschung und Wissenschaftsförderung der Universität Hamburg sowie während meiner Zeit als wissenschaftlicher Mitarbeiter der Gruppe TGI (Theoretische Grundlagen der Informatik) des Fachbereiches Informatik der Universität Hamburg entstanden. In diesem Zusammenhang möchte ich insbesondere meinen beiden Betreuern Dr. Daniel Moldt und Prof. Dr. Rüdiger Valk für ihre Unterstützung danken. Die Vielzahl an Diskussionen, Denkanstößen und kritischen Fragen einerseits und der mir zugestandene Freiraum bei der Ausrichtung des Forschungsthemas und der inhaltlichen Gestaltung der Arbeit andererseits, stellten für mich eine hervorragende Mischung dar. Außerdem möchte ich mich bei Prof. Dr. Bernd Page und Prof. Dr. Winfried Lamersdorf bedanken, die sich beide recht kurzfristig bereit erklärten, eine Begutachtung beziehungsweise den Vorsitz des Promotionsprüfungsverfahrens zu übernehmen.

Ich bedanke mich bei allen aktuellen sowie ehemaligen Mitarbeitern, Promovenden und Studenten der Gruppe TGI. Eine bessere Arbeitsatmosphäre hätte ich mir nicht wünschen können. Dies gilt auf fachlicher Ebene, wobei ich viele Anregungen für unterschiedlichste Aspekte meiner Arbeit erhalten habe und viele gemeinsame Veröffentlichungen entstanden sind. Dies gilt aber auch ausdrücklich auf persönlicher Ebene. Die gemeinsame freundschaftliche Zeit wird mir immer als lieb gewonnener Kontext meiner Promotionszeit in bester Erinnerung bleiben. Einen besonders herzlichen Gruß richte ich an die folgenden (langjährigen) Wegbegleiter: Lawrence Cabac, Michael Köhler-Bußmeier, Frank Heitmann, Michael Duvigneau, Heiko Rölke, Valentin Seegert, Cristine Reese, Tobias Betz, Marcin Hewelt. Zusätzlich bedanke ich mich speziell bei denjenigen, die mir durch das Korrekturlesen von Teilen meiner Arbeit sehr geholfen haben: Eva Müller, Erik Flick, Thomas Wagner, Jan Ortmann.

Bei Familie und Freunden bedanke ich mich für die notwendigen gänzlich dissertationsfreien Oasen. Dies richtet sich ganz besonders an meine vier Frauen: Meine Mutter Edith Wester-Ebbinghaus, meine beiden Schwestern Johanna und Franziska Wester-Ebbinghaus und natürlich mein Mädchen Astrid Breite. Meiner Mutter ist die vorliegende Arbeit im besonderen gewidmet. Es ist schwer vorstellbar für mich, wie ungleich mühseliger mein Studium und insbesondere das Unterfangen Dissertation ohne ihre Unterstützung, Liebe und ihr unerschütterliches Vertrauen in mich gewesen wären.

Zusammenfassung

In der aktuellen Forschung ergeht die Forderung nach grundsätzlichen *konzeptionellen Modellen* für die Beherrschbarkeit großangelegter Softwaresysteme mit dem Charakter als *Systeme von Systemen*, die durch operationale und administrative Unabhängigkeit/Eigenständigkeit der Teilsysteme gekennzeichnet sind.

In der vorliegenden Arbeit wird die Auffassung vertreten, der Einbezug von Konzepten und Mechanismen aus dem Bereich sozialer Organisationen für die Gestaltung von Softwaresystemen sei zu diesem Zweck besonders gut geeignet. Diese Idee hat sich insbesondere im Bereich der Multiagentensysteme bereits auf breiter Basis durchgesetzt. Allerdings wird das Konzept der Organisation hier vor allem als Kontext für die Handlungen individueller Agenten und mithin als oberer Systemabschluss verwendet. Ein besonderes Merkmal sozialer Organisationen ist hingegen ihr Wesen als *kollektive Akteure*. Dieses Merkmal ermöglicht die Betrachtung von großangelegten sozialen Organisationssystemen unter Bezugnahme auf multiple Ebenen des organisierten Handelns. Um das Potential der Organisationsmetapher für die Softwareentwicklung umfassender ausnutzen zu können, wird in der vorliegenden Arbeit daher ein Wechsel der konzeptionellen Herangehensweise von der Agentenorientierung zur Organisationsorientierung vorgeschlagen. In *organisationsorientierten Softwaresystemen* steht die Organisation als Softwareentität im Mittelpunkt und weist die beiden Merkmale als Kontext und als Akteur in Kombination auf.

Ein wichtiger Teil der Arbeit ist einer ausführlichen Motivation von organisationsorientierten Softwaresystemen gewidmet. Damit einher geht die sorgfältige Identifizierung der Kernkonzepte und -mechanismen aus dem Bereich sozialer Organisationen, die dafür als konzeptionelle Grundlage dienen sollen. Besonders wichtig ist dabei der *janusgesichtige* Charakter von Organisationen (oder allgemeiner: von *organisierten Kollektivitäten*) in sozialen Organisationssystemen.

Die Arbeit geht nicht mit dem Anspruch einher, eine allgemeingültige und verbindliche Definition von, beziehungsweise Perspektive auf organisationsorientierte Softwaresysteme einzunehmen. Stattdessen ist der Hauptteil der Arbeit der Präsentation einer konkreten Form von organisationsorientierten Softwaresystemen gewidmet. Es handelt sich dabei um *Multiorganisationssysteme*, die als konzeptionelle Erweiterung von (bereits organisationsorientierten) Multiagentensystemen eingeführt werden. Als neuartige Entität wird die *Organisationseinheit* eingeführt. Dabei handelt es sich genau um die geforderte janusgesichtige Entität, die sowohl organisatorischer Kontext als auch organisatorischer Akteur ist. Organisationseinheiten als Basisbausteine führen zu einer inhärent rekursiven Struktur von Multiorganisationssystemen.

Für Organisationseinheiten und Multiorganisationssysteme werden verschiedene Modellierungstechniken vorgestellt, universelle Grundmodelle erarbeitet und schließlich eine Referenzarchitektur für Multiorganisationssysteme abgeleitet. Als Grundlage der Modellierung dient der höhere Petrinetzformalismus der Referenznetze. Referenznetze kombinieren die für Petrinetze übliche anschauliche Darstellung der Verzahnung von Netzstrukturen und nebenläufigem Systemverhalten (Markenspiel) mit dem höheren Konzept der *Netze in Netzen*. Auf diese Weise werden Synchronisationsmechanismen über Systemebenen hinweg und dynamische Rekonfigurierbarkeit von Systemstrukturen als erstrangige Modellkonzepte angeboten. Die verschiedenen vorgestellten Modellierungstechniken erlauben eine systematische Modulation der bevorzugten Modellabstraktion und erlauben auf diese Weise die Erstellung von abstrakten „Denkmodellen“ bis hin zu technischen und direkt ausführbaren „Programmiermodellen“.

Abstract

Current research efforts lead to the request for *conceptual models* for the controllability of large-scale software systems that are *systems of systems*, where a system of systems is characterized by the operational and managerial independence of its component systems.

For this purpose, it is proposed in this work to draw upon concepts and mechanisms from the field of social organizations in order to incorporate them into the arrangement of software systems. This idea has already taken hold in the field of multi-agent systems on a broad basis. But the concept of an organization is mainly understood and utilized as a context for the actions of individual agents and an agent organization is consequently typically seen as an upper system closure. However, one special attribute of social organizations is their nature as *collective actors*. This attribute allows to conceive large-scale social organized systems as incorporating multiple analysis levels of organized behaviour. In order to exploit the potential of the organizational metaphor for software engineering more comprehensively, a change in the conceptual approach from agent-orientation to organization-orientation is proposed in this work. In *organization-oriented software systems*, the central software entity is that of an organization that exhibits the attributes as a context and as an actor in combination.

An important contribution of this work is dedicated to the presentation of an elaborated motivation for organization-oriented software systems. This is complemented by a thorough identification of core concepts and mechanisms from the field of social organizations that shall serve as a conceptual basis. Most central is the *Janus-faced* character of organizations (or more generally, of *organized collectivities*) in social organized systems.

It is not intended in this work to provide the one and only definition for organization-oriented software systems or to present a mandatory perspective. Instead, the main part of the work is dedicated to present a particular form of organization-oriented software systems as a representative. This particular form is called *multi-organization systems* and they are introduced as a conceptual extension of (organization-oriented) multi-agent systems. *Organizational units* are introduced as novel entities. They are exactly the required Janus-faced entities that are organizational contexts as well as organizational actors. Organizational units as basic building blocks of multi-organization systems lead to an inherently recursive overall system structure.

Various modelling techniques for organizational units and multi-organization systems are presented, standard models of organizational units are developed and a reference architecture for multi-organization systems is inferred. The high-level Petri net formalism of reference nets constitutes the foundation for all the modelling efforts in this work. Reference nets allow to combine the descriptive illustration of the interleaving of net structure and concurrent net behaviour that is common for Petri nets with the high-level concept of *nets-within-nets*. In this way, synchronization mechanisms across multiple system levels and dynamic reconfiguration of system structures are introduced as first-order modelling concepts. The various modelling techniques presented in this work allow for a systematic modulation of the preferred abstraction level for models and thus allow for the creation of abstract „thinking models“ all the way to technical and directly executable „programming models“.

Inhaltsverzeichnis

Abbildungsverzeichnis **11**

1 Einleitung **15**
1.1 Motivation . . . 15
1.2 Vorbereitende Überlegungen zum Ziel und Inhalt der Arbeit . . . 18
1.3 Übersicht . . . 22

I Grundlagen und Forschungskontext **23**

2 Softwaresysteme im Großen als Systeme von Systemen **25**
2.1 Charakteristika . . . 26
2.1.1 Moderne Softwaresysteme im Großen . . . 26
2.1.2 Das Konzept „System von Systemen“ . . . 26
2.1.3 Verschiedene Formen von Systemen von Systemen . . . 28
2.1.4 Entstehung von Systemen von Systemen . . . 29
2.2 Herausforderungen . . . 30
2.2.1 Heuristiken für die Entwicklung komplexer Systeme . . . 30
2.2.2 Dezentralisierung . . . 32
2.2.3 Multiple Systemebenen . . . 33
2.2.4 Sozio-technische Integration . . . 34
2.2.5 Software mit Selbst-Eigenschaften . . . 36
2.3 Zusammenfassung . . . 37

3 Motivation eines Organisationsparadigmas der Softwareentwicklung **39**
3.1 Organisationen als Meso-Strukturen zwischen Mikro- und Makroebene . . . 40
3.2 Organisationen als Systeme der Stabilität und Ordnung . . . 43
3.3 Organisationen als sozio-technische Systeme . . . 45
3.4 Organisationen als offene Systeme und kollektive Akteure . . . 48
3.5 Vorteile gegenüber anderen konzeptionellen Modellen . . . 50
3.6 Zusammenfassung . . . 51

4 Konkretisierung von organisationstheoretischen Konzepten und Mechanismen **53**
4.1 Generelle Vorüberlegungen und Begriffsklärung . . . 53
4.2 Offene Systeme . . . 55
4.2.1 Grenzen und Grenzziehung . . . 56
4.2.2 Charakteristika von Umgebungen . . . 57
4.2.3 Einflussnahme von Seiten der Umwelt . . . 58
4.2.4 Einflussnahme auf die Umwelt . . . 60
4.2.5 Hierarchie . . . 61

4.3 Ziele, Steuerung und Kontrolle, Macht . . . 62
4.3.1 Ziele und Strategien . . . 63
4.3.2 Steuerungs-/Kontrollsysteme und formalisierte Macht . . . 66
4.3.3 Strukturelle Komplexität . . . 67
4.3.4 Kybernetik . . . 71
4.4 Zusammenfassung . . . 73

5 Organisationsstrukturen in Softwaresystemen – *State of the Art* 75
5.1 Agenten und Multiagentensysteme . . . 75
5.1.1 Kanonische Sichtweise auf Multiagentensysteme . . . 76
5.1.2 Agentenorientierung in Abgrenzung gegenüber der Objektorientierung . . . 77
5.1.3 Entwicklung der Organisationsperspektive in Multiagentensystemen . . . 79
5.2 Organisationen in Multiagentensystemen . . . 80
5.2.1 Klassifikation von Organisationsansätzen in Multiagentensystemen . . . 80
5.2.2 Organisationsorientierte Programmierung in Multiagentensystemen . . . 84
5.2.3 Bewertung der Organisationsorientierung in Multiagentensystemen . . . 87
5.3 *Viable System Model* und *Viable Software* . . . 90
5.4 Betriebliche Softwaresysteme . . . 91
5.4.1 (IT-)Anwendungslandschaft und Geschäftsarchitektur . . . 91
5.4.2 Dienstorientierte Architekturen . . . 92
5.4.3 Bewertung . . . 94
5.5 Zusammenfassung . . . 95

6 Petrinetzmodellierung 97
6.1 Von S/T-Netzen zu Referenznetzen . . . 97
6.1.1 S/T-Netze . . . 98
6.1.2 Gefärbte Petrinetze . . . 101
6.1.3 Referenznetze . . . 103
6.1.4 Weitere grafische Elemente in Renew . . . 107
6.2 Referenznetze für die Modellierung sozialer Systeme . . . 109
6.3 Abstraktionsstufen der Modellierung in dieser Arbeit . . . 113
6.4 Zusammenfassung . . . 113

II Modellierungstechniken und Grundmodelle 115

7 Eingrenzung und Anforderungen 117
7.1 Inhaltliche Eingrenzung und verwendete Terminologie . . . 117
7.2 Anforderungen . . . 123

8 Abstrakte Modelle von Organisationseinheiten 127
8.1 Akteure, Aktivitäten und Kollektivitätseinheiten . . . 128
8.1.1 Kollektivitätseinheiten als Kontexte . . . 128
8.1.2 Kollektivitätseinheiten als kollektive Akteure . . . 134
8.1.3 Abstraktionsstufen der Kollektivitätseinheitsmodellierung . . . 137
8.2 Organisationseinheiten . . . 148
8.2.1 Steuerung und Kontrolle . . . 149

8.2.2 Organisationseinheit: Offenheit und Grenzmanagement . . . 153
8.3 Zusammenfassung und Diskussion . . . 156

9 Operationale Modelle von Organisationseinheiten 159
9.1 Kollektivitätseinheiten und rekursive Akteurssysteme . . . 160
9.1.1 Grundlegende Mechanismen . . . 160
9.1.2 Technische Details . . . 163
9.1.3 Synchronisationsdynamik anhand von Beispielen . . . 168
9.1.4 Dynamik von Einbettungsbeziehungen . . . 173
9.1.5 Varianten und Erweiterungen . . . 178
9.2 Operationalisierung von Steuerungs- und Kontrollaspekten . . . 181
9.2.1 Grundmodell einer Organisationseinheit . . . 182
9.2.2 Technische Details . . . 186
9.2.3 Varianten und Erweiterungen . . . 187
9.3 Diskussion: Kerneigenschaften und Anforderungen . . . 192
9.4 Zusammenfassung . . . 196

III Beispiele und Referenzarchitektur 197

10 Konkrete Organisationseinheiten 199
10.1 Vorbereitung . . . 200
10.2 Organisationseinheiten nach SONAR . . . 201
10.2.1 Der SONAR-Ansatz: Grundlegende Konzepte und Modell-Dimensionen . 202
10.2.2 Ausgestaltung von Organisationseinheiten gemäß SONAR . . . 210
10.3 Organisationseinheiten nach $\mathcal{M}$OISE$^+$ . . . 224
10.3.1 Der $\mathcal{M}$OISE$^+$-Ansatz: Grundlegende Konzepte und Modelldimensionen 224
10.3.2 Organisationseinheiten gemäß $\mathcal{M}$OISE$^+$ . . . 231
10.4 Organisationseinheiten nach ISLANDER . . . 235
10.4.1 Der ISLANDER-Ansatz: Grundlegende Konzepte und Modelldimensionen 236
10.4.2 Organisationseinheiten gemäß ISLANDER . . . 241
10.5 Diskussion: Vergleich und hybride Kombination . . . 247
10.6 Zusammenfassung . . . 250

11 Referenzarchitektur Organ für Multiorganisationssysteme 251
11.1 Architekturgerüst . . . 252
11.2 Organisationseinheitstyp *Abteilung* . . . 257
11.2.1 Organisationstheoretischer Bezug . . . 257
11.2.2 Abteilungsmodell . . . 258
11.3 Organisationseinheitstyp *Organisation* . . . 260
11.3.1 Organisationstheoretischer Bezug . . . 260
11.3.2 Organisationsmodell . . . 261
11.4 Organisationseinheitstyp *Organisatorisches Feld* . . . 262
11.4.1 Organisationstheoretischer Bezug . . . 263
11.4.2 Modell eines organisatorischen Feldes . . . 264
11.5 Organisationseinheitstyp *Gesellschaft* . . . 265
11.5.1 Organisationstheoretischer Bezug . . . 265

11.5.2 Gesellschaftsmodell 266
11.6 Diskussion: Einsatz der Referenzarchitektur 267
11.6.1 Softwaretechnische Implikationen 267
11.6.2 Multiperspektivischer Einsatz der Referenzarchitektur 270
11.6.3 Konkrete Entwicklungsansätze für Organisationseinheitstypen 271
11.7 Zusammenfassung 272

12 Schlussbetrachtung **273**
12.1 Zusammenfassung und Ergebnisse 273
12.1.1 Kapitelübersicht 273
12.1.2 Veröffentlichungen 279
12.2 Ausblick 280

A Ausführbare Referenznetze **283**

Literaturverzeichnis **287**

Abbildungsverzeichnis

1.1 Grundlegendes janusgesichtiges Konzept einer Organisationseinheit 18

4.1 Zusammenspiel von Machtverteilung, Zielsetzung und Kontrollsystemen 63

5.1 Kanonische Sichtweise auf Multiagentensysteme nach Jennings [Jen00] 76
5.2 Klassifikation von Organisationsansätzen für Multiagentensysteme, adaptiert von Boissier et. al. [BHS07] 81
5.3 Allgemeine Architektur für Organisationsinfrastrukturen in offenen Multiagentensystemen nach Hübner et. al. [HBKR10] 85

6.1 Ein einfaches S/T-Netz vor und nach dem Schalten der Transition `t1` 99
6.2 Beispielnetz: Herstellung und Versand eines Produktes 100
6.3 Ein gefärbtes Petrinetz vor und nach dem Schalten der Transition `t1` 101
6.4 Schalten der Transition `t1` unter Einbezug von Variablenanschriften 102
6.5 Bepacken einer Kiste über einen synchronen Kanal 105
6.6 Bepacken einer Kiste über zwei synchrone Kanäle 106
6.7 Verschiedene Typen von Kanten in Renew 108
6.8 Verwendung einer virtuellen Stellen 109
6.9 Multiagentensystemarchitektur Mulan nach [Röl04, S. 157] 111
6.10 Sonar-Architektur als Interpretation von Mulan, nach [KB09, S. 451] 112
6.11 Drei Abstraktionsstufen der Referenznetzmodellierung in dieser Arbeit 113

7.1 Kanonische Sichtweise auf Multiorganisationssysteme 118
7.2 Grundlegendes Verständnis von Organisationseinheiten 123
7.3 Konzeptionelle Richtungen bei der Betrachtung von Organisationseinheiten . . 124

8.1 Einfache Kollektivitätseinheit und exemplarische Einbettung von Akteuren . . . 129
8.2 Exemplarische Einbettung eines individuellen Akteurs 131
8.3 Ausschnitt einer komplexeren Kollektivitätseinheit 132
8.4 Exemplarische Einbettung einer Kollektivität als Akteur 135
8.5 Grundprinzip der Aktionsrekursion in Multiorganisationssystemen 137
8.6 Mehrfache Einbettung einer Kollektivität 138
8.7 Abstraktion von den Aktivitätsabläufen aus Abbildung 8.3 139
8.8 Einbettung kollektiver Akteure am Beispiel (vergleiche Abbildung 8.4) 140
8.9 Einbettung kollektiver Akteure: allgemeines Schema 141
8.10 Kombination von AND/XOR-Konfigurationen (vergleiche Abbildung 8.6) . . . 142
8.11 Einbettungssituationen in Multiorganisationssystemen 143
8.12 Exemplarischer Übergang zu abstrakter Mengendarstellung 144
8.13 Alternative Verfeinerung von Perspektive (3) aus Abbildung 8.12 147
8.14 Multiorganisationssystem basierend auf Akteurs-/Aktivitätsmengen-Modellen . 148

8.15 Organisationseinheit intern: Governance, Integration, Operation 150
8.16 Konzeptdiagramm für das Zusammenspiel von Operation, Integration und Governance 151
8.17 Organisationseinheit: Peripherieaktivitäten 154
8.18 (a) Direkter Durchgriff, (b) Pufferung 155
8.19 Peripherieaktivitäten mit Integrations- oder Governance-Charakter 156
8.20 Unterschiedliche analytische Interpretationen von Aktivitäten am Beispiel . . . 157
8.21 Exemplarische Kombination der abstrakten Modellarten 158

9.1 Operationales Grundmodell einer Kollektivitätseinheit 161
9.2 Ausführung einer Aktion mit Effekten auf mehreren Systemebenen 162
9.3 Operationale Details für interne Aktionen 164
9.4 Operationale Details für periphere Aktionen 165
9.5 Operationale Details der Aktivitätsaktivierung 166
9.6 Netzkomponenten für die Aktivitätskomposition 167
9.7 Netzkomponente für Aktionen individueller Akteure 167
9.8 Beispielaktivität: send-receive 169
9.9 Periphere Aktionen in einer beispielhaften Aktivitätsverzahnung 172
9.10 Erweiterung des Kollektivitätseinheitsmodells um mobile Akteursreferenzen . . 174
9.11 Migration eines Akteurs „nach unten/innen“ 176
9.12 Expansion eines Akteurs 177
9.13 Gezielte Aktivitätsaktivierung über Akteursidentifikatoren 179
9.14 Periphere Aktionen mit doppelter Beschreibung 179
9.15 Periphere Aktivierungsaktion 180
9.16 Externe Aktivitätsaktivierung 181
9.17 Technische Realisierung gemeinsamer Aktionen 182
9.18 Erstes operationales Modell einer Organisationseinheit 183
9.19 Schnittstelle zum Systemregelnetz 186
9.20 Erzwungene Akteursmobilität (interner Fall) 188
9.21 Trennung von Systemregeln in Spezifikation und Zustandsbeschreibung 190
9.22 Duale Systemregeln mit externem Durchgriff 191

10.1 Einbettung einer tuProlog-Regelbasis 201
10.2 Eine Allianz als Delegationsnetz 202
10.3 Die Dienstnetze PC und MMAC 204
10.4 Mögliche Teams zum Delegationsnetz aus Abbildung 10.2 207
10.5 Organisatorische Transformationen über stratifizierte SONAR-Modelle 208
10.6 MULAN4SONAR Middleware-Konzept 209
10.7 Erstes Grobmodell einer SONAR-Organisationseinheit 210
10.8 Fragment der tuProlog-Regelbasis einer SONAR-Organisationseinheit 213
10.9 Bearbeitung der initialen Aufgabe {producer, consumer} aus Abbildung 10.2 . . 214
10.10 Umsetzung stratifizierter SONAR-Organisationen 216
10.11 Einbindung sozialer Akteure als Governance-Akteure 218
10.12 Verfeinerung der Position contracting-supplier 219
10.13 Das Dienstnetz MAC 220
10.14 Verzahnte Teamplanausführung über Systemebenen hinweg (1) 222
10.15 Verzahnte Teamplanausführung über Systemebenen hinweg (2) 223

10.16$\mathcal{M}$OISE$^+$ Modelldimensionen an einem Beispiel . . . 226
10.17Reorganisations-Gruppe in $\mathcal{M}$OISE$^+$ (ursprünglich von Hübner et. al. [HSB04] und hier angepasst) . . . 230
10.18$\mathcal{S}$-$\mathcal{M}$OISE$^+$ Middleware-Konzept . . . 230
10.19Grobmodell einer $\mathcal{M}$OISE$^+$-Organisationseinheit . . . 232
10.20Überführung von $\mathcal{M}$OISE$^+$-Plänen in Interaktionsmuster . . . 233
10.21ISLANDER Modelldimensionen an einem Beispiel . . . 237
10.22AMELI Middleware-Konzept . . . 241
10.23Grobmodell einer ISLANDER-Organisationseinheit . . . 242
10.24Aktivitätsnetz zur Auktionsszene aus Abbildung 10.21 . . . 244
10.25Skizze eines Multiorganisationssystems auf Basis von ISLANDER-Organisationseinheiten . . . 246

11.1 *Top-Down*- und *Bottom-Up*-Prozesse zwischen institutionellen Systemebenen in sozialen Gesellschaften, übernommen von Scott [Sco08, S. 192] . . . 253
11.2 ORGAN-Referenzarchitektur für Multiorganisationssysteme . . . 255
11.3 Exemplarischer Auszug der realweltlichen Analogie zu ORGAN-Softwaresystemen 256
11.4 Abstraktes Modell einer Abteilungs-Organisationseinheit . . . 258
11.5 Abstraktes Organisationseinheits-Modell einer Organisation . . . 261
11.6 Abstraktes Organisationseinheits-Modell eines organisatorischen Feldes . . . 264
11.7 Abstraktes Organisationseinheits-Modell der Gesellschaft . . . 266

A.1 Ausführbares Grundmodell einer Kollektivitätseinheit . . . 283
A.2 Ausführbares Grundmodell einer Organisationseinheit (1) . . . 284
A.3 Ausführbares Grundmodell einer Organisationseinheit (2) . . . 285
A.4 Ausführbares Systemregelnetz mit tuProlog-Regeln . . . 286

1 Einleitung

In der vorliegenden Arbeit beschäftigt sich der Autor mit der Modellierung komplexer Softwaresysteme. Dabei wird die Herangehensweise verfolgt, Konzepte und Mechanismen aus dem Bereich sozialer Organisationen auf die Gestaltung von Softwaresystemen zu übertragen. Die damit verknüpften Anschlusspunkte sind vielfältig und interdisziplinär. Neben einer allgemeinen Motivation des Vorhabens der Arbeit werden daher an dieser Stelle Vorüberlegungen zur konkreten Zielsetzung und inhaltlichen Eingrenzung vorgenommen.

1.1 Motivation

Informatiker setzen es sich nicht als vorrangiges Ziel, komplexe Softwaresysteme zu entwickeln. Komplexe Softwaresysteme entstehen aus Notwendigkeit oder sogar unabsichtlich. Der Grund dafür liegt häufig in der engen Verzahnung der („virtuellen“) Softwaresysteme mit ihren („realweltlichen“) Umgebungen. Diese Umgebungen schließen vor allem soziale Systeme ein. Die zunehmende Durchdringung des sozialen Lebens mit Informationstechnologie führt zu softwareintensiven, sozio-technischen Gesamtsystemen. Die Ko-Evolution von softwaretechnischen und sozialen Systemen ist durch reziproke Einflüsse gekennzeichnet. Anhand der Einschätzung von Keating et. al. [KRU+03, S. 36] (die sich auf ein Zitat von Beer [Bee79] beziehen) lässt sich die Kritik ablesen, (Informations-)Technologie trage weniger zur Beherrschung komplexer Systeme bei, sondern stelle vor allem eine eigene Quelle der Komplexität dar.

> „Beer's (1979) observation about complex systems still holds – we are dealing with a jigsaw puzzle that is about five miles across and we are standing on the ground trying to see how to fit it together. Since Beer's observation unprecedented explosive growth has taken place in technology, artifacts produced by technology, and information access has risen exponentially. However, we seem no more able to address Beer's puzzle than when he identified it over 20 years ago. In fact, it appears that we now have a problem centered on integrating multiple five-mile diameter jigsaw puzzles that change as we try to fit the pieces together.“

Northrop et. al. [Nor06, S. ix] sehen die wesentliche Herausforderung im Umgang mit zukünftigen Softwaresystemen in der Kombination von drei Aspekten: (1) die stetig zunehmende Größe von Softwaresystemen (Codegröße als auch Anzahl interagierender Softwareeinheiten betreffend), (2) ihr Charakter als Systeme von Systemen und (3) die Notwendigkeit sozio-technischer Integration. Das Konzept eines Systems von Systemen geht dabei mit der Unabhängigkeit/Eigenständigkeit von Teilsystemen eines Gesamtsystems einher.

In diesem Zusammenhang ergeht die Forderung von Northrop et. al. [Nor06, S. 4] nach neuen beziehungsweise innovativen *konzeptionellen Modellen* für das Verständnis und die Gestaltung von Softwaresystemen. Die Erstellung von Softwaresystemen ist maßgeblich durch das verwendete *Entwicklungsparadigma* beeinflusst. Ein Paradigma ist vor allem an Konzepten orientiert, die den grundsätzlichen Aufbau von Softwaresystemen übergreifend über konkrete Systeme

prägen.[1] Duvigneau [Duv09, S. 13] spricht hier von *architekturprägenden Konzepten*. Diese werden auch als *Metaphern* bezeichnet. Beispiele für solche Metaphern sind Objekte, Plugins, Dienste oder Agenten.

Insbesondere die Agentenorientierung zeichnet sich bereits durch eine gute Eignung aus, den von Northrop et. al. genannten Herausforderungen zu begegnen. Das Konzept der Unabhängigkeit/Eigenständigkeit von Teilsystemen – hier den Agenten – ist einem Multiagentensystem inhärent. Darüber hinaus wird sozio-technische Integration durch die Verwandtschaft beziehungsweise Übereinstimmung von Konzepten und Terminologie auf softwaretechnischer und sozialer Seite unterstützt (vergleiche Cabac [Cab07]). Agenten sind dem Konzept des menschlichen Akteurs nachempfunden und werden häufig über ihre Ziele, ihr Wissen und ihre Pläne spezifiziert. Auch überindividuelle Strukturen eines Multiagentensystems sind sozialen Konzepten nachempfunden. Als besonders prominent hat sich die Herangehensweise herausgebildet, globale Systemstrukturen eines Multiagentensystems in Form von *Agentenorganisationen* zu gestalten. Die Spezifikation solcher Agentenorganisationen erfolgt in Form von Konzepten, die dem Bereich sozialer Organisationen entliehen werden: Positionen, Rollen, Dienste, Autorität, Delegation, Normen, Ablaufpläne, Gruppen, Teams etc. Auf diese Weise ergibt sich von Seiten des sozialen Systems aus ein intuitiver Zugang zu Aufbau und Funktionsweise des softwaretechnischen Systems.

Problematisch gestaltet sich die Eignung der Agentenmetapher für die Entwicklung großer, komplexer Systeme mit multiplen Systemebenen. Eine wichtige Erkenntnis der allgemeinen Systemtheorie ist, Hierarchie sei eine fundamentale Eigenschaft *aller* komplexen Systeme. Hierarchie ist hier im Sinne von Clustering (anstatt von Autorität) zu verstehen: Systeme einer Systemebene *bestehen* aus Systemen der darunter liegenden Systemebene. So hebt beispielsweise Simon [Sim62] die Bedeutung *stabiler Zwischenformen* für die Entwicklung komplexer Systeme hervor. Besonders plastisch sind die Ausführungen von Koestler [Koe67, S. 48] zu *Holonen* als Teile einer Systemhierarchie und dem dabei auftretenden *Janus-Effekt*. In der römischen Mythologie ist Janus der Gott der Durchgänge und Tore, der Anfänge und der Enden. Ein auffallendes Merkmal dieser Gottheit sind seine zwei Gesichter, die in entgegengesetzte Richtungen blicken. Wie Janus haben auch Holone als Zwischenstufen einer Systemhierarchie zwei Seiten, die in entgegengesetzte Richtungen blicken. Die erste Seite blickt in der Hierarchie nach „unten/innen“ und repräsentiert die Ganzheit der Entität in der Hierarchie. Als ein Ganzes ist die Entität in sich geschlossen, stabil und vollständig. Die andere Seite schaut in der Hierarchie als ein Teil eines Größeren nach „oben/außen“. In dieser Hinsicht zeigt die Entität den Willen zur Kooperation, um Teil eines größeren Ganzen zu werden.

Die Agentenmetapher unterstützt diese Sichtweise zumindest nicht inhärent. Nach Boealla et. al. [BvdT05] ist der *Komponentencharakter* des softwaretechnischen Konzeptes eines Objektes (Objekte können auf rekursive Weise aus anderen Objekten zusammengesetzt sein) beim Übergang zum Agenten verloren gegangen. Das Agentenkonzept ist dem Menschen als individuellem beziehungsweise „natürlichem“ Akteur nachempfunden, so dass sich eine Schachtelung nicht anbietet. Auch die weiter oben erwähnten Ansätze, überindividuelle Strukturen als Agentenorganisationen zu gestalten, schafft zunächst keine Abhilfe. Agentenorganisationen werden vornehmlich als *organisatorische Kontexte* verstanden, die das Verhalten der teilnehmenden Agenten rahmen (leiten, einschränken). Auch organisationsinterne Strukturelemente wie Gruppen und Teams werden vor allem als (Sub-)Kontexte verstanden. Die Agentenor-

[1] Im allgemeinen Fall beschreibt Kuhn [Kuh62, S. 10] Paradigmen als *„models from which spring particular coherent traditions of scientific research“*.

ganisation bildet mithin den oberen Systemabschluss. Damit wird allerdings das Potential, welches das Konzept der sozialen Organisation für den Bau von Multiagentensystemen birgt, nur zum Teil ausgenutzt. Neben ihrer Funktion als Kontexte sind soziale Organisationen *kollektive Akteure*. Scott [Sco03, S. 7+8] hebt diesen Aspekt besonders hervor und bezeichnet Organisationen als die primären Akteure in Gesellschaften moderner Prägung. Noch allgemeiner lässt sich feststellen, dass eine organisationsorientierte Betrachtung sozialer Gesellschaften multiple Ebenen des kollektiven Handelns und damit mehrere Formen kollektiver Akteure mit sich bringt. Unterhalb der Ebene der Organisation finden sich beispielsweise Abteilungen und Gremien. Oberhalb dagegen finden sich beispielsweise virtuelle Organisationen, strategische Allianzen, organisatorische Felder oder gesellschaftliche Institutionen. In der Folge der Arbeit wird der Oberbegriff der *organisierten Kollektivität* für diese Entitäten eingeführt. Insgesamt ergibt sich, dass die Organisationstheorie einen großen Fundus an Strukturen und Mechanismen bietet, die eine holonische Betrachtung von Multiagentensystemen nahe legt.

Daher ist es nicht überraschend, dass Bestrebungen innerhalb der Multiagentensystemforschung bestehen, das „klassische“ Verständnis von Softwareagenten als individuelle Akteure zu erweitern, um auch kollektive Akteure einzubeziehen. Beispielsweise hat sich das Konzept des Holons auch in der Agentenforschung festgesetzt (vergleiche beispielsweise Fischer et. al. [FSS03]). Nach Ansicht des Autors der vorliegenden Arbeit stellen die Bestrebungen einen Schritt in Richtung eines Paradigmenwechsels dar. Ein großer Teil der Multiagentensystemforschung geht mit einer kontinuierlichen Verschiebung des Fokus vom individuellen Agenten zur Agentenorganisation oder sogar zum kollektiven Akteur einher. Allerdings wird der Schritt nicht in letzter Konsequenz vollzogen. Beispielsweise handelt es sich bei existierenden Ansätzen für kollektive Akteure vor allem um Möglichkeiten der *technischen Realisierung* von kollektivem Akteurswesen und damit um Möglichkeiten, kollektive Akteure mit aktueller Agententechnologie zu implementieren. Fragen der *inhaltlichen Realisierung* kollektiver Akteure, die den Übergang von internen Konfigurationen einer Agentenorganisation zu ihrer Positionierung in der Umgebung und den damit einhergehenden kollektiven Zielen und Handlungen betreffen, werden vernachlässigt.

Darauf aufbauend wird in der vorliegenden Arbeit ein Paradigmenwechsel von der Agentenorientierung zur Organisationsorientierung vorgeschlagen. Für die Entwicklung von Softwarekomponenten steht nicht länger die Metapher des individuellen Akteurs, sondern die der Organisation beziehungsweise der organisierten Kollektivität im Fokus. In der Arbeit wird in diesem Zusammenhang ein konkreter Vorschlag für Softwaresysteme basierend auf sogenannten *Organisationseinheiten* präsentiert. Jede Organisationseinheit ist einerseits eine in sich geschlossene Systemorganisation und andererseits eine Einheit, die als Baustein umfassender Strukturen (wiederum Organisationseinheiten) verwendet werden kann. Anders ausgedrückt und um den sozialen Bezug deutlicher herauszustellen, ist jede Organisationseinheit sowohl ein organisatorischer Kontext als auch ein organisatorischer Akteur. Abbildung 1.1 illustriert das grundlegende janusgesichtige Konzept einer Organisationseinheit.[2]

Die Verwendung von Organisationseinheiten als Softwarebausteine führt nach Ansicht des Autors der vorliegenden Arbeit zu einer konzeptionellen Erweiterung von Multiagentensystemen. Die entstehenden Systeme werden daher als *Multiorganisationssysteme* bezeichnet. Im Gegensatz zu Multiagentensystemen weisen sie einen inhärent rekursiven Charakter auf. Die damit einhergehende Versprechung ist ein umfassenderes Ausschöpfen des Potentials des sozialen Organisationskonzeptes für die Gestaltung von Softwaresystemen. Damit werden die weiter

[2] Quelle des Janus-Bildes: http://upload.wikimedia.org/wikipedia/commons/f/f4/Janus-Vatican.JPG

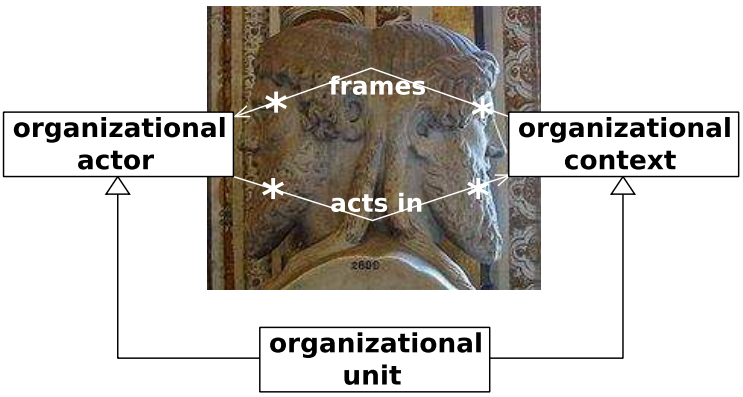

Abbildung 1.1: Grundlegendes janusgesichtiges Konzept einer Organisationseinheit

oben aufgeführten Stärken des Multiagentensystemansatzes (Entwicklung von Systemen von Systemen als Systeme aus autonomen Akteuren, Unterstützung sozio-technischer Integration durch Verwandtschaft/Übereinstimmung von Konzepten auf beiden Seiten) verstärkt und auf großangelegte Systeme von Systemen mit multiplen Systemebenen ausgeweitet. Zuvor bereits zitiert wurde die Forderung von Northrop et. al. nach innovativen konzeptionellen Modellen für das Verständnis und die Gestaltung von großangelegten Softwaresystemen der Zukunft. Nach Ansicht des Autors der vorliegenden Arbeit wird mit dem Konzept des Multiorganisationssystems genau ein solches konzeptionelles Modell geliefert.

1.2 Vorbereitende Überlegungen zum Ziel und Inhalt der Arbeit

Analog zu den bereits existierenden und etablierten Begriffen der objektorientierten, dienstorientierten oder agentenorientierten Softwaresysteme werden in dieser Arbeit *organisationsorientierte Softwaresysteme* betrachtet. Damit einher geht die Gestaltung der Struktur und des Verhaltens von Softwaresystemen anhand von sozialen Organisationskonzepten. Es geht also darum, Bauprinzipien, die sich in sozialen Organisationssystemen als leistungsfähig erwiesen haben, auf Softwaresysteme im Großen zu übertragen. Daraus resultieren Softwaresysteme, deren Komponenten sozial motiviert und fundiert sind. So entstehen Softwarekomponenten, die softwaretechnische Entsprechungen von Akteuren, Teams, Abteilungen, Unternehmen, strategischen Allianzen oder Märkten sind.

Das Konzept des organisationsorientierten Softwaresystems ist bislang nicht einheitlich gesetzt oder etabliert. Bisher finden sich in der Literatur vor allem erste Bemühungen der terminologischen Prägung des Konzeptes. Diese Bemühungen entstammen vornehmlich dem Forschungsbereich der Multiagentensysteme. Hier finden sich beispielsweise folgende Formulierungen: *Organization-Centred Multi-Agent Systems* [FGM03], *Organization-Based Systems* [DeL05], *Organisation-Oriented Software* [KNMB06], *Organization-Oriented Programming* [BHS07]. Der Autor der vorliegenden Arbeit hat zu diesen Bemühungen einen Artikel mit dem Titel *„Towards Organization-Oriented Software Engineering“* [WEMRM07] für die Software Engineering Konferenz 2007 beigetragen. Im Folgenden werden die konkreten Ziele für die vorliegende Arbeit in diesem Kontext beschrieben.

Das erste Ziel besteht darin, eine ausführliche Motivation für die Notwendigkeit beziehungsweise Angemessenheit eines Organisationsparadigmas der Softwareentwicklung vorzunehmen. Ein Beweis der Notwendigkeit/Angemessenheit kann jedoch nicht geführt werden, so dass die Argumentation für ein Organisationsparadigma hier vor allem qualitativ geführt wird. Zudem kann das Aufkommen eines Paradigmas eigentlich erst in Retrospektive als solches wahrgenommen und bewertet werden. Etwas schwächer formuliert kann man die in der Arbeit vorgeschlagene Sichtweise als eine Weitung des agentenorientierten Blickwinkels auf Softwaresysteme auffassen. In der Folge der Arbeit wird dennoch von einem Organisationsparadigma die Rede sein, denn es ist die Überzeugung des Autors, eine konsequente Organisationsorientierung könne in Zukunft zu einer eigenständigen Richtung der Softwareentwicklung führen. Daher wird die Motivation für ein Organisationsparadigma auch losgelöst von der aktuellen Multiagentensystemforschung vorgenommen. Neben der Motivation eines Organisationsparadigmas gilt es, die Kernkonzepte und -mechanismen zu identifizieren, die für diesen Zweck aus dem Bereich sozialer Organisationen herangezogen und in den Bereich der Entwicklung von Softwaresystemen überführt werden.

Die grundsätzliche Motivation eines Organisationsparadigmas und die Benennung von besonders wichtigen sozialen Organisationskonzepten bieten zunächst nur eine vage Vorstellung davon, wie organisationsorientierte Softwaresysteme tatsächlich aussehen. Dementsprechend ist der Hauptteil der Arbeit dem zweiten Ziel gewidmet, nämlich der Vermittlung eines klaren Verständnisses von organisationsorientierten Softwaresystemen und der Bereitstellung von Mitteln für die Modellierung der Systeme. Dabei geht mit der vorliegenden Arbeit ausdrücklich *nicht* der Anspruch einher, die eine oder die endgültige Definition von organisationsorientierten Softwaresysteme vorzunehmen. Stattdessen wird eine *mögliche* Konzeptionalisierung von organisationsorientierten Softwaresystemen vorgenommen. Diese basiert auf den im vorigen Abschnitt skizzierten Organisationseinheiten, die zu Multiorganisationssystemen als einer konkreten Form von organisationsorientierten Softwaresystemen führen.

Das zweite Ziel der Arbeit wird in Form von drei Teilzielen verfolgt: (1) Bereitstellung von Modellierungstechniken für Organisationseinheiten und Multiorganisationssysteme, (2) Erarbeitung universell einsetzbarer Grundmodelle von Organisationseinheiten und (3) Vorstellung eines Architekturkonzeptes für Multiorganisationssysteme auf Basis der Grundmodelle. Der Schwerpunkt der Arbeit liegt damit auf Modellierung. Dabei werden Modelle in Form von Diagrammen grafisch expliziert. Zunächst wird mit den grafischen Modellen der Zweck verfolgt, einen intuitiven Zugang zu Struktur und Verhalten von Organisationseinheiten und Multiorganisationssystemen auf Basis anschaulicher Illustrationen zu bieten. Dies ist die übliche Verwendung grafischer Modelle. Darüber hinaus wird in dieser Arbeit aber ein weiterer Zweck verfolgt: Es werden Modelle mit einer unterliegenden operationalen Semantik erstellt. Zunehmende Modellverfeinerung und -vereinheitlichung führen zu unmittelbar ausführbaren Modellen. Damit ergibt sich eine hohe Relevanz der Modelle für die Softwareentwicklung. Die ausführbaren Modelle vermitteln ein elaboriertes technisches Verständnis für die Funktionsweise von Organisationseinheiten und Multiorganisationssystemen. Sie leiten direkt auf Möglichkeiten der Implementierung hin und stellen selbst bereits voll funktionsfähige Prototypen dar. Rölke [Röl04, S. 247] betont die Bedeutung solcher prototypischen Implementierungen für die Klärung der Frage, welche Konzepte für ein Paradigma jeweils essentiell sind.

Um dem Anspruch an die Kombination aus Anschaulichkeit und operationaler Semantik bis hin zur Ausführbarkeit gerecht zu werden, ist die Wahl auf Petrinetze und insbesondere auf höhere Petrinetzformalismen als Basis aller in der Arbeit vorgestellten Modellierungstechniken und Grundmodelle gefallen. Diese Herangehensweise hat sich bereits an verwandter

Stelle als erfolgreich erwiesen. Moldt [Mol96] verwendet Petrinetze für die Erstellung anschaulicher sowie ausführbarer Modelle von Objekten und Objektsystemen. Rölke [Röl04] nimmt eine analoge Herangehensweise an die Modellierung von Agenten und Multiagentensystemen vor (Cabac [Cab10] bettet die ausführbaren Modelle Rölkes sogar in einen vollständigen und praxiserprobten Softwareentwicklungsansatz ein). Duvigneau [Duv09] verfolgt wiederum einen analogen Ansatz bei der Modellierung von Plugin-Systemen. Köhler-Bußmeier [KB09] verwendet Petrinetze für die Modellierung von Agentenorganisationen und hat unter Mitwirkung des Autors der vorliegenden Arbeit [KBWEM10, KBWE09b] Verfeinerungen der Modelle hin zur direkten Ausführbarkeit vorgenommen.

Eine besondere Stellung nimmt in diesem Zusammenhang die Petrinetzmodellierung unter Einbezug des Konzeptes der *Netze in Netzen* nach Valk [Val87, Val98, Val04] ein. Das Konzept der Netze in Netzen bietet die Möglichkeit der Modellierung von dynamischen Systemstrukturen auf mehreren Ebenen sowie von ebenenübergreifenden Synchronisationsmechanismen. Ganz konkret lässt sich festhalten, dass das Modellierungskonzept der Netze in Netzen dem Systemkonzept der Systeme von Systemen sehr nahe kommt und damit eine ideale Grundlage für die vorliegende Arbeit darstellt. In diesem Zusammenhang ergibt sich ein weiteres Ziel der Arbeit. Durch die spezifische Verwendung von Petrinetzen für die Modellierung von Organisationseinheiten und Multiorganisationssystemen werden konkrete Richtlinien für die Erstellung von Modellen und für die Kombination von Modellen erarbeitet. In Form solcher *organisationsorientierten Petrinetzmodelle* wird ein eigener Beitrag für die Petrinetzmodellierung geleistet (analog zu den zitierten Beiträgen in Form von objektorientierten, agentenorientierten und Plugin-basierten Petrinetzmodellen).

Die in der Arbeit verfolgten Ziele werden anhand der folgenden Aufzählung noch einmal zusammengefasst.

1. **Motivation eines Organisationsparadigmas der Softwareentwicklung**
   a) Allgemeine Motivation der Notwendigkeit beziehungsweise der Angemessenheit eines Organisationsparadigmas der Softwareentwicklung
   b) Identifizierung von Kernkonzepten und -mechanismen aus dem Bereich sozialer Organisationen und Organisationssysteme
2. **Vermittlung eines klaren Verständnisses von organisationsorientierten Softwaresystemen und Bereitstellung von Mitteln für die Modellierung**
   a) Bereitstellung von Modellierungstechniken für Organisationseinheiten und Multiorganisationssysteme (als konkreter Form von organisationsorientierten Softwaresystemen)
   b) Erstellung intuitiv zugänglicher Grundmodelle von Organisationseinheiten (und der darauf aufbauenden Multiorganisationssysteme) unter Berücksichtigung der identifizierten Kernkonzepte und -mechanismen aus dem Bereich sozialer Organisationen
   c) Unterbreitung eines Vorschlags für eine Referenzarchitektur von Multiorganisationssystemen auf Basis der erstellen Grundmodelle
   d) Durchgängige Fundierung aller Modellierungstechniken und Modelle auf operationaler Petrinetzsemantik, aufgrund derer ein sanfter Übergang von abstrakten zu direkt ausführbaren Modellen möglich ist
   e) Leistung eines Beitrags für die Petrinetzmodellierung in Form organisationsorientierter Petrinetzmodelle

Mit dem nun klarer skizzierten Vorhaben dieser Arbeit sind verschiedene Anschlusspunkte verknüpft. Von besonderer Bedeutung sind nach Ansicht des Autors die folgenden Punkte. Allerdings werden sie im Kontext der Arbeit nicht vertieft diskutiert, da es sich um eigenständige und umfangreiche Themen handelt. Der Autor sieht sie als tragende Säulen zukünftiger Forschungsbestrebungen. Um das Vorhaben der Arbeit klar abzugrenzen und zu positionieren, werden die Punkte hier knapp erläutert. Speziell die ersten beiden Punkte werden dabei im Laufe der Arbeit wiederholt aufgegriffen, um die Verbindungen aufzuzeigen.

1. *Modellierung von sozialen Systeme beziehungsweise von sozio-technischen Gesamtsystemen:* Mit dem Konzept des organisationsorientierten Softwaresystems ist in dieser Arbeit die Gestaltung von Softwaresystemen *selbst* unter Einbezug von Konzepten aus dem Bereich sozialer Organisationen assoziiert. Eine alternative Lesart ist die eines Softwaresystems, dessen Gestaltung an Anforderungen sozialer/realweltlicher Organisationen orientiert ist. Wenn notwendig, werden die beiden Lesarten in der Folge der Arbeit anhand der Charakterisierungen von „Softwaresystemen *als* Organisationen“ gegenüber „Softwaresystemen *für* Organisationen“ unterschieden. In der Arbeit wird an verschiedenen Stellen die Hypothese aufgegriffen, die Übereinstimmung von Konzepten und Terminologie auf softwaretechnischer und sozialer Seite würde eine sozio-technische Integration fördern (die Hypothese wird auch an anderer Stelle vertreten, vergleiche Cabac [Cab07] oder Bäumer [Bäu98]). Eine systematische und ausführliche Untermauerung der Hypothese wird jedoch nicht vorgenommen. Doch für die Zukunft stellt sich insbesondere die Frage, ob die in der Arbeit vorgestellten Modellierungstechniken und Grundmodelle für eine *gemeinsame/integrative* Modellierung von softwareintensiven, sozio-technischen Gesamtsystemen geeignet sind.

2. *Praktische Verwendung der Ergebnisse der Arbeit für die Entwicklung realer Softwareanwendungen:* Die Relevanz der in der Arbeit erstellten Modellierungstechniken und Modelle für die Softwareentwicklung wurde weiter oben betont. Allerdings ist festzuhalten, dass die praktische Einsetzbarkeit für die Entwicklung realer großangelegter Softwaresysteme nicht im Fokus steht. In erster Linie stehen hier klare Modellkonzepte und deren Orthogonalität im Mittelpunkt. Damit werden (teilweise sehr detaillierte, technische) *Modelle* der Funktionsweise von Organisationseinheiten und Multiorganisationssystemen geliefert. Die Übertragung der modellierten Konzepte und Mechanismen auf unterschiedliche (verteilte) Softwareumgebungen bedarf gegebenenfalls der Anpassung und eventueller Eingeständnisse. Darüber hinaus liefern die verschiedenen vorgestellten Modellierungstechniken eine Grundlage für die zukünftige Entwicklung einer *Sprache* für organisationsorientierte Softwaresysteme. Diese Entwicklung wird mit der vorliegenden Arbeit allerdings erst begonnen.

3. *Modellverifikation mit formalen Mitteln der Petrinetztheorie:* Die Verwendung von Petrinetzen als Modellierungstechnik führt zur Frage nach Verifizierbarkeit von erstellten Modellen auf Basis der Petrinetztheorie. Für die Erstellung der in dieser Arbeit vorgestellten Netze stand die Analysierbarkeit allerdings von vornherein nicht im Fokus. In erster Linie geht es darum, die volle Mächtigkeit des verwendeten Modellierungsformalismus auszunutzen. Für die Zukunft können vor allem die Arbeiten und Ergebnisse von Michael Köhler-Bußmeier (die mit seiner Dissertation [Köh04] ihren Anfang nahmen und fortdauern) zu Verifikationsverfahren für Objektnetzformalismen herangezogen werden.

1.3 Übersicht

Auf der gröbsten Ebene ist die Arbeit in drei Teile aufgeteilt: *Grundlagen und Forschungskontext*, *Modellierungstechniken und Grundmodelle* sowie *Beispiele und Referenzarchitektur*. Der erste Teil ist dabei vor allem dem ersten übergeordneten Ziel der Arbeit gewidmet (Motivation eines Organisationsparadigmas). Zusätzlich werden hier der *State of the Art* beleuchtet und Möglichkeiten der Petrinetzmodellierung als Grundlage für den weiteren Verlauf der Arbeit vorgestellt. Der zweite und dritte Teil sind gemeinsam dem zweiten übergeordneten Ziel gewidmet (Vermittlung eines klaren Verständnisses von organisationsorientierten Softwaresystemen, Mittel der Modellierung, Referenzarchitektur).

Der erste Teil ist in fünf Kapitel unterteilt. In Kapitel 2 werden Charakteristika und Problemstellungen von Softwaresystemen im Großen als Systeme von Systemen behandelt. Diese dienen in Kapitel 3 als Grundlage für die Motivation eines Organisationsparadigmas der Softwareentwicklung. Während im Zuge der Motivation die grundsätzlichen Versprechungen eines Organisationsparadigmas herausgestellt werden, bleiben die konkreten Konzepte und Mechanismen, die in diesem Zusammenhang für den Bau von Softwaresystemen herangezogen werden sollen, weitestgehend vage. Einer klaren Identifizierung dieser Konzepte und Mechanismen ist Kapitel 4 gewidmet. Ein besonderer Fokus liegt dabei auf der Betrachtung von kollektivem Akteurswesen. Auf Basis der Kapitel 3 und 4 erfolgt eine kritische Untersuchung des *State of the Art* im Bereich der organisationsorientierten Softwaresysteme in Kapitel 5. Dabei werden vor allem Defizite hinsichtlich der verwendeten Organisationskonzepte im Hinblick auf ihre Janusgesichtigkeit offengelegt. Der erste Teil endet mit Kapitel 6 und einer Vorstellung von Modellierungstechniken aus dem Bereich der Petrinetze und einer Herausstellungen ihrer Eignung und Bewährtheit für die informatiknahe Modellierung sozialer Systeme.

Der zweite Teil bildet das Herzstück der Arbeit. Hier werden Modellierungstechniken für und Grundmodelle von Organisationseinheiten und Multiorganisationssystemen vorgestellt. In Kapitel 7 wird zunächst die verwendete Terminologie geklärt, eine inhaltliche Einschränkung des Modellierungsvorhabens vorgenommen und ein konkreter Anforderungskatalog erarbeitet. In Kapitel 8 stehen anschließend abstrakte, anschauliche Modelle („Denkmodelle“) im Mittelpunkt, bevor der Fokus in Kapitel 9 auf technischere, ausführbare Modelle („Programmiermodelle“) gerichtet wird.

Der dritte Teil ist der Betrachtung konkreter Beispiele von Organisationseinheiten beziehungsweise von Organisationseinheitstypen und aus ihnen zusammengesetzten Multiorganisationssystemen gewidmet. In Kapitel 10 werden spezifische Ausprägungen von Organisationseinheiten entlang von verschiedenen Organisationsansätzen aus dem Multiagentensystembereich vorgenommen. Auf diese Weise werden die Tragfähigkeit der universellen Grundmodelle untermauert und erste Anhaltspunkte für die Komposition verschiedener Organisationseinheitstypen in Multiorganisationssystemen geliefert. Diese Anhaltspunkte werden in Kapitel 11 aufgegriffen und eine allgemeine Referenzarchitektur für Multiorganisationssysteme vorgeschlagen. Diese Architektur wird sehr abstrakt beschrieben. Es handelt sich um einen ersten Vorschlag, dessen Anwendbarkeit und Tauglichkeit noch gezeigt werden muss. Insofern stellt dieses Kapitel die Überleitung zu Zusammenfassung aus Ausblick der Arbeit in Kapitel 12 dar.

Teil I

Grundlagen und Forschungskontext

2 Softwaresysteme im Großen als Systeme von Systemen

In der Einleitung wurden Multiagentensysteme als der Ausgangspunkt der Forschung für die vorliegende Arbeit herausgestellt. Multiagentensysteme bringen bereits eine konkrete Perspektive auf Softwaresysteme mit sich und gehen mit dem konkreten Paradigma der Agentenorientierung einher. In diesem Kapitel erfolgt zunächst ein Schritt zurück. Hier stehen Charakteristika von großangelegten Softwaresystemen (der Kategorie System von Systemen) und den damit einhergehenden Problemstellungen im Allgemeinen im Fokus. Im folgenden Kapitel wird darauf aufbauend die Betrachtung von Softwaresystemen *als* Organisationen – also die Gestaltung von Softwaresystemen unter Einbezug von Konzepten und Mechanismen aus dem Bereich sozialer Organisationen – motiviert. Dies geschieht noch bevor Multiagentensysteme und die dort vorherrschenden Organisationsansätze später in der Arbeit betrachtet werden. Der Zweck dieser Herangehensweise liegt in der Beschreibung des Potentials einer organisationsorientierten Perspektive auf Softwaresysteme „in Reinform“, also bevor die Überführung von sozialen Konzepte auf die Softwareebene etwaige Verzerrungen bewirkt. Dies hilft anschließend bei der Bewertung des Standes der Forschung im Multiagentensystembereich. Außerdem wird so der Eindruck vermieden, die Organisationsorientierung stelle lediglich eine Notwendigkeit aufgrund von Folgeproblemen eines agentenorientierten Ansatzes dar.

Die Ausführungen dieses Kapitels beziehen sich vielfach auf Systeme, die nicht alleine durch die Betrachtung des softwaretechnischen Anteils begriffen werden können. Stattdessen müssen die Hardware für den Betrieb und die Kopplung der Softwaresysteme sowie gegebenenfalls weitere physikalische Systeme mit eigener fachlicher Funktionalität (beispielsweise Fließbänder, medizinisches Equipment, Aktoren/Sensoren) berücksichtigt werden. Darüber hinaus sind die technischen Systeme in der Regel nicht unabhängig von ihrem sozialen Kontext zu betrachten. Zumindest aber werden hier *softwareintensive* Systeme behandelt und die vorliegende Arbeit widmet sich den Herausforderungen und Problemstellungen, die sich für den softwaretechnischen Anteil ergeben. In der Folge der Arbeit ist daher der Einfachheit halber von Softwaresystemen die Rede solange die Einbettung in einen physikalischen und sozialen Kontext nicht explizit adressiert wird.[1]

In Abschnitt 2.1 werden Charakteristika von Softwaresystemen im Großen, insbesondere unter der Berücksichtigung des Charakters als *Systeme von Systemen*, herausgearbeitet. Davon ausgehend werden in Abschnitt 2.2 Herausforderungen und Forschungsrichtungen konkretisiert.

[1] Nach dem IEEE Standard 1362-1998 (S. 3) wird ein softwareintensives System wie folgt definiert (Definition 3.17, *software-intensive system*):
„A system for which software is a major technical challenge and is perhaps the major factor that affects system schedule, cost, and risk. In the most general case, a software-intensive system is comprised of hardware, software, people, and manual procedures.“
Demgegenüber ist ein Softwaresystem wie folgt definiert (Definition 3.20, *software system*):
„A software-intensive system for which software is the only component to be developed or modified.“

2.1 Charakteristika

Zunächst wird in diesem Abschnitt eine Betrachtung von Charakteristika großer Softwaresysteme und eine Erläuterung des Konzeptes eines Systems von Systemen vorgenommen.

2.1.1 Moderne Softwaresysteme im Großen

Die zunehmende Größe von modernen softwareintensiven Systemen wird von Northrop et. al. [Nor06] als Grundlage für die Forderung nach neuen softwaretechnischen Perspektiven und Paradigmen herangezogen. Größe beziehen Northrop et. al. (S. 11) dabei auf jede erdenkliche Dimension, von der Anzahl von Programmcodezeilen über die Anzahl interagierender Softwareeinheiten und die Anzahl von verschiedentlich am System beteiligten Personen, die Menge an gespeicherten Daten, die Anzahl an Verbindungen und Abhängigkeiten zwischen Systemanteilen bis hin zu benötigten/betroffenen physikalischen Systemen. Neben den von Northrop et. al. benannten *Ultra Large Scale (ULS) Systems* fangen auch andere Bezeichnungen den großangelegten Charakter moderner Softwaresysteme ein. – beispielsweise als *Anwendungslandschaften* nach Matthes, Lankes et. al. oder Humm et.al. [Mat08, LMW05, HVH06] oder als *Softwarestädte* nach Hess et. al. [HHVE07]. Insbesondere das Konzept beziehungsweise die Metapher der Softwarestadt wird auch von Northrop et. al. (S. 5+6) herangezogen, um den Unterschied zwischen traditionellen Softwaresystemen und ULS-Systemen zu verdeutlichen. Die Entwicklung traditioneller (gegebenenfalls auch recht großer) Softwaresysteme wird mit dem Bau eines einzelnen Gebäudes oder einer einzelnen Infrastruktur verglichen. Dagegen wird die Entwicklung eines ULS-Systems mit der Entwicklung einer ganzen Stadt verglichen. Der bestimmende Unterschied ist, dass das Gesamtbild einer Stadt im Gegensatz zu einem Gebäude nur in gewissen Rahmen *gezielt konstruiert* wird. Stattdessen *entwickeln* sich Städte (teilweise ungeplant) im Rahmen bestimmter Regularien.

Es zeigt sich bereits, dass es notwendig ist, über das Konzept der Größe an sich hinauszugehen. Große Softwaresysteme können sehr unterschiedlicher Natur sein, von monolithischen Systemen über geografisch verteilte aber zentral geplante/verwaltete Systeme bis hin zu sowohl geographisch verteilten als auch dezentral in Betrieb genommenen/verwalteten Systemen. Northrop et. al. [Nor06, S. 11–13] prophezeien für die Zukunft das verstärkte Aufkommen von Softwaresystemen als *Systeme von Systemen*. Diesen Begriff gilt es näher zu beleuchten. Zunächst liest sich die Definition eines Systems nach IEEE wie folgt (IEEE Standard 1362-1998, Definition 3.23 (A), *system*, S. 3): *„A collection of interacting components organized to accomplish a specific function or set of functions within a specific environment.“* Anhand dieser weitläufigen Definition können für viele Systeme Anteile identifiziert werden, die ihrerseits als Systeme verstanden werden können. Würden all solche Systeme unter die Kategorie System von Systemen fallen, so könnte der Begriff nicht als nützliche Grundlage taxonomischer Unterscheidungen herhalten. Ein System von Systemen muss derart definiert sein, dass es tatsächlich eine eigene Klasse innerhalb der größeren Klasse der Systeme impliziert.

2.1.2 Das Konzept „System von Systemen“

Die Ausführungen von Maier [Mai99] gehören zu den prominentesten und am häufigsten zitierten zum Thema Systeme von Systemen. Zunächst stellt Maier fest, dass die Größe eines Systems, die Komplexität oder die geografische Verteilung keine ausreichenden Kriterien für die Charakterisierung als System von Systemen darstellen. Stattdessen propagiert er, Systeme

von Systemen seien vor allem durch eine dezentrale Entwicklung, einen dezentralen Betrieb und ein dezentrales Management gekennzeichnet. Die Systeme sind einer *evolutionären Entwicklung* unterworfen. Das Gesamtsystem wird nicht als Ganzes (holistisch) geplant, entworfen und erstellt, sondern entwickelt sich dezentral, graduell und parallel zum Betrieb. Die Funktionsweise und das Zusammenspiel der Teilsysteme sind nicht uniform ausgerichtet, sondern weisen vielmehr einen kollaborativen Charakter auf. Die Teilsysteme sind in der Lage, unabhängig vom integrierten Ganzen zu operieren. Das Gesamtsystem entsteht und entwickelt sich aufgrund „bewusster“ Entscheidungen der Teilsysteme (beziehungsweise ihrer Entwickler/Administratoren), zu kooperieren. Maiers Charakterisierung eines Systems von Systemen lässt sich im Wesentlichen auf zwei Kriterien zurückführen. Systeme, die den Kriterien nicht genügen, werden nicht als Systeme von Systemen aufgefasst, unabhängig von Fragen der Größe, Komplexität oder geografischen Verteilung.

- **Operationale Unabhängigkeit:** Wenn das System aus Systemen in seine Teilsysteme zerlegt wird, so funktionieren die Teilsysteme unabhängig davon weiterhin und sind vor allem weiterhin in der Lage, eigenständig Dienstleistungen zu erbringen und einen nutzerbezogenen Zweck zu erfüllen.
- **Unabhängigkeit des Managements:** Die Teilsysteme eines Systems von Systemen *können* nicht nur unabhängig voneinander operieren, sie *tun* es tatsächlich. Teilsysteme werden unabhängig voneinander akquiriert und eingesetzt. Sie werden zwar ins Gesamtsystem integriert, behalten jedoch kontinuierlich ihre operationale Existenz unabhängig vom Gesamtsystem.

Auch wenn keine allgemeingültige Definition eines Systems von Systemen existiert, so hat sich die Charakterisierung Maiers anhand der beiden Kriterien doch weitestgehend durchgesetzt. Als weiteres Beispiel sei hier auf die (provisorische) Definition des Begriffs der *System-von-Systemen-Entwicklung* (*system of systems engineering, SoSE*) von Keating et. al. [KRU+03] verwiesen:

> „The design, deployment, operation, and transformation of metasystems that must function as an integrated complex system to produce desirable results. These metasystems are themselves comprised of multiple autonomous embedded complex systems that can be diverse in technology, operation, geography, and conceptual frame.“

Das umfassende System von Systemen wird hier also als Meta-System bezeichnet, analog zu Maier wird die Autonomie der Teilsysteme hervorgehoben. Im selben Artikel zitieren die Autoren weitere Auffassungen von Systemen von Systemen aus der Literatur, die sich alle um ähnliche Kernkonzepte drehen.

Das Forschungsfeld der Systeme von Systemen geht über die Betrachtung von Softwaresystemen hinaus. Keating et. al. kritisieren sogar explizit die vorherrschende Dominanz von informationstechnologischen Perspektiven auf Systeme von Systemen da auf diese Weise allgemeinere und weitsichtigere Perspektiven unabsichtlich ausgeschlossen werden. Beispielsweise können auch soziale Systeme anhand der bisher zitierten Charakterisierungen als Systeme von Systemen aufgefasst werden, wie in den Kapiteln 3 und 4 der vorliegenden Arbeit deutlich werden wird. Allerdings ist der Begriff des Systems von Systemen in diesem Forschungsfeld nicht gebräuchlich. Stattdessen wird der Begriff der sozialen Kollektivität verwendet. Der Begriff des Systems von Systemen ist traditionell technisch geprägt. In diesem Kapitel stehen

softwareintensive Systeme im Mittelpunkt, bei denen sich der Gesamtkontext als System von Systemen auf den Softwaresystemanteil überträgt, so dass man es mit Softwaresystemen von Softwaresystemen zu tun bekommt.

2.1.3 Verschiedene Formen von Systemen von Systemen

Konkrete Systeme von Systemen können entlang der beiden von Maier genannten Kriterien variieren, so dass ein Kontinuum von verschiedenen Systemen aus Systemen aufgespannt wird. Maier [Mai99] benennt drei Teilkategorien.

1. *Gerichtete Systeme von Systemen* zeichnen sich durch einen globalen Systemzweck und durch globale Steuerungs- und Kontrollinstanzen mit Weisungsbefugnis über die Teilsysteme aus. Als Beispiel nennt Maier militärische Luftabwehrsysteme. Mit betrieblichen Softwaresystemen wird in Kapitel 5 ein weiteres Beispiel betrachtet. In diesem Bereich ist bereits vielfach von Systemen von Systemen die Rede (vergleiche beispielsweise Matthes [Mat08]). Durch Bestrebungen um ein sogenanntes *Business-IT-Alignment* wird dabei die Informationstechnologie eines Unternehmens an den Geschäftszielen und der Geschäftsarchitektur ausgerichtet, so dass ein gerichtetes System von Systemen entsteht. Teilsysteme können innerhalb eines begrenzten Rahmens unabhängig voneinander operieren, erbringen eigenständige Funktionalität und werden im Kontext verschiedener realweltlicher Abteilungen beschafft und eingesetzt. Nichtsdestotrotz besteht ein gemeinsamer globaler Zweck und eine strikte globale Ordnung, denen die Teilsysteme untergeordnet sind.

2. *Kollaborative Systeme von Systemen* sind ebenfalls durch einen globalen Systemzweck und durch globale Steuerungs-/Kontrollinstanzen gekennzeichnet. Die Unterscheidung gegenüber gerichteten Systemen von Systemen besteht darin, dass die globalen Instanzen keine zwangsweise Weisungsbefugnis über die Teilsysteme haben. Die Teilsysteme partizipieren und kooperieren auf (mehr oder weniger) freiwilliger Basis aufgrund eines gemeinsamen Interesses am globalen Systemzweck. Als Beispiel nennt Maier intelligente Transportsysteme. Er bezieht sich vor allem auf Informationstechnologie für Echtzeitinformationen über Verkehrsbedingungen und Transport-/Reiseoptionen sowie für automatisches Verkehrsmanagement. Die entsprechenden Teilsysteme werden im Wesentlichen von kommerziellen Firmen entwickelt und vertrieben. Der Einsatz erfolgt teilweise auf staatlicher Seite, teilweise auf Seiten kommerzieller Firmen und teilweise auf Seiten individueller Benutzer. Trotz des gemeinsamen Systemzwecks können verschiedene Teilnehmer des Systems von Systemen (Staatliche Behörden, Unternehmen, Individuen) unterschiedliche und konfliktbehaftete Ziele haben und verfolgen. Als weiteres Beispiel können hier betriebliche Softwaresysteme gesehen werden, die unternehmensübergreifend eingesetzt werden (vergleiche Zirpins [Zir07] zur Anwendung dienstorientierter Softwaresysteme im Kontext der Planung und Steuerung von Produktionsvorgängen in virtuellen Unternehmensnetzwerken; allerdings verwendet Zirpins den Begriff des Systems von Systemen nicht explizit). Hier entstehen kollaborative Systeme von Systemen, deren Teilsysteme die jeweils unternehmensweiten gerichteten Systeme von Systemen sind.

3. *Virtuelle Systeme von Systemen* sind nicht nur durch das Fehlen globaler Instanzen mit Weisungsbefugnis über die Teilsysteme gekennzeichnet, sondern sogar durch das Fehlen eines klar umrissenen globalen Systemzwecks. Globales Systemverhalten *emergiert*. Auf

Ebene des Gesamtsystems können lediglich „relativ unsichtbare“ Mechanismen für das Hervorbringen und die Erhaltung bestimmter Systemcharakteristika eingesetzt werden (technische Standards, weitläufige Regulierung). Als Beispiel nennt Maier das Internet. Entwicklung, Operation und Aufsicht sind dezentralisiert. Die Art und Weise, auf die verschiedene Akteure das Internet nutzen, bestimmt unmittelbar, welche Dienste angeboten werden. Gleichzeitig beeinflussen die angebotenen Dienste, welche Aktionen von Seiten der Akteure im Kontext des Systems ausgeführt werden. Die Entwicklung des Gesamtsystems ist evolutionär, mit einem hohen Grad an spontaner Innovation und Fortentwicklung.

Weitere mögliche softwareintensive Anwendungsfelder für Systeme von Systemen sind Integration verschiedener Hierarchieebenen der Automatisierung in Fertigungsszenarien, Smart Cities, Health-Care-Systeme, Energieerzeugungs- und Verteilungssysteme, verteilte Netzwerksicherheit, Katastrophenmanagement, Steuerungs- und Kontrollsysteme für komplexe physikalische Systeme. Bei allen hier genannten Anwendungsfeldern stehen großangelegte, softwareintensive Systeme im Fokus. Es wurde aber bereits betont, dass Größe alleine noch kein System von Systemen bedingt. Daher handelt es sich um *mögliche* Anwendungsfelder.

2.1.4 Entstehung von Systemen von Systemen

Es stellt sich die Frage, warum Systeme von Systemen eingesetzt werden, beziehungsweise warum sie entstehen und inwiefern sie tatsächlich von Nutzen sind. Die Frage ist vor allem deswegen berechtigt, da zu erwarten steht, dass das Erreichen eines bestimmten Systemzwecks im Kontext eines Systems von Systemen ungleich aufwändiger und teurer gegenüber herkömmlichen, uniform ausgerichteten Systemen ist. Der Zweck eines Systems von Systemen steht immer im Zusammenhang mit den Systemzwecken der Teilsysteme. Ein monolithisches System kann von vorneherein so geplant und entworfen werden, dass alleine ein globaler Systemzweck erfüllt wird. In einem System von Systemen ist dies nicht ohne Weiteres möglich. Für die Teilsysteme muss ein Anreiz zur Partizipation beziehungsweise ein Anreiz für die Unterordnung gegenüber einem globalen Systemzweck bestehen. Ein zusätzlicher Kostenfaktor in Systemen von Systemen ist Redundanz: Durch die operationale Unabhängigkeit der Teilsysteme liegen gewisse Eigenschaften und Funktionen üblicherweise dupliziert vor.

Systeme von Systemen entstehen dann bewusst, wenn die Vorteile der Teilsystemunabhängigkeit die genannten Nachteile überwiegen. Beispielsweise kann die Tatsache, dass jedes Teilsystem auch für sich selbst gesehen von Wert ist, die zusätzlichen Kosten der Erhaltung des Systems von Systemen aufwiegen. In anderen Fällen werden die Kosten des Systems von Systemen gegebenenfalls nicht von einer einzigen Partei getragen und somit ist eine globale Kostenminimierung von geringerer Bedeutung. In solchen Fällen kann gegebenenfalls nicht einmal ein globaler Systemzweck eindeutig identifiziert werden. In wiederum anderen Fällen entsteht ein System von Systemen schlicht „zufällig“, beispielsweise ursprünglich resultierend aus einer eigentlich als temporär angedachten losen Kopplung einiger weniger Systeme. Pragmatische Gründe können dann in der Folge dazu führen, die Kopplung aufrecht zu halten und auf weitere Systeme auszuweiten.

Northrop et. al. [Nor06, S. 12+13, S.22–25] gehen noch über den Aspekt des zufälligen Entstehens eines Systems von Systemen hinaus. Stattdessen stellen sie fest, große softwareintensive Systeme der Zukunft würden *inhärent* mit dem Charakter als System von Systemen einhergehen. Die heutzutage bereits vorherrschende und weiter zunehmende nahezu vollstän-

dige Durchdringung des täglichen sozialen Lebens mit Informationstechnologie und die zunehmenden Möglichkeiten der flexiblen Vernetzung führen unweigerlich zu einer engen *sozio-technischen* Verzahnung von softwaretechnischen (sowie weiteren technischen) mit sozialen Systemen. Die sozialen Systemen ab einer gewissen Größe innewohnende Dezentralisierung und die verschiedenen Interessen und Anforderungen realweltlicher individueller sowie korporativer Akteure (vergleiche die Kapitel 3 und 4) übertragen sich auf Struktur und Abläufe der Softwaresysteme. Insofern stellt sich weniger die Frage, ob und wie großangelegte Softwaresysteme von Softwaresystemen gegebenenfalls zu vermeiden sind. Die Frage muss vielmehr von vorneherein lauten, wie solche Systeme zu beherrschen sind.

Diese Sichtweise von Northrop et. al. wird insofern gestützt, als dass das Konzept des Systems von Systemen bereits in den 1990er Jahren diskutiert wurde, ein verstärktes Interesse jedoch erst in den letzten Jahren aufkam. Seit dem Jahr 2006 findet die jährliche Konferenz *IEEE International Conference on System of Systems Engineering (SoSE)* statt, De Laurentis et. al. [DLDD+07] haben ein Internationales Konsortium für System-von-Systemen-Entwicklung vorgeschlagen und von verschiedener Seite aus wurden Vorschläge für eine integrierte Forschungsausrichtung vorgeschlagen (vergleiche Sousa-Poza et. al. [SPKK08], Keating [Kea05], Ring und Madni [RM05]).

Die bisherigen Ausführungen deuten bereits an, dass weder Größe, Komplexität noch die Eigenschaft als System von Systemen für sich genommen notwendigerweise für außergewöhnliche Problemstellungen im Umgang mit Softwaresystemen sorgen. Es ist die *Kombination* aller Aspekte, die nach Northrop et. al. [Nor06, S. 12+13, S.22–25] anspruchsvolle Herausforderungen für die Handhabung heutiger und vor allem zukünftiger Softwaresysteme bedingt.

2.2 Herausforderungen

Die Kombination aus Größe, Komplexität und der Eigenschaft als Systeme von Systemen stellt die Grundlage für Herausforderungen im Umgang mit modernen und zukünftigen Softwaresysteme dar. In diesem Abschnitt werden konkrete Problemstellungen benannt, an denen sich der im Verlauf der vorliegenden Arbeit propagierte Ansatz der Organisationsorientierung für den Bau von Softwaresystemen messen lassen muss.

Zunächst werden einige allgemeine Heuristiken der Entwicklung komplexer Systeme zitiert (Teilabschnitt 2.2.1). Diese werden in der Folge des Abschnitts wiederholt aufgegriffen. Konkrete Problemstellungen und Herausforderungen werden entlang der Dimensionen *Dezentralisierung* (Teilabschnitt 2.2.2), *multiple Systemebenen* (Teilabschnitt 2.2.3), *sozio-technische Integration* (Teilabschnitt 2.2.4) und *Software mit Selbst-Eigenschaften* (Teilabschnitt 2.2.5) klassifiziert und behandelt. Die Problemstellungen sind sehr umfangreich und repräsentieren eigentlich jeweils eigenständige Forschungsthemen. Im Kontext der vorliegenden Arbeit ist es nicht möglich, alle von ihnen umfassend zu adressieren. Der in der Arbeit vorgestellte Lösungsansatz der Organisationsorientierung bietet allerdings eine konzeptionelle Grundlage, auf Basis derer alle Problemstellungen betrachtet werden können. In den späteren Kapiteln der Arbeit liegt der Fokus auf der Adressierung der beiden Problemstellungen der Dezentralisierung und der multiplen Systemebenen.

2.2.1 Heuristiken für die Entwicklung komplexer Systeme

Die Existenz von Heuristiken, Entwurfsprinzipien, *Best Practices* oder Entwurfsmustern sind Manifestationen der Tatsache, dass bestimmte Regeln der Systementwicklung mit dem Er-

folg, leistungsfähige Systeme zu erhalten, korrelieren. Rechtin und Maier [RM00, S. 126-133] benennen einige Entwicklungsheuristiken für komplexe Systeme. Diese Heuristiken sind nicht exklusiv für Systeme von Systemen, können allerdings für diese Systemkategorie gezielt angewendet werden.

- **Stabile Zwischenformen** Die Entwicklung und Weiterentwicklung komplexer Systeme profitiert von stabilen Zwischenformen. Stabilität bezieht sich hier darauf, dass Zwischenformen „für sich stehen können“. Auf diese Weise können Fehlfunktionen früh festgestellt und das Potential zur Weiterentwicklung besser eingeschätzt werden. Bereits Simon [Sim62] betont, die Entwicklung von einfachen zu komplexen Systemen vollziehe sich schneller wenn stabile Zwischenformen erreicht werden.

  Die Idee der stabilen Zwischenformen bezieht sich im Falle eines Systems von Systemen sowohl auf Systemanteile als auch auf das Gesamtsystem. Stabile Zwischenformen von Systemanteilen sind einem System von Systemen aufgrund der unabhängigen Teilsysteme bereits inhärent. Auch die Integration mehrerer Teilsysteme in das Gesamtsystem kann über stabile Zwischenformen geschehen. (dieser Aspekt hängt unmittelbar mit der Entstehung multipler Systemebenen zusammen, vergleiche hierzu Teilabschnitt 2.2.3).

  Auf Ebene des Gesamtsystems sind stabile Zwischenformen zugleich von großer Wichtigkeit und schwer zu erreichen. Der Charakter als evolutionäres, kollaboratives System führt dazu, dass die Verfügbarkeit von Teilsystemfunktionen nicht dauerhaft angenommen werden kann. Für solche Fälle müssen Rückfall-Modi vorgesehen sein. Es muss sichergestellt werden, dass Komponenten vom Gesamtsystem getrennt werden können, ohne dass das gewünschte globale Systemverhalten in sich zusammenfällt.

- **Policy Triage** Der Wortlaut dieser Heuristik lautet bei Rechtin und Maier [RM00, S. 127]: *„Let the dying die. Ignore those who will recover on their own. And treat those who would die without help“.*

  Diese Heuristik dient als Hilfestellung, Prioritäten bezüglich der Auswahl, Weiterentwicklung und Unterstützung von Systemanteilen zu setzen. Es geht hierbei auch verstärkt um die Frage, was es in einem komplexen System zu steuern und zu kontrollieren gilt und was nicht. Im Falle von Systemen von Systemen bekommt diese Heuristik ein spezielles Gewicht. Ihr kollaborativer Charakter bedingt, dass zu umfassende Versuche der Steuerung aufgrund der fehlenden Autorität über die Teilsysteme scheitern müssen. Zu wenig Steuerung führt auf der anderen Seite zur Elimination eines integrierten Ganzen. Ein geeigneter Mittelweg ist notwendig, um die gewünschte Kooperation der Teilsysteme zu sichern.

- **Sicherstellen der Kooperation** Diese Heuristik steht im Zusammenhang mit der vorherigen. Wenn ein System auf dem Prinzip der freiwilligen Kooperation basiert, so müssen die Mechanismen der Kooperation und der Anreiz zur Kooperation vorhanden sein. Der Kosten/Nutzen-Faktor einer Kollaboration muss dem Kosten/Nutzen-Faktor des unabhängigen Handelns überlegen sein. Ein gutes Beispiel stellt das Internet dar. Die Kosten zur Teilnahme (Nutzung standardisierter Ausrüstung, Einhalten von Protokoll-/Adressierungs-Konventionen) sind recht gering, während der Nutzen (Zugang zu Backbone-Netzwerken) hoch ist.

- **Einflussnahme über die Schnittstellen** Mit zunehmender Unabhängigkeit der Teilsysteme eines Systems kommt den Schnittstellen eine immer größere Bedeutung zu. Im

Falle von Systemen von Systemen kann sogar die Sichtweise eingenommen werden, die Schnittstellen *seien* die Systemarchitektur. Die Schnittstellen bestimmen die Möglichkeiten der Kooperation und Koordination zwischen Teilsystemen und implizieren auf diese Weise, welche Struktur ein integriertes System von Systemen annehmen kann.

2.2.2 Dezentralisierung

Systeme von Systemen sind durch Dezentralisierung gekennzeichnet. In Teilabschnitt 2.1.2 wurde dargelegt, dass sich diese Dezentralisierung auf Unabhängigkeit von Management und Operation der Teilsysteme bezieht. Sie bezieht sich nicht unbedingt darauf, dass im System von Systemen keine globalen oder sogar zentralisierten Steuerungs- und Kontrollmechanismen angewandt werden. Es möglich, unabhängig voneinander eingesetzte Systeme durch eine zentrale Steuerungsinstanz zu koordinieren, um die gemeinsame Erbringung einer neuartigen Funktionalität zu erreichen. Diese Herangehensweise zeichnet insbesondere gerichtete Systeme von Systemen aus. Allerdings ist die Herangehensweise (zumindest in Reinform) nur begrenzt einsetzbar. Mit zunehmender Größe eines Systems von Systemen wird die Aufrechterhaltung einer umfassenden zentralen Steuerung unmöglich. Es gibt zu viele Entscheidungen zu treffen und zu viel Information zu verarbeiten. Erschwerend kommt das Entstehen verschiedener Abstraktions-/Granularitätsebenen und die Teilnahme von Teilsystemen an möglicherweise mehreren umfassenden Systemen hinzu (siehe den folgenden Teilabschnitt). Dies kann zu hierarchischen Modellen der Steuerung führen (vergleiche beispielsweise die Arbeit von Lygeros [Lyg96]). Doch analog zu Hierarchien in sozialen Organisationen (vergleiche Kapitel 4, insbesondere Teilabschnitt 4.3.2 der vorliegenden Arbeit) stoßen auch diese ab einer gewissen Größenordnung und Komplexität des zu steuernden Systems an ihre Grenzen. So folgern auch Northrop et. al. [Nor06, S. 13], Softwaresysteme der Kategorie ULS-System bedürften dezentralisierter Mechanismen der Steuerung und Kontrolle. Dies bedeutet nicht unbedingt einen Verzicht auf globale Autoritätsinstanzen. Es bedeutet allerdings das Treffen von Steuerungsentscheidungen auf verschiedenen Ebenen der Systemabstraktion, die Delegation von Hoheit über Entscheidungen sowie die Existenz multipler Autoritätslinien.

Für Systemen von Systemen außerordentlicher Größe kommt also eine weitere Dimension der Dezentralisierung hinzu. Die Implikationen sind durchaus schwerwiegend. Vor allem wird der evolutionäre Charakter und damit einhergehend das Auftreten emergenten Verhaltens verstärkt. Die Entwicklung des Gesamtsystems wird von Seiten verschiedener Akteure vorangetrieben. Für diese Akteure ergibt sich ein breites Spektrum, von realweltlich/sozialen individuellen sowie kollektiven Akteuren (vergleiche Teilabschnitt 2.2.4) bis hin zu künstlichen/softwaretechnischen Akteuren (vergleiche Teilabschnitt 2.2.5). (Weiter-)Entwicklungen des Systems finden nicht in diskreten Intervallen, sondern kontinuierlich und parallel zum Betrieb des Systems statt. Die Vielfalt der Quellen von Softwarekomponenten führt zu heterogenen Systemen. Teile der Software werden in verschiedenen Sprachen geschrieben, sind auf verschiedene (Hardware-/Software-)Plattformen zugeschnitten, und entspringen verschiedenen Entwicklungsmethodologien. Einige Teile des Gesamtsystems befinden sich in ständigem Fluss und liegen teilweise in verschiedenen Versionen vor. Insgesamt können dadurch unterschiedliche und sogar widersprüchliche Anforderungen der Teilsysteme an das Gesamtsystem gestellt werden. Die Anforderungen in der Gegenrichtung, vom Gesamtsystem an die Teilsysteme, sind in einigen Fällen nicht vollständig bekannt, bevor die Teilsysteme in Betrieb sind. Das Auftreten von Konflikten und Fehlerfällen ist unvermeidbar. Doch können Konflikte ohne zentrale Kontrolle nicht immer in uniformer Weise aufgelöst werden.

Es ist möglich, einigen der genannten Aspekte positive Seiten abzugewinnen. Beispielsweise führt Heterogenität der Systemanteile dazu, dass sie nicht gleichermaßen verwundbar gegenüber Störfällen oder Angriffen sind. Insgesamt ist allerdings festzuhalten, dass Dezentralisierung im Grunde genommen die Anwendbarkeit aller Heuristiken aus Teilabschnitt 2.2.1 negativ beeinflusst. Die Bildung stabiler Zwischenformen, die Definition einheitlicher Schnittstellen und die Zusicherung von Kooperation werden durch Heterogenität, verschiedene Interessenvertreter und widersprüchliche Anforderungen erheblich erschwert. Eine besondere Aufmerksamkeit kommt der Heuristik *Policy Triage* zu. Es stellt sich die Frage nach alternativen Steuerungs- und Kontrollmechanismen gegenüber der direkten Einflussnahme durch eine zentrale Instanz. Ein grundlegender Vorschlag wird von Northrop et. al. [Nor06, S. 15] unterbreitet: Die Evolution des Systems sollte durch Regeln und Vorgaben derart eingeschränkt und geleitet werden, dass lokale Bedürfnisse lokal befriedigt werden können, ohne dabei die Integrität des Gesamtsystems zu gefährden. Die Grundidee ist also, direkte Steuerung durch *Regulierung* zu ersetzen. Hier ergibt sich wieder die Analogie zur in Abschnitt 2.1 zitierten Metapher der Evolution einer Stadt gegenüber der gezielten Konstruktion eines Gebäudes.

Im folgenden Kapitel 3 der vorliegenden Arbeit wird argumentiert, dass eine organisationsorientierte Perspektive der Softwaresystemgestaltung verschiedene Modi der Steuerung und Kontrolle mit sich bringt und somit Möglichkeiten der Selektion je nach Kontext und Systemanteil bietet. Eine differenzierte Herangehensweise an den Aspekt der Steuerung und Kontrolle wird nicht zuletzt durch die in Abschnitt 2.1 zitierte weitere Unterteilung der Kategorie der Systeme von Systemen deutlich. In einem virtuellen System von Systemen wie dem Internet sind Steuerungsanforderungen deutlich geringer als in gerichteten beziehungsweise kollaborativen Systemen von Systemen wie beispielsweise Health-Care-Systemen oder Systemen zur Integration von Geschäftsprozessen.

2.2.3 Multiple Systemebenen

Bei den Ausführungen zu Systemen von Systemen in Teilabschnitt 2.1 ist in der Regel vom Gesamtsystem und den Teilsystemen die Rede. Dieser Sichtweise liegt dementsprechend zunächst eine Unterscheidung von zwei Systemebenen zu Grunde. Allerdings ist es möglich, das Konzept des Systems von Systemen rekursiv anzuwenden, indem die Teilsysteme wiederum Systeme von Systemen darstellen. Somit ist die Möglichkeit der Betrachtung verschiedener Systemebenen der Idee der Systeme von Systemen inhärent. Damit wird auch Simons [Sim62] klassischer Feststellung, Hierarchie sei eine Eigenschaft *aller* komplexen Systeme, Rechnung getragen. Hierarchie nach Simon bezieht sich zunächst nicht auf Macht-, Autoritäts- beziehungsweise Weisungsbefugnisse. Hierarchie ist eher als *Clustering-Mechanismus* zu verstehen. Systeme *bestehen aus* anderen Systemen. Die einzelnen zusammengesetzten Systeme sind dabei nicht zwingend disjunkt. Es kann sein, dass ein bestimmtes System zur Konstitution mehrerer umschließender Systeme beiträgt (und damit gegebenenfalls auch auf verschiedenen Systemebenen in Erscheinung tritt).

Allerdings bedarf es nach Ansicht des Autors der vorliegenden Arbeit einer *systematischen* Behandlung verschiedener Abstraktionsebenen eines Systems von Systemen. Insbesondere ist der kollaborative Charakter der Systeme zu beachten. Im vorigen Abschnitt 2.1 wurde wiederholt von „freiwilliger“ Kooperation zwischen Teilsystemen gesprochen. Bei der Betrachtung von Teilsystemen, die wiederum Systeme von Systemen und damit Supersysteme für weitere Teilsysteme sind, wird der Aspekt der freiwilligen Kooperation komplizierter. Es stellt sich die Frage nach den Zusammenhängen zwischen der Autonomie des Supersystems und der Au-

tonomie der Teilsysteme. Hier ist vor allem die Heuristik der *Zusicherung von Kooperation* aus Teilabschnitt 2.2.1 betroffen. Entsprechende Mechanismen und Anreize zur Kooperation müssen über mehrere Systemebenen hinweg propagiert werden.

Weitere Fragestellungen schließen sich an. Bisher war vor allem von der Unabhängigkeit der Teilsysteme gegenüber dem Gesamtsystem von Systemen die Rede. Nun rückt allerdings auch die Unabhängigkeit eines Teilsystems gegenüber dessen Teilsystemen in den Blickpunkt. Beispielsweise könnte ein Supersystem speziell im Namen/Auftrag seiner Teilsysteme existieren. Andererseits wäre es möglich, dass ein Supersystem unabhängig von konkreten Teilsystemen ist, solange deren Positionen kontinuierlich durch alternative Teilsysteme besetzt werden können. Auf diese Weise kann das Supersystem die Teilnahme vieler verschiedener Teilsysteme überdauern. Eine andere Frage ist, ob ein Supersystem eigene Ressourcen besitzt oder ob die einzigen Ressourcen die beteiligten Teilsysteme sind. Zusammenfassend formuliert betreffen diese Fragestellungen den Grad an eigener Existenz, die ein Supersystem gegenüber seinen Teilsystemen hat. Dies betrifft also unmittelbar die Anwendbarkeit der Heuristik der *Entwicklung stabiler Zwischenformen*. Insgesamt ergibt sich ein Spektrum an verschiedenen Formen von Systemkompositionen. Neben allgemeinen Untersuchungen kann es nach Ansicht des Autors der vorliegenden Arbeit sehr hilfreich sein, *Archetypen* von Systemen, die sowohl als Teilsysteme als auch als Supersystem eingesetzt werden können, zu definieren und ihre Anwendbarkeit in konkreten Anwendungsszenarien zu determinieren.

Ein weiterer Aspekt in diesem Zusammenhang ist die Repräsentation beziehungsweise Verkörperung von Systemen auf den verschiedenen Systemebenen. Einerseits könnte ein Supersystem lediglich virtuell/logisch existieren, nämlich als verteilte Repräsentation innerhalb seiner Teilsysteme. Demgegenüber könnte die Bildung eines Supersystems mit neuen softwaretechnischen Artefakten einhergehen, so dass das neue System softwaretechnisch reifiziert/verkörpert wird. Beispiele für beide Möglichkeiten finden sich im Bereich der Multiagentensysteme wieder, wie in Abschnitt 5.2 der vorliegenden Arbeit erläutert wird. In der vorliegenden Arbeit wird vor allem die Herangehensweise propagiert, jedes Teilsystem eines Systems von Systemen explizit softwaretechnisch zu verkörpern Dies fördert insbesondere die Anwendbarkeit der Heuristik *Einflussnahme über die Schnittstellen*. Eine wohlüberlegte Definition von Schnittstellen nicht nur für die horizontale sondern auch für vertikale Kopplung erlaubt eine direkte Einflussnahme darauf, was für Formen von Systemen von Systemen über multiple Systemebenen hinweg entstehen.

2.2.4 Sozio-technische Integration

In diesem Kapitel wurde an verschiedenen Stellen auf die Ko-Existenz von Softwaresystemen und sozialen Systemen hingewiesen. Die Berücksichtigung von beteiligten realweltlichen Akteuren in Form von Personen, Gruppen und Organisationen sind zentral für das Verständnis der Effekte und der Dynamik eines Softwaresystems im Großen.

Die Integration sozialer und technischer Kontexte ist in verschiedenen Bereichen von großer Wichtigkeit. Beispielsweise stellen auch soziale Organisationen sozio-technische Systeme dar. Dieses Thema wird in Abschnitt 3.3 aufgegriffen und stellt dort eines der Kernargumente für eine Organisationsperspektive auf Softwaresysteme im Großen dar. Auch im Bereich der Softwareentwicklung sind sozio-technische Perspektiven nicht neu. Als Beispiele dienen unter anderem *Pervasive/Ubiquitous Computing* [Mat01] *Ambient Intelligence* [IST05], *Information Cities* [SLA04] oder das Mikropolis-Modell [Rol08]. Die Anwendungsfelder der Multiagentensysteme und der betrieblichen Softwaresysteme werden in Kapitel 5 gesondert aufgegriffen.

Für große Softwaresysteme von Softwaresystemen nimmt die Notwendigkeit einer soziotechnischen Perspektive eine besonders imperative Stellung ein. Northrop et. al. [Nor06, Abschnitt 2.5] sprechen von einer *Erosion der Mensch/Softwaresystem-Grenze* im Kontext von ULS-Systemen. Die vereinfachende Auffassung realweltlicher/sozialer Akteure als Nutzer oder Entwickler ist nicht länger angemessen. Es geht darum, *alle* informationsverarbeitenden Mechanismen und Verhaltensweisen systematisch *gemeinsam* zu berücksichtigen. Das Softwaresystem ist nicht nur ein Gegenstück zum sozialen System. Stattdessen handelt es sich bei Softwaresystemen und sozialen Systemen um Dimensionen eines integrierten softwareintensiven Systems von Systemen. Eine wichtige Rolle nimmt hier wiederum die evolutionäre Entwicklung eines Systems von Systemen ein. Wenn sich softwaretechnische Konfigurationen und Eigenschaften ändern, so müssen Mechanismen und Prozeduren verfügbar sein, die den sozialen Akteuren etwaige neue Möglichkeiten aufzeigen und sie unterstützen, ihre aktuellen Ziele weiterhin zu verfolgen. Wenn sich andererseits die Muster, nach denen soziale Akteure Gebrauch vom Softwaresystem machen, ändern, müssen auf Seiten des Softwaresystems Mechanismen und Prozeduren vorhanden sein, eine Adaption an die neuen Gegebenheiten vorzunehmen. Gerade die Größe eines Systems von Systemen der Kategorie ULS-System eröffnet Möglichkeiten der Analyse und Verbesserung, die in kleineren Systemen oft nicht möglich sind. Das Verhalten individueller sozialer Akteure ist sehr schwer analysierbar und vorhersagbar. Das kollektive Verhalten einer großen Anzahl von Akteuren dagegen ist einer Analyse zugänglicher. Anhand einer großen Anzahl von Interaktionen zwischen sozialem und softwaretechnischem System können Daten bezüglich beobachteter Regularitäten gesammelt werden. Darauf aufbauend können statistisch verlässliche Modelle der verschiedenen benötigten Dienstleistungen und Funktionen erstellt werden.

Von den in Teilabschnitten 2.2.1 genannten Heuristiken sind im Grunde genommen alle vom Aspekt der sozio-technischen Integration betroffen. Besondere Aufmerksamkeit kommt allerdings der Heuristik *Einflussnahme über Schnittstellen* zu. Im Hinblick auf Schnittstellen geht es nicht nur um Schnittstellen zwischen softwaretechnischen Komponenten, sondern verstärkt um Schnittstellen zwischen softwaretechnischen Komponenten und sozialen Akteuren. Nach Ansicht des Autors der vorliegenden Arbeit darf eine solche Betrachtung nicht mit der bloßen Definition von Aktoren und Sensoren für die Bedienung und den Informationsaustausch enden. Stattdessen wird in der vorliegenden Arbeit die Auffassung vertreten, Betrachtungen der Passung und damit auch der Schnittstellen zwischen softwaretechnischem und sozialem System würden davon profitieren, beide Seiten in Form von gemeinsamen Konzepten und gemeinsamer Terminologie zu begreifen. Die softwaretechnische Entwicklung der vergangenen Jahrzehnte hat bereits den Weg dafür bereitet. Die Evolution von Assembler-Sprachen über prozedurale Abstraktionen, deklarative und objektorientierte Programmierung bis hin zu Komponenten und dienstorientierten Architekturen offenbart einen kontinuierlichen Wandel von maschinennahen Entwicklungskonzepten und -metaphern hin zu solchen, deren Abstraktionen ihren Ursprung in realweltlichen Konzepten haben. Mit dem Konzept des Softwareagenten (vergleiche Kapitel 5) gibt es sogar Softwareentitäten, die konzeptionell an soziale Akteure angelehnt sind und somit die gemeinsame Betrachtung von Softwaresystem und sozialem Kontext unmittelbar unterstützt. In den folgenden Kapiteln wird die Argumentation geführt, die Organisation als zentrales Theorieelement der Softwareentwicklung sei ein nächster logischer Schritt in der beschriebenen Entwicklung.

2.2.5 Software mit Selbst-Eigenschaften

Eng verknüpft mit dem Thema der sozio-technischen Integration ist der Aspekt der Software mit sogenannten „Selbst-Eigenschaften“. Die Größe der hier betrachteten Systeme von Systemen, der erhöhte Grad an Verteilung und Vernetzung, Dezentralisierung in multiplen Dimensionen sowie die evolutionäre Entwicklung und einhergehende Heterogenität beschwören unweigerlich ein Problem herauf, das bereits an verschiedener Stelle angedeutet wurde. Kephart und Chess [KC03] (als Vertreter des *IBM Research Center*) argumentieren, Softwaresysteme der Zukunft würden zu groß und komplex werden, um alleine von Personen als Systementwicklern/-administratoren installiert, konfiguriert, optimiert, verwaltet oder zusammengeführt zu werden. Die Anzahl der vernetzten beziehungsweise eingebetteten Softwaresysteme übersteigt die Anzahl der beteiligten sozialen Akteure bei Weitem. Daher sind letztere mit der Aufgabe, zeitnahe und wohlüberlegte Entscheidungen im Angesicht des kontinuierlichen Stroms an wechselnden Anforderungen zu treffen, kognitiv überfordert. Die einzig mögliche Lösung besteht nach Kephart und Chess im von ihnen so genannten *Autonomic Computing*. Dieser Ansatz bezeichnet Softwaresysteme mit der Fähigkeit, sich unter der Vorgabe hochrangiger/deklarativer Systemziele und -regeln selber zu verwalten. Zu einem solchen Selbstmanagement zählen Kephard und Chess im Wesentlichen folgende Eigenschaften.

- *Selbst-Konfiguration:* Unter der Vorgabe hochrangiger Policies findet die Konfiguration von Komponenten und Systemen automatisiert statt.
- *Selbst-Optimierung:* Die Komponenten und Systeme suchen kontinuierlich Möglichkeiten, ihre Leistung und Effizienz zu verbessern.
- *Selbst-Heilung:* Die Komponenten und Systeme identifizieren, diagnostizieren und beseitigen/reparieren Hardware- und Softwareprobleme selbständig.
- *Selbst-Schutz:* Die Komponenten und Systeme schützen sich selber gegen Angriffe und kaskadierende Fehler. Es werden Frühwarnsysteme eingesetzt, um systemweite Fehler möglichst zu antizipieren und zu verhindern.

Eine analoge Sichtweise nimmt auch Tennenhouse [Ten00] ein. Im Zuge seines *Proactive Computing*-Ansatzes fordert er, Personen aus den alltäglichen Betriebsinteraktionen von Softwaresystemen herauszutrennen und sie stattdessen in Positionen für die Aufsicht und das Treffen von Vorgaben zu hieven. Herring [Her02, S. 1+2] sieht das Proactive Computing nach Tennenhouse als nächsten Schritt einer Evolution des Zusammenspiels zwischen Mensch und Softwaresystem. Mit *Impersonal Computing* bezeichnet Herring den Beginn der Computertechnologie, als viele Personen gemeinsam ein beziehungsweise wenige Softwaresysteme benutzt haben. Mit *Personal Computing* bezeichnet er die Ära der Computertechnologie, in der jede Person auch im Grunde genommen ihr eigenes Softwaresystem genutzt hat. Proactive Computing bezeichnet die aktuelle und zukünftige Entwicklung, bei der auf jede Person eine Vielzahl von Softwaresystemen kommt. Herring [Her02, speziell Kapitel 5] stellt zudem seinen konkreten Ansatz der *Viable Software* vor, der als Instanz des Proactive Computing beziehungsweise des Autonomic Computing verstanden werden kann. Als verwandter Ansatz zur vorliegenden Arbeit wird Viable Software in Kapitel 5 aufgegriffen.

Die Notwendigkeit von Software mit Selbst-Eigenschaften bringt einen wichtigen Aspekt in die Überlegungen zur sozio-technischen Systemintegration ein. Bezogen auf die Heuristiken aus Teilabschnitt 2.2.1 sind vor allem *Policy Triage* und *Entwicklung stabiler Zwischenformen*

betroffen. Es stellt sich nicht nur die Frage, was es in einem System von Systemen zu steuern/kontrollieren gilt und was nicht. Zusätzlich gilt es zu bedenken, welche Steuerungs- und Kontrollfunktionalität man von Seiten der sozialen Akteure aus der Hand gibt und stattdessen auf die Ebene der Softwaresysteme selber verlagert. Die hier zitierten Quellen zu Software mit Selbst-Eigenschaften fußen alle auf dem Prinzip, soziale Akteure hochrangige/deklarative Systemregeln definieren zu lassen. Diese werden auf Seiten des Softwaresystems interpretiert und in entsprechende Steuerungs- und Kontrollmechanismen umgesetzt. Nach Ansicht des Autors der vorliegenden Arbeit ist es in diesem Zusammenhang unerlässlich, dass die Art und Weise, auf die sich ein Softwaresystem selber verwaltet, für soziale Akteure intuitiv nachvollziehbar bleibt. Dafür müssen die entsprechenden Konzepte, Prinzipien und Mechanismen auf Softwareebene ihre Entsprechungen in der Steuerung und Kontrolle realweltlicher/sozialer Systeme haben. Dieses Argument ist analog zu dem des vorigen Abschnittes bezüglich der Schnittstellen zwischen den sozialen und softwaretechnischen Systemanteilen. Darüber hinaus nimmt die Entwicklung stabiler Zwischenformen offensichtlich eine noch wichtigere Stellung als ohnehin bereits ein. Es ist kaum möglich, Softwaresysteme mit derart elaborierten Selbst-Eigenschaften auszustatten, dass sie auf *alle möglichen* Einflüsse angemessen reagieren können. Stattdessen ist es unerlässlich, dass die sozialen Akteure von Zeit zu Zeit die deklarativen Vorgaben oder die Art und Weise der Interpretation auf Seiten des Softwaresystems anpassen. Für solche Fälle sind temporäre Rückfallmodi für die Erbringung eingeschränkter Funktionalität notwendig.

2.3 Zusammenfassung

Als Grundlage für den weiteren Verlauf der Arbeit wurden in diesem Kapitel die Charakteristika großer softwareintensiver Systeme von Systemen zusammengetragen und darauf aufbauend eine Klassifikation der softwaretechnischen Problemstellungen erarbeitet. Problemstellungen wurden entlang der Dimensionen (1) Dezentralisierung, (2) multiple Systemebenen, (3) sozio-technische Integration und (4) Software mit Selbst-Eigenschaften erläutert. Zur besseren Bewertung der verschiedenen Problemstellungen wurden etablierte Heuristiken für die Entwicklung komplexer Systeme herangezogen und untersucht, inwiefern deren Anwendbarkeit im Kontext von Systemen von Systemen schwieriger beziehungsweise deren Anwendung noch wichtiger wird. Die identifizierten Problemstellungen sind interdisziplinär und weisen aktuell viele offene Enden auf.

Die im Laufe des Kapitels vorgebrachten Literaturverweise verdeutlichen, dass weder das Konzept der Systeme von Systemen noch die herausgearbeiteten Problemstellungen gänzlich neu sind. Die Informatik weist bereits Forschungsbestrebungen auf, sich den genannten Herausforderungen zu stellen. Allerdings argumentieren Northrop et. al. [Nor06], wirklich große und sozio-technisch integrierte Softwaresysteme von Softwaresystemen seien gerade erst dabei, sich in einer gewissen Vielfalt zu entwickeln. Die Bedeutung und Belastung der genannten Problemstellungen werden erst in der Zukunft offen zu Tage treten. Neben der Weiterentwicklung beziehungsweise der Verbesserung existierender Ansätze der Systementwicklung plädieren Northrop et. al. [Nor06, S. 4+5] vor allem für neuartige/innovative Ansätze. Dies betrifft insbesondere neuartige *konzeptionelle Modelle* für das Begreifen des Problembereiches. Genau diesem Aspekt widmet sich die vorliegende Arbeit vornehmlich. Es ist nicht möglich, im Rahmen dieser Arbeit umfassende Lösungsvorschläge für alle genannten Problemstellungen zu liefern. Stattdessen geht es um eine neuartige Perspektive für das Begreifen von Softwaresystemen im Großen. Diese Perspektive erachtet der Autor der vorliegenden Arbeit als Basis und

Verortungsmöglichkeit für daran anschließende, weiterführende Forschungsaktivitäten.

Im Laufe des Kapitels wurden bereits einige Vorschläge für konzeptionelle Modelle von Softwaresystemen zitiert. So wurden die Metaphern der *Softwarestadt* und der *Anwendungslandschaft* genannt. Eine weitere Metapher ist die des *Ökosystems* nach Northrop et. al. [Nor06, S. 6+7]. An verschiedenen Stellen des Kapitels wurde die Ansicht des Autors der vorliegenden Arbeit angedeutet, eine sozial inspirierte, organisationsorientierte Herangehensweise an die Gestaltung von Softwaresystemen würde besondere Vorteile mit sich bringen. Während es bislang lediglich bei Andeutungen blieb, ist das folgende Kapitel der ausführlichen Motivation einer solchen Herangehensweise gewidmet.

3 Motivation eines Organisationsparadigmas der Softwareentwicklung

Im vorigen Kapitel wurde die Forderung nach innovativen Perspektiven, konzeptionellen Modellen und Lösungsansätzen für die Beherrschung von Softwaresystemen im Großen, die einen Charakter als Systeme von Systemen aufweisen, zitiert. In der Einleitung der Arbeit wurde die vom Autor der Arbeit vertretene umfassende Hypothese bereits vorweggenommen: Der Einbezug von sozialen und insbesondere organisationstheoretischen Konzepten und Mechanismen für die Gestaltung von Softwaresystemen ist geeignet, den Weg für Lösungen der im vorigen Kapitel identifizierten Problemstellungen und Herausforderungen im Umgang mit großangelegten Softwaresystemen von Softwaresystemen zu bereiten.

Ausgehend von maschinennahen Assembler-Sprachen über prozedurale Abstraktionen, Objekte und Komponenten bis hin zu Diensten und Agenten hat die Informatik im Laufe der Jahrzehnte verschiedene Metaphern und Theorieelemente für die Betrachtung von Softwareentitäten und deren Komposition hervorgebracht. Dabei ist ein steigender Abstraktionsgrad der Metaphern, eine immer höhere Granularität der betrachten Basisbausteine und eine immer deutlichere Verwandtschaft/Übereinstimmung softwaretechnischer und realweltlicher Konzepte zu beobachten. Auf neue Denkmodelle folgten neue Programmiersprachen beziehungsweise Frameworks und Entwicklungsumgebungen für die unmittelbare softwaretechnische Umsetzung. Auf diese Weise entstanden verschiedene Softwareentwicklungsparadigmen. In der vorliegenden Arbeit vertritt der Autor die Auffassung, die Metapher der sozialen Organisation (allgemeiner: der organisierten Kollektivität) für die Betrachtung von Softwareentitäten und deren Zusammenspiel sei der logische nächste Schritt, um insbesondere den Herausforderungen von Systemen von Systemen begegnen zu können. Damit wird also ein Organisationsparadigma der Softwareentwicklung propagiert. Ausgehend von der Unterscheidung in der Einleitung der Arbeit geht es hier also ausdrücklich um organisationsorientierte Softwaresysteme im Sinne von Softwaresystemen *als* Organisationen (und nicht vorrangig um Softwaresysteme *für* Organisationen).

Wie in der Einleitung der Arbeit dargelegt, können die Angemessenheit, Nützlichkeit und Erfolgsaussichten eines Organisationsparadigmas nicht „bewiesen“ werden. Stattdessen ist dieses Kapitel einer qualitativen Argumentation für Softwaresysteme mit Organisationsstrukturen gewidmet. Fallstudien und Erfahrungswerte müssen später zeigen, ob das Paradigma seine Versprechungen halten kann. Allerdings gilt, dass mit der Organisationsorientierung keine gänzlich neuartige, revolutionäre Perspektive auf dem Prüfstand steht. Stattdessen wird in Kapitel 5 aufgezeigt, dass organisationsorientierte Softwareentwicklungskonzepte und das Konzept der Organisation als Metapher für Softwarebausteine bereits auf den Weg gebracht wurden.

Bevor allerdings existierende Ansätze beleuchtet, kritisiert und darauf aufbauend ein eigener, erweiterter Ansatz vorgestellt wird, wird in diesem Kapitel zunächst ein *grundsätzliches*

Plädoyer für eine Organisationsorientierung dargelegt. Es basiert ausschließlich auf konzeptionellen Betrachtungen der Merkmale realweltlicher/sozialer Organisationen und deren prinzipiellen Versprechungen hinsichtlich der im vorigen Kapitel identifizierten softwaretechnischen Problemstellungen. Der Wert einer solchen Diskussion liegt in der Annäherung an das Wesen von Organisationen sowie ihres Potentials für die Softwareentwicklung „in Reinform“. Nach Kenntnisstand des Autors findet sich eine Motivation in dieser grundsätzlichen Form bisher nicht in der Literatur. Die Argumentation wird dabei entlang von vier Hauptargumenten geführt. Diese wiederum orientieren sich an vier charakteristischen Merkmalen sozialer Organisationen.

1. Organisationen sind Meso-Strukturen zwischen Mikro- und Makro-Ebene. Dieses Argument stellt im Grunde genommen den Ausgangspunkt und die Grundlage für die weiteren Argumente dar.

2. Organisationen schaffen und erhalten Stabilität und Ordnung.

3. Das Konzept der Organisation unterstützt eine sozio-technische Perspektive inhärent.

4. Die Organisationsperspektive bringt multiple Ebenen des kollektiven Handelns mit sich.

Jedem der Argumente wird ein eigener Abschnitt gewidmet. Dabei beleuchtet jeder der Abschnitte auch einen Teil des Wesens des Konzeptes der sozialen Organisation. Eine allgemeingültige Definition des Begriffes Organisation gibt es nicht. Scott [Sco03, S. 9] beschreibt das Studium von sozialen Organisationen vor allem als im Spannungsfeld multidisziplinärer Forschung verortet. Verschiedene Wissenschaftsbereiche wie die Soziologie, die Betriebswirtschaftslehre (Management) oder die Politikwissenschaften betonen jeweils verschiedene Organisationseigenschaften mehr oder weniger stark. In der vorliegenden Arbeit werden soziologische Studien besonders betont, doch auch die Perspektiven anderer Wissenschaftsbereiche werden berücksichtigt. In jedem Abschnitt wird ein bestimmtes Merkmal von sozialen Organisationen betrachtet und darauf aufbauend eine Bewertung des Potentials für die Softwareentwicklung abgeleitet.

3.1 Organisationen als Meso-Strukturen zwischen Mikro- und Makroebene

Zunächst erfolgt an dieser Stelle ein Rückgriff auf einige der in Kapitel 2 zusammengetragenen Kernkonzepte im Zusammenhang mit großangelegten Systemen von Systemen.

- Die einzelnen Teilsysteme sind unabhängig in Bezug auf Management und Operation.
- Die Entwicklung eines Systems von Systemen ist evolutionär.
- Das Zustandekommen eines integrierten Ganzen basiert auf dem Prinzip der „freiwilligen“ Kooperation der Teilsysteme.
- Globale Steuerungs- und Kontrollmechanismen sorgen für die Entstehung gewünschten beziehungsweise erlaubten Systemverhaltens. Steuerungs- und Kontrollmöglichkeiten sind dabei durch die Autonomie der Teilsysteme eingeschränkt.

Diese Eigenschaften implizieren eine unmittelbare und starke Analogie zwischen Charakteristika von Softwaresystemen im Großen und der Mikro/Makro-Problematik der Soziologie. Nach Weber [Web72, S. 1] ist die Soziologie eine Wissenschaft, die soziales Handeln deutend verstehen und dadurch in seinem Ablauf und in seinen Wirkungen ursächlich erklären will. Die Mikro/Makro-Problematik nimmt dabei eine prominente Stellung ein. Für einen umfassenden Überblick über den Hintergrund und die Geschichte der Mikro/Makro-Debatte in der Soziologie sei hier auf den Artikel von Alexander und Giesen [AG87] verwiesen. Eine allgemeine Charakterisierung der Thematik ist hier ausreichend, da in diesem Kapitel das speziellere Phänomen der Organisation im Mittelpunkt steht.

Als Grundlage der Mikro/Makro-Problematik dient der weithin akzeptierte Wirkungszusammenhang von sozialem Handeln und sozialer Struktur. (1) Soziale Strukturen sind das Produkt sozialen Handelns, des sozialen Zusammenwirkens mehrerer. (2) Soziales Handeln wiederum wird wesentlich durch soziale Strukturen geprägt. Soziale Strukturen beziehen sich dabei auf musterbehaftete oder regelmäßige Aspekte der Beziehungen zwischen sozialen Akteuren. Sie treten in vielen Formen auf, beispielsweise in informellen Gruppen, Gemeinden, Organisationen oder ganzen Gesellschaften. Als generischen Oberbegriff all dieser Phänomene wird in der vorliegenden Arbeit das soziologische Konzept der *Kollektivität* verwendet. Scott [Sco03, S. 20+186] bezeichnet eine Kollektivität als *eine spezifische Instanz sozialer Ordnung*. Dies bedeutet im Wesentlichen, dass eine Kollektivität eine soziale Struktur in Form eines begrenzten Netzwerkes sozialer Relationen zwischen Akteuren aufweist. Die soziale Struktur setzt sich wiederum aus drei Unterstrukturen zusammen: (1) Einer *normativen Struktur* für generalisierte Regeln bezüglich angebrachten beziehungsweise erwarteten Verhaltens, (2) einer *kulturell-kognitiven* Struktur hinsichtlich gemeinsamer Werte und Begriffsinhalte sowie (3) einer *Verhaltensstruktur*, die das tatsächliche Verhalten der Kollektivität beschreibt (von Interesse sind hier wiederkehrende, regelmäßige Verhaltensmuster).

Stellvertretend beschreibt Blau [Bla87] nun, Mikro-Soziologie und Makro-Soziologie würden unterschiedliche Perspektiven auf soziales Leben einnehmen und es daher auch auf unterschiedliche Weise erklären. Zunächst einmal sind die Ebenen der Analyse verschieden. Der Ausgangspunkt bei Mikro-Soziologie sind Individuen, bei Makro-Soziologie sind es hingegen Kollektivitäten. Dementsprechend sind auch die betrachteten Konzepte und Variablen verschieden. Mikro-Soziologie betrachtet Attribute von Menschen, Makro-Soziologie hingegen emergente Eigenschaften von Kollektivitäten. Dementsprechend liefern Mikro- und Makro-Soziologie auch verschiedene Erklärungen sozialer Relationen und der darauf aufbauenden komplexeren sozialen Muster und Phänomene. Mikro-Soziologie analysiert soziale Prozesse, die Relationen zwischen *Menschen* erzeugen. Der Fokus liegt auf sozialen Interaktionen und Kommunikation, wichtige Konzepte sind Reziprozität, Obligationen, Austausch, Abhängigkeit. Makrosoziologie hingegen analysiert Strukturen zwischen verschiedenen *Positionen* in einer Kollektivität und deren beschränkender Wirkung auf soziale Relationen. Der Fokus liegt auf externen Limitationen der sozialen Umwelt menschlicher Beziehungen, wichtige Konzepte sind Differenzierung, Ungleichförmigkeit, Institutionen und Heterogenität.

Zusammenfassend formuliert wird in der Mikro-Soziologie eine Aufgliederung der *internen* Dynamik sozialer Relationen vorgenommen, während in der Makro-Soziologie die *externen* Einflüsse auf und Beschränkungen von sozialen Relationen betrachtet werden. Den beiden Perspektiven wohnt ein unterschiedliches Verständnis des Bedingungsverhältnisses von Handlung und Struktur inne. Der System-/Strukturorientierung der Makro-Soziologie liegt die Annahme zu Grunde, kontextübergreifende Strukturen würden kontextspezifischem Handeln vorausgehen und das Handeln wesentlich prägen. Der Handlungs-/Akteursorientierung der

Mikro-Soziologie liegt die Annahme zu Grunde, das Aufeinandertreffen sozialer Akteure würde Strukturen kontextsensitiv erzeugen. Auf der einen Seite werden Strukturen als Bedingung für Handlungen angesehen, auf der anderen Seite als handlungsabhängige Folge. Dabei gilt, dass weder die mikro- noch die makro-soziologische Perspektive als die alleinig richtige oder wichtigere anzusehen ist. Auch wenn klassische soziologische Theorien häufig eine klare Tendenz zu einer der beiden Perspektiven aufweisen, so existieren doch Ansätze, die sich um die Aufhebung einer solchen Einseitigkeit bemühen. So kritisiert beispielsweise Giddens [Gid84, S. 16–28] die Konnotation des Statischen im Zusammenhang mit dem Begriff Struktur. Stattdessen schlägt er den Begriff der *Strukturation* vor. Damit wird der *duale Charakter* sozialer Strukturen charakterisiert: Sie sind einerseits das Medium, andererseits das Ergebnis von sozialen Handlungen. Strukturen wirken einschränkend (allerdings gleichermaßen ermöglichend beziehungsweise anregend) auf soziale Handlungen. Gleichzeitig gilt, dass konkrete soziale Strukturen nur existieren, solange sie von sozialen Handlungen produziert und reproduziert werden. Brechen soziale Handlungen aus den durch die Struktur induzierten Mustern aus, so kann dies zu Strukturmodifikationen führen. Konkrete soziale Strukturen operieren nur zeitlich und örtlich begrenzt. Das Verhältnis von sozialen Strukturen und sozialen Handlungen ist eines der *wechselseitigen Konstitution*.

Die Analogie zu den Herausforderungen, die sich im Kontext des Umgangs mit Softwaresystemen von Softwaresystemen stellen, werden offenbar. Einerseits sind Teilsysteme unabhängig vom Gesamtsystem, partizipieren auf freiwilliger Basis und Interaktionen zwischen Teilsystemen lassen Systemstrukturen auf höherer Systemebene erst entstehen. Andererseits müssen Steuerungs- und Kontrollmechanismen etabliert werden, die zu gewünschtem beziehungsweise erlaubtem Systemverhalten führen. Nach Ansicht des Autors der vorliegenden Arbeit stellt die Mikro/Makro-Dynamik der Soziologie ein hervorragendes konzeptionelles Denkzeug für Systemen von Systemen dar. Einige Systeme nehmen dabei die Rolle individueller Akteure ein, andere wiederum die Rolle von Kollektivitäten. Es ist festzuhalten, dass soziale Gesellschaften trotz der Autonomie, Diversität und Eigeninteressiertheit ihrer Akteure ein erstaunliches Maß an Stabilität, Ordnung und Steuerbarkeit aufweisen. Hier ergibt sich also ein großes Potential, Inspirationen aus dem Studium sozialer Gesellschaften für den Umgang mit großangelegten Softwaresystemen zu ziehen. Dieses Potential wurde beispielsweise im Bereich der Multiagentensystemforschung und speziell dem Forschungsfeld der Sozionik erkannt, wie in den Kapiteln 5 und 6 der vorliegenden Arbeit erläutert wird.

Die in diesem Abschnitt aufgezeigte Analogie und das damit verbundene Potential des Einbezugs von sozialen Konzepten für die Gestaltung von Softwaresystemen stellt die Grundlage für die weitere Argumentation dieses Kapitels dar. Die Frage ist nun, inwiefern speziell das Konzept der sozialen Organisation eine wichtige Hilfestellung liefern kann. Zum einen stellt die Einbettung individueller Organisationsteilnehmer in einen organisatorischen Kontext bereits für sich gesehen eine spezielle Instanz von Mikro/Makro-Dynamik dar. Allerdings greift diese Sichtweise zu kurz. Im Rahmen organisationstheoretischer Betrachtungen werden üblicherweise multiple Analyseebenen betrachtet (siehe beispielsweise Scotts [Sco03, S. 16+17] Klassifizierung oder Abschnitt 3.4 dieses Kapitels). Insbesondere gehen diese Analyseebenen über die Betrachtung *einer* Organisation hinaus. In diesem Zusammenhang tut sich eine noch vielversprechendere Perspektive auf. Organisationen sind auf einer *Meso-Ebene* zwischen Mikro- und Makroebene angeordnet. Die Betrachtung von Meso-Strukturen ist insbesondere dann hilfreich, wenn Makro-Phänomene auf sehr hoher Ebene von Interesse sind. Ein unmittelbarer Übergang von der Mikro- zur Makro-Ebene ist dann unpraktikabel beziehungsweise lässt außer Acht, dass die soziale Realität verschiedene Kontexte und Ebenen des Handelns beinhaltet. So

beschreibt auch Johnson [Joh08, S. 9], Meso-Strukturen als soziale Formationen seien genau das geeignete Mittel, um die *Angesicht-zu-Angesicht-Interaktionen* der Mikro-Ebene mit den weitläufigen und umfassenden Makro-Strukturen und -Mustern der gesamten Gesellschaft in Beziehung setzen zu können. In Bezug auf Softwaresysteme von Softwaresystemen ergibt sich demnach das Potential, Erkenntnisse hinsichtlich der Beziehung zwischen den Teilsystemen, die den unteren Systemabschluss bilden, und dem Gesamtsystem von Systemen zu erlangen.

Organisationen sind natürlich nicht die einzige Form von Kollektivitäten auf Meso-Ebene. Stattdessen lassen sich bei der Untersuchung sozialer Gesellschaften multiple Instanzen der wechselseitigen Konstitution zwischen sozialen Strukturen und sozialen Handlungen auf multiplen Ebenen der Systemabstraktion identifizieren. Im weiteren Verlauf des Kapitels wird herausgearbeitet, warum gerade Kollektivitäten mit „Organisationscharakter“ vielversprechend im Hinblick auf ihre Hilfestellung für die Softwareentwicklung sind.

3.2 Organisationen als Systeme der Stabilität und Ordnung

Wenn in der Organisationstheorie das Aufkommen von Organisationen in modernen Industriegesellschaften thematisiert wird, so bedeutet dies selbstverständlich nicht, vorherige gesellschaftliche Arrangements seien unorganisiert gewesen. Der Kontrast ergibt sich vielmehr aus unterschiedlichen Typen von Gesellschaftsstrukturen. Wesentliche Unterschiede zwischen modernen und früheren Strukturen betreffen (1) die Beziehung zwischen Individuen und korporativen Einheiten und (2) die Beziehung zwischen korporativen Einheiten untereinander. Coleman [Col90, Kapitel 20] beschreibt, dass frühere korporative Einheiten ihre individuellen Mitglieder „umfassten“, ein Höchstmaß an Autorität über sie besaßen und selber wiederum in einer strikten Hierarchie angeordnet waren. Diese Beziehungen haben sich über die Jahrhunderte verändert. Moderne korporative Einheiten umfassen ihre Mitglieder nicht länger. Ihr Einfluss auf das Verhalten ihrer Mitglieder beschränkt sich auf den spezifischen Teil, zu dem sich die Mitglieder (gegebenenfalls vertraglich) verpflichten, wenn sie der korporativen Einheit beitreten. Akteure sind nur noch *partiell involviert* in typischerweise multiplen korporativen Einheiten, entsprechend ihrer zahlreichen Interessen und Rechte. Unter den korporativen Einheiten herrscht dadurch ein hohes Maß an Diversität und Spezialisierung, sie nehmen den Charakter zweckgerichteter *korporativer/kollektiver Akteure* an. Das Konzept des kollektiven Akteurs wird in Abschnitt 3.4 wieder aufgegriffen und vertieft.

Im Gegensatz zu früheren Gesellschaftsstrukturen weist die moderne Gesellschaft demnach ein beträchtlich höheres Maß an struktureller Vielfalt auf und birgt damit ein größeres Potential für Unordnung und Instabilität. Die Problematik lässt sich anhand der Ausführungen aus Kapitel 2 unmittelbar auf großangelegte Softwaresysteme von Softwaresystemen übertragen. Insbesondere der dort hervorgehobene Aspekt der Dezentralisierung in Bezug auf Operation, Management und Steuerung von Systembestandteilen findet sich analog in der obigen Beschreibung moderner Gesellschaftsstrukturen wieder. Auch für soziale Gesellschaften bedeutet Dezentralisierung im Normalfall nicht den Verzicht auf globale Autoritäten. Allerdings werden Steuerungsentscheidungen betreffend vielfältiger Aspekte an verschiedenen Stellen auf verschiedenen Ebenen von verschiedenen Akteuren getroffen. In sozialen Gesellschaften moderner Prägung sind es gerade die Organisationen unter den korporativen Einheiten, die trotz der geschilderten Charakteristika Ordnung schaffen und erhalten. Deswegen sind sie für die Verwaltung der unzähligen Aktivitäten des sozialen Lebens so gefragt. Es ist wenig überraschend, dass sich die charakteristischen Merkmale von Organisationen in sozialen Gesellschaften nach Han-

nan und Carroll [HC95] wie wünschenswerte Eigenschaften von Softwaresystemen im Großen lesen: Organisationen sind *langlebig*, *verlässlich* und *rechenschaftsfähig/nachvollziehbar*.

- *Langlebigkeit:* In höherem Maß als für andere sozialen Strukturen gilt für Organisationen, dass sie langlebig sind sowie routinemäßig und fortlaufend Bemühungen unterstützen, um eine bestimmte Menge beziehungsweise Art von Aktivitäten auszuführen.

- *Verlässlichkeit:* Organisationen etablieren Steuer- und Kontrollmechanismen (unter ihnen Formalisierung, Autoritätsstrukturen, Regelsysteme, Routinen, kulturelle Einflussnahme oder spezialisierte technische Systeme). Auf diese Weise wird die Verlässlichkeit in Bezug auf die Verrichtung der organisatorischen Aktivitäten gefördert.

- *Rechenschaftsfähigkeit und Nachvollziehbarkeit:* Organisatorische Aktivitäten sind eingebettet in ein Gerüst aus Regeln und Richtlinien, die den Mitgliedern eine Hilfestellung sowohl im Hinblick auf zukünftige Entscheidungen und Handlungen als auch im Hinblick auf die Rechtfertigung vergangener Entscheidungen und Handlungen bieten. In modernen Gesellschaften ist dieses Gerüst typischerweise verbunden mit und unterstützt durch globalere Richtlinien, die den Einfluss und die Grenzen von Organisationen innerhalb umschließender Kontexte definieren.

Abstrakt formuliert wird hier also die Funktion von Organisationen als Ordnungsstifter in verteilten, dezentralisierten Systemen mit autonomen, zweckgerichteten Teilkomponenten hervorgehoben. Konzepte und Mechanismen zur Stützung dieser Funktion existieren in sozialen Gesellschaften. Diese auf die Softwareebene zu überführen leistet nach Ansicht des Autors der vorliegenden Arbeit einen wesentlichen Beitrag für die Beherrschbarkeit von großen Softwaresystemen. Während der Nutzen der Merkmale der Langlebigkeit und Verlässlichkeit unmittelbar offensichtlich ist[1], so sei die Bedeutung des Merkmals der Rechenschaftsfähigkeit hier explizit hervorgehoben. Gerade für Softwaresysteme von Softwaresystemen mit ihren unabhängigen Teilsystemen, der evolutionären Entwicklung und den damit einhergehenden dynamisch entstehenden Funktionen und Anforderungen haben Rahmenwerke für die Anleitung und Rechtfertigung von Entscheidungen und Handlungen einen hohen Wert. Dieser Aspekt korrespondiert zu der in Kapitel 2 mehrfach zitierten Forderung, direkte Steuerung bevorzugt durch Regulierung zu ersetzen. Vorgaben und Beschränkungen engen den Handlungsspielraum von beteiligten Systemen ein, ohne direkt in ihr Handeln einzugreifen. Gleichzeitig dienen sie den Entwicklern von Teilsystemen als Hilfestellung bei der Bewertung, Ausrichtung und Rechtfertigung der jeweiligen Systemfunktionalität (im Falle von Softwaresystemen mit Selbst-Eigenschaften kommt der zusätzliche Aspekt hinzu, dass Regeln und Richtlinien maschinenlesbar vorliegen und von den Softwaresystemen selber interpretiert werden müssen).

Stabilität, Verlässlichkeit und Rechenschaftsfähigkeit hängen wesentlich mit *Formalisierung* beziehungsweise *formalen Organisationsstrukturen* und dem damit einhergehenden soziotechnischen Charakter von Organisationen zusammen. Dieser Aspekt wird im Folgenden Abschnitt näher betrachtet.

[1] Der Umstand, dass beide Merkmale keinen reinen Segen darstellen, sei an dieser Stelle nur erwähnt und nicht vertieft diskutiert. Beispielsweise sind Faktoren, die Langlebigkeit und Verlässlichkeit sicherstellen, häufig (wenn auch nicht notwendigerweise) mit Rigidität assoziiert.

3.3 Organisationen als sozio-technische Systeme

In Abschnitt 3.1 wurde die wechselseitige Konstitution zwischen der sozialen Struktur einer Kollektivität und den sozialen Handlungen ihrer Mitglieder betont. Im Falle von Organisationen als Kollektivitäten erhalten die sozialen Strukturen ein besonderes Gewicht durch *Formalisierung*. Nach Scott [Sco03, S. 35] bezieht sich Formalisierung (im soziologischen Sinne) auf zwei Aspekte. (1) Formalisierung bedeutet eine explizite und präzise Spezifikation von verhaltensbestimmenden Regeln. (2) Formalisierung bedeutet die Spezifikation von Positionen, Rollen und einhergehenden sozialen Beziehungen unabhängig von den persönlichen Attributen und Beziehungen der Akteure, die diese Rollen tatsächlich einnehmen.

Natürlich können nicht alle sozialen Strukturen einer Organisation formalisiert werden, zumal auch die Formalisierung selber als Auslöser für die Entstehung weiterer, nicht vorhergesehener sozialer Strukturen dienen kann. In Organisationen findet sich daher ein Nebeneinander beziehungsweise Widerspiel von formalen und informellen Strukturen.[2] Diesen Umstand hebt Trist [Tri81] als eine kennzeichnende Eigenschaft von Organisationen hervor: Organisationen umfassen sowohl technische als auch soziale Systeme und die Relation zwischen beiden stellt die Kernschnittstelle des Gesamtsystems dar. Mit dem technischen/technisierten Anteil einer Organisation sind in diesem Kontext weniger physikalische Komponenten gemeint. Vielmehr ist das durch Formalisierung technisierte Wesen von Organisationen an sich gemeint. Dies bezieht sich unter anderem auf Aufgabenspezialisierung, Standardisierung, Gruppierung oder Autoritätshierarchien. Formalisierung dient der Standardisierung und Vorhersagbarkeit von Verhalten in Organisationen. Der Grad an Formalisierung spielt eine wichtige Rolle in Mintzbergs [Min83] Gegenüberstellung verschiedener Organisationsformen. Je dominanter die aus Formalisierung hervorgegangenen technischen Faktoren einer Organisation sind, als desto *bürokratischer* charakterisiert Mintzberg [Min83, S. 36] sie. Organisationsformen mit einem sehr hohen Maß an Formalisierung charakterisiert er als *maschinelle Bürokratien* [Min83, Kapitel 9]. Bürokratischen Strukturen haftet häufig eine negative Assoziation an, da sie die Entscheidungshoheit und das Verhalten betroffener Akteuren wesentlich einschränken. Allerdings ist diese Sichtweise etwas einseitig. Denn gleichzeitig kann eine Menge von Einschränkungen, im Kontext derer Entscheidungen getroffen und Handlungen ausgeführt werden, auch als notwendige Hilfestellung angesehen werden. Bürokratischen Strukturen gegenüber sieht Mintzberg [Min83, S. 36] *organische Strukturen*, die mit einem geringen Maß an Standardisierung auskommen. Als Gegenentwurf zur maschinellen Bürokratie beschreibt Mintzberg [Min83, Kapitel 12] die im folgenden Kapitel etwas näher erläuterte *Adhoc-kratie* (*adhocracy*) als hochorganische Organisationsform. Der Grad an Formalisierung spannt ein Kontinuum an Organisationsformen auf, an dessen Enden maschinelle Bürokratien und Adhoc-kratien stehen.

Anhand Scotts [Sco03, Kapitel 2+3] Gegenüberstellung der Perspektiven auf Organisationen als *rationale* und als *natürliche Systeme* wird der sozio-technische Charakter besonders deutlich.

[2] Die Übersetzung der englischen Adjektive *formal* und *informal* in ihrer soziologischen Verwendung variiert in deutschsprachigen Texten je nach Quelle. So finden sich als Übersetzungen *formal/informal*, *formell/informell* sowie Mischformen, wie beispielsweise in der Kapitelüberschrift „Formale und informelle Rollen in Organisationen“ des Buches von Isa Jahnke [Jah07, Kapitel 3]. Tatsächlich bevorzugt der Autor der vorliegenden Arbeit die dritte Variante. Einerseits ist speziell der Begriff der *Formalstruktur* einer Organisation weit verbreitet und akzeptiert. Andererseits ist der Gebrauch des Adjektivs *informal* gegenüber dem des Adjektivs *informell* nach dem Sprachgefühl des Autors weniger verbreitet. Daher wird im folgenden der Arbeit von *formalen* und *informellen* sozialen Strukturen die Rede sein. Das Adjektiv *formalisiert* wird synonym zum Adjektiv *formal* verwendet.

- *Organisationen als rationale Systeme:* Die Perspektive auf Organisationen als rationale Systeme geht mit der Betonung der unverwechselbaren Kriterien einher, die Organisationen von anderen Kollektivitäten unterscheiden. Im Wesentlichen handelt es sich bei diesen Kriterien um *(1) Ausprägung spezifischer Ziele* und *(2) ein hohes Maß an Formalisierung*. Die Merkmale und die Funktion von Formalisierung wurden bereits erläutert. Zielspezifität sagt aus, dass Organisationen zweckgerichtete Kollektivitäten sind. Die Thematik des Setzens und Verfolgens von Organisationszielen wird im folgenden Kapitel ausführlicher behandelt. Beide Kriterien sind als Variablen aufzufassen, entlang derer verschiedene Organisationen variieren. Insgesamt unterscheidet es Organisationen von anderen Kollektivitäten wie beispielsweise Familien, Gruppen, Kommunen oder sozialen Bewegungen, dass sie beide Kategorien in einem höheren Maß und in Kombination aufweisen.

- *Organisationen als natürliche Systeme:* Demgegenüber geht die Perspektive auf Organisationen als natürliche Systeme mit der Betonung der Gemeinsamkeiten mit allen Formen von Kollektivitäten einher. Die Unterscheidungsmerkmale müssen nicht zwingend die wichtigsten oder einflussreichsten sein. Organisationen mögen spezifische Ziele haben und eine Formalstruktur aufweisen. Das bedeutet aber nicht, dass diese auch tatsächlich einen alles andere außer Kraft setzenden Einfluss auf die Mitglieder hätten. Stattdessen sind Mitglieder motiviert durch ihre eigenen Interessen und versuchen, diese auf die Organisation zu übertragen. Die Organisation wird als eine wichtige Ressource angesehen, um eigene Interessen zu wahren oder zu verfolgen. Informelle und zwischenmenschliche Strukturen werden als wichtiger und bestimmender als die Formalstruktur angesehen.

Keine der beiden Perspektiven ist richtiger oder wichtiger als die andere, beide repräsentieren „halbe Wahrheiten“. Sie weisen auf unterschiedliche Kräfte und Einflüsse hin, die *gemeinsam* auf *jede* Organisation wirken.

Die soziale Organisationen ist also ein eingehend untersuchter Anwendungsfall soziotechnischer Kopplung und kann damit als universelle Inspirationsquelle und Anleitung für die Realisierung sozio-technischer Kopplung an anderer Stelle dienen. Die in Abschnitt 3.1 begonnene Argumentation für eine sozial fundierte Sichtweise auf Softwaresysteme fußt darauf, Softwaresysteme und -komponenten als individuelle Akteure, Kollektivitäten oder als Kombination aus beidem (kollektive Akteure) zu konzeptionalisieren. Anhand der Ausführungen dieses Abschnittes wird deutlich, dass die spezifischere organisationsorientierte Perspektive dabei eine erhebliche Hilfestellung bietet. Stärker als im Falle anderer Kollektivitäten findet im Falle von Organisationen eine Formalisierung von sozialen Strukturen sowie eine Explizierung globaler Ziele statt. Diese beiden Aspekte sorgen in realweltlichen Gesellschaften für eine höhere Sichtbarkeit von Organisationen gegenüber informellen Kollektivitäten und für eine bessere Planbarkeit, Steuerbarkeit und Nachvollziehbarkeit ihres kollektiven Verhaltens. Gleichzeitig fällt damit nach Ansicht des Autors der vorliegenden Arbeit eine Überführung der Konzepte und Mechanismen von sozialen Organisationen auf softwaretechnischen Programmcode leichter als im Falle von Kollektivitäten, die auf informellen Strukturen beruhen. Hierbei können zwei Aspekte sozio-technischer Integration unterschieden werden.

- *Integration zwischen softwaretechnischen/künstlichen und realweltlichen/natürlichen sozialen Systemen:* Die in Teilabschnitt 2.2.4 formulierte Forderung nach soziotechnischer Integration bezieht sich auf softwaretechnische Systeme und natürliche/realweltliche soziale Systeme. Mit der Übertragung realweltlicher Organisations-

konzepte und -mechanismen auf Softwaresysteme ergeben sich auch die Möglichkeiten für eine solche Integration unmittelbar. So können soziale Akteure beispielsweise mittels einer Benutzungsschnittstelle als Akteure im Kontext einer softwaretechnischen Organisation auftreten. Die Gleichartigkeit von Terminologie und Konzepten auf sozialer und softwaretechnischer Seite eröffnet sozialen Akteuren dabei ein intuitives Verständnis ihrer Position/Stellung gegenüber, beziehungsweise im Kontext des Softwaresystems (beispielsweise als Kunde, Operator oder Manager).

Gleichzeitig wird auch die Integration auf höherer Granularitätsebene gefördert. Konzepte wie Ziele, Strategien und Regeln sowie die analytische Aufteilung in verschiedene Systembereiche (beispielsweise Ausführung von Kernaufgaben, Support, Managementebenen) existieren auf softwaretechnischer und sozialer Seite, so dass eine *gemeinsame* Ausrichtung und Optimierung unterstützt werden.

Analog zu realweltlichen Organisationen gibt es ein Spektrum zwischen sehr bürokratischen und sehr organischen Softwarestrukturen. Bürokratische Strukturen lassen lediglich eine eingeschränkte und fixe Interaktion mit dem Softwaresystem beziehungsweise vermittelt durch das Softwaresystem zu. Auf der anderen Seite gewinnt das System dadurch an Verlässlichkeit und Ablaufeffizienz. Organische Strukturen erlauben flexible Interaktionen und sind gegebenenfalls modifizierbar. Damit wird allerdings dem Einfluss der komplexen Kräfte und Wirkungen sozialer Kollektivitäten Zugang gewährt.

- *Integration künstlicher sozialer Systeme:* Der sozio-technische Ansatz kann auch auf die Integration rein softwaretechnischer Systemanteile untereinander angewandt werden. Eine sozial fundierte Perspektive auf Softwaresysteme und -komponenten als Akteure und Kollektivitäten führt zu *künstlichen sozialen Systemen*. Damit kann der sozio-technische Ansatz auch als Grundlage für die Integration softwaretechnischer Akteure in softwaretechnische Kollektivitäten dienen. Auch hier kommt es zum Widerspiel formaler und informeller (künstlicher) sozialer Strukturen Diese Herangehensweise spielt aktuell bereits in der Multiagentensystemforschung eine wichtige Rolle (vergleiche Kapitel 5.2). Allgemeiner spielt dieser Aspekt eine wichtige Rolle für die in Teilabschnitt 2.2.5 behandelten Softwaresysteme mit Selbst-Eigenschaften.

Die beiden Aspekte sind natürlich verzahnt. In großangelegten Softwaresystemen ist zu erwarten, dass natürliche und künstliche soziale Akteure nebeneinander und miteinander agieren. Analog zu realweltlichen Organisationen stellt sich natürlich auch im Falle softwaretechnischer Organisationen die Frage nach eher bürokratischen oder organischen Strukturen. Wiederum bietet sich der Erfahrungsschatz der Organisationstheorie an, dieser Problematik Herr zu werden. So betont beispielsweise Mintzberg [Min83, Kapitel 6], dass in der Hauptsache die Charakteristika der Umgebung einer Organisation die Favorisierung von eher rationalen oder eher natürlichen Einflüssen bestimmen. Stark vereinfacht ausgedrückt sind bürokratische Organisationsformen für stabile Umgebungen angemessen während organische Organisationsformen für dynamische Umgebungen angemessen sind. Als weiteren Ansatz machen sowohl Thompson [Tho03, S. 10–13] als auch Mintzberg [Min83, S. 9–23] die Nützlichkeit mehr oder weniger formaler Strukturen an unterschiedlichen Ebenen innerhalb einer Organisation wie dem operationalen Kern, dem mittleren Management oder dem strategischen Apex fest. Wiederum stark vereinfacht ausgedrückt sind organische Strukturen desto angemessener, je weniger repetitiv und je flexibler die Handlungen der jeweiligen Akteure sind. Dies bedeutet, dass auf immer höheren Ebenen des Managements immer organischere Strukturen anzutreffen

sind. Die Konzepte der Umgebung und unterschiedlicher Systemebenen finden sich analog in Softwaresystemen von Softwaresystemen wieder, so dass sich hier Ansätze finden, zwischen unterschiedlichen Systemanteilen sozio-technisch zu differenzieren.

3.4 Organisationen als offene Systeme und kollektive Akteure

Die Ausführungen in Kapitel 2 und insbesondere Teilabschnitt 2.2.3 haben die Notwendigkeit des Einbezugs unterschiedlicher Abstraktionsebenen bei der Handhabung von großen Softwaresystemen von Softwaresystemen hervorgehoben. Der Abstraktionsgrad bezieht sich dabei auf die Granularität der auf einer Ebene betrachteten Systemeinheiten.

Die Organisationstheorie hat im Laufe der Jahrzehnte verschiedene Abstraktionsebenen beziehungsweise Analyseebenen der organisationsorientierten Betrachtung sozialer Gesellschaften identifiziert und systematisch studiert. Die Differenzierung beschränkt sich dabei nicht auf die offensichtliche Abgrenzung zwischen Organisation und Individuum. So propagiert Scott [Sco03, Kapitel 4] neben der Perspektive auf Organisationen als rationale und natürliche Systeme die zusätzliche (beziehungsweise orthogonale) Perspektive auf Organisationen als *offene Systeme*. Die zentrale Einsicht dieser Perspektive ist, dass alle Organisationen „unvollständig“ sind. Alle sind von Interaktionen mit weiteren Systemen abhängig. Alle sind in Umgebungen eingebettet und von diesen beeinflusst.

So unternimmt beispielsweise Pfeffer [Pfe82] eine Klassifikation organisationstheoretischer Forschung entlang der Dimension der Analyseebene (unter anderem). Pfeffer unterscheidet die *individuelle Ebene / Untereinheitenebene* von der *Organisationsebene*. Im ersten Fall bezieht sich Pfeffer auf die internen Anteile von Organisationen, unter ihnen Individuen und Gruppen wie Gremien oder organisationsinterne Koalitionen. Im zweiten Fall bezieht sich Pfeffer auf die Organisation als holistische Einheit und betrachtet auch Gruppen von Organisationen.

Eine analoge, aber verfeinerte Klassifizierung findet sich bei Scott [Sco03, S. 16–18]. Die *sozio-psychologische Ebene* bezeichnet die Analyseebene, auf der das Verhalten von individuellen Mitgliedern und deren zwischenmenschlichen Beziehungen untersucht werden. Organisatorische Charakteristika werden als Kontext oder Umgebung aufgefasst, deren Einfluss auf persönliche Haltungen und Verhaltensweisen es zu erforschen gilt. Als identifizierbare Einheit taucht die Organisation zum ersten Mal auf der *organisatorischen Strukturebene* auf. Diese beschreibt strukturelle Eigenschaften und soziale Aktivitäten, die die Organisation und ihre Untereinheiten charakterisieren. Untersuchungen befassen sich mit den verschiedenen Anteilen einer Organisation (wie Abteilungen, Arbeitsgruppen oder Teams) oder ihren analytischen Aspekten (Spezialisierung, Kommunikationsnetzwerk oder Hierarchie). Die *ökologische Ebene* fokussiert auf Charakteristika und Aktionen von Organisationen als kollektive Akteure, die in einem noch umfassenderen Netzwerk aus Beziehungen operieren. Für diese Ebene sind weitere Differenzierungen möglich. Unter ihnen bezeichnet das *organisatorische Feld* das umfassendste Konzept der unmittelbaren Umgebung einer Organisation. Nach DiMaggio und Powell [DP83] besteht ein organisatorisches Feld im Wesentlichen aus einer Menge von Organisationen, die zusammen einen anerkannten Bereich institutionellen Lebens (beispielsweise Anbieter und Konsumenten spezifischer Leistungen sowie Behörden für die Regulierung des Leistungsaustausches) begründen. Die *Gesellschaft* dient der Integration organisatorischer Felder in ein überspannendes Gesetzeswerk und benutzt die Felder als Vermittler für Verordnungen und Sanktionen, die eigentlich auf individuelle Organisationen abzielen.

Die Klassifikation nach Scott wird in Kapitel 11 in größerer Detailtiefe erneut aufgegrif-

fen. Sie steht dort Pate für eine Referenzarchitektur organisationsorientierter Softwaresysteme. Doch unabhängig von konkreten Charakterisierungen verschiedener Analyseebenen haben Studien, die Organisationen als holistische Einheiten in ihrer Umgebung thematisieren, die Auffassung von *Organisationen als Akteure* gemein. Nach Auffassung von Scott [Sco03, S. 7+8] kann man die Bedeutung von Organisationen für soziale Gesellschaften nicht ausreichend begreifen solange man sie lediglich als Kontexte für die Aktivitäten individueller Akteure auffasst. Organisationen sind selber *kollektive Akteure*. Sie können agieren, Gebrauch von Ressourcen machen, Verträge schließen und sie haben Besitz.

Bereits in Abschnitt 3.2 wurde der Begriff des kollektiven Akteurs aufgegriffen. Dort wurde auf Colemans [Col90, Kapitel 20] Untersuchungen hinsichtlich des Verhältnisses individueller Akteure und korporativer Einheiten verwiesen. In diesem Zusammenhang charakterisiert Coleman Organisationen im Gegensatz zu früheren Formen von korporativen Einheiten als zielgerichtete, ihre Mitglieder lediglich partiell einbindende kollektive Akteure. Coleman beschreibt die graduelle Entwicklung kollektiver Akteure seit dem Mittelalter bis hin zu einem Punkt, an dem es heutzutage gerechtfertigt ist, von zwei grundsätzlich verschiedenen „Personen“ zu sprechen, nämlich *natürlichen* Personen und *kollektiven* beziehungsweise *juristischen* Personen. Kollektive Akteure gehen über die Lebenszeit, Ressourcen, Energien und Macht der sie erschaffenden natürlichen Personen hinaus. Ihr Zweck ist, die Macht ihrer Mitglieder zu erhöhen und ihnen die Verfolgung ihrer Interessen und Ziele (beziehungsweise einer Teilmenge davon) zu erleichtern.[3] Das Konzept des kollektiven Akteurs birgt diverse strittige Aspekte. Einige von ihnen werden im folgenden Kapitel behandelt. Grundsätzliche Einigkeit innerhalb des Forschungsfeldes der Organisationstheorie besteht aber zumindest hinsichtlich der Notwendigkeit, kollektive Akteure in Theorien über moderne Gesellschaften einzubeziehen.

Das Konzept des kollektiven Akteurs bringt einen zusätzlichen Aspekt der Betrachtung von Kollektivitäten mit sich. Dieser Aspekt wurde bereits an mehreren Stellen des Kapitels angedeutet, jedoch bisher nicht weiter ausgeführt. Die Idee der wechselseitigen Konstitution von sozialer Struktur und sozialen Handlungen einer Kollektivität ist nicht auf Individuen als sozial handelnde und strukturell geprägte Akteure eingeschränkt. Stattdessen können die Mitglieder einer Kollektivität wiederum Kollektivitäten sein. Organisationen sind dabei nicht die einzige Form von kollektiven Akteuren. Allerdings sind sie mit der Kombination von Zielausprägung und Formalisierung „greifbarer“ in ihrer Funktion als identifizierbare, zweckgerichtete Akteure gegenüber anderen Formen kollektiver Akteure. Erst diese duale, janusgesichtige Auffassung von Organisationen – als Umgebungen für soziale Akteure sowie als Akteure in sich selbst – entfaltet ihr wahres Potential als Meso-Strukturen für den sukzessiven Übergang von der Mikro- zur Makro-Ebene.

Das Konzept des kollektiven Akteurs stellt nach Ansicht des Autors der vorliegenden Arbeit genau das zentrale Theorieelement dar, welches es für einen organisationsorientierten Ansatz der Gestaltung von Softwaresystemen im Großen zu berücksichtigen gilt und der bislang vernachlässigt wurde. Anhand der Ausführungen in Kapitel 5 wird offenbar, dass organisatorische Konzepte und Mechanismen bereits vielfach Einzug in die Softwareentwicklung gehalten haben. Dies ist nicht überraschend und ganz im Sinne des in diesem Kapitel vorgetragenen Plädoyers für einen organisationsorientierten Ansatz der Integration (teil-) autonomer, heterogener und verteilter Softwarebausteine. Sowohl die organisatorische Funktion des Schaffens

[3] Allerdings ist die Thematik, welche Ziele ein kollektiver Akteur aufweist und in welchem Zusammenhang diese zu den Interessen der Mitglieder stehen, ein komplexes Thema, das im folgenden Kapitel eingehender behandelt wird.

von Ordnung als auch die Funktion der sozio-technischen Integration finden bereits Berücksichtigung in verschiedenen Ansätzen. Es ist der Aspekt der Offenheit und des kollektiven Akteurswesens, der in Kapitel 5 als bislang unzureichend berücksichtigt identifiziert wird. Dabei stellt der janusgesichtige Charakter von Organisationen nach Ansicht des Autors genau das fehlende Glied dar, um existierende Softwareentwicklungsideen insoweit zu bereichern, dass sie für die Herausforderungen von Softwaresystemen der Kategorie ULS-System gewappnet sind. In diesem Zusammenhang bietet die Organisationstheorie einen reichhaltigen Fundus an Konzepten und Mechanismen. So wird nicht nur das grundlegende Konzept des kollektiven Akteurs behandelt. Darüber hinaus existieren Untersuchungen und Bewertungen ganz konkreter Systemkonfigurationen bestehend aus mikro-, meso- und makroperspektivischen Strukturelementen. Für jede Ebene existieren eingehend studierte Konzepte und Mechanismen, die sich in sozialen Gesellschaften als funktionstüchtig erwiesen haben. Sie können als Leitfaden für die Orchestrierung großer Softwaresysteme auf unterschiedlichen Ebenen der Abstraktion dienen.

3.5 Vorteile gegenüber anderen konzeptionellen Modellen

Nachdem in diesem Kapitel eine ausführliche Motivation der Gestaltung von Softwaresystemen als künstliche soziale Organisationssysteme propagiert wurde, erfolgt abschließend eine knappe Benennung von Vorteilen gegenüber anderen konzeptionellen Modellen von großangelegten Softwaresystemen. In Kapitel 2 wurden drei solcher Modelle benannt: Anwendungslandschaften, Softwarestädte und Ökosysteme.

Zunächst lässt sich festhalten, dass das konzeptionelle Modell des Ökosystems das konzeptionelle Modell des sozialen Organisationssystems im Grunde subsumiert (wenn menschliche Ökosysteme gemeint sind). Insofern kann man die gesamte in diesem Kapitel vorgenommene Motivation auch auf das konzeptionelle Modell des Ökosystems übertragen. Allerdings stellt die Sichtweise auf Softwaresysteme als Organisationssysteme eine deutliche Schärfung dar und ist auf diese Weise ausdrucksstärker in ihrer Funktion, die Gestaltung von Softwaresystemen zu leiten.

Nach Ansicht des Autors der vorliegenden Arbeit können im Wesentlichen zwei Vorteile eines Organisationsparadigmas gegenüber den konzeptionellen Modellen der Anwendungslandschaft und der Softwarestadt identifiziert werden.

1. Bei der Betrachtung des Konzeptes eines Systems von Systemen wurde insbesondere auf die Unabhängigkeit der Teilsysteme hingewiesen. Das Gesamtsystem basiert auf freiwilliger Teilnahme und Kooperation der Teilsysteme. Dennoch muss eine Integration der Teilsysteme auf der darüber liegenden Systemebene entlang globaler Zwecke und Richtlinien erreicht werden. Eine organisationsorientierte Sichtweise drängt sich hier geradezu auf. Teilnehmer einer sozialen Organisationen partizipieren freiwillig. Es muss also ein Anreiz für sie bestehen, an der Organisation teilzunehmen. Dieser Anreiz liegt beispielsweise in materieller Entlohnung, in der Übereinstimmung von Zielen oder der Unfähigkeit, eigene Ziele alleine zu erreichen. Über die Schaffung solcher Anreize gelingt es Organisationen, Macht und Kontrolle über ihre Mitglieder auszuüben. Darüber hinaus erlangt eine Organisation (speziell anhand des Mechanismus der Formalisierung) eine eigenständige Existenz unabhängig von ihren Mitgliedern, so dass sie den Beitritt und Austritt vieler Teilnehmer überdauern kann.

   Weder das konzeptionelle Modell der Anwendungslandschaft noch das der Softwarestadt unterstützen eine Diskussion des Aspektes der freiwilligen Kooperation und der daraus

entstehenden Problemstellungen inhärent. In beiden Fällen geht es eher um die Anordnung von Teilsystemen und der infrastrukturellen Verbindungen zwischen ihnen.

2. Die konzeptionellen Modelle des Organisationssystems, der Anwendungslandschaft und der Softwarestadt unterstützen alle die Betrachtung von dynamischen Softwaresystemstrukturen. Landschaften und Städte können umgestaltet/umgebaut werden. Dennoch ist der Autor der vorliegenden Arbeit der Meinung, die Sichtweise auf Softwaresysteme als Organisationssysteme erlaube eine flexiblere Behandlung von Dynamik. Dies liegt vor allem an dem großen Facettenreichtum der von der Organisationsmetapher unmittelbar abgedeckten Dynamiken: Einstellen/Entlassen von Mitgliedern, spontane Bildung von Projektteams, Eröffnen/Schließen von Abteilungen, Business Process Reengineering, strukturelle Reorganisation, Ausweiten des Betätigungsfeldes einer Organisation auf eine neue Domäne, Gründung neuer Organisationen, Bildung von strategischen Allianzen oder virtuellen Organisationen, institutioneller Wandel auf organisatorischen Feldern und viele mehr.

Bei beiden genannten Punkten handelt es sich um Kernaspekte des Konzeptes eines Systems von Systemen (unabhängige Teilsysteme, evolutionäre und damit hochdynamische Entwicklung) und bei beiden Punkten zeigt sich die Eignung der Sichtweise auf Softwaresysteme als künstliche Organisationssysteme besonders deutlich.

3.6 Zusammenfassung

In diesem Kapitel wurden die Kernmerkmale realweltlicher/sozialer Organisationen herausgearbeitet und deren Potential für die Anwendung auf die Gestaltung von Softwaresystemen offengelegt. Auf diese Weise wurde ein Organisationsparadigma der Softwareentwicklung motiviert. Die Vorteile der Gestaltung von Softwaresystemen als künstliche Organisationssysteme wurden dabei anhand der Problemstellungen aus Kapitel 2 im Zusammenhang mit softwareintensiven Systemen von Systemen deutlich gemacht. Die Kombination der Organisationsmerkmale als Meso-Strukturen und kollektive Akteure, als Ordnungsstifter sowie als Systeme mit inhärent sozio-technischer Kopplung erlauben gemeinsam die Adressierung der Problemstellungen Dezentralisierung, multiple Systemebenen, sozio-technische Integration und Software mit Selbst-Eigenschaften.

Während das in diesem Kapitel vorgetragene Plädoyer die grundsätzliche Eignung eines sozial fundierten Organisationsparadigmas der Softwareentwicklung motiviert hat, so sind Möglichkeiten der konkreten Überführung von sozialen Konzepten und Mechanismen auf die Softwareebene eher vage geblieben. Für ein Organisationsparadigma muss vor allem klar sein, wie die Bausteine von Softwaresystemen aussehen beziehungsweise welche Kerneigenschaften sie haben. So sind die Bausteine von objekt- beziehungsweise agentenorientierten Softwaresystemen durch die Konzepte des Objektes beziehungsweise des Agenten (relativ) klar charakterisiert. Für Multiorganisationssysteme als konkreter Form von organisationsorientierten Softwaresystemen wurde in der Einleitung der Arbeit die Organisationseinheit als Baustein in Aussicht gestellt. Wie eine solche Organisationseinheit konkret aussieht, wurde bislang nicht beschrieben. Dies erfolgt erst im zweiten Teil der Arbeit. Das folgende Kapitel widmet sich zunächst einer Konkretisierung und Klassifizierung von organisationstheoretischen sowie (noch grundlegenderen) systemtheoretischen Konzepten und Mechanismen, die im zweiten Teil der

Arbeit als Grundlage für die Erstellung von konkreten, softwaretechnisch nutzbaren Modellen dienen.

4 Konkretisierung von organisationstheoretischen Konzepten und Mechanismen

Die grundlegenden Argumente für ein Organisationsparadigma der Softwareentwicklung wurden im vorigen Kapitel vorgetragen. Der Weg für die Erstellung von Softwaresystemen und ihrer Bausteine entlang eines solchen Paradigmas ist damit allerdings noch nicht geebnet. Dafür war die qualitative Argumentationsführung zu abstrakt. Es fehlt bislang an einer Identifikation und Konkretisierung der Konzepte und Mechanismen aus dem Bereich sozialer Organisationen, die als Grundlage eines Organisationsparadigmas angesehen werden sollen. In diesem Kapitel wird diese Konkretisierung vorgenommen und damit das konzeptionelle Fundament für den weiteren Verlauf der Arbeit gelegt.

Der Zweck des Kapitels ist, *grundlegende Aussagen* bezüglich ausgewählter Kernaspekte von Organisationen zu identifizieren und dabei insbesondere ihren janusgesichtigen Charakter zu beleuchten. Es geht nicht um eine Bewertung verschiedener konkreter Organisationsformen oder den Vergleich „moderner“ mit „althergebrachten“ Formen von Organisationen. Solche Betrachtungen können sich zwar für die Anwendung eines Organisationsparadigmas als nützlich erweisen, das Fundament selber muss aber vor allem vom *Wesen* des Konzeptes der sozialen Organisation abgeleitet sein. Daher entstammen die in diesem Kapitel herangezogenen Quellen einem recht großen Zeitspektrum.

Die Erläuterungen dieses Kapitels sind an einigen Stellen ausführlicher als es für die Identifizierung von Kerneigenschaften und expliziten Modellelementen des softwaretechnischen Konzeptes einer Organisationseinheit im zweiten Teil der Arbeit notwendig wäre. Allerdings geht es in diesem Kapitel auch mehr darum, grundlegende Fragestellungen, Eigenschaften und Möglichkeiten aufzuzeigen, die mit dem Konzept der sozialen Organisation assoziiert sind und die für den Fall eines Organisationsparadigmas der Softwareentwicklung zumindest potentiell auf die Betrachtung von Strukturen und Abläufen in Softwaresystemen übertragen werden.

4.1 Generelle Vorüberlegungen und Begriffsklärung

Im zweiten Teil der Arbeit werden *Multiorganisationssysteme* als konkrete Form von organisationsorientierten Softwaresystemen eingeführt. Dabei werden Modelle erstellt, die ausgehend von eher abstrakt-analytischen „Denkmodellen“ bis hin zu technischen, ausführbaren „Programmiermodellen“ den Brückenschlag von sozialen und organisationstheoretischen Konzepten zur Informatik darstellen. Angesichts der Charakterisierung der hier adressierten Softwaresysteme als Systeme von Systemen geht es dabei ausdrücklich nicht um die Modellierung *einer* Organisation im Sinne eines Systems, das genau *eine* Ebene der Organisation von Systembestandteilen aufweist. Stattdessen werden Multiorganisationssysteme im zweiten Teil der Arbeit als Weiterführung von Multiagentensystemen beschrieben. Als neuartige Entität wird dabei die *Organisationseinheit* eingeführt. Organisationseinheiten sind janusgesichtige Entitä-

ten. Sie stellen gleichzeitig einen Organisationskontext für weitere Organisationseinheiten dar und treten selber als Organisationsakteure im Kontext umschließender Organisationseinheiten auf. Auf diese Weise ergibt sich eine rekursive Struktur von Multiorganisationssystemen.[1]

Die Ausführungen des vorangehenden Kapitels haben deutlich gemacht, dass die Organisationstheorie die dafür benötigten Konzepte und Mechanismen bietet. Spezielles Gewicht wurde zum Ende des vorigen Kapitels auf die Perspektive auf Organisationen als offene Systeme und kollektive Akteure gelegt. Auch das Konzept der sozialen Organisation ist also mit einem janusgesichtigen Charakter assoziiert. Organisationen sind Kontexte für Akteure und kollektive Akteure in einem. Dementsprechend können Organisationen als Akteure im Kontext anderer Organisationen auftreten und selber einen Kontext für weitere Organisationen bereitstellen. Nach Ansicht des Autors der vorliegenden Arbeit ist diese rekursive Sichtweise allerdings mit einem terminologischen Problem (oder eher einem „Unwohlsein“ hinsichtlich der Terminologie) verknüpft. Der Begriff der Organisation ist zumindest traditionell mit Entitäten der Granularitätsstufe eines betrieblichen Unternehmens oder einer Firma assoziiert. Kollektivitäten unterhalb und oberhalb dieser Granularitätsstufe erhalten andere Bezeichnungen: Projektteams, Abteilungen, Gremien, Divisionen, strategische Allianzen, virtuelle Organisationen, Organisationsverbünde, Märkte, Industrien, organisatorische Populationen, organisatorische Felder und (inter-)gesellschaftliche Institutionen. Nichtsdestotrotz handelt es sich um Kollektivitäten, für die die im vorigen Kapitel als kennzeichnend für Organisationen genannten Kriterien gelten, nämlich *Formalisierung* und *Zielspezifität*.[2]

Um etwaige Missverständnisse zu vermeiden, wird die gemeinsame Verwendung des Begriffes der Organisation für diese Kollektivitätstypen in der Folge der Arbeit vermieden. Stattdessen verwendet der Autor den Begriff *organisierte Kollektivität* als gemeinsamen Oberbegriff. Organisierte Kollektivitäten und aus ihnen zusammengesetzte Organisationssysteme stehen im zweiten Teil der Arbeit Pate für die Modellierung von Organisationseinheiten und Multiorganisationssysteme. Natürlich bestehen durchaus beträchtliche Unterschiede zwischen verschiedenen Typen von organisierten Kollektivitäten und diese Diversität kann man sich zu nutze machen. So liefert die Organisationstheorie bereits Richtlinien dafür, welche Konfigurationen verschiedener organisierter Kollektivitäten sich in realweltlichen Organisationssystemen durchgesetzt und als leistungsfähig erwiesen haben. Der dritte Teil der vorliegenden Arbeit (insbesondere Kapitel 11) widmet sich der Aufgabe, diese Richtlinien auf ihre Eignung für die Softwareentwicklung zu prüfen und in eine Referenzarchitektur für Multiorganisationssysteme zu überführen. Zuvor gilt es jedoch, die Grundlage dafür zu schaffen.

Bevor eine Differenzierung verschiedener Organisationseinheiten vorgenommen werden kann, werden im zweiten Teil der Arbeit universelle Grundmodelle von Organisationseinheiten erarbeitet. Für diese universellen Modelle werden im Folgenden des Kapitels die organisationstheoretischen Grundlagen konkretisiert. So haben organisationstheoretische Untersuchungen von Aspekten wie sozialer Struktur, Zielen/Strategien, Grenzdefinition und -management, Steuerung, Kontrolle und Macht unter einer offenen Systemperspektive Relevanz auf allen Systemebenen für alle auf den jeweiligen Ebenen angesiedelten organisierten Kollektivitäten. Im Folgenden werden die generischen Charakteristika solcher Aspekte zusammengetragen. Der

[1] Die Verwendung gerade der beiden Begriffe *Multiorganisationssystem* und *Organisationseinheit* wird im zweiten Teil der Arbeit ausführlicher motiviert.

[2] Diese beiden Indikatoren wurden im Zusammenhang mit der rationalen Organisationsperspektive genannt. Allerdings wurde zugleich beschrieben, dass auch die natürliche Organisationsperspektive die *Existenz* von Formalisierung und Zielen anerkennt, ihre hohe Relevanz hinsichtlich des tatsächlichen Verhaltens der organisatorischen Mitglieder jedoch anzweifelt.

Autor der vorliegenden Arbeit ist der Ansicht, die in diesem Kapitel zitierten organisationstheoretischen Aussagen und Ergebnisse seien derart gewählt, dass sie universelle Bedeutung für organisierte Kollektivitäten hätten. Dies gilt auch für Literaturstellen, die sich originär auf Organisationen im klassischen Sinne von Unternehmen beziehen und demzufolge auch der Begriff der Organisation verwendet wird. Auf eine wiederholte Relativierung der Begriffe Organisation und organisierte Kollektivität wird im weiteren Verlauf des Kapitels verzichtet. Im Kontext der Arbeit wird sie hingegen punktuell wieder aufgegriffen werden.

Es bietet sich an, neben der Organisationstheorie auch Ergebnisse der *allgemeinen Systemtheorie* heranzuziehen, zumal diese ohnehin als Grundlage für viele organisationstheoretische Betrachtungen herhält. Die grundlegenden Aussagen betreffen dabei das Spektrum und das Wesen des Konzeptes System an sich. Ludwig von Bertalanffy [Ber56] gilt als Begründer der allgemeinen Systemtheorie. Die allgemeine Systemtheorie war zu jener Zeit als Reaktion auf eine zunehmende Abschottung wissenschaftlicher Disziplinen voneinander zu verstehen. Von Bertalanffy und seine Kollegen zeigten sich besorgt darüber, dass Physiker, Biologen, Psychologen und Soziologen sich jeweils in eigenen Universen bewegen und somit einen interdisziplinären Austausch verhindern würden. Dabei haben bestimmte Kernkonzepte Relevanz über ein breites Spektrum von Disziplinen hinweg. Insbesondere fallen viele der wichtigsten von Wissenschaftlern studierten Entitäten unter die gemeinsame Rubrik des Systems. Mit der universellen Herangehensweise sowie dem Bestreben, Gemeinsamkeiten zwischen Systemen zu identifizieren, eignet sich die allgemeine Systemtheorie als zusätzliche Inspirationsquelle für das Vorhaben der vorliegenden Arbeit.

Die beiden folgenden Abschnitte des Kapitels werden genutzt, um das Konzept organisierter, agierender Kollektivitäten von verschiedenen Seiten aus zu betrachten. Ein besonderes Augenmerk liegt darauf, bei der Betrachtung kollektiver Akteure davon abzusehen, der Kollektivität anthropomorphe Eigenschaften zuzusprechen. Eine Kollektivität ist immer ein System, das durch seine Teile konstituiert wird. Eine Kollektivität mag von außen als holistisch und „als Ganzes“ handelnd wahrgenommen werden und diese Perspektive ist in vielen Fällen ein adäquates und hilfreiches Abstraktionsmittel. Nichtsdestotrotz sind die Handlungen einer Kollektivität auf der Analyseebene der Kollektivität selber durch die Handlungen ihrer Mitglieder verkörpert. Dies gilt gegebenenfalls über mehrere Einbettungsebenen zwischen Kollektivitäten hinweg bis hin zu individuellen Akteuren. Ähnliche Überlegungen gilt es bezüglich der Ziele und Strategien einer Kollektivität anzustellen. Diese entspringen nicht der Existenz eines metaphysischen korporativen Verstandes der Kollektivität. Andererseits sind sie auch nicht lediglich die Summe der Ziele der Mitglieder. Die Wahrheit liegt dazwischen.

Im Folgenden steht also das Verhältnis der Interna einer Kollektivität zu ihrem kollektiven Akteurswesen nach außen hin im Blickpunkt. Im folgenden Abschnitt 4.2 werden vor allem organisationstheoretische Studien offener Systeme herangezogen. Es geht um das Zusammenspiel eines offenen Systems mit seiner Umgebung, welchen Anforderungen und Herausforderungen sich das offene System dabei gegenüber sieht und welche Implikationen sich daraus für die Umgebung ergeben. Auch wenn es dabei bereits vereinzelt um die Aspekte Macht, Ziele, Steuerung und Kontrolle geht, werden diese Punkte gesondert in Abschnitt 4.3 behandelt.

4.2 Offene Systeme

Die Systemeigenschaft der Offenheit zur Umgebung spielt im Kontext der in dieser Arbeit betrachteten Systeme von Systemen eine tragende Rolle. Studien zur Offenheit von Systemen

spielen in vielen Forschungsfeldern eine Rolle, die hier zitierten Studien stammen hauptsächlich aus der Organisationstheorie im Speziellen und teilweise aus der Soziologie im Allgemeinen.

Offenheit ist eine Grundeigenschaft vieler Systeme. So präsentiert Boulding [Bou56, S. 200–207] eine neunstufige Systemtopologie, bei der die Systeme jeder Stufe wesentliche Eigenschaften der Systeme auf den tieferen Stufen inkorporieren. Während „symbolverarbeitende Systeme“ (beispielsweise Menschen) und soziale Systeme erst auf der siebten beziehungsweise achten Stufe eingeführt werden, werden offene Systeme bereits auf der vierten Stufe angesiedelt. Als charakteristische Eigenschaft (vor allem für hinreichend komplexe offene Systeme) gilt die Fähigkeit zur Selbsterhaltung auf Basis des Durchsatzes von Ressourcen aus der Umwelt. Im Falle sozialer Systeme handelt es sich bei diesen Ressourcen verstärkt um Informationen und Wissen anstatt Materialien oder Energie.

Pondy und Mitroff [PM79] argumentieren, es sei *gerade* der Durchsatz von Ungleichförmigkeit aus der Umwelt, der die differentielle Struktur offener Systeme bewahre. In der Folge werden die wichtigsten Aspekte der Betrachtung offener Systeme und speziell der Betrachtung von Organisationen als offenen Systemen behandelt.

4.2.1 Grenzen und Grenzziehung

Offene Systeme *haben* Grenzen und verwenden Energien darauf, sie zu erhalten. So verwendet Scott [Sco03, S. 123+185] eine anschauliche metaphorische Charakterisierung von Organisationen als offenen Systemen. Organisationen sind weder *Festungen* – undurchdringbar gegenüber etwaigen Einflüssen ihrer Umwelt – noch *Windkanäle* – völlig offen und responsiv gegenüber jeglichen äußeren Unruhen. Stattdessen gleicht eine Organisation eher einem *Sieb* – erwünschte Durchsätze zulassend und zugleich unzweckmäßige oder gar schädliche Elemente ausschließend. Organisationen konstruieren und rekonstruieren Grenzen, über die hinweg sie sich mit der Außenwelt in Beziehung setzen. Dabei gibt es in der Regel nicht nur eine Grenze, sondern mehrere und sie werden höher und undurchdringlicher je näher man dem Teil der Organisation kommt, in dem die Kernaufgaben ausgeführt werden.

Allerdings stellt sich an dieser Stelle die Frage, wie Grenzen offener sozialer Systeme überhaupt spezifiziert werden. Laumann et. al. [LMP83] zeigen in diesem Zusammenhang verschiedene Herangehensweisen auf. Gerade die Offenheit eines Systems macht die Grenzdefinition schwierig und scheinbar willkürlich. Laumann und seine Kollegen heben im Wesentlichen drei Kriterien für die Grenzdefinition hervor: *Akteure*, *soziale Relationen* und *Aktivitäten*.[3]

So ist die vielleicht naheliegendste Option, Grenzen über die Akteure einer Kollektivität zu definieren, beispielsweise darüber, welche Akteure Mitglieder der Kollektivität sind und welche nicht. Allerdings wurde bereits in Abschnitt 3.2 auf den Umstand hingewiesen, dass Organisationen als „moderne“ Kollektivitäten heutiger Gesellschaften ihre Mitglieder eben gerade nicht vollständig umschließen, sondern stattdessen lediglich einen bestimmten Teil ihrer

[3] An dieser Stelle stellt sich gegebenenfalls die Frage, warum *physikalische* Grenzkriterien nicht betrachtet werden. Schließlich können Organisationen der realen Welt in vielen Fällen tatsächlich „gesehen“ werden. Scott [Sco03, S. 188] argumentiert in diesem Fall, dass die Kriterien der sozialen Relationen und Aktivitäten mit ihren Fokus auf Verhalten physikalische Grenzen implizieren. Verhalten benötigt Raum und Zeit und verweist daher auf die räumlichen Grenzen einer Organisation (Zäune, Mauern, Eingänge) und außerdem auf ihre zeitlichen Grenzen (Arbeitszeiten, Stundenpläne). Allerdings verweist Scott gleichzeitig darauf, dass dies eine eher traditionelle Sichtweise widerspiegelt. In vielen modernen Organisationsformen, die vor allem eine offene Netzwerkperspektive als Grundlage haben, existieren gegebenenfalls keine räumlichen und/oder zeitlichen Grenzen mehr. Zumindest aber markieren sie vielfach nicht länger die Grenzen organisatorischen Verhaltens.

Verhaltensweisen und Aktivitäten. Eine akteurszentrierte Grenzdefinition muss daher eher in Form gemeinsamer Akteursattribute wie beispielsweise Interessen, Zielen, Training (im Falle individueller Akteure) oder Struktur und Strategien (im Falle kollektiver Akteure) geschehen.

Eine alternative Herangehensweise besteht darin, statt Akteursattributen soziale Relationen in den Mittelpunkt der Grenzdefinition zu stellen. Ein weit verbreiteter Indikator von Verbundenheit ist die Frequenz von Interaktionen. In der Regel ist keine Kollektivität nach dieser Auffassung vollständig isoliert. Allerdings können „dünne Stellen“ in einem Interaktionsnetzwerk als Grundlage der Grenzdefinition herhalten. Zusätzlich zur Frequenz als Indikator können gegebenenfalls weitere, unterstützende Indikatoren herangezogen werden. Beispiele sind die Bedeutung von Interaktionen (und damit die Stärke sozialer Beziehungen) oder die Positions-/Rollenassoziation von Interaktionen.

Als drittes Kriterium der Grenzdefinition können Aktivitäten herangezogen werden. Dieses Kriterium wird beispielsweise von Pfeffer und Salancik [PS78, S. 31+32] favorisiert. Sie plädieren generell für die Auffassung, es seien Aktivitäten anstatt Akteuren, die bei der Erschaffung von Organisationen strukturiert werden. Natürlich bedarf es Akteuren, um Aktivitäten auszuführen. Allerdings ist es üblicherweise möglich, Akteure zu ersetzen, während die Aktivitäten erhalten bleiben. Auf diese Weise ergibt sich auch die Möglichkeit, die Mitgliedschaft von Akteuren zu charakterisieren: Der Grad des Einbezugs eines Akteurs in die Verhaltensstruktur einer Kollektivität kann als der Anteil seines Verhaltens zur Mitwirkung an dieser speziellen Kollektivität geteilt durch sein gesamtes Verhalten zur Mitwirkung an allen Kollektivitäten definiert werden. Demnach ist eine Organisation die Gesamtmenge an interstrukturierten Aktivitäten und die Grenze verläuft dort, wo andere Organisationen ein höheres Maß an Kontrolle über Aktivitäten haben als die gerade betrachtete Organisation.[4]

Zusätzlich zu diesen drei Kriterien der Grenzdefinition Laumann et. al. [LMP83] hebt Scott [Sco03, S, 188] ein viertes Kriterium hervor. Er betont die Grenzen, die „in den Köpfen“ der Mitglieder einer Kollektivität existieren. Damit werden die zentralen, dauerhaften und charakteristischen Überzeugungen und Werte bezeichnet, die von Mitgliedern geteilt werden und die eine Kollektivität von anderen Kollektivitäten abhebt. Scott wählt die Bezeichnungen des *kulturell-kognitiven Einvernehmens* beziehungsweise der *organisatorischen Identität* (übernommen von Albert und Whetten [AW85]).

Akteure/Mitgliedschaft, soziale Relationen, Aktivitäten und kulturell-kognitives Einvernehmen als Kriterien der Grenzdefinition sind natürlich nicht unabhängig voneinander, doch Scott [Sco03, S. 188] weist auf die Problematik hin, dass sie häufig nicht koinzidieren. Für verschiedene Kollektivitäten sind auch verschiedene Kriterien mehr oder weniger bestimmend.

4.2.2 Charakteristika von Umgebungen

Erst Grenzen lassen das Konzept einer Umwelt aus Sicht eines offenen Systems überhaupt zu. Zwischen einer Kollektivität und ihrer Umwelt besteht in der Folge eine Wechselwirkung. Die Organisationstheorie bietet einen reichhaltigen Fundus an Studien zur Beziehung von Organisationen zu ihrer Umwelt. Dabei ist in der Regel nicht von einer, sondern von mehreren

[4] In dem hier zitierten Buch von Pfeffer und Salancik [PS78] geht es vor allem um externe Einflüsse, denen Organisationen ausgesetzt sind. Die Befugnisse/Entscheidungsfreiheit von Organisationen, Mitgliederverhalten über Aktivitäten zu steuern, unterliegt in der Regel vielfältigen externen Einschränkungen. Als Beispiel führen Pfeffer und Salancik unter anderem General Motors an (S. 32). Deren Entscheidungsfreiheit bezüglich Aktivitäten der Automobilentwicklung wird erheblich durch Umweltorganisationen, das Verkehrsministerium und verschiedene Kundengruppen eingeschränkt.

Umgebungen (beziehungsweise Domänen) einer Organisation die Rede. Diese Umgebungen können sich überlagern und wechselseitigen Abhängigkeiten unterworfen sein. Analytisch ist es von Vorteil, verschiedene Umgebungen mit verschiedenen Charakteristika zu unterscheiden. So können für eine bestimmte Organisation beispielsweise die Umgebungen in Form eines organisatorischen Feldes und in Form einer strategischen Allianz mit anderen Organisationen unterschieden werden, selbst wenn die strategische Allianz (teilweise) selber im Kontext des Feldes operiert. In der Folge wird der Begriff der Umwelt für die Gesamtheit der differenzierten Umgebungen einer Kollektivität verwendet.

Für die Betrachtung konkreter Umgebungen schlägt Scott [Sco03, S. 133–141] eine analytische Unterscheidung zweier grundlegender Facetten vor: *Material-Ressourcen-Eigenschaften* und *institutionelle Eigenschaften*. Material-Ressourcen-Eigenschaften betonen die Perspektive auf Organisationen als Produktionssysteme. Die Ziele einer Organisation stehen in engem Zusammenhang mit der Verrichtung bestimmter Arten von Tätigkeiten und diese ergeben sich aus der Transformation von Eingaben in Ausgaben. Entsprechend der offenen Systemperspektive ist dabei keine Organisation vollständig selbstgenügsam. Stattdessen müssen Anbieter benötigter Ressourcen auf der einen und Abnehmer produzierter Ressourcen auf der anderen Seite gesichert werden. Die Umgebung wird dementsprechend als *Aufgabenumgebung*, als ein Bestand von Ressourcen (Rohstoffe, Produkte, Dienstleistungen sowie Informationen) aufgefasst. Dem gegenüber stehen die institutionellen Eigenschaften einer Umgebung. Scott [Sco08, S. 50–95] beschreibt eine *Institution* als charakterisiert durch *regulative*, *normative* und *kulturell-kognitive* Elemente, die das Verhalten der Institutionsmitglieder leiten und die in Kombination mit Aktivitäten und Ressourcen zu ihrer Durchsetzung Stabilität und Bedeutung in sozialen Systemen herbeiführen. Die meisten organisatorischen Umgebungen stellen sich als eine Kombination aus Aufgabenumgebung und Institution dar. Insbesondere gilt, dass alle Aufgabenumgebungen auf institutionellen Fundamenten fußen.

Das Verhältnis von Organisationen und Institutionen wird in organisationstheoretischen Studien häufig aufgegriffen. Scott [Sco08, S. 150+151] unterscheidet drei wiederkehrende, zumindest analytisch unterscheidbare Perspektiven. (1) Organisationen agieren im Kontext von Institutionen. (2) Organisationen sind selber Institutionen. (3) Institutionelle Prozesse durchdringen die gesamte Gesellschaft und Organisationen sind dementsprechend institutionalisiert beziehungsweise eine institutionelle Form innerhalb der modernen Gesellschaft. Zusammenfassend zeigt sich wieder der janusgesichtige Charakter von Organisationen. Sie sind eingebettet in Umgebungen, die sich als Aufgabenumgebung und Institution darstellen, spannen beide Aspekte aber auch wiederum für ihre internen Anteile auf.

4.2.3 Einflussnahme von Seiten der Umwelt

In Hinsicht auf die angedeutete Wechselwirkung zwischen Organisation und Umwelt gilt zunächst, dass Umwelteinflüsse einen erheblichen Einfluss auf Struktur, Performanz sowie Resultate einer Organisation ausüben. Speziell im Kontext der Strömung der *contingency theory* innerhalb der Organisationstheorie wird betont, Organisationen müssten bewusste Schritte unternehmen, sich ihren Umgebungen (Aufgabenumgebungen sowie Institutionen) anzupassen. Galbraith [Gal73, S. 2] formuliert die beiden wesentlichen Grundannahmen der *contingency theory*: (1) *„There is no one best way to organize“* und (2) *„Any way of organizing is not equally effective“*. Zusammen führen diese beiden Aussagen zum Schluss, der beste Weg des Organisierens hänge von Kontext der jeweiligen Organisation ab. Nach Pfeffers [Pfe82, S. 148] „Übereinstimmungshypothese“ (*consonance hypothesis*) gilt: *„[...] organizations that*

have structures that more closely match the requirements of the context are more effective than those that do not“.

Der Kontext wiederum ist dominiert von der Umwelt der Organisation. Scott [Sco03, S. 97] bietet zwei Anhaltspunkte dafür, wie Anpassung einer Organisation an Umweltanforderungen stattfinden kann. (1) Organisationen bilden spezialisierte Untereinheiten, um ganz bestimmten Anforderungen der Umwelt (beispielsweise speziellen Umgebungen) zu begegnen. (2) Die Differenzierung und Integration der Organisation als Ganzem ist der Gesamtkomplexität der Umwelt angepasst, in der die Organisation operiert. Anpassung heißt dabei allerdings auch, den Anforderungen nicht nur *mindestens*, sondern möglichst *genau* gerecht zu werden. In diesem Zusammenhang formulieren Pondy und Mitroff [PM79, S. 7] das *Gesetz der eingeschränkten Vielfalt* (*law of limited variety*) für offene Systeme: *„A system will exhibit no more variety than the variety to which it has been exposed in its environment“.*

Allerdings bedeutet die Anpassung von Organisationen an ihre Umwelt ein Problem in sich. Je vielfältiger die Umweltanforderungen sind, desto differenzierter und komplexer muss die Organisationsstruktur sein. Für eine sehr komplexe und differenzierte Organisationsstruktur gilt jedoch, dass die Koordination von Aktivitäten zwischen den einzelnen Untereinheiten sehr aufwändig ist. Außerdem sind solche Strukturen schwierig anzupassen wenn sich Umweltanforderungen wieder ändern.[5] Dementsprechend gibt es verschiedene Ansätze, Anpassung und Wandel einer Organisation zu kanalisieren, beziehungsweise bestimmte Bereiche oder Aspekte zu schaffen, die möglichst persistent und stabil bleiben. Scott [Sco03, 197–220] beschreibt, wie Organisationen ihren Umgebungen mit *Pufferungsmechanismen* einerseits und *Brückenschlagsmechanismen* anderseits begegnen. Beides gilt sowohl für den Aspekt einer Umgebung als Aufgabenumgebung als auch für den Aspekt der Umgebung als Institution.

Im Falle der Aufgabenumgebung sorgen Pufferungsmechanismen dafür, den Teil der Organisation, in dem ihre zentralen Aufgaben ausgeführt werden (*technischer/operationaler Kern*), vor etwaigen Turbulenzen der Umwelt abzuschirmen. Zu diesem Zweck werden gegebenenfalls die strukturellen Arrangements um den operationalen Kern herum verändert. Brückenschlagsmechanismen hingegen dienen der Erhaltung der gesamten Organisation. Dabei geht es vor allem darum, die Machtposition der Organisation in Relation zu den externen Kollektivitäten, mit denen ein Ressourcenaustausch besteht, zu festigen beziehungsweise auszubauen.

Im Falle des Institutionscharakters von Umgebungen geht es für eine Organisation vor allem darum, eine Balance zwischen Konformität mit institutionellen Vorgaben und kennzeichnender Eigenständigkeit zu finden. Pufferungsmechanismen sorgen beispielsweise dafür, die Formalstruktur der Organisation von ihrer tatsächlichen Verhaltensstruktur zu entkoppeln. Dies erlaubt, strukturelle Elemente konform zu institutionellen Konventionen zu inkorporieren und dabei gleichzeitig die eigene Verhaltensautonomie zu wahren. Damit ist auch gleichzeitig ein wichtiger Mechanismus des Brückenschlags angedeutet, nämlich Isomorphie. DiMaggio und Powell [DP83] bezeichnen Isomorphie als den *master bridging process* in institutionellen Umgebungen. Dabei geht es um das Inkorporieren institutioneller Vorgaben in die eigenen Strukturen. Dies fördert die Anerkennung der Organisation im Kontext der Institution. Organisationen in einer gemeinsamen institutionellen Umgebung werden einander strukturell

[5] Folgerichtig gibt es auch Organisationstheorien, bei denen die Grundannahme der *contingency theory*, Organisationen würden und müssten bewusste Schritte der Anpassung unternehmen, hinterfragt wird. Hannan und Freeman [HF77, S. 931–932] identifizieren eine große Anzahl an Beschränkungen für Anpassung in Organisationen wie beispielsweise finanzielle Zwänge, spezialisierte Ausbildung des Personals oder geltende Verträge mit externen Kollektivitäten. Solche Beschränkungen sorgen nach Hannan und Freeman in der Summe für eine *strukturelle Trägheit* besonders in großen Organisationen.

ähnlicher (da Organisationen allerdings üblicherweise in mehreren institutionellen Umgebungen operieren, bleibt ihre Struktur letztlich doch unterschiedlich). Die Ursachen, die Isomorphie in institutionellen Umgebungen leiten, lassen sich grob in drei Kategorien unterteilen: Zwang (Institutionen haben üblicherweise Governance-Strukturen, über die bestimmte Vorgaben durchgesetzt werden), Normen (Organisationen übernehmen bestimmte Strukturen oder Verhaltensweisen weil diese gemeinhin als richtig beziehungsweise erfolgsversprechend gehalten werden) und Nachahmung (Organisationen übernehmen/kopieren Struktur oder Verhalten anderer Organisationen, teilweise aus Unsicherheit).

4.2.4 Einflussnahme auf die Umwelt

Von der Wechselwirkung zwischen Kollektivität und Umwelt wurde bislang die Richtung der Einflussnahme von Seiten der Umwelt betont. Dazu passt die weiter oben zitierte Metapher, eine Organisation gleiche einem Sieb, Einflüsse aus der Umwelt filternd. Allerdings gilt auch die Gegenrichtung der Einflussnahme. Organisationen wählen ihre Umgebungen und modifizieren diese in der Folge. Ein wichtiger Schritt hin zu diesem Verständnis ist, die Umgebung einer Organisation nicht lediglich anhand ihrer *tatsächlichen/gegenständlichen* Aspekte zu bewerten, sondern vor allem anhand der Aspekte, die von der Organisation *wahrgenommen* werden (*objective/archival measures of the environment* im Gegensatz zum *perceptual measures* nach Boyd et. al. [BDR93]). Die Betonung der wahrgenommenen Aspekte spielt eine wichtige Rolle, denn schließlich kann nur Wahrgenommenes in die Entscheidungsfindung einfließen. Diese Sichtweise spiegelt sich offensichtlich bereits in den weiter oben beschriebenen Pufferungs- und vor allem den Brückenschlagsmechanismen wieder.

Das Konzept einer wahrgenommenen Umgebung ist allerdings noch zu schwach. So geht Weick [Wei79, Kapitel 6] einen Schritt weiter und führt das Konzept des *enactment* beziehungsweise des *enacted environment* ein. Als deutsche Übersetzung verwendet der Autor der vorliegenden Arbeit den Begriff der *Inbetriebnahme*. Weick unterstreicht, das Konzept einer in Betrieb genommenen Umgebung sei nicht synonym mit dem einer wahrgenommenen Umgebung. Das Konzept der Inbetriebnahme ist so zu verstehen, dass Organisationen die tatsächlichen/gegenständlichen Aspekte ihrer Umgebungen konstruieren, umordnen, aussondern oder auch beschädigen. Der Prozess der Inbetriebnahme ist einer, im Zuge dessen das Subjekt mit seinem Objekt interagiert und es teilweise konstituiert. Dabei können durchaus unvorhergesehene und unerwünschte Effekte auftreten.

In diesem Kontext betont Scott [Sco03, S. 142+143] das Zusammenspiel von Informationssystem, Zuwendungsstruktur (*attention structure*) und Inbetriebnahmeprozessen einer Organisation. Beim Informationssystem handelt es sich um Arrangements (spezialisierte Untereinheiten sowie Routinen) für das Sammeln und Verarbeiten von Informationen. Aus dem regelmäßigen Sammeln von Informationen und der Tatsache, dass einige dieser Informationen häufiger gesammelt und gründlicher aufbereitet oder analysiert werden müssen, ergibt sich die Zuwendungsstruktur. Gegebenenfalls ist es schlicht die Verfügbarkeit, die bestimmte Informationen besonders relevant macht. March [Mar94, S. 10] weist darauf hin, dass die Ressourcen (insbesondere Zeit) einer Organisation, sich verschiedenen Aspekten zuzuwenden, begrenzt sind. Zu viele Dinge sind relevant für die Entscheidungsfindung. Aufgrund dieser Limitationen können Theorien der organisatorischen Entscheidungsfindung häufig eher als Theorien der Zuwendung denn als Theorien der rationalen Auswahl beschrieben werden. Die wesentliche Frage ist, nach welchen Mechanismen die knappen Zuwendungsressourcen einer Organisation verteilt werden. Erst anhand der Zuwendungsstruktur können Inbetriebnahmeaktivitäten einer

Organisation untersucht werden.

Abschließend zu diesem Abschnitt sei allerdings noch explizit darauf hingewiesen, dass eine Konzeptionalisierung der Umgebung allein über Wahrnehmung und Inbetriebnahme nicht ausreicht. Die gegenständlichen/tatsächlichen Aspekte einer Umgebung dürfen nicht vernachlässigt werden. Pfeffer und Salancik [PS78, S. 62–63] stellen fest, dass Umgebungscharakteristika die Wirkung von Organisationen auf die Umgebung beeinflussen, unabhängig davon, ob dies von der Organisation wahrgenommen wird oder nicht.[6] Eine Organisation produziert *Ausgaben* – Waren und Dienste, über deren Beschaffenheit die Organisation ein sehr hohes Maß an Kontrolle hat. Die damit assoziierte *Wirkung* ist allerdings das gemeinsame Produkt aus diesen Ausgaben und der Reaktion der Umgebung darauf. Außerdem haben die tatsächlichen Aspekte einer Umgebung natürlich einen unmittelbaren Einfluss auf Zuwendungsstruktur, Informationssystem und Domänenwahl einer Organisation.

4.2.5 Hierarchie

Es ist nicht ausreichend, die Umwelt einer Organisation schlicht als „alles andere“ aufzufassen. Die Beschreibung einer Organisation muss eine explizite Referenz ihres Kontextes beinhalten da dieser die Charakteristika des betrachteten Systems wesentlich bestimmt. Bezüglich der Beschaffenheit der Umwelt wurden bereits konkrete Aussagen getroffen. So wurde die Umwelt als bestehend aus mehreren konzeptionellen Umgebungen beschrieben. Für jede Umgebung kann eine Differenzierung von Charakteristika nach Aufgabenumgebung und Institution vorgenommen werden. Zudem war wiederholt von anderen Systemen/Kollektivitäten in den Umgebungen die Rede.

Speziell der letzte Punkt soll an dieser Stelle hervorgehoben werden. Dazu passen Aussagen Luhmanns [Luh84, S. 243], der sich nicht nur auf Organisationen, sondern auf soziale Systeme im Allgemeinen bezieht. In der Umwelt eines fokussierten Systems existieren weitere Systeme, die das fokussierte System selbst wiederum als Teil ihrer Umwelt betrachten. Alles, was vorkommt, ist immer zugleich zugehörig zu einem System (oder mehreren Systemen) und zugehörig zur Umwelt anderer Systeme. Parsons unterstreicht die zusätzliche Systematik, die sich für das Studium des Verhältnisses zwischen offenen Systemen und Umgebungen ergibt, wenn die Umgebung selber als System verstanden wird. So beschreibt Parsons [Par60, S. 19], die Ziele einer Organisation würden aus Sicht der umschließenden Gesellschaft häufig als differenzierte beziehungsweise spezialisierte Funktionen aufgefasst werden. Dadurch ergibt sich eine funktionale Kopplung, die den Platz oder die Rolle des Teilsystems innerhalb des Wertesystems des umschließenden Systems legitimiert und unter anderem die Anerkennung und Ressourcenberechtigung des Teilsystems zur Folge haben kann.

[6] An dieser Stelle ergibt sich ein Bezug zur vorherigen Fußnote. Gerade Organisationstheorien, die das Konzept der bewussten Adaption im Sinne der *contingency theory* hinterfragen, betonen stattdessen die Bedeutung von Zufall als Ursache organisatorischen Wandels. Anstelle von inkrementeller, strategisch gesteuerter Adaption einer Organisation wird der evolutionäre Charakter von Organisationen und ihren Umgebungen hervorgehoben. Gegenüber Adaption bezieht sich *Selektion* als Mechanismus des Wandels darauf, ein gegebenes System schlicht durch ein anderes zu ersetzen. Allerdings betont Scott [Sco03, S. 222], die Debatte, ob Organisationen sich in der Hauptsache durch Adaption oder Selektion verändern, habe sich mit der Zeit gesetzt. Eine strikte Trennung zwischen Wandel, der adaptiv oder selektiv geschieht, ist häufig nicht möglich. Stattdessen vollzieht sich Wandel häufig durch einen dreistufigen Prozess, der beide Aspekte einbezieht: (1) *Variation* – das Aufkommen neuer Elemente, sei es durch Intention oder durch Zufall, (2) *Selektion* – die differenzierte Eliminierung einiger dieser Elemente und schließlich (3) Retention – die Aufrechterhaltung, Verstärkung und Verbreitung der Elemente, die sich als tauglich erwiesen haben.

Diese Ausführungen decken sich mit den Ausführungen aus Kapitel 3 und Abschnitt 4.1 dieses Kapitels bezüglich Organisationen (beziehungsweise allgemeiner, organisierten Kollektivitäten) als janusgesichtigen Entitäten, die sowohl Kontexte (und damit Umgebung) als auch Akteure sind. Die daraus resultierenden hierarchischen Systemstrukturen finden sich auch in anderen Studien offener Systeme wieder. Insbesondere gilt dies für konzeptionelle Modelle der allgemeinen Systemtheorie. So ist beispielsweise Koestlers [Koe67, S. 48] *Holon* eine janusgesichtige Entität mit zwei Identitäten – eine als Supersystem in Relation zu seinen umschlossenen Anteilen und eine als Teilsystem in Relation zu umschließenden Systemen. Im Kontext des *Viable System*-Modells von „(über-)lebensfähigen Systemen“ nach Beer [Bee79] enthält jedes *Viable System* weitere *Viable Systems* und ist selber wiederum enthalten in *Viable Systems*.

Anhand dieser rekursiven Sichtweise wird die Beziehung offener Systeme zu ihren Umgebungen auf eine hierarchische Schachtelung von Systemen zurückgeführt. Es handelt sich nicht notwendigerweise um eine eindeutige/disjunkte Schachtelung. Vielmehr kann es zu Überlappungen und Durchdringungen kommen. Weiterhin ist Hierarchie nicht im Sinne von Status, Macht oder Autorität misszuverstehen. So stellt Simon [Sim62] fest, Hierarchie sei eine fundamentale Eigenschaft *aller* komplexen Systeme, und bezieht sich dabei auf Hierarchie vor allem als einen Clustering-Mechanismus. Die verschiedenen Systeme *bestehen auseinander*. Auf jeder Hierarchieebene sind die Verbindungen und Abhängigkeiten innerhalb einzelner Systemanteile straffer und von höherer Dichte als zwischen Systemanteilen auf der nächst höheren Ebene.

Grundsätzlich kann eine solche Systemhierarchie beliebig tief sein. Soll jedoch ein konkretes System von Systemen entwickelt oder bewertet werden, stellt sich die Frage nach unteren und oberen Systemabschlüssen. Grundsätzlich handelt es sich hier um eine Frage des gewählten Paradigmas. Beispielsweise kann man sich fragen, bis zu welcher Systemebene organisationstheoretische Konzepte und Mechanismen ihre Anwendbarkeit behalten: Markt/Feld, Gesellschaft, Staat, supra-nationale Systeme oder gar noch darüber hinaus. Speziell im dritten Teil der vorliegenden Arbeit wird diese Frage im Hinblick auf ein Organisationsparadigma der Softwareentwicklung aufgegriffen.

4.3 Ziele, Steuerung und Kontrolle, Macht

Nach Swanson [Swa76] handelt es sich um eine der wesentlichen Problemstellungen der Soziologie, Organisationen so aufzufassen und zu studieren, dass ihr Verhalten weder vermenschlicht noch auf die Summe des individuellen Verhaltens ihrer Mitglieder reduziert wird. Die Konzeptionalisierung von Organisationen als kollektive Akteure bedingt ein sorgfältiges Studium der Transition von Mitgliederverhalten zum Verhalten der Kollektivität. Im vorigen Abschnitt über offene Systeme wurden in diesem Zusammenhang bereits die Problemstellungen der Grenzdefinition behandelt. Auch bei der Beschreibung des Widerspiels von Umwelt und Organisation wurde der Charakter von Organisationen als zusammengesetzte Systeme gewahrt. Allerdings wurde dabei teilweise eine *Black Box*-Sicht eingenommen und das Verhalten von Organisationen ohne Referenz auf deren Ursprung beschrieben. So betont die Aussage, Organisationen würden ihren Umgebungen mit Mechanismen der Pufferung und des Brückenschlags begegnen, das Vorhandensein einer bewusst gewählten inneren Struktur der Organisation. Welche Mechanismen gewählt, wie sie Instand gesetzt werden und wie sich daraus in der Folge das kollektive Verhalten der Kollektivität ergibt, bleibt dabei allerdings verborgen. Je nach Abstraktionsebene ist eine solche Sichtweise angemessen oder sogar notwendig.

In diesem Abschnitt hingegen wird das Zustandekommen interner Strukturen einer Organi-

sation explizit behandelt. Eine einfache Veranschaulichung der folgenden Ausführungen ist in Abbildung 4.1 abgebildet. Wie zu Beginn des Kapitels festgehalten, werden hier organisierte

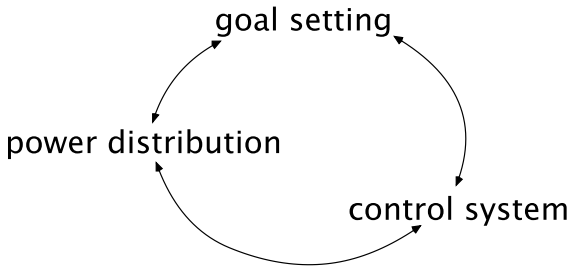

Abbildung 4.1: Zusammenspiel von Machtverteilung, Zielsetzung und Kontrollsystemen

Kollektivitäten behandelt, die sich von anderen Kollektivitäten durch einen höheren Grad an Zielspezifität und Formalisierung abheben. Beide Aspekte sind in der Abbildung präsent. Das Verhalten von Mitgliedern wird in einer Art und Weise gesteuert, dass es dem Zweck beziehungsweise den korporativen Zielen der Organisation zuträglich ist. Zielsetzung zieht demnach die explizite Schaffung eines Steuerungs- und Kontrollsystems[7] nach sich. Mittels der Etablierung solcher Systeme erfolgt eine Formalisierung von Machtstrukturen innerhalb der Organisation. Machtverteilung in Organisationen ist zumindest zum Teil kein Akteursattribut, sondern ein Attribut der Position, die ein Akteur im Kontext der Organisationsstruktur einnimmt. Die Machtverteilung wiederum bestimmt in der Hauptsache, welche Akteure auf welche Weise an der organisatorischen Zielsetzung beteiligt sind. Damit ergibt sich ein Zyklus hinsichtlich der Beziehung zwischen Zielsetzung, Steuerungs-/Kontrollsystem und Machtverteilung. Allerdings wird im Laufe des Abschnitts deutlich werden, dass tatsächlich eine wechselseitige Beeinflussung zwischen allen drei Aspekte untereinander existiert.

4.3.1 Ziele und Strategien

Das Konzept des organisatorischen Ziels gehört zu den am meisten diskutierten Themen im Kontext der Organisationstheorie. Scott [Sco03, S. 292+293] beschreibt, dass Organisationsziele je nach Perspektive unterschiedlichen Zwecken dienen. Am häufigsten wird die *kognitive Funktion* von Zielen betont. Ziele liefern Kriterien für die Generierung von und Auswahl zwischen verschiedenen Verhaltensalternativen. Auf diese Weise leiten Ziele die Entscheidungsfindung von Organisationsmitgliedern. Daneben können Ziele beispielsweise eine *motivierende Funktion* (Ziele dienen als Quelle der Identifikation und Motivation der Mitglieder), eine *symbolische Funktion* (an Akteure außerhalb der Organisation gerichtete Ziele, dienen der Sicherung des Stellenwertes und der Legitimierung der Organisation in ihren Umgebungen) oder eine *rechtfertigende Funktion* (Verhalten geht der Zieldefinition voraus, Ziele werden im Nachhinein definiert, um vorheriges Verhalten mit einer Rechtfertigung zu untermauern.)

[7] Der englische Begriff *control* wird gleichermaßen für die beiden deutschen Begriffe *Steuerung* und *Kontrolle* verwendet. Dementsprechend steht der Begriff *Control System* in Abbildung 4.1 für die Kombination aus Steuerungs- und Kontrollsystem.
Die Texte in den Abbildungen der Arbeit sind durchgängig in englischer Sprache gehalten. In späteren Kapiteln dient dies vor allem, um die Benennung von Modellelementen in den Abbildungen prägnant vom erklärenden Text abzuheben.

Es wird deutlich, dass für eine gegebene Organisation eine Vielzahl unterschiedlicher Ziele existieren kann. Ein Großteil davon existiert nur temporär und hat lediglich einen geringen Einfluss im Gesamtkontext der Organisation. Hinsichtlich der Funktion von Organisationen als kollektive Akteure besteht ein Bedarf daran, von „den Zielen" der Organisation sprechen zu können. Daher wird das Konzept der *Strategie* dem des Ziels vielfach vorgezogen da es stärker mit Dauerhaftigkeit/Langlebigkeit assoziiert ist. Chandler [Cha62, S. 13] definiert Strategie wie folgt: *„Strategy can be defined as the determination of the basic long-term goals and objectives of an enterprise and the adoption of course of action and the allocation of resources necessary for carrying out these goals."*. Der primäre Fokus von Strategien liegt auf externen Belangen und damit darauf, Organisationen und ihre Umgebungen in Bezug zu setzen. Dabei geht es jeweils um (1) die Auswahl einer Domäne/Umgebung und um (2) die Auswahl einer wettbewerbsfähigen Haltung in dieser Domäne. Mintzberg [Min87] schlägt darüber hinaus vor, eine Unterscheidung hinsichtlich intendierten *Strategien* (Pläne), *emergenten* Strategien (ungeplante, aber wiederkehrende Verhaltensmuster) und *umgesetzten* Strategien vorzunehmen. Emergente Strategien sind bereits per Definition auch immer umgesetzt. Intendierte Strategien hingegen können, müssen aber nicht umgesetzt werden.

Der Rest des Teilabschnittes ist der Frage gewidmet, wer für die Bestimmung der intendierten Strategien und damit für den „bewussten" Anteil der Systemsteuerung verantwortlich ist. Dazu gehört auch, emergente Strategien zu intendierten Strategien „umzuwandeln". So kann die oben zitierte Charakterisierung von Strategien nach Mintzberg nach Ansicht des Autors der vorliegenden Arbeit um *anerkannte* Strategien angereichert werden. Emergente Strategien können sich beispielsweise auf niedrigrangigen Ebenen der Organisationshierarchie herausbilden. Erweisen sie sich als leistungsfähig, werden sie gegebenenfalls von Akteuren der mittleren Managementebenen selektiert und unterstützt und schließlich von hochrangigen Entscheidungsträgern anerkannt und legitimiert.

Zur Klärung der Frage, wer in einer Organisation tatsächlich für die Zielsetzung (vor allem der Strategien) verantwortlich ist, gilt es zunächst, die Unterscheidung von Zielen der Mitglieder und von organisatorischen Zielen herauszustreichen. Simon [Sim64] ist einer der Organisationstheoretiker, die diese Unterscheidung besonders betonen und nimmt damit eine rationale Systemperspektive ein. Demgegenüber betont die natürliche Systemperspektive, Mitglieder würden ihre eigenen Interessen auf die Organisation projizieren (vergleiche Kapitel 3). Während Simon diese Sichtweise anerkennt, besteht er dennoch darauf, die obige Unterscheidung aufrecht zu erhalten. Die Anforderungen von Organisationen an ihre Mitglieder sind häufig orthogonal zu den Motiven der Mitglieder. Aus diesem Grund gibt es Mechanismen der Be-/Entlohnung in Organisationen. Letztlich ist das Konzept der organisatorischen Rolle und ihres Einflusses auf das Verhalten des Rolleninhabers eine Manifestation der Tatsache, dass individuelle und organisatorische Ziele nicht notwendigerweise übereinstimmen. Dementsprechend wäre es falsch, Ziele der Mitglieder einer Organisation zu aggregieren, um eine Beschreibung der Organisationsziele zu erhalten. Wie in den Vorüberlegungen des Kapitels argumentiert, wäre es ebenso falsch, die Existenz eines metaphysischen korporativen Verstandes der Organisation zu postulieren. Auf diese Weise würde die Vergegenständlichung der Organisation zu weit gehen. Die Antwort auf die Frage, wer organisatorische Ziele setzt und wie sie gesetzt werden, liegt zwischen den beiden genannten Extremen.

Eine der anerkanntesten Grundlagen für das Studium organisatorischer Zielsetzung stammt von Cyert und March [CM63, S. 27–32]. In jeder Organisation verfolgen verschiedene Akteure gemeinsame Interessen. Schließen sich solche Akteure zu Interessengemeinschaften – in der Folge der Einfachheit halber als Gruppen bezeichnet – zusammen, so entsteht ein gewisses

Machtgefüge zur Stützung dieser Interessen. Die verschiedenen Gruppen einer Organisation sind üblicherweise nicht disjunkt. Jede Gruppe versucht, ihre Präferenzen und Ziele auf die Organisation zu übertragen. Dabei ist typischerweise keine einzelne Gruppe vollständig erfolgreich. Stattdessen suchen sie Alliierte in anderen Gruppen mit zumindest ähnlich gelagerten Interessen. So macht beispielsweise eine Gruppe einer anderen ein Zugeständnis, um deren Kooperation zuzusichern. Ein Zugeständnis einer Gruppe kann immer als Ziel einer anderen Gruppe und umgekehrt verstanden werden. Durch Kompromissbildung entstehen also aus den ursprünglichen Gruppen sukzessive größere und einflussreichere Koalitionen. Schließlich entsteht eine Koalition, die zu einer für alle beteiligten Gruppen akzeptablen Einigung gekommen und zugleich einflussreich genug ist, um als die *dominante Koalition* der Organisation betrachtet werden zu können. Die Interessen aller Gruppen, die dieser Koalition angehören, müssen berücksichtigt werden und tragen somit zu den Zielen der Organisation bei. Daraus folgt, dass Organisationsziele vor allem als (gegebenenfalls recht komplexe) Präferenzaussagen aufgefasst werden können. Um für alle an der dominanten Koalition beteiligten Gruppen akzeptabel zu sein, müssen Organisationsziele einer ganzen Menge an Anforderungen genügen. So beschreibt auch Simon [Sim64], obwohl häufig eine der vielen Anforderungen hervorgehoben und als „das Ziel“ bezeichnet werde, so sei es doch angemessener, sich bei der Charakterisierung des Ziels auf die gesamte Menge an Anforderungen zu beziehen.

Das Konzept der dominanten Koalition hilft bei der Vermeidung vieler Probleme, die sich an anderer Stelle häufig bei der Beschreibung von organisatorischer Zielsetzung ergeben. Scott [Sco03, S. 297–303] fasst die Vorteile zusammen.

- Das Problem der übertriebenen Vergegenständlichung wird vermieden. Ziele der Organisation werden explizit als auf den Zielen der Mitglieder beruhend aufgefasst.
- Es wird anerkannt, dass Mitglieder und Gruppen die Ziele der Organisation bestimmen. Dabei gilt allerdings keine Annahme der Gleichberechtigung. Die Machtverteilung innerhalb der Organisation bestimmt, welche Akteure welchen Einfluss auf die Zielsetzung geltend machen können.
- Nur in wenigen Fällen ist ein einzelnes Mitglied oder eine kleine Gruppe mächtig genug, die Ziele der Organisation alleine zu bestimmen. Stattdessen sind organisatorische Ziele in der Regel verschieden von den Zielen jedes einzelnen Mitglieds.
- Das Beschließen von Kompromissen bei der Koalitionsbildung dient der Abschwächung von Interessenskonflikten. Vollständig aufgelöst werden können Konflikte in der Regel nicht. Der Entstehung von konfliktbehafteten Zielen wird somit Rechnung getragen.
- Die Größe und Zusammensetzung der dominanten Koalition variiert von einer Organisation zur nächsten auf Basis verschiedener Faktoren: Größe der Organisation, Machtverteilung, Einfluss externer Akteure und diverse weitere.
- Neue Interessen können kontinuierlich entstehen. Sie führen zu neuen Interessensgruppen und Koalitionen. Die dominante Koalition ist also zumindest konzeptionell ein dynamisches Gebilde.
- Der Einfluss externer Akteure auf die Organisation kann unmittelbar berücksichtigt werden. Je größer der Einfluss eines externen Akteurs ist, desto einflussreicher sind auch Organisationsmitglieder, die Grenzpositionen für das Management der Beziehungen zu diesem Akteur besetzen (Brückenschlag/Pufferung).

4.3.2 Steuerungs-/Kontrollsysteme und formalisierte Macht

Organisationen etablieren Steuerungs- und Kontrollsysteme, um die Verfolgung ihrer Ziele sicherzustellen. Die Definition von Steuerungs-/Kontrollsystemen macht einen wesentlichen Teil der Formalstruktur einer Organisation aus. In gewisser Weise steuern und kontrollieren *alle* Kollektivitäten ihre Mitglieder. Wie bereits in Abschnitt 4.1 geschildert, kann ohne eine normative und kulturell-kognitive Struktur gar nicht erst von einer Kollektivität gesprochen werden. Für den Fall der Organisation ist dieser Aspekt jedoch ein sehr prominenter als für andere Kollektivitäten. Etzioni [Etz64, S. 95] argumentiert, es sei gerade der künstliche Charakter von Organisationen, ihr spezielles Interesse an hoher Performanz und ihre Tendenz zu deutlich höherer Komplexität gegenüber „natürlichen“ Kollektivitäten, die ein Verlassen auf informelle Formen der Steuerung und Kontrolle unpraktikabel machen. In den meisten Fällen kann nicht einfach davon ausgegangen werden, dass die Mitglieder ihre Pflichten verinnerlichen und ihre Aufgaben freiwillig ausführen. Stattdessen weist jede Organisation zumindest in einem gewissen Rahmen formale Strukturen für die Verteilung von Steuerungs- und Kontrollbefugnissen und die Etablierung konkreter Mechanismen auf.

Der Aspekt der Steuerung und Kontrolle ist eng mit dem Aspekt der Macht verwoben. Schließlich sind es nicht formalisierte Steuerungs-/Kontrollsysteme selbst, die Einfluss ausüben, sondern die Akteure, die Positionen in diesen Systemen einnehmen. Ein Akteur kann einen anderen Akteur besonders dann effektiv steuernd oder kontrollierend beeinflussen, wenn er diesem gegenüber Macht besitzt. Emerson [Eme62] beschreibt, Macht ruhe implizit in der Abhängigkeit eines Akteurs von einem anderen Akteur. Macht eines Akteurs, einen anderen Akteur zu beeinflussen, liegt in der Kontrolle über Dinge, die der zweite Akteur wertschätzt. Diese „Dinge“ variieren abhängig von der betrachteten Beziehung. Es kann sich beispielsweise um Geld, Wissen, Fähigkeiten oder Reputation handeln. Ein wichtiger Aspekt von Emersons Charakterisierung ist, dass er Macht als relational ansieht, also nicht als Attribut eines Akteurs selber, sondern als Attribut der sozialen Beziehung zu einem anderen Akteur.

In informellen Kollektivitäten basiert Macht häufig auf den Charakteristika einzelner Akteure. In Organisationen hingegen ist Macht und Machtverteilung zumindest teilweise das Ergebnis einer bewussten Konstruktion: Das Potential zur Beeinflussung anderer wird Positionen innerhalb der Formalstruktur der Organisation zugeschrieben und ist damit jedem Akteur, der die entsprechende Position innehat, zugänglich. Damit einher geht zunächst die in der rationalen Systemperspektive verwurzelte Annahme, man könne die Machtverteilung einer Organisation derart entwerfen, dass sie die organisatorischen Anforderungen hinsichtlich zielgerichteter Steuerung und Kontrolle der Mitglieder unterstütze. Formalisierte Steuerungs- und Kontrollsysteme dienen dazu, die „persönlichen/individuellen“ Elemente in Machtbeziehungen zumindest teilweise zu eliminieren.

Allerdings kann es keiner Organisation gelingen, alle Machtverteilung gänzlich zu formalisieren. Dieser Umstand wird im Kontext der natürlichen Systemperspektive explizit betont. Schließlich werden Positionen mit Akteuren besetzt, die immer individuelle Charakteristika mitbringen. Die damit zusammenhängenden informellen Strukturen einer Organisation sind Quellen informeller Machtverteilung. Desweiteren sind der Zugang zu Informationen und angeeignete Fähigkeiten wichtige Ressourcen in Organisationen. Deren Auftreten kann nicht vollständig geplant werden. Stellvertretend kann hier Galbraith [Gal02, S. 80+81] zitiert werden. Er argumentiert, die weitverbreiteten Organigramme von Organisationen würden leicht zu der Auffassung verleiten, Macht fließe ausschließlich von oben nach unten in der Organisationshierarchie. Stattdessen gilt, dass jede Art von Entscheidung bestimmte Informationen

benötigt, die verteilt vorliegen. Dementsprechend geht immer ein Teil der Macht von Entscheidungsträgern über an Akteure, die relevante Informationen besitzen.

Ein weiterer Aspekt im Zusammenhang mit Macht und formalisierten Strukturen ist *Legitimität*. Suchmann [Suc95, S. 574] betrachtet verschiedene Definitionen/Charakterisierungen von Legitimität in organisationstheoretischen Studien und leitet eine eigene Definition ab: *„Legitimacy is a generalized perception or assumption that the actions of an entity are desirable, proper, or appropriate within some socially constructed system of norms, values, beliefs, and definitions.“* Nach Scott [Sco03, S. 313–318] bedürfen Organisationen zusätzlich zu Machtbeziehungen vor allem eines Regel/Normen-Systems, das die Verteilung von Macht, ihre Ausübung sowie die Reaktion darauf reguliert. In diesem Zusammenhang charakterisiert Scott *Autorität* als legitimierte Macht und legitimierte Macht als normativ regulierte Macht. Die Art der Legitimation im Rahmen des Normensystems einer Organisation kann unterschiedlich sein. Dabei tritt speziell in Organisationen der Fall der *autorisierten Autorität* auf. In formalen Organisationshierarchien gibt es relativ zum Inhaber einer Machtposition untergeordnete wie auch übergeordnete Akteure (Positionen). Organisationsmitglieder haben die Möglichkeit, die ihnen gegenüber ausgeübte Macht in Frage zu stellen. Eine typische Sicherheitsvorkehrung in Organisationsstrukturen besteht darin, dass sich Organisationsmitglieder für den Fall der Wahrnehmung einer ungerechtfertigten Machtausübung an den Vorgesetzten ihres unmittelbar Vorgesetzten wenden können. Mit autorisierter Autorität bezeichnet Scott genau diese Situation, in der die Macht eines Akteurs von einem ihm vorgesetzten Akteur reguliert wird. Diese Propagierung von Machtlegitimation endet nicht an der Spitze der Organisationshierarchie. Organisationen sind in institutionelle Umgebungen eingebettet, denen gegenüber eine Organisation als Ganzes ihre Legitimation sichern muss.

Zusammenfassend bleibt festzuhalten, dass die Formalstruktur einer Organisation und die darin eingeflochtenen Steuerungs- und Kontrollsysteme zwar nicht dazu dienen können (und sollen), eine allumfassende Kontrolle der Machtverteilung zu erlangen. Nichtsdestotrotz stellt sie das zentrale Instrument der bewussten Steuerung des Organisationsverhaltens entlang der Organisationsziele dar. Der folgende Teilabschnitt widmet sich dem Thema der strukturellen Komplexität einer Organisation genauer.

4.3.3 Strukturelle Komplexität

Nach Mintzberg [Min83, S. 2] ist jedwedes organisierte Verhalten in sozialen Systemen zwei fundamentalen und gegenläufigen Anforderungen unterworfen: (1) der Arbeitsteilung in verschiedene Aufgaben und (2) der Koordination der verteilten Ausführung dieser Aufgaben. Dementsprechend sieht Mintzberg die Struktur einer Organisation zunächst abstrakt als die Gesamtsumme der Mechanismen, die die Arbeitsteilung und die anschließende Koordination vornehmen. Mintzberg geht dabei mit den Ausführungen des vorigen Abschnittes konform, nach denen die Organisationsstruktur zu einem Großteil als Steuerungs- und Kontrollsystem dient. Beispielsweise identifiziert er bezüglich der Koordination fünf grundlegende Mechanismen: (1) Wechselseitige Abstimmung, (2) Beaufsichtigung, (3–5) Standardisierung von Aktivitäten, Resultaten/Ausgaben und Fertigkeiten. Mindestens die letzten vier Punkte sind unmittelbar mit formalen Steuerungs-/Kontrollstrukturen assoziiert.

In diesem Teilabschnitt werden die Quellen struktureller Komplexität in Organisationen konkretisiert und ihre Auswirkungen näher beleuchtet. Dabei wird Scott [Sco03, Kapitel 9+10] folgend die analytische Unterscheidung zwischen dem *technischen Kern* und den *peripheren Komponenten* einer Organisation vorgenommen. Mintzberg [Min83, S. 10–19] nimmt eine ana-

loge Trennung vor. Er identifiziert einen *operationalen Kern* und benennt vier weitere strukturelle Organisationsanteile, die nach Scott den peripheren Komponenten zuzuordnen wären: Managementhierarchie (bei Mintzberg als *middle line* bezeichnet), Support, Technostruktur (verantwortlich für Standardisierung) und strategischer Apex (verantwortlich für die strategische Ausrichtung).

Die strukturelle Komplexität des technischen Kerns ist vor allem auf das Wesen der auszuführenden Kernaufgaben zurückzuführen Die strukturelle Komplexität der peripheren Komponenten ergibt sich zum einen aus der Größe der Organisation, zum anderen aus den Anforderungen, die aus den Aufgabenumgebungen und institutionellen Umgebungen der Organisation herrühren.

4.3.3.1 Der technische Kern

Der technische Kern ist mit der Ausführung der organisatorischen Kernaufgaben verknüpft. Um im Kontext der Gesamtmenge organisatorischer Aktivitäten eine spezifische Benennung zu finden, werden die damit assoziierten Aktivitäten in der Folge schlicht als *Arbeitsvorgänge* bezeichnet. Von besonderem Interesse sind hier die Konzepte des *technischen Systems* und der *Technologie*. Das technische System bezeichnet dabei Ausrüstung, Maschinen und Instrumente für die Transformation von Ein- in Ausgaben (vergleiche Mintzberg [Min83, S. 128-135]). Gegebenenfalls werden noch Methoden einbezogen (vergleiche Sproull und Goodman [SG90, S. 255]). Über die durch das technische System ermöglichten physikalischen Prozesse hinaus bezieht sich Technologie vor allem auf intellektuelle Prozesse und die Anwendung von Wissen für die Produktion von Ein- in Ausgaben (vergleiche Hulin und Roznowski [HR85, S. 47], Mintzberg [Min83, S. 128], Scott [Sco03, S. 231+232]). Dabei kann Technologie auf komplexen technischen Systemen aufbauen oder aber keine oder nur sehr einfache technische Systeme in Anspruch nehmen. Für beide Aspekte spielt die Umgebung der Organisation eine wichtige Rolle. Einerseits stellt sie die Quelle von Eingaben als auch von Abnehmern des Produzierten dar (Aufgabenumgebung). Andererseits ist sie außerdem die Quelle technischen und fachlichen Wissens, spezifischer Arbeitsmethoden und innovativer Werkzeuge (Institution).

Unter den vielfältigen Möglichkeiten, Technologie zu messen beziehungsweise zu bewerten, identifiziert Scott [Sco03, S. 232–233] drei allgemeine Dimensionen.

- *Komplexität/Diversität* bezieht sich auf die Vielfalt verschiedener Aspekte und Elemente, die bei Arbeitsvorgängen simultan berücksichtigt werden müssen.
- *Unsicherheit/Unvorhersehbarkeit* bezieht sich darauf, inwieweit Arbeitsvorgänge beziehungsweise die Reaktionen der prozessierten Elemente vorausgesehen werden können.
- *Interdependenz* bezieht sich auf wechselseitige Abhängigkeiten zwischen mehreren Arbeitsvorgängen und den dabei prozessierten Elementen.

Diese Dimensionen sind so interessant, weil sich aus ihnen Vorhersagen über strukturelle Eigenschaften einer Organisation ableiten lassen. Scott [Sco03, S. 233] nennt drei Faustregeln. (1) Hohe technische Diversität führt zu hoher struktureller Komplexität. (2) Hohe technische Unsicherheit verringert die Anwendbarkeit einfacher Steuerungsmechanismen wie Ablaufstandardisierung. (3) Hohe technische Interdependenz verlangt hohen Koordinationsaufwand.

Galbraith [Gal74] schlägt eine uniforme Handhabung der vielfältigen Anforderungen von Technologie an Organisationsstrukturen vor. Er stellt Informationsverarbeitung in den Mittelpunkt. Die drei von Scott identifizierten Faustregeln lassen sich derart interpretieren, dass

erhöhte technische Komplexität, Unsicherheit oder Interdependenz jeweils zu erhöhtem Aufwand an Informationsverarbeitung führen. Scott [Sco03, 234–244] folgt Galbraiths Ausführungen und klassifiziert Koordinationsmechanismen und deren Strukturbedingungen nach wachsenden Anforderungen an Informationsverarbeitung.

Für einfache, vorhersehbare Arbeitsvorgänge sind *Programme* und *Ablaufpläne* ausreichend. Sie adressieren die grundlegenden Fragen des „Was?“, „Wie?“ und „Wann?“ der Aufgabenbearbeitung. Bei steigender Anzahl von Arbeitsvorgängen und des damit einhergehenden Informationsflusses werden darauf aufbauende Strukturen notwendig. *Abteilungsbildung* dient in der Hauptsache der Gruppierung von Arbeitsvorgängen. Dabei gilt es, eine Abwägung zu treffen. Einerseits sollte eine Gruppierung den Koordinationsaufwand möglichst minimieren. Dies bedeutet, reziprok und sequentiell interdependente Arbeitsvorgänge möglichst gemeinsam beziehungsweise adjazent zu gruppieren. Auf diese Weise entstehen weitestgehend selbstgenügsame Produktabteilungen. Funktionale/prozessorientierte Gruppierung von ähnlichen Arbeitsvorgängen hingegen fördert die gemeinsame Nutzung technischer Systeme. Auch Mintzberg [Min83] unterscheidet *marktbasierte* gegenüber einer *funktionalen* Gruppierung (S. 58–62) und analysiert ihre jeweilige Angemessenheit entlang verschiedener Anteile einer Organisation (S. 62–65) sowie entlang verschiedener Organisationsformen (Kapitel 8–12).

Hierarchien erweitern das Abteilungsprinzip um multiple Ebenen. Auf diese Weise können immer umfassendere Koordinationsaufgaben bewältigt werden. Hierarchien werden häufig mit dem Prinzip der *Delegation* verknüpft. Ausgehend von einer Hierarchieebene erfolgt die Koordination der untergeordneten Ebenen nicht mittels direkter, feingranularer Steuerung, sondern mittels der Spezifikation von gewünschten Ergebnissen und Zielen. Verantwortung über die tatsächliche Vorgehensweise wird delegiert. Auf diese Weise wird die Kopplung von Hierarchieebenen gelockert und damit der Koordinationsaufwand verringert.

Mit steigender Abhängigkeit zwischen Abteilungen und dem damit einhergehenden höheren Informationsfluss werden vertikale Hierarchien überladen, da Koordinationsfragen und Konflikte häufig eine Hierarchieebene höher gereicht werden müssen (das führt zusätzlich zu dem Problem der unzureichenden Informiertheit des Akteurs, der sich schließlich mit den Problemen befassen muss). Zusätzlich zu den bisher genannten grundlegenden Koordinationsmechanismen benennen Galbraith und Scott daher einige *fortgeschrittene Mechanismen*. Dabei kann einerseits versucht werden, das Aufkommen an Informationsverarbeitung an sich zu verringern. Vor allem aber wird die Strategie verfolgt, informationsverarbeitende Kapazitäten zu erhöhen. Dabei gilt die Aufmerksamkeit vor allem der Anreicherung der traditionellen vertikalen Hierarchien um zusätzliche horizontale Verbindungen. Diese erlauben eine direkte Interaktion zwischen den betroffenen Parteien. Solche lateralen Verbindungen reichen von lokal (und häufig auch zeitlich) begrenzten Liaisonpositionen, Integrationsmanagern, Task Forces, und Projektteams über organisationsweite Performanzkontroll- und Aktionsplanungssysteme bis hin zu Matrixorganisationen mit multiplen formalen Autoritätslinien.

Bis hierher wurde vor allem die rationale Systemperspektive betont. Von Vertretern der natürlichen Systemperspektive werden zusätzliche Aspekte betont. Hier wird die Bedeutung informeller Strukturen hervorgehoben, insbesondere im Angesicht von technischer Unsicherheit. Winter [Win90] argumentiert, organisatorische Fähigkeiten seien fragmentiert, verteilt und eingebettet in organisatorische Routinen, so dass kein einzelner Akteur wirklich wisse, wie die Organisation ihre Aufgaben bewerkstellige oder welche Alternativen es gebe. Darüber hinaus wird die rationale Grundannahme, Technologie würde Strukturen determinieren, in Frage gestellt. Beispielsweise beschreibt Child [Chi72], Technologie sowie weitere Umwelteinflüsse würden lediglich vage und allgemeine Beschränkungen der Struktur einer Organisation

bestimmen. Ein gegebener Kontext lässt viele adaptive Antworten in Form von Organisationsstrategien und -strukturen zu. Die Entscheidungsträger der Organisation (dominante Koalition) haben die Möglichkeit der *strategischen Wahl* (*strategic choice*).

Ein weiterer Aspekt, der eng mit der natürlichen Systemperspektive verknüpft ist, betrifft die Beobachtung, Organisationen würden mehr und mehr Gewicht auf Aktivitäten/Prozesse statt auf Strukturen legen. Scott [Sco03, S. 258] beschreibt, in den letzten Jahren des 20. Jahrhunderts hätten Organisationen neuartige Möglichkeiten der Flexibilität entwickelt, unter ihnen reduzierte Hierarchien, losere und flexiblere Kopplung zwischen Abteilungen sowie die Etablierung temporärer Projektteams. Mintzberg [Min83, Kapitel 12] definiert als Gegenentwurf zur klassischen Bürokratie eine *Adhoc-kratie* (*Adhocracy*) als hochorganische Konfiguration mit äußerst geringem Maß an Formalisierung. Akteure werden je nach Kontext und Aufgabenstellung in temporären, produktorientierten Projektteams beschäftigt. Allerdings ist der vollständige Verzicht auf formale Strukturen insbesondere für große Organisationen nicht praktikabel. Brown und Eisenhardt [BE97, S. 28] schlagen *Semi-Strukturen* als Mittelweg vor. In diesem Kontext findet sich bei Morabito et. al [MSB99, S. 66–74] eine Gegenüberstellung verschiedener grundsätzlicher Ansätze des organisatorischen Entwurfs. Dabei werden verschiedene Konstruktionsprioritäten (Struktur, Prozess oder Hybrid) unterschieden.

4.3.3.2 Die peripheren Komponenten

Die Unterscheidung zwischen technischem Kern und den peripheren Komponenten ist in vielen Fällen recht willkürlich. Es geht um eine analytische Unterscheidung zwischen Organisationsanteilen, die auf verschiedene Kräfte reagieren. Der technische Kern ist unmittelbar von technologischen Anforderungen geprägt. Vereinfacht ausgedrückt handelt es sich bei den peripheren Komponenten schlicht um die Organisationsanteile, für die technologische Zwänge weniger prägend sind. Eine absolute Definition ist nicht möglich.

In vielerlei Hinsicht ist es schwierig, allgemeingültige Aussagen hinsichtlich struktureller Konfigurationen der peripheren Komponenten zu treffen. Eine häufig gestellte Frage ist beispielsweise, wie die Größe einer Organisation mit ihrem Grad an Bürokratie – gemessen als die relative Größe der administrativen Komponente der Organisation – zusammenhängt. Allerdings beschreibt Blau [Bla70], dass Größe der Organisation je nach konkretem Anwendungsfall gegensätzliche Effekte auf die administrative Komponente haben kann. Einerseits kann Größe mit struktureller Differenzierung/Diversifikation in Form einer größeren Anzahl verschiedenartiger Untereinheiten einher gehen. Der erhöhte Koordinationsaufwand führt dementsprechend zu einer Vergrößerung der administrativen Komponente. Andererseits kann Größe auch lediglich das Volumen homogener Arbeitsvorgänge und damit die Größe einzelner Untereinheiten erhöhen. Je größer die Anzahl der Akteure ist, die die gleiche oder ähnliche Arbeit verrichten, desto geringer ist der Koordinationsaufwand. Dementsprechend hat dieser Fall einen gegenteiligen Effekt auf die administrative Komponente.

Für den Fall der Diversifikation lässt sich ein weiterer Aspekt beobachten. Wenn eine Organisation immer weitere Domänen erschließt und dabei über einen gewissen Punkt hinaus wächst, so kann dies zu einer signifikanten Reorganisation führen. Chandler [Cha90, S. 38–45] propagiert, Organisationen, die immer neue Märkte erschließen und dementsprechend hinsichtlich Skopus anstatt Skalierung wachsen, würden von einer Umstrukturierung in eine *M-Form* (multidivisionale Form) profitieren. In einer M-Form werden weitestgehend autonome Divisionen etabliert. Diese zeichnen für Produktion oder Dienstleistungen in einer bestimmten Domäne oder gegenüber einem bestimmten Markt verantwortlich. Darüber wird eine neue

Hierarchieebene eingeführt. Die Akteure der *allgemeinen Unternehmenszentrale* (*General Corporate Office*) sind von alltäglichen Entscheidungen befreit und legen stattdessen die globale Strategie fest, beziehungsweise teilen den einzelnen Divisionen Ressourcen zu. Interessanterweise finden sich hier zwei Spezialfälle (im Großen) der weiter oben für den technischen Kern skizzierten Koordinationsmechanismen wieder: Produkt-/marktbasierte Gruppierung und Delegation. Dieses Beispiel unterstreicht, dass bestimmte Koordinationsmechanismen Relevanz über die konkrete Art der Tätigkeit hinaus haben (Arbeitsvorgänge des technischen Kerns im Gegensatz zu administrativen Aktivitäten der peripheren Komponenten).

Trotz der Schwierigkeit, konkrete und gleichzeitig hinreichend allgemeingültige Aussagen bezüglich der strukturellen Konfiguration der peripheren Komponenten einer Organisation zu treffen, so lassen sich doch zumindest einige universelle Richtlinien identifizieren. Bereits in Abschnitt 4.2 wurde auf Pufferungs- und Brückenschlagsmechanismen hingewiesen, die Organisationen sowohl ihren Aufgabenumgebungen als auch ihren institutionellen Umgebungen gegenüber anwenden. Ohne auf konkrete Mechanismen einzugehen, lässt sich feststellen, dass die Anwendung jedes solcher Mechanismen Änderungen der Organisationsstruktur mit sich bringt. Die Mechanismen können als Versuch der Organisation verstanden werden, einen Teil der Komplexität und Unsicherheit der Umwelt zu internalisieren. Scott [Sco03, S. 279] unterteilt *Internalisierungsaktivitäten* in zwei Kategorien: *Absorption* und *Abbildung*.

Beispiele für Absorption treten beispielsweise in Form von Pufferungsmechanismen gegenüber Aufgabenumgebungen auf. Um Materialien auf der Eingabenseite des technischen Kerns trotz variabler Verfügbarkeit in der Umwelt zu sichern, können Lager eingerichtet werden. Auf der Ausgabenseite können dauerhafte Wege der Distribution und des Vertriebs etabliert werden. Beispiele der Abbildung treten besonders gegenüber institutionellen Umgebungen auf. Organisationen erhöhen ihre Legitimität gegenüber ihrer Umwelt durch Isomorphie, das Inkorporieren/Spiegeln institutioneller Vorgaben von außen in ihre eigene institutionelle Struktur.

Im Zusammenhang mit Brückenschlagsmechanismen ist ein weiterer Aspekt hervorzuheben. In einigen Fällen gilt, dass sich die formalen Grenzen der Beteiligten verändern. Ein Beispiel sind Firmenfusionen. In anderen Fällen geht es eher um eine mehr oder weniger lose Kooperationen, bei denen die formalen Grenzen der Beteiligten gewahrt bleiben. Ein Beispiel sind strategische Allianzen. Nach Powell [Pow90] können Verbindungen des zweiten Falls zu einer eigenen Organisationsform führen – der *Netzwerkform*. Sie rangiert zwischen den traditionellen Organisationsformen der Hierarchie und des Marktes. Vereinfacht formuliert sind Akteursverbindungen in Netzwerkformen stabiler als auf Märkten, zugleich aber wechselseitiger und egalitärer als in Hierarchien. Als wesentlicher Vorteil der Netzwerkform gegenüber der Hierarchie gilt Flexibilität und damit verbunden eine höhere Geschwindigkeit bei der Anpassung an dynamische Umweltcharakteristika. Gegenüber dem Markt sind die Reduktion von Unsicherheiten sowie die Reduktion von Produktions- und Transaktionskosten als Vorteile zu nennen.

4.3.4 Kybernetik

Aspekte der Steuerung und Kontrolle werden natürlich nicht nur im Rahmen der Organisationstheorie behandelt. Sie spielen beispielsweise auch in der allgemeinen Systemtheorie eine tragende Rolle. Dabei wird in diesem Teilabschnitt demonstriert, dass organisationstheoretische Betrachtungen von Steuerung und Kontrolle zu einem großen Teil als Ausprägungen der Betrachtungen der allgemeinen Systemtheorie aufgefasst werden können. Dies ist auch die erwartete Beobachtung, schließlich wurde das Ziel der allgemeinen Systemtheorie bereits zu

Beginn dieses Kapitels als die Identifizierung von Gemeinsamkeiten über die Grenzen aller Formen von Systemen hinweg benannt.

Steuerung und Kontrolle wird vor allem im Rahmen der *Kybernetik* als einem Teilbereich der allgemeinen Systemtheorie adressiert. Als Begründer der Kybernetik gilt Norbert Wiener. Wiener [Wie65, S. 11] leitet den Begriff Kybernetik von dem griechischen Wort *Kubernetes* für *Steuermann* ab. Die Kybernetik hat eine große Bedeutung innerhalb der Systemtheorie und auch innerhalb der Organisationstheorie eingenommen. Beer [Bee85, S. ix] charakterisiert sie als die Wissenschaft des effektiven Organisierens. Zwei wesentliche Aspekte der Kybernetik lassen sich anhand Herrings [Her02, S. 11] Sichtweise auf den Zweck der Kybernetik identifizieren. Er formuliert das Ziel der Kybernetik als das Verstehen und die Formalisierung der grundlegenden Prinzipien von Systemen, die durch einen konsistenten *Systemzweck* charakterisiert sind und auf *Rückkopplungssteuerung* zurückgreifen, um diesen Zweck zu erreichen.

Die Essenz der Rückkopplungssteuerung beschreibt Herring [Her02, S. 25] damit, bestimmte Aspekte der Ausgabe des *gesteuerten Anteils* des Systems mit dem *steuernden Anteil* des Systems rückzukoppeln. Der steuernde Anteil wägt die Ausgabe gegen den Systemzweck ab und trifft Entscheidungen in Form von Kommandos an den gesteuerten Anteil, um dessen zukünftiges Verhalten zu steuern. Dabei bezieht sich die kybernetische Sichtweise auf offene, in eine Umwelt eingebettete Systeme. Es sind im Wesentlichen Einflüsse der Umwelt, die die Ausgabe des gesteuerten Anteils in einer Weise beeinflussen, die den steuernden Anteil zum Einschreiten zwingt.

Als grundlegend für die Gestaltung des steuernden Anteils eines Systems gilt das *Gesetz der angemessenen Vielfalt* (*law of requisite variety*) nach Ashby [Ash56, S. 207][8]: *„Variety can destroy variety.“* Damit ist hier gemeint, dass eine Systemsteuerung desto mehr Störungen in einem Steuerungsprozess ausgleichen kann, je größer ihre eigene Handlungsvielfalt ist (die Handlungsvielfalt der Steuerung „zerstört“ die Vielfalt an auftretenden Störungen).

Es existieren verschiedene Ausprägungen der Rückkopplungssteuerung (eine Übersicht findet sich beispielsweise bei Herring [Her02, Abschnitt 2.3]). Eine populäre Variante besteht darin, eine weitere Zweiteilung des steuernden Anteils vorzunehmen. Ein Teil trifft hochrangige Steuerungsentscheidungen während der andere für die Operationalisierung dieser Entscheidungen zuständig ist. Swinth [Swi74, S. 23] bezeichnet die beiden Teile als *Policy-Zentrale* (*policy center*) und *Steuerung* (*control*).[9] Die Policy-Zentrale setzt die Ziele des Gesamtsystems unter dem Einfluss von Präferenzen und Anforderungen der Umwelt. Anschließend übermittelt sie die Ziele in Form von Teilzielen, Plänen oder Performanzstandards an die Steuerung. Die Steuerung transformiert diese Instruktionen in konkrete Programme für den gesteuerten Anteil des Systems. Das Modell weißt zwei Rückkopplungskreise auf. Die Steuerung überwacht die Ausgaben des gesteuerten Anteils und vergleicht sie qualitativ und quantitativ mit den Vorgaben der Policy-Zentrale. Diskrepanzen sind Auslöser für Modifikationen der Programme. Darüber hinaus kann die Reaktion der Umwelt auf die Ausgaben des Systems dazu führen, dass die Policy-Zentrale die Systemziele anpasst beziehungsweise ändert.

Nach Ashby [Ash60, Kapitel 7] befasst sich der erste Rückkopplungskreis mit Störeinflüssen in Bezug auf Ausmaß und der zweite mit Störeinflüssen in Bezug auf Gattung. Im ersten Fall handelt es sich um Störeinflüsse, die anhand bestehender Regeln behandelt werden können

[8] Beer [Bee84] stuft Ashbys Gesetz in Bezug auf die Wissenschaft des Managements als ebenso wichtig ein wie Newtons Gesetz in Bezug auf die Physik.

[9] Terminologisch bestehen Unterschiede zwischen verschiedenen Autoren bei der Zweiteilung des steuernden Anteils eines Systems. So verwendet beispielsweise Herring [Her02, S. 14] die Begriffe *Steuerung* und *Regulierung* jeweils für die Policy-Zentrale und die Steuerung nach Swinth.

(Ashby formuliert es auch so, dass die Werte kritischer Systemvariablen innerhalb vorgesehener/erwarteter Bereiche bleiben). Im zweiten Fall handelt es sich um Störeinflüsse, die zur Modifikation von Systemregeln beziehungsweise zur Bildung neuer Systemregeln führen (die Werte kritischer Systemvariablen überschreiten vorhergesehene/erwartete Bereiche).[10] Solch adaptives Verhalten, das nicht nur in veränderten Aktivitäten sondern auch in veränderten Regeln zur Auswahl dieser Aktivitäten resultiert, wird von Argyris und Schön [AS78, S. 2+3] als *kontinuierliches Lernen* (*double-loop learning*) bezeichnet.

Nach Scott [Sco03, S. 86] stellt dieses Prinzip ein mächtiges und generelles Modell der Steuerung zur Verfügung, das in organisatorischen Szenarien weithin Anwendung findet. Und tatsächlich lassen sich unter dem Dach dieser abstrakten Charakterisierung von Systemverhalten nach dem Prinzip der doppelten Rückkopplungssteuerung die in diesem Abschnitt betrachteten Aspekte der Steuerung und Kontrolle in Organisationen weitestgehend verorten: Ziel-/Strategiesetzung, Errichtung formaler Steuerungs-/Kontrollsysteme, Machtverteilung, Umwelteinflüsse und Adaption. Jeder dieser Aspekte wurde im Laufe des Abschnittes mit einer Vielzahl an einhergehenden Facetten vorgestellt. So ergibt sich ein großes Spektrum an Ausprägungen der kybernetischen Steuerungs- und Kontrollprinzipien für den Fall sozialer Organisationen.

4.4 Zusammenfassung

Dieses Kapitel war der Konkretisierung von fundamentalen Konzepten und Mechanismen aus dem Bereich sozialer Organisationssysteme gewidmet, die für die Gestaltung von organisationsorientierten Softwaresystemen herangezogen werden können (oder sollten). Im zweiten Teil der Arbeit dienen sie als Grundlage für die Erarbeitung des Konzepts einer Organisationseinheit als Basisbaustein von Multiorganisationssystemen (als konkreter Form von organisationsorientierten Softwaresystemen). Das vorige Kapitel endete mit der Feststellung, es sei gerade die janusgesichtige Eigenschaft von Organisationen als Kontexte sowie als kollektive Akteure, die den Einbezug von Organisationskonzepten in die Gestaltung von Softwaresystemen von Softwaresystemen so vielversprechend macht.

Dementsprechend stand das vorliegende Kapitel ganz im Zeichen dieser Janusgesichtigkeit. Zunächst wurde eine Relativierung der Begriffe der Organisation und der organisierten Kollektivität vorgenommen. Da der Organisationsbegriff traditionell vielfach mit betrieblichen Unternehmen/Firmen assoziiert ist, wird die organisierte Kollektivität hier als Oberbegriff für jegliche Form von formalisierter und zielgerichteter Kollektivität in einem mehrschichtigen Organisationssystem verwendet. Bei der Referenzierung der verschiedenen organisationstheoretischen Studien wurde jedoch häufig den Originalstellen der Literatur gefolgt und der Begriff der Organisation verwendet. Dabei wurden die zitierten Konzepte und Mechanismen aber durchgängig so gewählt, dass sie einen universellen Charakter aufweisen und auf ein breites Spektrum an organisierten Kollektivitäten anwendbar sind.

[10] Im Zusammenhang mit der doppelten Rückkopplungssteuerung steht auch die Charakterisierung von Stabilität und Ultrastabilität von Ashby [Ash60, insbesondere Kapitel 7]. Beide beziehen sich auf die Fähigkeit eines Systems, nach Störeinflüssen aus der Umgebung wieder in einen Zustand zu konvergieren, in dem das gewünschte Verhalten beziehungsweise der gewünschte Wert einer kritischen Systemvariablen wieder hergestellt ist. Im Falle der Stabilität geschieht dies ohne eine Änderung der internen Systemorganisation. Damit können aber lediglich Störeinflüsse in Bezug auf Ausmaß abgefangen werden. Ultrastabilität bedeutet Stabilität einer logisch höheren Ordnung, indem durch Modifikation der internen Systemorganisation auch Störeinflüssen in Bezug auf Gattung begegnet werden kann.

Die hier zitierten Studien wurden in zwei Kategorien unterteilt, die jeweils eine Seite des Konzeptes der organisierten Kollektivität betreffen. Zunächst wurde das Zusammenspiel organisierter Kollektivitäten mit ihren Umgebungen adressiert. Betrachtet wurden dabei Kriterien der Grenzdefinition, Charakteristika von (organisatorischen) Umgebungen als Aufgabenumgebungen sowie als Institutionen, Einflüsse aus Umgebungen und Reaktionen darauf, die Inbetriebnahme von und das Agieren in Umgebungen und schließlich die Rekursivität des Inbezugsetzens von offenen Systemen und ihrer Umwelt.

Darauf folgend wurden interne Konfigurationen einer organisierten Kollektivität behandelt. Im Fokus standen hier Ziele/Strategien einer Kollektivität und deren Zustandekommen, die Errichtung von Steuerungs-/Kontrollsystemen für die Verfolgung der Ziele sowie die daraus folgenden Aspekte der strukturellen Komplexität. Schließlich wurden die zuvor vornehmlich organisationstheoretisch geprägten Konzepte in Bezug zu allgemeineren Erkenntnissen der Kybernetik gesetzt.

Die in diesem Kapitel zusammengetragenen Konzepte und Mechanismen werden vom Autor der vorliegenden Arbeit als Grundlage für die Diskussion organisierter Kollektivitäten angesehen. Im Hinblick auf die Softwareentwicklung stellen sie die Grundlage für eine Bewertung dar, inwieweit das Potential der Organisationsmetapher ausgenutzt wird oder nicht. Im folgenden Kapitel erfolgt zunächst eine kritische Untersuchung, inwieweit die hier benannten Konzepte und Mechanismen im Kontext bereits existierender Ansätze der organisationsorientierten Gestaltung von Softwaresystemen Berücksichtigung finden. Aus dem Ergebnis dieser Untersuchung ergibt sich die Zielsetzung des zweiten Teils der Arbeit. Dort steht die vertiefte Betrachtung von Multiorganisationssystemen als einer konkreten Form von organisationsorientierten Softwaresystemen im Fokus. Die Konzepte und Mechanismen des vorliegenden Kapitels stehen dabei Pate für die Formulierung konkreter Anforderungen.

5 Organisationsstrukturen in Softwaresystemen – *State of the Art*

Bis hierher wurde auf Basis der für Softwaresysteme von Softwaresystemen herausgearbeiteten Charakteristika und Herausforderungen ein Plädoyer für ein Organisationsparadigma der Softwareentwicklung vorgetragen. Zusätzlich wurden organisationstheoretische Kernkonzepte und -mechanismen als Grundlage für ein solches Paradigma expliziert. In diesem Kapitel wird nun beleuchtet, welche Bestrebungen hinsichtlich einer organisationsorientierten Gestaltung von Softwaresystemen bereits existieren. Diese werden charakterisiert und anhand der Ausführungen der vorigen beiden Kapitel dahingehend bewertet, inwiefern das Potential der Organisationsmetapher ausgenutzt wird. Die Ergebnisse des vorliegenden Kapitels stellen mithin die Ausgangslage für die Ausrichtung des Modellierungsvorhabens im zweiten Teil der vorliegenden Arbeit dar.

Es geht in diesem Kapitel um Ansätze, die den Einzug von Organisationsstrukturen in Softwaresystemen auf Grundlage des sozialen Konzeptes einer Organisation umsetzen. Diese Herangehensweise hat sich vornehmlich in der Multiagentensystemforschung durchgesetzt. Dies ist nicht überraschend da das agentenorientierte Softwareentwicklungsparadigma ohnehin von sozialen Konzepten geprägt ist und somit einen idealen Nährboden für organisationsorientierte Ideen bietet. Multiagentensysteme und Organisationsorientierung werden in den Abschnitten 5.1 und 5.2 behandelt. Als Ergänzung wird mit *Viable Software* einer der wenigen Ansätze für Softwaresysteme mit Organisationsstrukturen, der nicht in der Multiagentensystemtechnik verwurzelt ist, in Abschnitt 5.3 vorgestellt. Interessanterweise bietet speziell dieser Ansatz eine teilweise bessere Ausnutzung des Potentials der Organisationsmetapher als es in Multiagentensystemansätzen der Fall ist.

In der Einleitung der Arbeit wurde betont, der Begriff des *organisationsorientierten Softwaresystems* sei mit zwei Lesarten assoziiert. Statt *Organisationsorientierung* auf die Übertragung der Bauprinzipien von sozialen Organisationssystemen auf Softwaresysteme zu beziehen, kann auch die Ausrichtung von Informationstechnologie an den Anforderungen realweltlicher Organisationen gemeint sein. In der Einleitung wurden die beiden Lesarten als „Softwaresysteme *als* Organisationen“ beziehungsweise als „Softwaresysteme *für* Organisationen“ charakterisiert. Auch wenn die zweite Interpretation nicht im Fokus der vorliegenden Arbeit liegt, ist ihr der Abschnitt 5.4 gewidmet. Dem Thema der Softwaresysteme *für* Organisationen wird eine große Aufmerksamkeit in der aktuellen Forschung zuteil und der Autor sieht eine Verknüpfung mit den Ergebnissen der vorliegenden Arbeit als besonders lohnenden Zweig zukünftiger Forschung.

5.1 Agenten und Multiagentensysteme

Die agentenorientierte Softwareentwicklung hat sich als sozial fundiertes Softwareentwicklungsparadigma etabliert. Ihren Ursprung hat die Agentenorientierung in der Künstlichen Intelligenz beziehungsweise vor allem in der Verteilten Künstlichen Intelligenz. Weiß [Wei99,

Prologue, S. 1] definiert die Verteilte Künstliche Intelligenz (*distributed artificial intelligence, DAI*) und Multiagentensysteme gemeinsam: *„DAI is the study, construction, and application of multiagent systems, that is, systems in which several interacting, intelligent agents pursue some set of goals or perform some set of tasks.“*

Ihrem Ursprung in der (Verteilten) Künstlichen Intelligenz ist die Agentenorientierung mittlerweile entwachsen. Stattdessen hat sich das Konzept des Agenten als stellvertretend für eine eigenständige Klasse von universell einsetzbaren Softwareentitäten herauskristallisiert. Die damit einhergehende agentenorientierte Softwareentwicklung wird weithin als Fortführung der objektorientierten Softwareentwicklung verstanden (vergleiche Teilabschnitt 5.1.2). Zusätzlich zu den bereits für das Objektkonzept bekannten Prinzipien der Kapselung von Daten und Funktionen, wird im Falle von Agenten das Prinzip der Kapselung von Ausführungssteuerung betont (Prinzip der Autonomie). Damit entsteht eine unmittelbare inhaltliche Nähe zur Thematik der Systeme von Systemen mit ihren unabhängigen Teilsystemen.

5.1.1 Kanonische Sichtweise auf Multiagentensysteme

Abbildung 5.1 zeigt die von Jennings [Jen00] in seinem einflussreichen Referenzartikel *„On Agent-Based Software Engineering“* postulierte und vielfach zitierte *kanonische Sichtweise auf Multiagentensysteme*. Dabei werden die Konzepte der Interaktionen zwischen Agenten, die

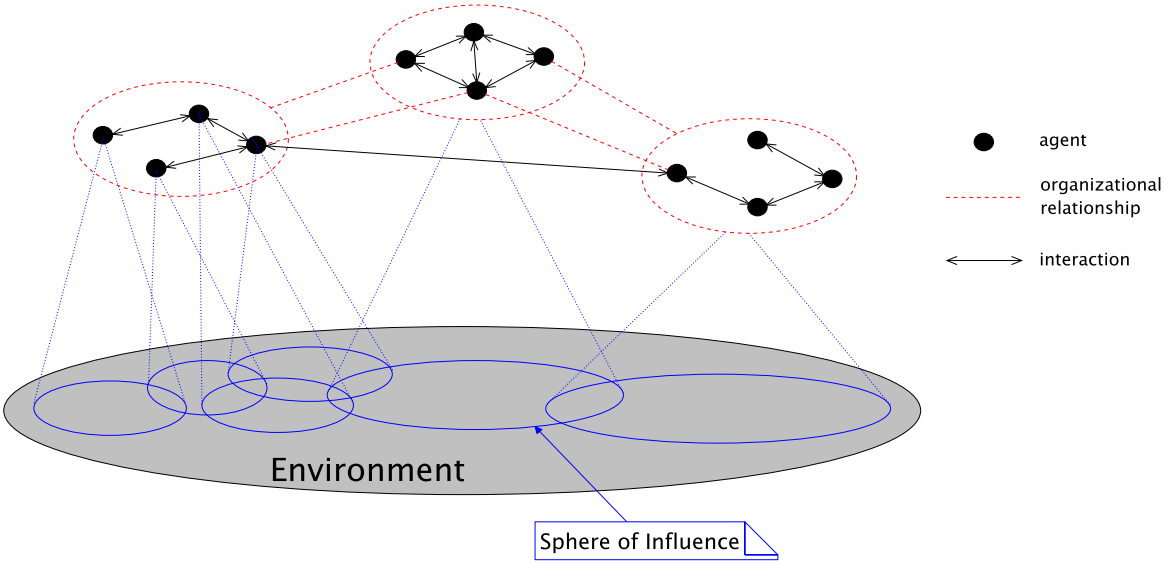

Abbildung 5.1: Kanonische Sichtweise auf Multiagentensysteme nach Jennings [Jen00]

daraus resultierenden organisatorischen Beziehungen, das Konzept der Umgebung und die Idee der verschiedenen Einflusssphären abstrakt illustriert. Ein Multiagentensystem ergibt sich vor allem aus flexiblen Interaktionen zwischen den eigenständigen, zielgerichteten Agenten. Im Allgemeinen ist mit einem Agenten eine autonome, (künstlich) intelligente und sowohl reaktive als auch proaktive Softwareentität bezeichnet. Während das Konzept der Autonomie prägend ist, wird den anderen Eigenschaften je nach Kontext ein mehr oder minder schweres Gewicht zugesprochen. Da Agenten in Multiagentensystemen in Interaktion zueinander treten, müssen sie eine gewisse „soziale Kompetenz“ besitzen. Agenten setzen ihre Fähigkeiten ein,

um in potentiell dynamischen und unsicheren Umgebungen ihre Ziele zu erreichen. Sie treffen selbständig Entscheidungen, führen Handlungen aus und koordinieren diese mit denen anderer Agenten (in Form von Kooperation oder Konkurrenz).

Die Abbildung adressiert eher die logischen Zusammenhänge in einem Multiagentensystem. Die technische Umsetzung dieser logischen Zusammenhänge kann auf unterschiedliche Weise erfolgen. Dabei ist das Konzept der Umgebung von besonderer Bedeutung. Agenten werden auf *Plattformen* ausgeführt und diese Plattformen werden durch eine Kommunikations- und Migrationsinfrastruktur gekoppelt.[1] Damit ergibt sich zunächst eine technische Deployment-Umgebung für Agenten. Inhaltliche/anwendungsbezogene Umgebungen und damit logische Plattformen können auf unterschiedliche Weise realisiert werden. Einen Überblick bieten beispielsweise Viroli et. al. [VHR+07]. Als grundlegende Differenzierung gilt dabei, ob die inhaltlichen Aspekte eines Multiagentensystems ausschließlich in Form von Agenten umgesetzt werden oder ob zusätzliche Programme, Anwendungen oder ganze Systeme einbezogen sind. Im ersten Fall werden Umgebungen schlicht durch Agenten und Beziehungen zwischen ihnen verkörpert. Im zweiten Fall werden Teile der Umgebung extern zu Agenten softwaretechnisch verkörpert und Agenten haben Zugriff auf die von Seiten der Umgebung angebotenen Daten und Dienste (vergleiche beispielsweise den *Agents & Artifacts*-Ansatz nach Omicini et. al. [ORV08]). Von besonderem Interesse für die vorliegende Arbeit sind Ansätze, die eine Verkörperung organisatorischer Umgebungen/Kontexte ermöglichen. Diese werden im folgenden Abschnitt 5.2 behandelt.

Die soziale Fundierung der Agentenorientierung spiegelt sich sowohl in der vom Menschen als Akteur inspirierten Metapher des Softwareagenten als auch in den Beschreibungen von Interaktion, Koordination und Kooperation zwischen mehreren Agenten wider. So werden Agenten häufig in Form *künstlicher mentaler Zustände* wie Überzeugungen, Wünschen oder Intentionen spezifiziert (vergleiche exemplarisch Georgeff et. al. [GPP+99] oder Cohen und Levesque [CL90a, CL90b]). Auch bei der konkreten Entwicklung bis hin zur Programmierung von Agenten wird die Verwendung mentaler Zustände unterstützt (vergleiche exemplarisch Bordini et. al. [BHW07], Bellifemine et. al. [BCG07], Braubach [Bra07], Pokahr [Pok07]). Auch Mechanismen der Kommunikation und Kooperation, die mehrere Agenten einschließen, werden häufig basierend auf mentalen Zuständen (hier auch *gemeinsamen* mentalen Zustände) und unter zusätzlichem Einbezug von Sprechakttheorie spezifiziert (vergleiche exemplarisch kooperatives Problemlösen nach Jennings [Jen95], Teamwork nach Cohen et. al [LCN90, CLS98] beziehungsweise Tambe [Tam97], gemeinsame Planausführung nach Grosz und Kraus [GK96] oder Konversationsrichtlinien nach Smith et. al. [SCBG98]).

5.1.2 Agentenorientierung in Abgrenzung gegenüber der Objektorientierung

Im Einleitungskapitel der Arbeit wurde betont, Softwareentwicklungsparadigmen würden sich vor allem anhand ihrer jeweiligen *architekturprägenden* Konzepte voneinander abheben. Diese Konzepte dienen als Metapher, anhand derer sich Techniken für Handhabung von Systemkomplexität (Dekomposition, Abstraktion, Organisation) ableiten lassen.

[1] Es gibt eine große Fülle an verschiedenen Agentenplattformen mit verschiedenen Charakteristika. Einige Beispiele sind die Plattformen JADEX (http://jadex-agents.informatik.uni-hamburg.de/, Standalone oder als BDI-Erweiterung von JADE, http://jade.tilab.com/), JACK (http://aosgrp.com/products/) oder JASON (http://jason.sourceforge.net/Jason/Jason.html) für BDI-Agenten (*Beliefs, Desires, Intentions*, siehe unten zu Softwareagenten mit künstlichen mentalen Zuständen), JIAC (http://www.jiac.de/) für eine möglichst enge Kopplung von Agenten- und Service-Konzepten oder Mulan/Capa (vergleiche Kapitel 5) mit petrinetzbasierter Prozessunterstützung.

In diesem Zusammenhang lässt sich die Idee der Agentenorientierung besonders anschaulich durch eine Gegenüberstellung mit der Objektorientierung begreifen. Das bedeutet also eine Gegenüberstellung der beiden Kernmetaphern *Objekt* und *Agent*. Zambonelli et. al. [ZJW03] argumentieren, es handle sich beim Übergang vom Objekt zum Agenten nicht lediglich um eine einfache Spezialisierung sondern um einen Paradigmenwechsel. Mit der Metapher des Agenten als Softwareentität werden Konzepte wie Autonomie, Situiertheit und Proaktivität als erstrangig eingeführt. Diese Eigenschaften gelten für „traditionelle" Objekte nicht. Internes Objektverhalten wird durch Methodenaufrufe von einem externen Steuerungsstrang ausgelöst. Dementsprechend sind Objekte eines proaktiven Verhaltens ebenso wenig fähig wie der autonomen Entscheidungsfindung darüber, was in bestimmten Situationen zu tun ist. Bei der Entwicklung von Objektsystemen wird dementsprechend Wert auf *Kapselung von Daten und Funktionalität* gelegt, während die Kapselung von Ausführungssteuerung und -kontrolle vernachlässigt wird. Demgegenüber spielt dieser Aspekt in Multiagentensystemen eine wichtige Rolle. *Kapselung und Delegation von Ausführungskontrolle* gehören zu den Kernkonzepten. Da Agenten neben Daten und Algorithmen zusätzlich die Kontrolle über ihr eigenes Verhalten kapseln, sind sie in der Lage, der Dynamik einer komplexen Umgebung Herr zu werden und Interdependenzen mit anderen Entitäten zu reduzieren. Auf diese Weise ergibt sich eine klarere Trennung von Entwurfsdimensionen für agenteninterne (Komponentenebene) und -externe (Systemebene) Belange sowie von aktiven (Agenten) und passiven (Ressourcen der Umgebung) Komponenten eines Softwaresystems.

Allerdings weisen Zambonelli et. al. [ZJW03] gleichzeitig darauf hin, dass „modernere" Auffassungen von Objekten die konzeptionelle Grenze zum Agenten verschwimmen lassen. Traditionelle Objektabstraktionen können durch interne Steuerungsstränge, Ereignis-/Ausnahmenbehandlung und Kontextabhängigkeit erweitert werden. Auf diese Weise ergibt sich eine übergeordnete Objektabstraktion in Form einer (potentiell) aktiven, in sich abgeschlossenen Softwarekomponente mit einem weitem Skopus. Für Objektsysteme bedeutet dies multiple Loki der Ausführungskontrolle und flexiblere Objektinteraktionen. Auf diese Weise rücken Objekte und Objektsysteme in die Nähe von Agenten und Multiagentensystemen. Allerdings verschleiert die Anreicherung des konventionellen Objektverständnisses mit neuartigen Konzepten und Charakteristika lediglich die Tatsache, dass Autonomie, Aufgabenorientierung, Situiertheit und flexible Interaktionen nicht zu den *inhärenten* Konzepten des Objektparadigmas zählen. Objektorientierte Ansätze beschäftigen sich zumindest nicht originär mit dem Entwurf von Verhandlungsalgorithmen zur Interaktionssteuerung oder einem geeigneten Verhältnis von reaktivem und proaktivem Verhalten in dynamischen, unsicheren Umgebungen. Für agentenorientierte Ansätze hingegen handelt es sich dabei um natürliche Fragestellungen, die im Agentenparadigma unmittelbar verwurzelt sind. Die Agentenorientierung bietet daher für bestimmte softwaretechnische Fragestellungen und Herausforderungen eine angemessenere „konzeptionelle Passung" gegenüber der Objektorientierung und bezieht aus diesem Umstand ihre Rechtfertigung als eigenes Paradigma.

Jennings [Jen00] stellt die Vorteile der Agentenorientierung im Hinblick auf Techniken zur Komplexitätsreduktion heraus. (1) Die agentenorientierte *Dekomposition* eines Problemraums mittels autonomer, flexibler Problemlöser gibt klare Hinweise, entlang welcher Linien eine Trennung in Systembestandteile vorgenommen wird, nämlich entlang von Ressourcen, Wissen, Fähigkeiten usw. (2) Die der Agentenorientierung innewohnenden (sozial fundierten) Abstraktionen bieten eine aus Sicht menschlicher Systementwickler natürliche und intuitive Herangehensweise, bestimmte Aspekte eines Systems hervorzuheben und gleichzeitig andere auszublenden. (3) Schließlich bringt Jennings ein drittes Argument vor. Es bezieht sich auf

den Aspekt der bisher nicht thematisierten organisatorischen Beziehungen zwischen Agenten aus Abbildung 5.1. Die soziale Fundierung des Softwareagenten legt Mechanismen für überindividuelle Systemstrukturen unmittelbar nahe: Zwischen Agenten entstehen organisatorische Beziehungen, die ebenfalls mittels sozialer Konzepte beschrieben und begriffen werden. Wie in Kapitel 4 aufgezeigt wurde, ergibt sich ein großes Reservoir an Organisationsmechanismen realweltlicher sozialer Systeme als Grundlage für den Aufbau großer Multiagentensysteme.

5.1.3 Entwicklung der Organisationsperspektive in Multiagentensystemen

Der Bau großer Multiagentensysteme ist nicht die einzige Motivation für die sorgfältige Betrachtung überindividueller Systemstrukturen. Gerade die Kombination von Agentenautonomie und flexibler Interaktion zwischen Agenten bedingt ein inhärentes Problem des agentenorientierten Ansatzes der Softwareentwicklung. Agentenautonomie erschwert die Vorhersagbarkeit von Mustern und Ergebnissen von Agenteninteraktionen. Noch allgemeiner beschreibt Milner [Mil93], wie interaktive Komposition in Verhaltensphänomenen resultiert, die sich nicht alleine über das individuelle Verhalten der beteiligten Entitäten oder der Aggregation der individuellen Verhaltensanteile erklären lassen. Stattdessen folgt *emergentes Verhalten* auf nichtlineare Weise aus den Interaktionen der Entitäten. Das „Ganze“ ist mehr als die „Summe seiner Anteile“. Gerade in Anbetracht der (zumindest potentiellen) Elaboriertheit und Flexibilität der Komponenten von Multiagentensystemen ergibt sich ein beträchtlicher Variationsrahmen für emergentes Verhalten. Eine analoge Problematik wurde bereits anhand der evolutionären Entwicklung von Systemen von Systemen in Kapitel 2 identifiziert.

Ausgehend von diesen Schwierigkeiten fordert Jennings [Jen00] die systematische Berücksichtigung einer *sozialen Ebene* der agentenorientierten Softwareentwicklung. Diese ist oberhalb der Ebene, auf der individuelle Agenten charakterisiert werden, anzusiedeln und soll soziale Prinzipien und Fundamente für Multiagentensysteme bereitstellen. Als konkreten Vorschlag unterbreitet Jennings, die Entitäten der sozialen Ebene als *Agentenorganisationen* zu betrachten und deren *Komponenten* (Agenten, Interaktionskanäle, Abhängigkeiten, organisatorische Beziehungen), *Kompositionsgesetze* (Rollen, organisatorische Regeln), *Verhaltensgesetze* (organisatorische Rationalität) und *Medien* (organisatorische und soziale Verpflichtungen, Mittel der Beeinflussung Anderer, Mittel zur Veränderung der Organisationsstruktur) entsprechend zu charakterisieren.

Nach Horling und Lesser [HL05] gehen nahezu alle Multiagentensysteme ohnehin mit einer Form der Organisation einher. Dementsprechend lassen sich Gedanken und Ansätze zu Organisationen in Multiagentensystemen seit Anbeginn der Forschung auf diesem Gebiet finden: Hierarchien [Fox81, MD93, MRS02], Matrizen [DLPW95, HMS+03], Koalitionen [SL97, SGL03], Teams [Tam97, SPS+04], Föderationen [Gen97, ML04], soziale Gesetze [MT95, DK99]. Allerdings weisen Horling und Lesser auch darauf hin, organisatorische Prinzipien seien häufig implizit umgesetzt und nur informell beschrieben. Analoge Kritik wird von verschiedenen Autoren geübt. Mao und Yu [MY04] kritisieren, organisatorische und soziale Konzepte würden sich in der Literatur zu agentenorientierter Softwareentwicklung ungeordnet und ohne einheitliche Terminologie finden. Gasser [Gas01] beklagt, die Idee der Organisation sei zu lange Zeit lediglich ein peripheres Thema der Agentenforschung gewesen, eher eine Spezialtechnik der Koordination. Ebenso kritisiert Ferber [FGM03], von den beiden wesentlichen Aspekten in Multiagentensystemen, nämlich dem einzelnen Agenten (Mikro-Ebene) und der Agentengesellschaft (Makro-Ebene), sei der Gesellschaft zu wenig Aufmerksamkeit zu Teil geworden. Der Schwerpunkt lag auf Seiten des einzelnen Agenten, seinem internen Zustand, Entschei-

dungsmechanismen und dem daraus resultierenden Verhalten.

Allerdings hat sich im Angesicht solcher Kritiken eine Trendwende innerhalb der letzten Jahre ergeben. Mittlerweile existiert eine Fülle von Ansätzen, bei denen die Organisationsmetapher als zentrales Instrument für die überindividuelle Gestaltung von Multiagentensystemen begriffen wird. Die Organisation rückt in den Mittelpunkt, von individuellen Agenten wird teilweise sogar gänzlich abstrahiert. Dieser „neuen Generation“ von Multiagentensystemansätzen ist der folgende Abschnitt ausführlich gewidmet. Ein umfassender Überblick über aktuelle Forschungsbestrebungen und Ergebnisse wird im von Dignum [Dig09] herausgegebenen *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models* gegeben. Viele der im folgenden Abschnitt zitierten Autoren sind mit Beiträgen in dem Handbuch vertreten.

5.2 Organisationen in Multiagentensystemen

Es existieren viele verschiedene Varianten von Organisationsansätzen für Multiagentensysteme. Bevor eine der Varianten als besonders relevant im Kontext der vorliegenden Arbeit gekennzeichnet und anschließend etwas genauer untersucht wird, erfolgt hier eine allgemeine Klassifikation der verschiedenen Varianten.

5.2.1 Klassifikation von Organisationsansätzen in Multiagentensystemen

Für eine systematische Betrachtung von Organisationsansätzen der Multiagentensystementwicklung wird hier auf die Charakterisierung von Boissier et. al. [BHS07] zurückgegriffen. Ansätze werden dabei entlang zweier Dimensionen klassifiziert.

1. Zunächst wird danach unterschieden, ob Organisationskonzepte eine dominierende Rolle spielen oder nicht. Diese explizite Trennung wurde zuvor bereits von Ferber et. al. [FGM03] vorgeschlagen. Sie bezeichnen „herkömmliche“ Multiagentensysteme als *agentenzentriert* (ACMAS, *agent-centered multi-agent systems*) und eine neue Generation von Multiagentensystemen als *organisationszentriert* (OCMAS, *organization-centered multi-agent systems*). Bezogen auf die sozialen und organisationstheoretischen Ausführungen der beiden vorigen Kapitel kann man die Unterscheidung auch dahingehend charakterisieren, ob eine Agentenorganisation *formalisiert* wird oder bestenfalls *informell* „in den Köpfen“ der Agenten entsteht und existiert.

2. Desweiteren wird danach unterschieden, ob Agenten organisatorisches Wissen besitzen oder nicht. Dies betrifft vor allem die Frage, ob es sich bei der Organisation eines Multiagentensystems lediglich um ein Entwurfskonzept (Agenten ohne organisatorisches Wissen) oder aber um ein Laufzeitkonzept (Agenten mit organisatorischem Wissen) handelt.

Abbildung 5.2 zeigt eine Veranschaulichung der verschiedenen Arten von Organisationsansätzen anhand einer Matrix. Die vier Quadranten werden in der Folge näher betrachtet.

Quadrant (a) sind Ansätze zuzuordnen, bei denen weder eine explizite Spezifikation/Formalisierung von Organisationen stattfindet, noch Agenten eine interne Repräsentation organisatorischer Beziehungen aufbauen. Es kann trotzdem der Fall sein, dass ein externer Beobachter mit einer globalen Perspektive organisiertes Verhalten feststellt. Dies tritt beispielsweise bei Multiagentensystemen auf, bei denen das Agentenverhalten auf biologischen

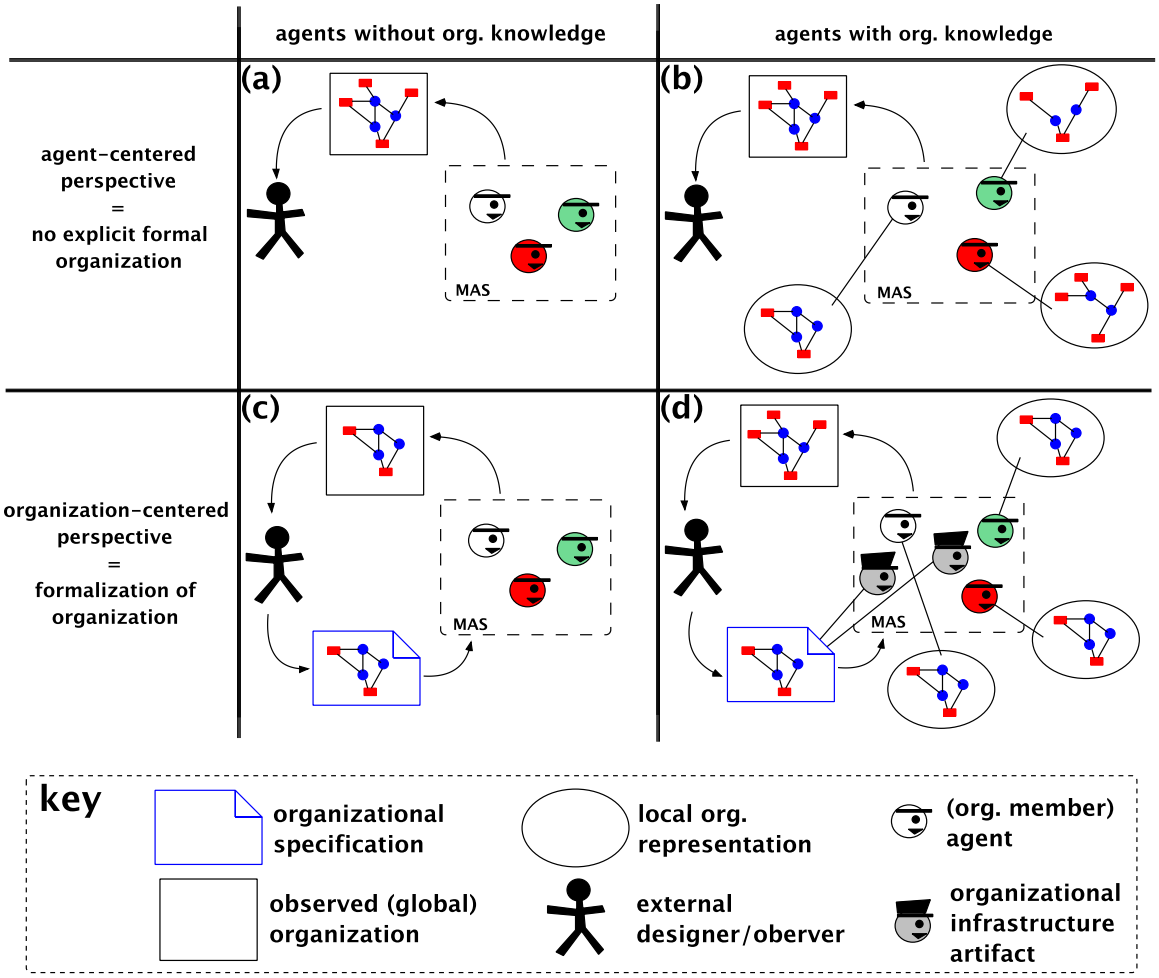

Abbildung 5.2: Klassifikation von Organisationsansätzen für Multiagentensysteme, adaptiert von Boissier et. al. [BHS07]

Selbstorganisationsmechanismen basiert (beispielsweise Futtersuche, Nestbau, Herdenverhalten). Hier wird koordiniertes Verhalten durch die Spezifikation von Reiz-Reaktion-Schemata *implizit* programmiert. Allerdings ist es auch möglich, solche Mechanismen „bewusst“ einzusetzen. Sudeikat et. al. [SBP+09] stellen einen Ansatz vor, bei dem ausgehend von bestimmten Anforderungen eine *Selbstorganisationsmetapher* bewusst gewählt und anschließend implementiert wird. (Sudeikat et. al. beziehen sich dabei vor allem auf die von Mamei [MMTZ06] genannten Selbstorganisationsmetaphern aus dem biologischen Bereich, schließen aber auch physikalische oder soziale Selbstorganisationsphänomene nicht aus).

Quadrant (b) sind Ansätze zuzuordnen, bei denen zwar immer noch keine explizite Spezifikation/Formalisierung von Organisationen stattfindet, Agenten aber elaboriert genug sind, um interne Repräsentationen ihrer organisatorischen Beziehungen zu anderen Agenten aufzubauen und diese in ihre Entscheidungsfindung einzubeziehen. Als Beispiele können das *Contract Net Protocol* nach Smith [Smi80] und der Ansatz des *Social Reasoning* in Abhängigkeitsnetzwerken nach Sichman et. al. [SCCD94] dienen. Im Falle des Contract Net Protocol werden hierarchi-

sche Organisationsstrukturen aufgebaut, indem Agenten sukzessive Aufgaben übernehmen, in Teilaufgaben aufteilen und *Verträge* mit anderen Agenten hinsichtlich der Erledigung dieser Teilaufgaben schließen. Im Falle des Social Reasoning sammeln Agenten Informationen über andere Agenten im Bezug auf deren Ziele, Aktionen, Ressourcen und Pläne. Jeder Agent speichert diese Informationen und nutzt sie, um seine Abhängigkeit gegenüber anderen (oder deren Abhängigkeiten ihm selber gegenüber) entlang einer Taxonomie von Abhängigkeitsrelationen zu bestimmen. Abhängigkeiten beziehen sich immer jeweils auf ein bestimmtes Ziel, so dass der Agent Rückschlüsse über mögliche Überschneidungen oder wechselseitige Ausschlüsse seiner eigenen Ziele und Pläne mit denen anderer Agenten ziehen kann. Die beiden hier skizzierten Ansätze können als Stellvertreter für einen Großteil der weiter oben als „herkömmlich“ charakterisierten Multiagentensystemansätze verstanden werden. Organisationsstrukturen im Sinne künstlicher sozialer Beziehungen entstehen und werden von Agenten intern repräsentiert. Es findet allerdings keine Formalisierung der Organisation im Sinne des in Kapitel 3 eingeführten soziologischen Konzeptes der Formalisierung statt. Es erfolgt keine überindividuelle Explizierung globaler Strukturen. Vor allem entstehen keine Strukturen, die *unabhängig* von den konkreten Agenten sind. Stattdessen „sind die Agenten die Organisation“ und die Organisation existiert lediglich „in den Köpfen der Agenten“.[2]

Dieser Umstand ändert sich beim Übergang von der ersten in die zweite Zeile der Matrix. Bei Ansätzen, die dieser Zeile zuzuordnen sind, werden formale Organisationskonzepte wie Rollen, Interaktionsprotokolle, Gruppen, organisatorische Regeln und ähnliche als *First-Order Concepts* für den Entwicklungsprozess von Multiagentensystemen verwendet. Die Organisationsmetapher wird expliziert, indem eine Trennung der Agenten vom Rahmenwerk des Gesamtsystems vorgenommen wird. Anstatt die Organisation „zu sein“, besetzen Agenten *Positionen* oder *Rollen*, die das Agentenverhalten für spezifische organisatorische Kontexte beschränken und leiten. Organisationsspezifikationen sind extern zu den Agenten. Es findet eine *Formalisierung* der Agentenorganisationen statt, ganz entsprechend des soziologischen Konzeptes. Auf diese Weise verschiebt sich der Fokus bei der Betrachtung eines Multiagentensystems von den einzelnen Agenten auf die globale Organisation. Im Zuge der verschiedenen Ansätze werden dabei verschiedene Dimensionen der Formalisierung betont, so dass je nach Ansatz strukturelle, funktionale, interaktionsbezogene oder normative Spezifikationen dominieren.

Quadrant (c) sind Ansätze zuzuordnen, bei denen zwar eine explizite Spezifikation/Formalisierung der Agentenorganisation vorgenommen wird, die Agenten diese jedoch nicht (oder zumindest nur sehr eingeschränkt) intern repräsentieren. Die Organisation existiert extern zu den Agenten als formalisiertes Schema, welches durch Systementwickler entworfen wurde. Die Agenten nehmen die Organisation jedoch nicht als überindividuelle Entität wahr, sondern kennen lediglich ihren eigenen ganz spezifischen Kontext in Form von Aufgaben, Ressourcen oder Interaktionsmöglichkeiten. Agenten folgen der formalen Organisationsspezifikation schlicht deswegen, weil sie entlang ihres organisatorischen Kontextes und dessen Einschränkungen programmiert werden. Dementsprechend sind Organisationsstrukturen fix und können zur Laufzeit nicht adaptiert werden. Zwei besonders prominente Vertreter dieser Variante von Organisationsansätzen sind die Entwicklungsmethodologien GAIA nach Zambo-

[2] Es lässt sich diskutieren, ob eine besonders verfestigte Repräsentation von sozialen Strukturen „in den Köpfen der Agenten“ eine Formalisierung im soziologischen Sinne darstellt. Beispielsweise könnte unter den Agenten Einigkeit über die Existenz und Funktion bestimmter Positionen/Rollen herrschen, so dass diese Positionen/Rollen über die Zeit mit verschiedenen konkreten Agenten besetzt werden können.

nelli et. al. [ZJW03] und MaSE nach DeLoach [DeL05].[3] In beiden Fällen ist die *Rolle* zunächst das Kernkonstrukt. Darauf aufbauend werden Ziele, Verantwortlichkeiten und Rechte im Zusammenhang mit Rollen sowie Rollenbeziehungen und Interaktionen zwischen Rolleninhabern spezifiziert. Dabei können gegebenenfalls mehrere Verfeinerungszyklen durchlaufen werden. Ausgehend von Rollen und darauf aufbauenden Spezifikationen erfolgt die Transition zu implementierungsnäheren Konstrukten, die schließlich in der Spezifikation und dem Deployment konkreter Agenten münden. Diese Agenten sind somit unmittelbar aus der Spezifikation der formalen Organisation abgeleitet und erfüllen ihre organisatorische Funktion daher auf „natürliche“ Weise. Es ist also kein Zufall, dass es sich bei den Ansätzen dieses Quadranten vor allem um *Entwicklungsmethodologien* handelt. Ein Multiagentensystem wird hier vollständig „aus einem Guss“ geplant, entworfen und implementiert. Die Autonomie eines Agenten bezieht sich insofern ausdrücklich nicht auf Entscheidungsfreiheit bezüglich seiner Mitgliedschaft in der Organisation oder bezüglich seiner konkreten organisatorischen Einbettung, sondern lediglich darauf, wie er seine jeweilige Organisationsfunktion in einem konkreten Anwendungskontext erfüllen kann. So etwas wie eine informelle Struktur zwischen den Agenten einer Organisation (analog zum Widerspiel von Formalstruktur und informeller Struktur in realweltlichen Organisationen) kann es hier also gar nicht geben. Hier ergibt sich auch, dass die Agenten nicht länger „die Organisation sind“. Ein konkreter Agent kann ausgetauscht werden, während seine Rolle mitsamt den an sie geknüpften Spezifikationen weiter existiert und einfach von einem anderen Agenten eingenommen wird.

Als Übergang zwischen Quadrant (c) und (d) können die Entwicklungsansätze TROPOS (vergleiche Bresciani et. al. [BPG+04]) und OMNI/OperA (vergleiche Dignum et. al. [DVSD04]) verstanden werden. Gegenüber GAIA und MaSE wird hier ein reicheres Reservoir an Organisationskonzepten für Planung, Entwurf und Modellierung von Multiagentensystemen bereitgestellt. Darüber hinaus werden Organisationskonzepte speziell im Falle OMNI/OperA auch zur Laufzeit unterstützt, beispielsweise in Form von Mechanismen für die Besetzung von Rollen und die Durchsetzung von Normen. Gegenüber den Ansätzen aus Quadrant (d) ist die analytische Trennung zwischen Mitgliedern und Organisationsinfrastruktur jedoch weniger klar.

Quadrant (d) sind Ansätze zuzuordnen, bei denen die formale Organisationsspezifikation sowohl als Entwicklungs- als auch als Laufzeitkonzept geführt wird. Agenten haben eine interne Repräsentation der Organisationsspezifikation (beziehungsweise zumindest eines Teils davon) und nutzen diese, um über ihren jeweiligen organisatorischen Kontext zu räsonieren. Der wesentliche Unterschied zu Ansätzen des Quadranten (b) besteht darin, dass zusätzlich zu diesen internen Repräsentationen der einzelnen Agenten eine zusätzliche externe und globale Repräsentation der Organisation im Softwaresystem existiert. Die Organisation wird *softwaretechnisch reifiziert* und erhält damit eine eigene (softwaretechnische) Existenz unabhängig von konkreten Agenten als Mitgliedern. Zu diesem Zweck sind in Abbildung 5.2 zusätzlich zu den Mitgliedsagenten der Organisation *Artefakte der Organisationsinfrastruktur* aufgeführt. Sie sorgen für eine Manifestation der formalen Organisation im Multiagentensystem. Auch hier gilt also, dass Agenten nicht „die Organisation sind“. Allerdings ist dieser Aspekt hier noch stärker ausgeprägt als bei Ansätzen des Quadranten (c). Da die formale Organisation als

[3] Hier ist jeweils ein Artikel zur „zweiten Generation“ der beiden Ansätze zitiert. Diese gehen bereits mit einer stärkeren Betonung der Organisationsperspektive einher, als es in den ursprünglichen Versionen von GAIA nach Wooldridge et. al. [WJK00] und MaSE nach DeLoach [DWS01] der Fall ist. Damit ist auch ein höheres organisatorisches Bewusstsein der entwickelten Agenten verbunden, so dass bereits eine Annäherung an Quadrant (d) aus Abbildung 5.2 zu beobachten ist.

Softwarekonstrukt zur Laufzeit existiert, müssen Agenten nicht von vorneherein auf die Erfüllung ihrer spezifischen Organisationsfunktion hin programmiert werden. Stattdessen wird für Ansätze dieses Quadranten häufig eine *offene Multiagentensystemperspektive* propagiert. Agenten (verschiedener Entwickler) können das System betreten und verlassen. Entschließt sich ein Agent, der Organisation beizutreten, so kann er zur Laufzeit Zugang zu der für ihn relevanten Organisationsspezifikation erhalten und diese inkorporieren. Natürlich stellt sich hier die Frage, inwiefern von Organisationsseite aus sichergestellt werden kann, dass Mitgliedsagenten ihrer jeweiligen Organisationsfunktion tatsächlich nachkommen. Dies hängt wiederum mit der allgemeineren Frage zusammen, wie die Reifizierung der formalen Organisation in Form einer Organisationsinfrastruktur eigentlich aussieht.

Insgesamt lässt sich festhalten, dass Ansätze des Quadranten (d) die für die vorliegende Arbeit interessanteste und relevanteste Variante von Organisationsansätzen für Multiagentensysteme aufspannen. Hier sind Organisationen sowohl zur Entwicklungs- als auch zur Laufzeit expliziert. Dies unterstützt eine unmittelbare Behandlung von Mikro-Makro-Dynamiken und das „Zulassen“ von informellen Strukturen neben der Formalstruktur einer Organisation. Die softwaretechnische Reifizierung der Formalanteile von Organisationen erlaubt darüber hinaus die Behandlung von Organisationen als konkrete und eigenständige Softwareentitäten. Daher wird den Ansätzen dieses Quadranten zusätzlich zu den bisherigen Ausführungen der folgende Teilabschnitt für eine ausführlichere Betrachtung gewidmet.

5.2.2 Organisationsorientierte Programmierung in Multiagentensystemen

Boissier et. al. [BHS07] charakterisieren Ansätze des Quadranten (d) als Ansätze des *organisationsorientierten Programmierens* von Multiagentensystemen. Der Fokus verschiebt sich also von der softwaretechnischen Entwicklung der Agenten zur softwaretechnischen Entwicklung der Organisation. Hübner et. al. [HBKR10] stellen die Gemeinsamkeiten aller diesbezüglichen Ansätze in Form einer *allgemeinen Architektur für Organisationsinfrastrukturen in offenen Multiagentensystemen* heraus. Diese Architektur ist in Abbildung 5.3 zu sehen.

Die Architektur spezifiziert eine Organisationsinfrastruktur in Form einer dreischichtigen *Middleware*. Domänen-/Anwendungsagenten, die als Organisationsmitglied fungieren, benötigen Zugang zu einer *Proxykomponente* und Berechtigung zu deren Benutzung. Nur auf diese Weise ist es möglich, als Mitglied der Organisation aufzutreten und mit eigenen Handlungen Effekte auf den *Organisationszustand* ausüben zu können. Der Organisationszustand wird von der Middleware explizit verfolgt und überwacht. Er setzt sich aus der *Organisationsspezifikation* und den Ausprägungen dieser Spezifikation zusammen. Die Ausprägungen manifestieren sich zum einen in der Besetzung bestimmter Organisationsfunktionen (meist beschrieben mittels organisatorischer Rollen oder Positionen) mit Domänenagenten, zum anderen in der Verfolgung bestimmter Organisationsereignisse (beispielsweise Teamformation oder gemeinsamer Planausführung) und der Belegung bestimmter Organisationsvariablem mit konkreten Werten (beispielsweise einer Liste aktiver Teams oder instanziierter Pläne). Damit organisationsrelevante Aktivitäten der Mitgliedsagenten reibungslos und in Konformanz mit den Organisationsspezifikationen ablaufen, werden bestimmte *Organisationsdienste* direkt von der Middleware angeboten und können/müssen von den Domänenagenten (vermittelt über den Umweg ihres jeweiligen Proxys) genutzt werden. Um welche Dienste es sich konkret handelt, hängt vom jeweils unterstützten Organisationsansatz ab. Im Wesentlichen geht es um Unterstützung/Beschränkung von Koordination zwischen Mitgliedern, die Zuweisung von Rollen an Mitglieder und auf höherer Ebene auch um Reorganisation und Sanktionen.

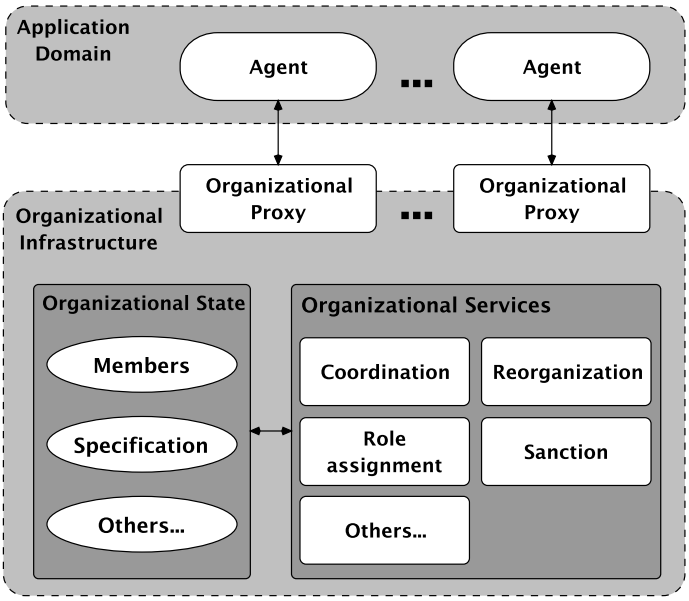

Abbildung 5.3: Allgemeine Architektur für Organisationsinfrastrukturen in offenen Multiagentensystemen nach Hübner et. al. [HBKR10]

Ansätze des organisationsorientierten Programmierens in Multiagentensystemen bringen immer zwei Dinge mit sich, nämlich (1) eine Technik oder Sprache für die Modellierung/Spezifikation einer Agentenorganisation und (2) eine Organisationsinfrastruktur als Ausprägung des soeben beschriebenen allgemeinen Konzeptes. Beides muss aufeinander abgestimmt sein und es existieren eine Reihe konkreter Ansätze für die organisationsorientierte Programmierung von Multiagentensystemen.

- $\mathcal{M}$OISE$^+$ nach Hübner et. al. [HSB02] bietet die Möglichkeit, Organisationsstrukturen im Form von Rollenbeziehungen zu beschreiben und diese mit Verantwortlichkeiten für die Ausführung von gemeinsamen Plänen zu assoziieren. $\mathcal{M}$OISE$^+$-Spezifikationen werden durch die $\mathcal{S}$-$\mathcal{M}$OISE$^+$-Middleware nach Hübner et. al [HSB04] unterstützt. Allerdings existiert mittlerweile auch die neuere Middleware-Unterstützung ORA4MAS nach Hübner et. al. [HBKR10]. Sie basiert auf dem *Agents&Artifacts*-Ansatz nach Omicini et. al [ORV08] und die Organisation wird dabei in Form von Material-/Werkzeugartefakten, Kommunikationsartefakten sowie institutionellen Artefakten verkörpert.

- ISLANDER nach Esteva et. al. [EdlCS02] ist ein Editor, der eine spezielle Modellierungstechnik für *elektronische Institutionen* von Agenten bietet. Dabei werden einzelne Interaktionsszenarien modelliert und mittels einer übergeordneten Routinginfrastruktur und zusätzlichen Normen verknüpft. ISLANDER-Modelle werden durch die AMELI-Middleware nach Esteva et. al. [ERARA04] unterstützt.

- Der AGR-Ansatz nach Ferber et. al. [FGM03] erlaubt eine minimalistische, rein strukturorientierte Modellierung von Agentenorganisationen in Form von Gruppen, Rollen und Agenten. AGR-Spezifikationen werden durch die MadKit-Agentenplattform nach Gutknecht und Ferber [GF00] unterstützt.

- Das TEAMCORE-Framework nach Pynadath und Tambe [PT03] bietet einem (menschlichen) Benutzer eine grafische Oberfläche für die Spezifikation von *teamorientierten Programmen* für Agenten, bestehend aus einer Organisation und ihren Teamplänen (*team-oriented programming interface*, TOPI). Die Spezifikationen werden an die Middleware-Komponente KARMA weitergeleitet. Diese koordiniert in der Folge (gegebenenfalls unter weiterem Einbezug des menschlichen Benutzers) die Teamaktivitäten.

- SONAR nach Köhler-Bußmeier [Köh06, KB09] ist ein mathematischer Ansatz der Beschreibung von Organisationen auf der Basis von Petrinetzen. Dabei werden Aufgabenhierarchien, Netzwerke aus organisatorischen Positionen für die Verteilung der Aufgaben und Interaktionen für die verteilte Ausführung von Aufgaben beschrieben. Aus der SONAR-Spezifikation einer Organisation kann direkt eine für diese Organisation spezifische Middleware-Komponente generiert werden, die im Kontext des allgemeinen Middleware-Prototypen MULAN4SONAR nach Köhler-Bußmeier und Wester-Ebbinghaus [KBWEM10, KBWE09b] eingesetzt wird. Die MULAN4SONAR-Middleware basiert auf dem in der Einleitung der vorliegenden Arbeit skizzierten und in Kapitel 6 näher erläuterten Agenten-Framework MULAN/CAPA.

Es handelt sich hierbei nur um eine sehr grobe Skizze der jeweiligen Ansätze. In Kapitel 10 der vorliegenden Arbeit werden drei der Ansätze ($\mathcal{M}$OISE$^+$/$\mathcal{S}$-$\mathcal{M}$OISE$^+$, ISLANDER/AMELI, SONAR/MULAN4SONAR) herausgegriffen und unter Zuhilfenahme der im zweiten Teil der Arbeit erstellten Modelle ausführlicher betrachtet.

Wie die Organisationsinfrastruktur in den jeweiligen Ansätzen tatsächlich technisch implementiert ist, variiert. Teilweise wird auch hier Agententechnologie eingesetzt. So sind zumindest die Proxys in der Regel über Agenten realisiert. Dazu kommen je nach Ansatz verschiedene *Manageragenten*, die für die Kapslung und Wahrung der Organisationsspezifikationen verantwortlich sind und die jeweiligen Organisationsdienste anbieten (vergleiche hierzu die nähere Betrachtung in Kapitel 10).

Die softwaretechnische Verkörperung einer Agentenorganisation über eine Infrastrukturschicht dient insbesondere dem Zweck, heterogene Agenten eines offenen Multiagentensystems in Organisationen einbinden zu können. Die Heterogenität hat dabei sowohl einen technischen als auch einen (künstlichen) sozialen Aspekt. Im technischen Sinne stellt ein Proxy ein Interface bereit, über das ein Mitgliedsagent Zugriff auf die Organisation erhält. Über den inneren Aufbau eines Mitgliedsagenten in Form von Agentenarchitektur, Wissensrepräsentation und Schlussfolgerungsmechanismen werden dabei keinerlei Annahmen gemacht. Wichtig ist nur, dass der Agent in der Lage ist, mit seinem Proxy zu interagieren.

Der soziale Aspekt der Heterogenität bezieht sich auf das „organisatorische Bewusstsein“ der Mitgliedsagenten. Sie können beispielsweise schlicht auf die blinde Ausführung bestimmter organisatorischer Aufgaben hin entwickelt werden. Sie können aber stattdessen auch derart elaboriert sein, die Organisationsspezifikation interpretieren und gegen ihre eigenen Ziele abwägen zu können (sofern die Organisationsspezifikationen zugänglich sind). Damit ist die Grundlage für ein Wechselspiel zwischen Formalstruktur der Organisation und informellen Beziehungen gegeben. Selbst wenn die Mitgliedsagenten durch die Infrastrukturtrennung in ihrem unmittelbaren Einfluss auf die Organisation eingeschränkt sind (analog der beschränkenden Wirkung einer Formalstruktur in realweltlichen sozialen Organisationen), so können sie doch zusätzlich in einer Weise interagieren (kooperieren oder konkurrieren), die von Seiten der Organisation weder vorgesehen ist noch zur Laufzeit zwangsläufig wahrgenommen wird.

Die bisherigen Ausführungen machen deutlich, dass die mögliche Heterogenität der Mitgliedsagenten soweit gehen kann, auch realweltliche/soziale Akteure einzuschließen. Ihnen muss lediglich eine Schnittstelle zur Kommunikation mit den Proxys verfügbar sein. So ergibt sich die Möglichkeit einer unmittelbaren sozio-technischen Integration.

Eine sich herauskristallisierende informelle Struktur hat automatisch Einfluss auf die Funktionsweise der Organisation. Darüber hinaus kann sie ein wesentlicher Faktor im Hinblick auf die Evolution der Organisation in Form von dynamischer Reorganisation zur Laufzeit sein. Inwiefern Reorganisation überhaupt zugelassen wird, variiert von Ansatz zu Ansatz. Boissier et. al. [BHS07] differenzieren zwischen Ansätzen, die Reorganisation (1) nicht zulassen, (2) exogen zulassen (durch eine Instanz verschieden von den eigentlichen Mitgliedsagenten) oder (3) endogen zulassen (durch die Mitgliedsagenten selber). Ansätze, die das Konzept einer softwaretechnisch verkörperten Organisation mit der Möglichkeit einer endogenen Reorganisation kombinieren, erfüllen genau die beiden Bedingungen, die Schillo et. al. [SFF+02] an ein soziologisch korrektes Konzept der *Selbstorganisation* in Multiagentensystemen stellen. (1) Es existieren (formale) soziale Strukturen, die unabhängig von der Existenz einzelner individueller Agenten bestehen und die die Aktionen eingebundener Agenten rahmen (beschränken, ermöglichen). (2) Die Entstehung, Evolution und Adaption dieser Strukturen resultiert aus den Aktionen der eingebundenen Agenten, allerdings müssen die Strukturen nicht notwendigerweise die Ziele einzelner Agenten reflektieren. Auf diese Weise wird die Organisation zur Ressource ihrer Mitglieder, die im Rahmen ihrer Möglichkeiten versuchen, eigene Ziele und Interessen auf die Organisation zu übertragen.

5.2.3 Bewertung der Organisationsorientierung in Multiagentensystemen

In Kapitel 3 der vorliegenden Arbeit wurden vier Hauptargumente für ein Organisationsparadigma der Softwareentwicklung vorgetragen: (1) Organisationen als Ordnungsstifter, (2) Organisationen als inhärent sozio-technische Systeme, (3) Organisationen als Meso-Strukturen, (4) Organisationen als offene Systeme und kollektive Akteure. Mit diesen vier Argumenten wurden auch die wesentlichen Funktionen der Organisationsmetapher für die Softwareentwicklung verknüpft. Es gilt nun zu beleuchten, inwiefern die immer dominanter werdende Organisationsperspektive in der Multiagentensystemforschung bereits dazu beigetragen hat, dass diese Funktionen auch tatsächlich genutzt werden.

Die Erläuterungen zur Entwicklung der Organisationsperspektive in der Multiagentensystemforschung in Unterabschnitt 5.1.3 machen deutlich, dass gerade die Funktion von Organisationen als Ordnungsstifter überhaupt erst die treibende Kraft hinter den Bemühungen für die Betrachtung von Agentenorganisationen war. Insofern wird diese Funktion in existierenden Ansätzen auch vollständig ausgenutzt. Dies gipfelt in einer expliziten softwaretechnischen Verkörperung und Unterstützung von Organisationsspezifikationen als Middleware-Schicht.

Diese Middleware-Ansätze sind es auch, die der Formalisierung einer Organisation eine erstrangige Rolle einräumen und damit das Wechselspiel von Formalstruktur und informellen Strukturen überhaupt erst softwaretechnisch ermöglicht/unterstützt. Damit wird einerseits die Etablierung von Organisationsstrukturen in offenen Multiagentensystemen mit heterogenen, dynamisch/unvorhergesehen auftretenden, aus verschiedenen Quellen stammenden und gegebenenfalls informelle Beziehungen unterhaltenden *Softwareagenten* ermöglicht. Andererseits wird aber auch die Integration einer softwaretechnischen Organisation und *realweltlichen/natürlichen sozialen Akteuren* unterstützt. Natürliche soziale Akteure können ebenso wie künstliche soziale Akteure als Mitglieder einer Softwareorganisation auftreten. Da die

Konzepte und Mechanismen auf Softwareseite deckungsgleich zu den realweltlichen Konzepten und Mechanismen sind, haben die natürlichen sozialen Akteure einen intuitiven Zugang zu ihrer Rolle gegenüber dem Softwaresystem beziehungsweise zu ihrer Stellung im Kontext der künstlichen Organisation. Auch die Organisationsfunktion der sozio-technischen Integration wird in der aktuellen Multiagentensystemforschung also genutzt.

Nach Ansicht des Autors der vorliegenden Arbeit treten hinsichtlich der beiden Funktionen von Organisationen als Meso-Strukturen und als kollektive Akteure Defizite bei der Nutzung durch die Multiagentensystemforschung zutage. Einerseits werden Agentenorganisationen durchaus als Meso-Strukturen zwischen der Ebene der individuellen Agenten und der globalen Ebene des gesamten Multiagentensystems verwendet. Andererseits hat man es dann mit lediglich drei Systemebenen zu tun. Bestenfalls können weitere Ebenen wie Teams oder Gruppen innerhalb einer Agentenorganisation identifiziert werden. Diese werden allerdings vornehmlich als Variablen der internen Strukturierung einer Agentenorganisation verwendet und stellen keine eigenständig umgesetzten Systemebenen dar. Die Erläuterungen in den Kapiteln 3 und 4 der vorliegenden Arbeit unterstreichen dagegen deutlich, dass eine organisationsorientierte Perspektive auf soziale Systeme multiple Meso-Ebenen mit sich bringt. Diese manifestieren sich in organisierten Kollektivitäten wie Abteilungen, Unternehmen, virtuellen Organisationen, strategischen Allianzen, Märkten oder Industrien. Es fällt insbesondere auf, dass diese Kollektivitäten teilweise *oberhalb* der Ebene der klassischen Organisation im Sinne eines Unternehmens anzusiedeln sind. Damit einher geht also das Merkmal von Organisationen (und weiteren organisierten Kollektivitäten) als offene Systeme. Diese Offenheit bezieht sich weniger darauf, dass Mitglieder der Organisation beitreten und sie verlassen (dieser Aspekt wird innerhalb der Multiagentensystemforschung zur Genüge abgedeckt), sondern auf das kollektive Akteurswesen von Organisationen. Soziale Organisationen sind sowohl Kontexte als auch Akteure. Nur unter Einbezug dieses janusgesichtigen Charakters ist es möglich, verschieden abstrakte Meso-Ebenen systematisch zu unterscheiden und in Beziehung zu setzen.

Gerade dieser so wichtige Aspekt spielt in der Multiagentensystemforschung allerdings bislang eine untergeordnete Rolle. Organisationen werden vorrangig als Kontexte für ihre Mitglieder aufgefasst. In dieser Funktion treten sie durchaus als „nach innen gerichtete“ Akteure auf, indem explizite Middleware-Schichten eingezogen werden und Mitgliederverhalten beschränkt beziehungsweise unterstützt wird. Agentenorganisationen als „nach außen gerichtete“ Akteure, die auf höheren Systemebenen agieren, werden jedoch kaum behandelt. Nach Ansicht des Autors der vorliegenden Arbeit bedeutet gerade dieser Schritt von der Betrachtung organisatorisch eingebetteter individueller Akteure zur Betrachtung organisierender und gleichzeitig wiederum organisatorisch eingebetteter kollektiver Akteure einen *Paradigmenwechsel von der Agenten- zur Organisationsorientierung*. Solange dieser Schritt nicht vollzogen ist, bewegt man sich weiterhin im Paradigma der Agentenorientierung. Dies ist in Übereinstimmung zu der bereits in Kapitel 3 zitierten Auffassung von Scott [Sco03, S. 7], das Konzept der realweltlichen Organisation sei nur teilweise begriffen, solange man sie lediglich als Kontext für individuelle Akteure, nicht jedoch als Akteure in sich betrachte.

Es sei allerdings darauf hingewiesen, dass entgegen der hier geäußerten Kritik durchaus einige Ansätze und Gedanken zu Agentenorganisationen als kollektiven Akteuren vorgebracht wurden. So präsentieren beispielsweise Hahn et. al. [HMMF08] ein Meta-Modell für Multiagentensysteme, in dem eine Agentenorganisation als kollektiver Akteur spezifiziert wird. Allerdings geschieht dies einfach dadurch, indem das Konzept *Organisation* als sowohl vom Konzept *Kooperation* als auch vom Konzept *Agent* erbend spezifiziert wird. DeLoach [DeL05] wählt eine ähnliche Herangehensweise. In seinem Meta-Modell für Multiagentensysteme wird

das Konzept *Organisationsagent* als von den Konzepten *Agent* und *Organisation* erbend angegeben. In beiden Fällen bleibt unklar, wie sich die Transition von einem ordnenden Kontext zu einem eigenständigen Akteur vollziehen soll. Als Ergänzung existieren allerdings Ansätze, die eine technische Realisierung kollektiver Akteure aufzeigen/ermöglichen. Rölke [Röl04, S. 154–157, S. 181–183] skizziert das Konzept von Agentenplattformen, die selber Agenten sind (Plattformagenten). Fischer et. al. [FSS03] propagieren den föderativen Ansatz der *holonischen Multiagentensysteme* (angelehnt an das bereits in Kapitel 4 skizzierte Holonenkonzept nach Koestler [Koe67]). Eine Menge von Agenten – der *Körper* des Holons – wird dabei von einem oder wenigen Agenten – dem *Kopf* des Holons – nach außen hin vertreten. Auf diese Weise kann ein Agenten-Holon im Kontext anderer Agenten-Holone agieren. Ähnlich gelagert ist der von Stockheim et. al. [SNSS04] vorgeschlagene föderative Ansatz für *Multi-Multiagentensysteme*, bei denen mehrere Multiagentensysteme durch Gateway-Agenten verknüpft werden. Ein ausgereifter Ansatz für kollektive Akteure in Multiagentensystemen wird durch das *JACK Intelligent Agents* Framework (http://aosgrp.com/products/jack/index.html) und insbesondere durch die Erweiterung JACKTeams (http://aosgrp.com/products/jackteams/index.html) bereitgestellt (ein Überblick wird beispielsweise in Kapitel 15 des Buches *Agentenorientierte Softwareentwicklung* von Weiß und Jakob [WJ05] geboten). JACKTeams erweitert JACK unter anderem um die Konzepte *Team, Rolle* und *Teamplan*. Ein Team ist dabei eine ausgeprägte Entität, die im Wesentlichen durch Rollen charakterisiert ist, nämlich (externe) Rollen, die das Team selber einnimmt, und (interne) Rollen, deren Besetzung durch Andere vom Team benötigt wird. Die Formation eines Teams geschieht durch die Assoziation von Subteams mit internen Rollen des gerade betrachteten Teams. Da diese Subteams gegebenenfalls ihrerseits mit internen Rollen assoziiert sind, entsteht eine Hierarchie von Teams. Diese Hierarchie muss nicht eindeutig sein: Ein Team kann Rollen in verschiedenen anderen Teams einnehmen.

Der weiter oben aufgezeigte Missstand, die Transition vom Konzept der Agentenorganisation als Kontext zum Konzept der Agentenorganisation als kollektiver Akteur sei unklar, wird durch die genannten Ansätze nur bedingt geschmälert. Die hier zitierten Ansätze zeigen deutlich: In der Multiagentensystemforschung existieren Ansätze für eine *technische Realisierung* kollektiven Akteurswesens. Plattformagenten, holonische Multiagentensysteme und die Möglichkeit, Teams Rollen einnehmen zu lassen, zeigen Wege für die Implementierung kollektiver Akteure auf. Fragen zur *inhaltlichen Realisierung* kollektiven Akteurswesens bleiben jedoch offen. In der Agentenorientierung wird die Thematik, wie Integration und Steuerung einer Agentenorganisation zum systematischen Grenzmanagement und dem kollektiven Verhalten der Organisation „als Ganzem“ in Beziehung stehen, nicht originär behandelt. Ebenso wenig befasst sich die Agentenorientierung mit der Differenzierung verschiedener Ebenen des kollektiven Handelns und deren Verhältnis zueinander. Beide Aspekte werden jedoch im Studium sozialer Gesellschaften mit Organisationen als ihren primären sozialen Akteuren als von zentraler Bedeutung angesehen. Ansätze der „Agentifizierung“ von Agentengruppierungen und deren Auftreten als Agent in wiederum anderen Agentengruppierungen können als Versuche begriffen werden, agierende Kollektivitäten in das Paradigma der Agentenorientierung zu transferieren. Nach Ansicht des Autors der vorliegenden Arbeit ergibt sich hier eine Analogie zu der in Abschnitt 5.1 behandelten Verwischung der Grenzen zwischen Objekt- und Agentenorientierung durch „modernere“ Auffassungen des Objektkonzeptes. Die Rechtfertigung für die Organisationsorientierung als eigenständiges Softwareentwicklungsparadigma ergibt sich nun analog zur Rechtfertigung der Agenten- gegenüber der Objektorientierung: Die Organisationsorientierung führt eine neuartige Ebene der Systemabstraktion ein und bietet auf diese Weise für bestimmte softwaretechnische Problemstellungen eine höhere „konzeptionelle Pas-

sung“ gegenüber der Agentenorientierung. Die softwaretechnischen Fragestellungen sind dabei genau die in Abschnitt 2.2 im Zusammenhang mit Softwaresystemen von Softwaresystemen genannten.

Nach Ansicht des Autors der vorliegenden Arbeit stellt der Übergang von der agenten- zur organisationsorientierten Softwareentwicklung demnach den nächsten logischen Schritt dar. Weiter oben wurden Ansätze des Quadranten (d) aus Abbildung 5.2 als Ansätze des *organisationsorientierten Programmierens von Multiagentensystemen* charakterisiert. Damit hat sich die Agentenorientierung einem eigenständigen Paradigma der Organisationsorientierung bereits stark genähert.

5.3 *Viable System Model* und *Viable Software*

Der *Viable Software*-Ansatz nach Herring [Her02] baut auf dem *Viable System Model* von Beer [Bee79, Bee84] auf. Das *Viable System Model* ist ein systemtheoretisches Modell für die Organisation von „lebensfähigen“ Systemen. Es handelt sich um ein kybernetisches Modell, dass speziell in seiner Anwendung auf den Bereich der sozialen Organisationen große Aufmerksamkeit erlangt hat. Allgemein gesprochen ist ein lebensfähiges System auf eine Art und Weise organisiert, den Anforderungen des Überlebens in einer dynamischen, sich verändernden Umwelt gerecht zu werden. Eine der wichtigsten Eigenschaften eines lebensfähigen Systems ist demnach seine Adaptionsfähigkeit. Im Kontext des *Viable System Model* werden fünf interagierende Teilsysteme für ein lebensfähiges System gefordert. Vereinfacht dargestellt beschäftigen sich die Systeme 1 - 3 mit dem aktuellen operativen Betrieb des Systems, System 4 mit strategischen Antworten auf externe Umwelteinflüsse und Zukunftsplanung und System 5 mit übergreifenden und allgemeingültigen Richtlinien und Gesetzen des Gesamtsystems.

Herring überträgt mit seinem *Viable Software*-Ansatz die Prinzipien und Mechanismen des *Viable System Model* auf die Entwicklung (speziell den Entwurf) von Softwaresystemen. Das Ziel ist dabei ein „intelligenter Ansatz der Steuerung“ durch die Migration von menschlichen Managementfähigkeiten in Softwaresysteme, die dadurch über die Zeit gesehen selber intelligente Selbstkontrolle und -steuerung ausüben sollen. Entlang der fünf Teilsysteme eines lebensfähigen Systems nach Beer führt Herring sein (UML-)Modell eines lebensfähigen Softwaresystems in fünf Schritten ein.

1. Im ersten Schritt erfolgt die klassische kybernetische Trennung der Zuständigkeiten in steuernden (*Controller*) und gesteuerten Anteil (*Process*) des Gesamtsystems. Ein menschlicher Akteur füttert die Steuerung mit Anweisungen. Die Steuerung implementiert die Anweisungen und überwacht deren Ausführung durch den gesteuerten Anteil über einfache Rückkopplung. In diesem und in allen folgenden Schritten ist außerdem die Umgebung des Systems repräsentiert.

2.+3. In den Schritten Zwei und Drei erfolgt eine erste Elaborierung der Steuerung mittels expliziter Regulierungs- und Auditorinstanzen (*Regulator, Auditor*).

4. Im vierten Schritt wird der Steuerung eine weitere Instanz vorgeschaltet. Anstatt konkrete Anweisungen an die Steuerung zu übermitteln, formuliert der menschliche Benutzer nun Regeln in abstrakterer Form an eine adaptive Steuerung (*Adaptive Controller*). Diese übersetzt die Regeln kontextabhängig in konkrete Anweisungen für die Steuerung.

5. Im letzten Schritt wird der adaptiven Steuerung eine weitere Instanz vorgeschaltet. Es handelt sich dabei um eine Art Hauptsteuerprogramm (*Supervisor*), dem der menschliche Benutzer gewünschtes Systemverhalten auf *deklarative* Art und Weise in Form von *Policies* übermittelt. Diese wiederum werden in Regeln für die adaptive Steuerung transformiert.

Herring übernimmt das Konzept der Rekursivität von Beers *Viable System Model*. Softwaresysteme nach dem *Viable Software*-Ansatz setzen sich aus sogenannten *Viable Components* zusammen. Eine solche Komponente enthält interne Komponenten entsprechend der weiter oben beschriebenen Aufteilung. Die interne Komponente, die den niedrigrangigen gesteuerten Teil verkörpert (*Process*), kann dabei wiederum aus *Viable Components* bestehen. Die beschriebene Rolle des menschlichen Akteurs, der eine *Viable Component* mit Anweisungen füttert, kann in einem rekursiv aufgebauten System auch vom Hauptsteuerprogramm (*Supervisor*) der einbettenden *Viable Component* übernommen werden.

Im Gegensatz zu den im vorigen Abschnitt vorgestellten rekursiven Ansätzen aus dem Multiagentensystembereich (Holone, JACKTeams) erfolgt für den Fall einer *Viable Component* eine explizite Berücksichtigung der Konzepte Umgebung und Einbettung. Interne Steuerungsstrukturen einer *Viable Component* werden systematischer mit der Umgebung und übergreifenden Steuerungsstrukturen in Bezug gesetzt. Demgegenüber steht der Nachteil, dass es sich beim unterliegenden *Viable System Model* mit seinen fünf Teilsystemen bereits um ein recht spezifisches Modell handelt und damit ein recht enger Rahmen für die Ausgestaltung konkreter Ausprägungen vorgegeben wird. Ein weiterer Nachteil besteht nach Ansicht des Autors der vorliegenden Arbeit darin, dass lediglich für eines der fünf Teilsysteme einer *Viable Component* die Möglichkeit der rekursiven Verfeinerung vorgesehen ist. In sozialen Organisationssystemen können auch steuernde/kontrollierende Akteure kollektiver Natur sein (wie beispielsweise im Fall einer Regulierungsbehörde auf Ebene eines organisatorischen Feldes).

5.4 Betriebliche Softwaresysteme

Der Aspekt der Softwaresysteme *für* Organisationen soll hier nicht unerwähnt bleiben. Zum einen wurde bereits in der Einleitung der Arbeit die Hypothese aufgestellt, Softwaresysteme *als* Organisationen seien auch der sozio-technischen Integration von Softwaresystemen *für* Organisationen zuträglich. Auch wenn diese Thematik im Kontext der vorliegenden Arbeit nicht vertieft wird, so ergibt sich hier zumindest ein wichtiger Aspekt zukünftiger Forschung. Zum anderen stellt sich die Frage, ob Softwaresysteme *für* Organisationen nicht automatisch Strukturen und Verhalten mit sich bringen, die sie in den Bereiche der Softwaresysteme *als* Organisationen rücken.

5.4.1 (IT-)Anwendungslandschaft und Geschäftsarchitektur

Mit Softwaresystemen *für* Organisationen sind hier betriebliche Softwaresysteme für IT-anwendende realweltliche Organisationen gemeint. Die zugehörige Literatur ist umfangreich und die verwendete Terminologie nicht immer eindeutig. Die Ausführungen des vorliegenden Abschnittes beziehen sich vor allem auf die beiden Konzepte der (1) *(IT-)Anwendungslandschaft* (*Enterprise Information System*) und der (2) *Geschäfts-/Unternehmensarchitektur* (*Enterprise Architecture*). Zwei komplette Handbücher

zu den beiden Themen wurden beispielsweise von Bernus et. al. als Editoren [BNS03, BMS06] herausgegeben.

Mit einer Anwendungslandschaft bezeichnen Humm et. al. [HVH06] die Gesamtheit der betrieblichen IT-Anwendungen eines Unternehmens inklusive ihrer Vernetzung über Schnittstellen und Daten. Die Anwendungslandschaft dient der Unterstützung des *Geschäfts* eines Unternehmens. Dieses wiederum wird durch die Geschäftsarchitektur des Unternehmens beschrieben. Die Geschäftsarchitektur umfasst dabei (unter anderem) die Geschäftsstrategie, die Geschäftsprozesse und die Informationen, die im Rahmen der Geschäftsprozesse benötigt und erzeugt werden. Matthes [Mat08, S. 528] legt einen stärkeren Fokus auf das Wesen von Anwendungslandschaften in Unternehmen und beschreibt sie als *„langlebige hoch-komplexe Strukturen bestehend aus hunderten bis tausenden von miteinander vernetzten betrieblichen Informationssystemen, die von Personen mit sehr unterschiedlichen Interessen und Erfahrungshintergrund konzipiert, erstellt und modifiziert, betrieben, genutzt und finanziert werden“*. Darüber hinaus charakterisiert Matthes Anwendungslandschaften explizit als Systeme von Systemen. Hier wird also deutlich, dass es sich bei Anwendungslandschaften um einen Vertreter der Systemkategorie handelt, die in der vorliegenden Arbeit adressiert wird.

Verschiedene Ansätze der Beschreibung von Geschäftsarchitekturen sind entwickelt worden. Als Beispiele mögen hier CIMOSA (*Computer Integrated Manufacturing Open System Architecture*) des ESPRIT Consortiums AMICE [ESP93], ARIS (*Architektur integrierter Informationssysteme*) nach Scheer [Sch02] oder das auf Basis der gemeinsamen Eigenschaften verschiedener Ansätze abgeleitete Meta-Rahmenwerk GERAM der *IFIP–IFAC Taskforce on Architectures for Enterprise Integration* [IFI03] dienen. Humm et. al. [HVH06] betonen vor allem die folgenden Aspekte einer Geschäftsarchitektur: *Geschäftsstrategie*, *Geschäftsdomänen*, *Geschäftsprozesse* inklusive der Akteure in ihren verschiedenen Rollen, *Geschäftsfunktionen* und *Geschäftsobjekte*.

Die Geschäftsarchitektur bildet den Ausgangspunkt für die Gestaltung der Anwendungslandschaft. Unter *Business-IT-Alignment* wird die gegenseitige Ausrichtung von Geschäft und IT beziehungsweise die verzahnte Entwicklung von Software und (realweltlicher) Organisation verstanden (vergleiche Matthes [Mat08]). Der Zweck ist eine geeignete Unterstützung (bei manueller Beteiligung) beziehungsweise Realisierung (bei Vollautomatisierung) von Geschäftsprozessen durch die Anwendungslandschaft. Um das Business-IT-Alignment zu unterstützen wird vielfach verstärkt auf Visualisierung sowohl von realweltlichen als auch von softwaretechnischen Zusammenhängen und Prozessen zurückgegriffen. Dies hilft bei der Überwindung inhaltlicher und terminologischer Hürden zwischen Fach- und IT-Abteilungen. Beispiele sind Softwarekartographie nach Matthes [Mat08] oder die Visualisierung von Projektportfolios nach Schirmer und Zimmermann [SZ08].

5.4.2 Dienstorientierte Architekturen

Im Zuge eines Business-IT-Alignment steht die Unternehmens-IT vor der Herausforderung, stets neue Geschäftsprozesse unterstützen und auf Änderungen der Unternehmensstruktur flexibel reagieren zu können. Dabei kann eine Anwendungslandschaft beispielsweise durch große monolithische Systeme geprägt sein, wobei ein monolithisches System die Unterstützung einer Menge von Geschäftsprozessen vollständig abdeckt. Die Flexibilität dieser monolithischen Systeme bestimmt folglich die Flexibilität der Geschäftsprozesse. Übergänge zwischen Geschäftsprozessunterstützungen verschiedener Systeme müssen durch proprietäre Verbindungen realisiert werden. Diese bedingen häufig eine zu enge Kopplung der Systeme, Probleme bei

der Wiederverwendung von Schnittstellen, mangelnde Kapselung fachlicher Kompetenzen und Unklarheit über Datenhoheit.

Mit Ansätzen basierend auf *dienstorientierten Architekturen (service-oriented architcture, SOA)* wird in der Regel das Versprechen verbunden, eine angemessenere Herangehensweise zu bieten. Stellvertretend wird hier Richter [Ric05] zitiert, der charakteristische Merkmale dienstorientierter Architekturen zusammenfasst und als Minimalkonsens der verschiedenen Literaturstellen bezeichnet.

- Die dienstorientierte Architektur ist eigentlich ein Architekturmuster, das den Aufbau einer Anwendungslandschaft aus einzelnen Anwendungssystemen beschreibt.
- Die Anwendungssysteme sind lose miteinander gekoppelt, indem sie einander ihre Funktionalitäten in Form von Diensten anbieten.
- Ein Dienst ist eine feste, definierte Leistung, die als Baustein eines oder mehrerer größerer Verarbeitungsabläufe verwendet werden kann. Als solcher stellt der Dienst eine abstrakte Sicht auf das anbietende Anwendungssystem dar und verbirgt alle Implementierungsdetails. Die Definition eines Dienstes hat den Charakter einer vertraglichen Übereinkunft zwischen Dienstanbieter und Dienstnutzer. Dienste sind tendenziell grobgranular.
- Dienste werden über einen einheitlichen Mechanismus aufgerufen, der die Anwendungssysteme plattformunabhängig miteinander verbindet und alle technischen Details der Netzkommunikation verbirgt. Der Dienstnutzer adressiert bei seinem Aufruf eine – anonyme – Schnittstelle. Der Aufrufmechanismus beinhaltet das Auffinden eines geeigneten, konkreten Dienstanbieters.

Richter stellt fest, die Dienstorientierung würde vor allem altbewährte Prinzipien der Softwarearchitektur auf die Domäne der Anwendungslandschaften übertragen. Gleichzeitig kritisiert er, der Begriff des Dienstes sei nicht ausreichend präzise eingegrenzt. Im Grunde genommen kann jede beliebige Funktionalität als Dienst interpretiert werden. Eine solchen Beliebigkeit schwächt die Ausdruckskraft des Architekturkonzeptes der dienstorientierten Architektur. Der Einsatz einer dienstorientierten Architektur alleine bietet lediglich eine geringe Hilfestellung für die erfolgreiche Gestaltung einer Anwendungslandschaft. Vielmehr liegt die Hauptaufgabe in der Entwicklung einer konkreten *Architektur der Anwendungslandschaft.*

Ein exemplarischer Ansatz für die konkrete Ausprägung einer dienstorientierten Architektur für Anwendungslandschaften wird im Folgenden skizziert. Es handelt sich dabei um den von Humm et. al. [HVH06] beschriebenen dienstspezifischen Teil von *Quasar (Qualitäts-Software-Architektur) Enterprise* nach Engels et. al. [EHH+08].

Humm et. al. bezeichnen die *Anwendungsdomänen*, *Anwendungskomponenten* und *Anwendungsdienste* als Grundlage für die Strukturierung von Anwendungslandschaften. Eine Anwendungsdomäne umfasst eine Menge von Anwendungskomponenten, die aus Sicht der Geschäftsarchitektur sinnvollerweise gemeinsam betrachtet werden. Anwendungskomponenten sind die wesentlichen Einheiten der Planung, des Entwurfs und der Implementierung. Sie exportieren und importieren Schnittstellen und verbergen Implementierungsdetails. Anwendungsdienste bilden die Schnittstellen von Anwendungskomponenten. Ein Dienst bietet dabei eine fachliche Funktionalität an, die mehrere *Anwendungsdienstoperationen* umfassen kann. Der Dienst stellt eine abstrakte Sicht auf die exportierende Komponente dar und verbirgt alle Implementierungsdetails.

Die Gestaltung der Anwendungslandschaft erfolgt entlang der Strukturierung in Anwendungsdomänen und Dienstkategorien. Humm et. al. postulieren, in idealen Anwendungslandschaften seien die Anwendungsdomänen identisch mit den Geschäftsdomänen. Jede Geschäftsdomäne wird auf diese Weise explizit in der Anwendungsarchitektur berücksichtigt. Die Dienstkategorien stellen genau die weiter oben geforderte konkrete Ausprägung des dienstorientierten Architekturkonzeptes dar. Dienstkategorien geben sowohl einen Leitfaden für die Gestaltung einzelner Dienste als auch Kontexte für die Verknüpfung von Diensten vor.

Bestandskomponenten haben die Hoheit über jeweils einen Ausschnitt der Geschäftsdaten eines Unternehmens. Ihre Dienste umfassen die CRUD-Operationen (*create, read, update, delete*) und darauf aufbauend Pflegeoperationen und verschiedene Sichten. *Funktionskomponenten* bieten Dienste an, die die Geschäftsfunktionen eines Unternehmens unterstützen. Einige dieser Dienste sind für sich allein gesehen sinnvoll, andere wiederum lassen sich nur im Kontext und unter der Kontrolle von Geschäftsprozessen ausführen. Funktionskomponenten nutzen Dienste der Bestandskomponenten. *Prozesskomponenten* unterstützen jeweils einen oder mehrere Geschäftsprozesse. Sie steuern den Ablauf von Prozessen gemäß Geschäftsregeln über mehrere Funktionskomponenten hinweg. Es kann sich dabei sowohl um vollständig automatisierte Abläufe als auch um solche mit Benutzerinteraktionen handeln. Prozesskomponenten greifen auf Dienste anderer Prozesskomponenten sowie auf die Dienste von Funktions- und Bestandskomponenten zurück. Darüber hinaus ermöglichen *Interaktionskomponenten* menschlichen Anwendern den Zugang zu den Diensten einer Anwendungslandschaft. Nach außen hin bieten sie eine homogene Sicht auf die verschiedenen Komponenten und Anwendungen. Idealerweise sind Grenzen zwischen einzelnen Anwendungen für den Benutzer nicht mehr sichtbar. Interaktionskomponenten unterscheiden sich von den anderen Komponentenkategorien. Die Dienste von Interaktionskomponenten sind die Benutzerschnittstellen selber, während die Dienste der übrigen Komponenten Schnittstellen im softwaretechnischen Sinne sind.

Bereits anhand der knappen Beschreibung dieses Teilabschnittes wird deutlich, dass durch die Strukturierung nach Anwendungsdomänen und den verschiedenen Kategorien von Anwendungskomponenten/-diensten ein ausdrucksstärkerer Vorschlag für die Architektur von Anwendungslandschaften unterbreitet wird, als es durch die reine Propagierung einer dienstorientierten Architektur geschieht.

5.4.3 Bewertung

Zu Beginn des Abschnittes wurde die Frage aufgeworfen, ob Softwaresysteme *für* Organisationen automatisch in den Bereich Softwaresysteme *als* Organisationen fallen. Die Ausführungen zu betrieblichen Softwaresystemen in Form von Anwendungslandschaften machen deutlich, dass dies zumindest nicht notwendigerweise der Fall ist. Beispielsweise wurde darauf hingewiesen, der Einsatz großer monolithischer Systeme sei gegebenenfalls ausreichend. Es ist allerdings festzuhalten, dass Bemühungen um ein Business-IT-Alignment, wie es hier exemplarisch erläutert wurde, zumindest einen Schritt in die Richtung einer Auffassung von Softwaresystemen *als* Organisationen darstellt. Es entstehen zwar nicht unbedingt Softwaresysteme mit eigenständigen Organisationsstrukturen, zumindest aber werden Struktur und Verhalten des realweltlichen Organisationskontextes berücksichtigt und teilweise auf das Softwaresystem übertragen.

In diesem Zusammenhang ergibt sich, dass speziell Anwendungslandschaften basierend auf dienstorientierten Architekturen in eine konzeptionelle Nähe von Multiagentensystemen rücken. Die Dienstmetapher ist der Agentenmetapher ähnlich. In beiden Fällen geht es um Au-

tonomie, Kapselung, lose Kopplung und dynamisch realisierbare sowie flexible Kollaboration von Systemkomponenten. Die Agentenmetapher ist dabei die mächtigere. Die Idee, dass Agenten autonom und flexibel Dienste gegenüber anderen Agenten oder realweltlichen Akteuren anbieten, ist unmittelbar in der Metapher des (künstlich) sozialen Agenten/Akteurs assoziiert. Konzepte wie Zielverfolgung, Deliberation, das Eingehen von Verpflichtung oder soziale Beziehungen und soziale Interaktionen zwischen Komponente sind im Gegenzug jedoch nicht mit der Dienstmetapher assoziiert. Analog zu den in diesem Kapitel geführten Diskussionen bezüglich der Abgrenzungen zwischen den verschiedenen Paradigmen geht es auch hier weniger um das, was grundsätzlich *implementiert/programmiert* werden kann, sondern darum, was im Kontext des jeweiligen Paradigmas und seiner Metaphern eine *„natürliche" Herangehensweise* an Problemstellungen und Anwendungsfälle darstellt. In dieser Hinsicht kann die Agentenmetapher auf die Dienstmetapher „reduziert" werden.

Damit einher geht die Feststellung, dass die Gestaltung großer Softwaresysteme unter Einbezug organisationstheoretischer Konzepte und Mechanismen im Kontext dienstorientierter Architekturen nur begrenzt möglich ist. Konzepte wie Autonomie, Rollen und Kollaboration lassen sich mit der Dienstmetapher noch vereinen. Konzepte wie Autorität, verteilte Planung, Verpflichtung, Formalisierung, Normen oder Beaufsichtigung hingegen sind zwar unmittelbar mit der Agentenmetapher, nicht jedoch mit der Dienstmetapher assoziiert. Gegenüber der Organisationsmetapher fehlen darüber hinaus Konzepte, Mechanismen und Kategorien für die systematische Meso-Strukturierung von Systemen.

Zusammenfassend ist festzustellen, dass die Menge der Softwaresysteme *für* Organisationen zwar gegebenenfalls eine Schnittmenge mit der Menge der Softwaresysteme *als* Organisationen bildet, eine notwendige Engführung jedoch nicht existiert. Insbesondere führen Bemühungen um ein Business-IT-Alignment zwar zu einer Angleichung von Softwaresystemstrukturen und -prozessen an die realweltlichen Gegenstücke. Eine eigene Identität der Softwareorganisation gegenüber der realweltlichen Organisation ist allerdings nicht gegeben. Nichtsdestotrotz ist nach Meinung des Autors der vorliegenden Arbeit ein Trend im Bereich der betrieblichen Softwaresysteme hin zu einer Perspektive auf Softwaresysteme *als* Organisationen erkennbar. Die in Kapitel 3 herausgearbeiteten Versprechungen eines Organisationsparadigmas sind vor allem auf Softwaresysteme im Großen als Systeme von Systemen gemünzt. Damit einher geht somit die unmittelbare Hypothese, ein Organisationsparadigma der Softwareentwicklung sei gerade für große betriebliche Softwaresysteme von Vorteil. Eine systematische Untermauerung dieser Hypothese in Kombination mit praxisnahen Belegen steht allerdings zum gegenwärtigen Zeitpunkt aus und ist zukünftigen Forschungsbestrebungen vorbehalten.

5.5 Zusammenfassung

In den vorigen beiden Kapiteln wurden die Versprechungen eines Organisationsparadigmas der Softwareentwicklung sowie die dafür relevanten Konzepte und Mechanismen aus dem Bereich realweltlicher Organisation herausgearbeitet. Darauf aufbauend wurde in diesem Kapitel untersucht, ob und inwieweit existierende Ansätze der Softwaresystementwicklung das Potential der Organisationsmetapher bereits ausnutzen.

Der Fokus lag dabei auf Ansätzen aus dem Multiagentensystembereich, da die Idee von Softwareentitäten/-infrastrukturen als Verkörperung von Organisationen (in einem sozial fundierten Sinne) hier besonders nahe liegt und tatsächlich bereits weithin umgesetzt wird. Elaborierte formale Organisationsspezifikationen werden in Form von Organisationsinfrastrukturen

umgesetzt (Organisations-Middleware). Agenten können der Organisation beitreten und als Mitglieder agieren, sind in ihren organisatorischen Handlungen jedoch durch die von Seiten der Infrastruktur bereitgestellten Möglichkeiten und Dienste eingeschränkt. Auf diese Weise wird eine explizite Reifizierung von Organisationen vorgenommen und Organisationen werden als Meso-Strukturen zwischen Agenten- und Multiagentensystemebene eingesetzt.

Genau anhand des Aspektes der Meso-Strukturierung wurden jedoch auch Defizite identifiziert. Die Merkmale von Organisationen als offene Systeme und kollektive Akteure werden in der Multiagentensystemforschung bislang nicht ausreichend berücksichtigt. Auch wenn Möglichkeiten der technischen Realisierung kollektiver Akteure existieren, so werden inhaltliche Aspekte des Übergangs von einem organisatorischen Kontext zu einem kollektiven Akteur nicht vertieft behandelt. Der Weg zu einem rekursiven Verständnis von softwaretechnischen Organisationssystemen auf Basis janusgesichtiger Softwareentitäten, die auf dem Konzept der organisierten Kollektivität fußen, ist daher noch nicht vollends bereitet. Die Konzeptionalisierung solcher Softwareentitäten stellt nach Ansicht des Autors der vorliegenden Arbeit den Übergang von der Agenten- zur Organisationsorientierung dar und ist das vorrangige Thema des zweiten Teils der Arbeit.

Neben der Multiagentensystemforschung wurden der Ansatz der Viable Software und betriebliche Softwaresysteme betrachtet. Viable Software erlaubt bereits einen rekursiven, organisationsorientierten Aufbau von Softwaresystemen. Allerdings ist der Ansatz recht spezifisch und sieht nur eingeschränkte Rekursivität vor. Im Falle betrieblicher Softwaresysteme lässt sich im Kontext von Bemühungen um ein Business-IT-Alignment ein gewisser Trend für die Berücksichtigung von Organisationsstrukturen und -prozessen bei der Gestaltung von Softwaresystemen erkennen. Allerdings geht es vorrangig um die Anpassung/Angleichung von Softwarekonfigurationen an geschäftliche Anforderungen der realweltlichen Organisation. Die Behandlung von Softwaresystemen als eigenständige (künstliche) Organisationssysteme steht aktuell nicht im Fokus.

6 Petrinetzmodellierung

Bis zu diesem Punkt war die vorliegende Arbeit vor allem dem Vorhaben gewidmet, eine organisationsorientierte Perspektive auf Softwaresysteme von Softwaresystemen zu motivieren, Kernkonzepte und -mechanismen anhand der Betrachtung realweltlicher Organisationen zu identifizieren und verwandte Arbeiten im Kontext der Softwareentwicklung kritisch zu beleuchten. Im zweiten Teil der Arbeit wird aufbauend auf der bisher geleisteten Grundlagenarbeit eine ganz konkrete Modellierung der Struktur und des Verhaltens von organisationsorientierten Softwaresystemen vorgenommen. Es handelt sich dabei um einen spezifischen Vorschlag, der die klassische Auffassung von Multiagentensystemen um janusgesichtige Entitäten erweitert, die Organisationskontexte und -akteure in einem sind.

Es wird deutlich, dass die Wahl einer geeigneten Modellierungstechnik von hoher Bedeutung ist. Die gewählte Technik muss zunächst einmal dem Wesen der hier betrachteten Softwaresysteme als Systeme von Systemen gerecht werden. Darüber hinaus muss sie einen Brückenschlag von organisationstheoretischen Konzepten und Mechanismen zu einem informatiknahen und programmatischen Modell von Softwaresystemen als Organisationen leisten. Die Wahl für die vorliegende Arbeit ist auf *Petrinetze* und insbesondere den höheren Petrinetzformalismus der *Referenznetze* gefallen. Nach einer Einführung in die Modellierung mit Petrinetzen in Abschnitt 6.1 werden die Gründe für diese Wahl in Abschnitt 6.2 ausführlich dargelegt. Anschließend erfolgt eine explizite Unterscheidung verschiedener Abstraktionsstufen der Petrinetzmodellierung im Kontext der vorliegenden Arbeit in Abschnitt 6.3.

6.1 Von S/T-Netzen zu Referenznetzen

Im Folgenden wird sowohl die Modellierung basierend auf einfachen als auch auf höheren Petrinetzformalismen behandelt. Das intuitive Grundmodell der Petrinetze geht auf die grundlegenden Arbeiten von Petri [Pet62] zurück. Im Laufe der Zeit wurde eine Vielfalt von erweiterten Formalismen entwickelt. Von besonderer Wichtigkeit für die vorliegende Arbeit ist das Konzept der *Netze in Netzen* nach Valk [Val98, Val04] und der darauf aufbauende Formalismus der Referenznetze nach Kummer [Kum02]. Referenznetze erhalten Werkzeugunterstützung durch das Modellierungs- und Simulationswerkzeug Renew.[1] Geeignete Einschränkungen des Formalismus der Referenznetze führen zu einfacheren Formalismen wie S/T-Netzen und „herkömmlichen“ gefärbten Petrinetze, so dass auch diese Formalismen von Renew unterstützt werden. Alle Modelle des vorliegenden Kapitels und des gesamten Rests der Arbeit wurden mit Renew erstellt. Die Einführung in die Petrinetzmodellierung dieses Abschnittes ist dementsprechend an den Modellierungsmöglichkeiten und -eigenheiten von Renew orientiert. Darüber hinaus erfolgt eine Erläuterung einiger Mechanismen der Simulation von Petrinetzen mit Renew.

An dieser Stelle kann lediglich eine Einführung in die Thematik der Modellierung mit Petrinetzen gegeben werden. Dabei werden alle Aspekte abgedeckt, die im Rahmen der vorliegenden

[1] Der Internet-Auftritt von Renew (**Re**ference **Ne**ts **W**orkshop) ist unter http://www.renew.de zu finden.

Arbeit relevant sind. Für eine höhere Detailtiefe und zusätzliche Aspekte wird auf weiterführende Literatur verwiesen. Von besonders hoher Relevanz ist dabei die Theorie der Referenznetze nach Kummer [Kum02] und das Renew-Handbuch von Kummer et. al. [KWD09]. Eine umfassende Einführung in die „klassische“ Systemmodellierung mit Petrinetzen (ohne das Konzept der Netze in Netzen) findet sich beispielsweise im Band von Girault und Valk [GV03]. Ein großer Teil der Petrinetz-Forschergemeinde findet sich als Autoren der verschiedenen Kapitel wieder. Weitere Literatur wird im Verlauf des Abschnittes genannt.

Ein Nachteil der Verwendung von Renew muss an dieser Stelle erwähnt werden. Petrinetze haben neben ihrer grafischen Darstellung eine mathematische Repräsentation. Damit bietet sich die Möglichkeit der formalen Analyse von Petrinetzmodellen. Für Renew existiert zum aktuellen Zeitpunkt keine ausgereifte Unterstützung solcher Analysen. Es ist zwar ein Export von Modellen in Form verschiedener Formate zur Weiterverarbeitung mit anderen Werkzeugen möglich, aufgrund des speziellen Formalismus der Referenznetze sind dann aber wichtige Netzelemente nicht nutzbar. Für mit Renew erstellte Modelle ist somit lediglich eine Validierung von Petrinetzmodellen über Simulation möglich. Eine automatische Verifizierung hingegen ist nur sehr eingeschränkt möglich. Im Kontext der vorliegenden Arbeit steht die Möglichkeit der exakten Modellierung von organisationsorientierten Softwaresystemen sowie die operationale Semantik konkreter Modelle im Vordergrund. Die mathematisch fundierte Analyse stellt hingegen zwar einen relevanten Aspekt zukünftiger Forschung dar, wird in dieser Arbeit jedoch nicht behandelt. Daher wird in diesem Abschnitt auch auf die mathematische Repräsentation von Petrinetzen verzichtet.

6.1.1 S/T-Netze

Grundsätzlich ist ein Petrinetz ein gerichteter, bipartiter Graph. Die beiden unterschiedlichen Arten von Knoten sind *Stellen* und *Transitionen*. Valk [Val03a] identifiziert hier das *Prinzip der Dualität von Petrinetzen*. Eine Stelle wird dabei als *passives* Element aufgefasst und modelliert beispielsweise einen Speicher, eine Bedingung, eine Ressource oder einen Kanal. Eine Transition hingegen wird als *aktives* Element aufgefasst und modelliert beispielsweise ein Ereignis, eine Aktion oder einen Nachrichtenversand. Die Eigenschaft der Bipartition sorgt dafür, dass Kanten nur von Stellen zu Transitionen oder umgekehrt führen dürfen.

Um von der statischen Struktur eines Petrinetzes zu einem dynamischen Verhalten zu gelangen, werden Stellen mit *Marken* belegt. In diesem Teilabschnitt wird zunächst der grundlegende Formalismus der *Stellen/Transitions-Netze* (S/T-Netze) behandelt. Eine formale Betrachtung von S/T-Netzen wird von Jantzen und Valk [JV80] vorgenommen. In S/T-Netzen gibt es lediglich nicht unterscheidbare *schwarzen* oder *anonyme Marken*. Eine konkrete Belegung von Stellen eines Petrinetzes mit Marken heißt *Markierung* und stellt einen Zustand des Netzes dar. Jede Stelle ist dabei entweder mit keiner, einer oder mehreren Marken belegt.

Zwischen verschiedenen Markierung (Zuständen) eines S/T-Netzes sind Übergänge möglich. Für einen Übergang müssen bestimmte Bedingungen erfüllt sein.

- Ein Übergang ist mit dem *Schalten* einer Transition verknüpft. Dafür werden die mit der Transition über Kanten verbundenen Stellen betrachtet. Diese stellen die sogenannte *Lokalität* oder *Umgebung* der Transition dar. In diesem Zusammenhang formuliert Valk [Val03a] das *Prinzip der Lokalität von Petrinetzen*: Das Verhalten einer Transition hängt ausschließlich von seiner Lokalität und sich selber ab. Wie sich dieses Verhalten im Falle von S/T-Netzen ergibt, wird in den folgenden Spiegelpunkten erläutert.

- Führt eine Kante von einer Stelle zu einer Transition, so liegt die Stelle im *Vorbereich* der Transition. Führt eine Kante von einer Transition zu einer Stelle, so liegt die Stelle im *Nachbereich* der Transition.

- Damit eine Transition schalten kann, muss sie *aktiviert* sein. Eine Transition ist aktiviert, wenn jede Stelle ihres Vorbereiches mindestens so viele Marken enthält wie Kanten von der jeweiligen Stelle zur Transition führen.[2]

- Beim Schalten einer Transition werden von jeder Stelle ihres Vorbereiches so viele Marken abgezogen, wie Kanten von der jeweiligen Stelle zur Transition führen. Gleichzeitig werden jeder Stelle des Nachbereiches der Transition so viele Marken hinzugefügt wie Kanten von der Transition zur jeweiligen Stelle führen. Das Schalten einer Transition muss nicht markenerhaltend sein. Es können mehr Marken von Stellen des Vorbereiches abgezogen werden als den Stellen des Nachbereiches hinzugefügt werden und umgekehrt.

In Abbildung 6.1 sind ein beispielhaftes S/T-Netz und ein Schaltvorgang illustriert. Entsprechend des *Prinzips der grafischen Repräsentation von Petrinetzen* nach Valk [Val03a] werden Stellen als Kreise und Transitionen als Rechtecke dargestellt. Gegebenenfalls tragen sie einen Namen. Falls Namen für Stellen oder Transitionen angegeben sind, so werden sie in der vorliegenden Arbeit stets fettgedruckt dargestellt. Es handelt sich dann nicht um Anschriften mit semantischem Gehalt für die Funktionsweise des Netzes. Die linke und rechte Seite der

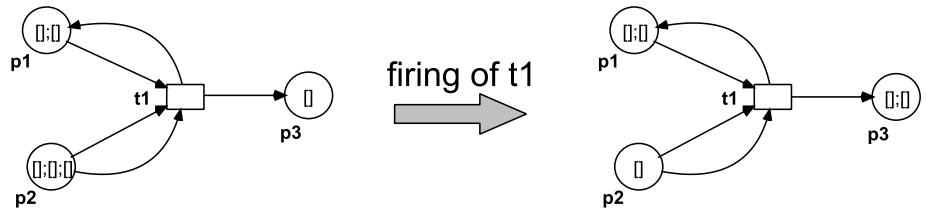

Abbildung 6.1: Ein einfaches S/T-Netz vor und nach dem Schalten der Transition `t1`

Abbildung stellen jeweils eine Markierung desselben S/T-Netzes und damit jeweils einen Zustand des Netzes dar. Eine schwarze Marke wird in Renew als öffnende und schließende eckige Klammer [] dargestellt. Mehrere Marken auf einer Stelle werden als durch Semikolon getrennt dargestellt. In der linken Seite der Abbildung ist die Transition `t1` aktiviert: (1) Von Stelle `p1` führt eine Kante zur Transition und die Stelle enthält zwei Marken, (2) von der Stelle `p2` führen zwei Kanten zur Transition und die Stelle ist mit drei Marken belegt. Der Schaltvorgang überführt die linke Seite der Abbildung in die rechte. Beim Schalten der Transition werden eine Marke von Stelle `p1` und zwei Marken von Stelle `p2` abgezogen. Gleichzeitig werden Marken auf Stellen des Nachbereiches der Transition hinzugefügt. Die Stellen `p1` und `p3` erhalten jeweils eine zusätzliche Marke. Insgesamt werden beim Schaltvorgang also drei Marken verbraucht und zwei Marken generiert. Für Stelle `p1` gilt, dass sie sowohl zum Vor- als auch zum Nachbereich der Transition `t1` gehört.

[2] Alternativ zu multiplen Kanten von einer Stelle zu einer Transition können auch gewichtete Kanten betrachtet werden.

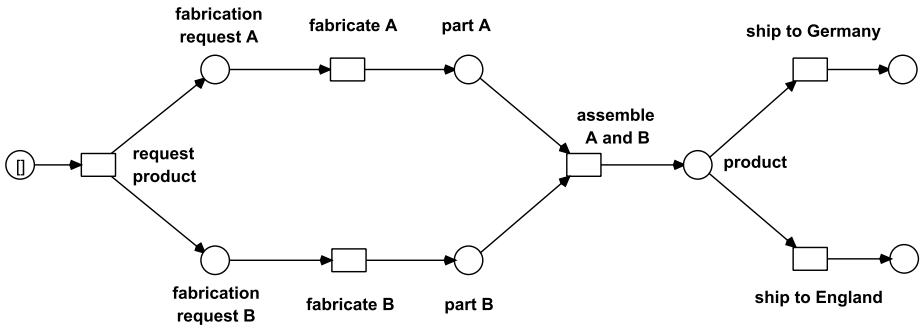

Abbildung 6.2: Beispielnetz: Herstellung und Versand eines Produktes

In der Markierung auf der rechten Seite der Abbildung ist die Transition t1 nicht mehr aktiviert. Zwar gilt weiterhin, dass die Stelle p1 die benötigten zwei Marken enthält. Die Stelle p2 enthält allerdings nur noch eine Marke, notwendig wären aber zwei Marken.

Einige weitere Konzepte werden anhand des etwas größeren Beispiels aus Abbildung 6.2 veranschaulicht. Hier wird folgender einfache Vorgang modelliert: Zunächst wird die Herstellung eines Produktes in Auftrag gegeben, dann werden zwei Teile des Produktes hergestellt und anschließend zusammengefügt, bevor das fertige Produkt schließlich verschifft wird. Anhand des Beispiels lassen sich einige grundlegende Konzepte beziehungsweise Grundmuster der Petrinetzmodellierung identifizieren.

- *Nebenläufigkeit:* Nebenläufigkeit ist vielleicht *das* Kernkonzept der Petrinetztheorie. Zwar kann man bei einer Markierung von einem globalen Systemzustand eines Petrinetzes sprechen. Aber eigentlich handelt es sich dabei um einen über verschiedene Zustandselemente – die Stellen und ihre Belegung – verteilten Zustand. Dieser verteilte Charakter des Systemzustandes führt zu einer verteilten, nebenläufigen Dynamik eines Petrinetzes. Ein Beispiel dafür ist in der Abbildung anhand der beiden Transitionen fabricate A und fabricate B zu sehen. Dem *Prinzip der Nebenläufigkeit in Petrinetzen* nach Valk [Val03a] entsprechend können diese beiden Transitionen unabhängig voneinander und damit nebenläufig schalten, weil sie disjunkte Lokalitäten haben. Das Prinzip der Nebenläufigkeit ergibt sich unmittelbar aus dem weiter oben zitierten Prinzip Lokalität.

- *Synchronisation:* Die Transition assemble A and B stellt den Fall dar, dass zuvor unabhängige Schaltsequenzen wieder zusammengeführt und synchronisiert werden.

- *Konflikt:* Die Situation mit der Stelle product und den Transitionen ship to Germany und ship to England modelliert einen Konflikt. Die Stelle liegt im Vorbereich beider Transitionen. Ist die Stelle mit nur einer Marke belegt, so kann lediglich eine der beiden Transitionen schalten.

Auf Basis der bisher eingeführten einfachen S/T-Netze werden im Folgenden höhere Petrinetzformalismen behandelt.

6.1.2 Gefärbte Petrinetze

In gefärbten Petrinetzen treten üblicherweise unterscheidbare Marken auf. Marken können nun als Datenstrukturen aufgefasst werden. Analog zu Datentypen in Programmiersprachen haben Marken auch einen Typ. Die Typen werden auch *Farben* genannt. Wiederum orientiert sich dieser Abschnitt an den Möglichkeiten von Renew, für eine formale Behandlung gefärbter Petrinetze sei beispielsweise auf die Arbeit von Jensen [Jen96] verwiesen.

Wie in Teilabschnitt 6.1.3 näher beschrieben wird, erreicht die Anschriftensprache von Renew nahezu die vollständige Mächtigkeit der Programmiersprache Java. Dementsprechend können gefärbte Marken in Renew beliebige Java-Objekte sein. Die Modelle der vorliegenden Arbeit kommen (neben Netzmarken, vergleiche den folgenden Teilabschnitt) nahezu vollständig mit elementaren Datentypen wie Zeichenketten, Zahlen und Wahrheitswerten sowie Gruppierungen in Form von Tupeln und Listen aus. Daher sind auch die Beispiele dieses Abschnittes auf elementare Datentypen eingeschränkt.

Schränkt man die Anzahl der Farben auf nur eine einzige ein, so erhält man wieder den Formalismus der S/T-Netze. Gibt es aber mehrere Farben, so muss dies bei der Betrachtung der Dynamik eines gefärbten Petrinetzes berücksichtigt werden. Beim Schalten einer Transition spielt es nun eine Rolle, welche der unterscheidbaren Marken von den Stellen des Vorbereiches der Transition abgezogen werden und mit welchen Marken die Stellen des Nachbereiches belegt werden. Zu diesem Zweck besteht eine Möglichkeit darin, Kanten direkt mit Datenwerten zu beschriften. Abbildung 6.3 zeigt ein Beispiel.

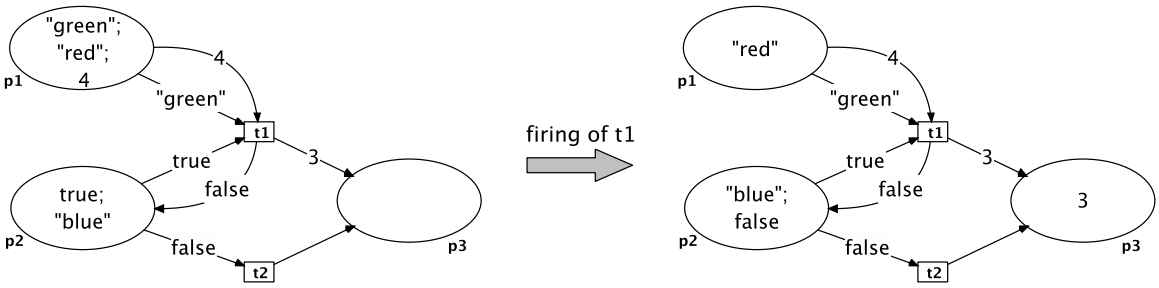

Abbildung 6.3: Ein gefärbtes Petrinetz vor und nach dem Schalten der Transition t1

Hier wird ein interessanter Umstand ersichtlich. Die reine Netzstruktur modelliert einen Konflikt. Die Stelle p2 liegt im Vorbereich beider Transitionen t1 und t2. Unter der Markierung auf der linken Seite der Abbildung besteht allerdings tatsächlich gar kein Konflikt. Nur die Transition t1 ist aktiviert, da beide Stellen des Vorbereiches mit den auf den jeweiligen Kanten genannten Datenwerten markiert sind ("green" und 4 im einen Fall, true im anderen Fall). Die Transition t2 hingegen ist nicht aktiviert, da die Vorbereichsstelle p2 keine Marke false führt. Erst mit dem Schalten der Transition t1 und der entstehenden Folgemarkierung auf der rechten Seite der Abbildung wird Transition t2 aktiviert.

Anstatt Kanten direkt mit Datenwerten zu beschriften, ist es auch möglich, sie mit Variablenanschriften zu versehen. Renew erlaubt sowohl die Verwendung ungetypter als auch getypter Variablen. Dabei ist allerdings keine Mischform erlaubt. Sobald eine einzige Variable getypt wird, so müssen auch alle anderen Variablen des Netzes getypt werden. Zusätzlich zu Variablenanschriften an Kanten können auch Transitionen mit Anschriften versehen werden.

Im Falle von Renew kann es sich dabei um nahezu beliebige Java-Befehle handeln (vergleiche Teilabschnitt 6.1.3). Hier werden zunächst nur Variablenzuweisungen als Anschriften betrachtet. Damit eine Transition, die selber oder deren Eingangs- und Ausgangskanten Variablenanschriften tragen, schalten kann, muss eine *aktivierende Variablenbindung* gefunden werden. Dies bedeutet zum einen, dass gleiche, aber mehrfach verwendete Variablenbezeichner auch an gleiche Werte gebunden werden. Zum anderen bedeutet es, dass die Variablen auf den Eingangskanten der Transition an Werte gebunden werden, die auch als Marken auf den entsprechenden Vorbereichsstellen vorhanden sind. Renew verwendet das *Unifikationskonzept*, wie es aus deklarativen Programmiersprachen wie PROLOG bekannt ist, um aktivierende Variablenbindungen zu finden. Unifikation findet dementsprechend über den verschiedenen Variablenbezeichnern und den Markenwerten auf den Vorbereichsstellen statt und muss gegebenenfalls zusätzliche Wertzuweisungen resultierend aus Transitionsanschriften berücksichtigen. Eine unifizierende Variablenbindung gilt dabei natürlich nur für einen einzigen Schaltvorgang. Abbildung 6.4 zeigt ein Beispiel.

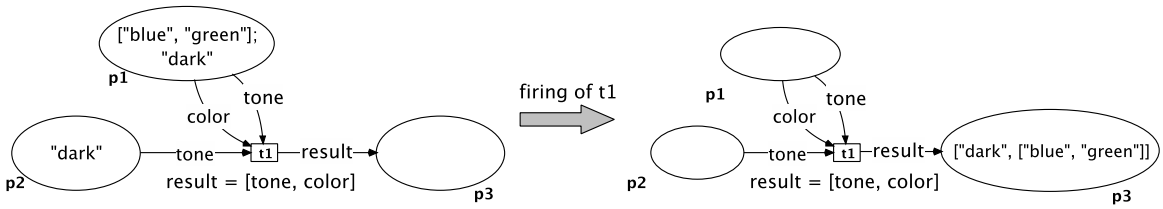

Abbildung 6.4: Schalten der Transition t1 unter Einbezug von Variablenanschriften

Bei diesem Beispiel sind mehrere Dinge erwähnenswert

- Grundsätzlich können als Variablenbezeichner beliebige Zeichenketten über den Buchstaben des Alphabets, den Zahlen 0 bis 9 und einiger weiterer Zeichen gewählt werden (Zeichenketten als Datenwerte/Marken sind durch zusätzliche umschließende Anführungsstriche gekennzeichnet). Allerdings sind einige Schlüsselwörter als Bezeichner ausgeschlossen. Zwei solcher Schlüsselwörter sind true und false da diese, wie im vorigen Beispiel veranschaulicht, Wahrheitswerte repräsentieren.

- Es kommt eine weitere Renew-Schreibweise zum Tragen. *Tupel* werden als durch Kommata getrennte, von eckigen Klammern umschlossene Datenwerte und Variablen repräsentiert. Eine ähnliche Repräsentation findet sich im Falle von *Listen*. Hier werden geschweifte Klammern verwendet und es besteht die Möglichkeit, über die Schreibweise {A:B} auf den Kopf A und den Rest B einer Liste zuzugreifen.

- Unifikation über Variablen und Datenwerten bezieht Tupel (und Listen) ein. Im Beispiel wird die Variable color mit dem Tupel ["blue", "green"] unifiziert.

- Bestünde der Vorbereich der Transition t1 lediglich aus der Stelle p1, so wären zwei verschiedene aktivierende Variablenbindungen für die Variablen tone und color möglich und eine davon würde zum Schalten der Transition nichtdeterministisch ausgewählt. Erst der Einbezug der Stelle p2 in den Vorbereich der Transition und die erneute Verwendung der Variablen tone auf der Kante von der Stelle zur Transition legt eine einzige mögliche aktivierende Variablenbindung eindeutig fest.

- In diesem Beispiel ist auch zu sehen, dass die Transition t2 eine Anschrift trägt, anhand derer eine weitere Variable result belegt wird.

Es bleibt festzuhalten, dass insbesondere die im vorigen Teilabschnitt zitieren Prinzipien der Lokalität und Nebenläufigkeit in Petrinetzen weiterhin gewahrt bleiben. Die Prüfung der Aktiviertheit einer Transition wird durch den Einbezug komplexer Datenstrukturen als Marken und die Verwendung von Variablen allerdings komplizierter.

Es ist möglich, die Verwendung getypter/gefärbter Marken mit den von S/T-Netzen bekannten anonymen Marken zu kombinieren. Allerdings können anonyme Marken in Renew nicht an Variablen gebunden werden und müssen daher über unbeschriftete Kanten abgezogen und hinzugefügt werden.

6.1.3 Referenznetze

Angesichts der bisherigen Ausführungen der vorliegenden Arbeit zu Systemen von Systemen, Mikro/Makro-Dynamik, Organisationsstrukturen und kollektiven Akteuren wird deutlich, dass ein Modellierungsansatz mit den bisher betrachteten „herkömmlichen“ Petrinetzformalismen nicht angemessen ist. Insbesondere die Beschränkung auf eine einzige Systemebene steht einer Skalierung im Wege. Eine erste Abhilfe schaffen Techniken wie Verfeinerung/Vergröberung von Petrinetzmodellen beziehungsweise Faltung/Auffaltung von (gleichartigen) Netzelementen (vergleiche beispielsweise Valk [Val03a, Val03b]). Beispielsweise spannen die Handlungen zur Erstellung eines Produktes in der Abbildung 6.2 aus Teilabschnitt 6.1.1 ein Teilnetz des gesamten Netzes auf. Es umfasst die Transitionen request product, fabricate A, fabricate B, assemble A and B und die Stellen fabrication request A, fabrication request B, part A, part B. Dieses Teilnetz ist *transitionsberandet* durch die Transitionen request product und assemble A and B: Kanten herführend von beziehungsweise hinführend zu Netzelementen außerhalb des Teilnetzes sind nur mit diesen beiden Transitionen verbunden. Transitionsberandete Teilnetze können zu einer einzigen Transition vergröbert werden. Im Falle des Beispiels könnte also gegebenenfalls eine Transition manifacture product entstehen, die den gesamten Herstellungsvorgang modelliert.

Allerdings wird auf diese Weise eher die Betrachtung verschieden abstrakter Modelle *eines* Systems unterstützt. Im Beispiel ergibt sich für das nun vergröberte (und daher teilweise verborgene) Teilnetz zur Produktfertigung nicht unbedingt die Auffassung als eigenständiges Teilsystem. Anders formuliert erfordert die intuitiv zugängliche Modellierung *multipler* Systeme im Kontext eines Gesamtsystems einen weiteren Schritt. Dieser nächste Schritt wird mit der Einführung des Konzeptes der *Netze in Netzen* geleistet. Das Konzept wurde von Valk in die Theorie der Petrinetze eingeführt. Valks Arbeiten erweitern die bereits im vorigen Teilabschnitt über gefärbte Petrinetze eingeführten Markentypen um den Typ der Petrinetze selber. Petrinetze können Marken – sogenannte *Netzmarken* – sein. Da Netzmarken als Petrinetze auch ein eigenes Verhalten aufweisen, handelt es sich um *aktive Marken*. Dies steht im Gegensatz zu den bisher betrachteten Markentypen. Erstmals wurde dieses Konzept von Valk [JV87] im Kontext der Auftragsverkehrsnetze formalisiert (vergleiche auch Valk [Val87]). Dabei modellieren Netzmarken den Abarbeitungszustand eines Auftrages. Aus diesem Ansatz resultiert Valks [Val98, Val04] Formalismus der *elementaren Objektnetze*. Der Formalismus bezieht sich auf genau zweistufige Systeme von umschließenden und eingebetteten Petrinetzen. Ein einbettendes Netz wird dabei als *Systemnetz* und ein eingebettetes Netz als *Objektnetz* bezeichnet. Köhler [Köh04] erweitert diesen Formalismus zum Formalismus der *Objektnetzsysteme*, wobei beliebige Schachtelungstiefe zugelassen ist.

Der in der vorliegenden Arbeit verwendete Formalismus der *Referenznetze* nach Kummer [Kum02] ist ein spezieller Objektnetzformalismus. Auch hier ist beliebige Schachtelungstiefe erlaubt. Der Zugriff auf Netzmarken erfolgt mittels Referenzsemantik. Das bedeutet, die Netzmarken sind eigentlich Referenzen auf die tatsächlichen Netze. Dementsprechend können auch nichtdisjunkte Hierarchien entstehen: Verschiedene umschließende Netze können Referenzen auf dasselbe (und daher mehrfach eingebettete) Netz besitzen. Darüber hinaus wird im Referenznetzformalismus ein Mechanismus analog zu objektorientierten Programmiersprachen verwendet: Netze (analog zu Klassen) sind *Schablonen* für *Netzinstanzen* (analog zu Objekten). Zwei Instanzen desselben Netzes sind vollständig unabhängig voneinander. Im Kontext der vorliegenden Arbeit wird der Einfachheit halber in den meisten Fällen schlicht von Netzen und Netzmarken die Rede sein, es sind aber eigentlich Referenzen auf Netzinstanzen gemeint. Nur wenn diese Aspekte von spezieller Bedeutung sind, wird explizit darauf Bezug genommen.

Zur sinnvollen Verwendung geschachtelter Netzsysteme bedarf es eines Kommunikationsmechanismus. Christensen und Hansen [CH94] haben *synchrone Kanäle* bereits für herkömmliche gefärbte Petrinetz eingeführt. Im Referenznetzformalismus dienen sie als Kommunikationsmittel zwischen verschiedenen Netzen (und können auch auf Ebene eines einzigen Netzes verwendet werden). Ein synchroner Kanal sorgt für eine Fusion zweier Transitionen für einen Schaltvorgang. Die Transitionen schalten also gleichzeitig. Dabei besteht die Möglichkeit eines Datenaustausches. Weitere Feinheiten werden anhand des illustrierenden Beispiels aus Abbildung 6.5 erläutert. Dabei wird ein Arbeitsschritt eines Fließbandes modelliert. Eine Kiste wird mit einem bestimmten Gegenstand bepackt. Sowohl Fließband als auch Kiste sind als Netze modelliert und die Kiste ist als Netzmarke in das Fließbandnetz eingebettet. Diese Einbettung wird hier durch gepunktete Pfeile modelliert.[3]

- Synchrone Kanäle tragen einen Namen. In diesem Fall gibt es nur den einzigen synchronen Kanal :pack(). Analog zu Methodenaufrufen in klassischen Programmiersprachen wird auf einer des Kanals die andere Seite explizit benannt. Hier trägt die Transition t1 des Netzes assembly line einen *Downlink*. Über diesen wird spezifiziert, dass die andere Seite des Kanals in dem Netz, das beim Schaltvorgang an die Variable box gebunden wird, zu finden ist. In diesem Netz muss es eine Transition mit dem passenden *Uplink* geben. Hier handelt es sich dabei um das Netz box1 und die Transition t2.

  Downlink und Uplink können sich auch in demselben Netz befinden. Beim Downlink wird in diesem Fall das Schlüsselwort this für die Bezeichnung des Netzes mit der Gegenseite des Kanals verwendet.

- Synchrone Kanäle sind nicht nur durch ihren Namen gekennzeichnet. Stattdessen haben sie eine Signatur, bestehend aus Name und Parameterstelligkeit. Hier hat der Kanal :pack() die Stelligkeit Zwei.

[3] In Renew wird eine Netzmarke als speziell gekennzeichnete Zeichenkette dargestellt. Mittels eines Mausklicks auf die Zeichenkette öffnet sich ein neues Fenster für die durch die Netzmarke referenzierte Netzinstanz.
Die Initialisierung der Netzmarken ist in diesem Beispiel nicht enthalten. Während die Initialisierung elementarer Datentypen einfach als Zeichenkettenanschrift an Stellen geschehen kann, ist für die Initialisierung einer Netzmarke das Schalten einer Transition mit spezieller Anschrift notwendig. Eine Transitionsanschrift x:new someNet sorgt für die Bindung einer neuen Instanz der Netzschablone someNet an die Variable x. Mittels einer Anschrift x:new someNet(param1, param2) können noch weitere Initialisierungsparameter übergeben werden.

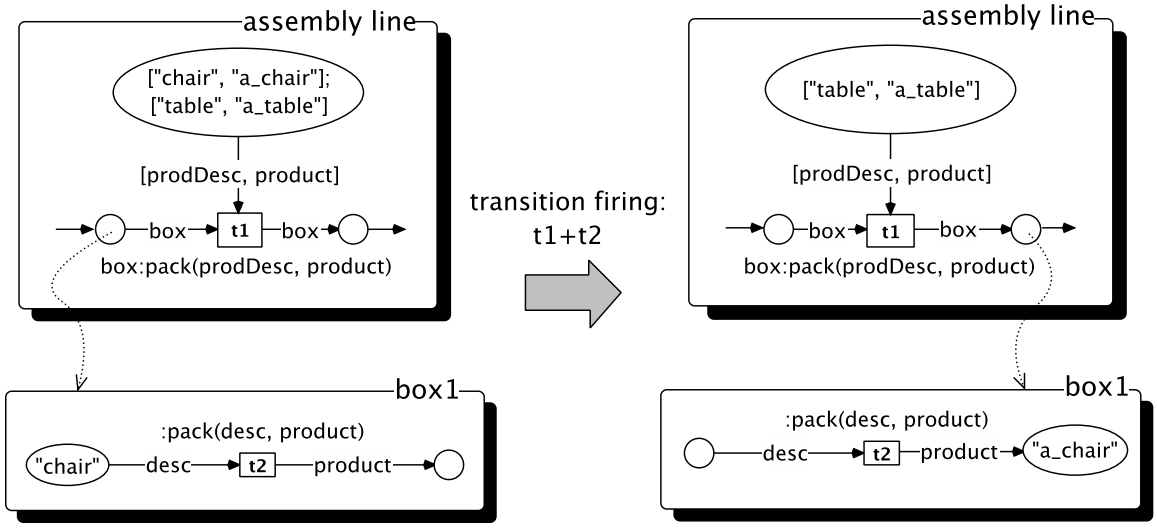

Abbildung 6.5: Bepacken einer Kiste über einen synchronen Kanal

- Damit über einen synchronen Kanal verbundene Transitionen schalten können, müssen beide aktiviert sein. Wie im Falle gefärbter Petrinetze mit Variablenanschriften wird für die Suche nach einer aktivierenden Bindung Unifikation verwendet. Allerdings gestaltet sich die Suche unter Einbezug synchroner Kanäle komplizierter, da nun zusätzlich die Kanalparameter einbezogen werden. Dabei sollte die Asymmetrie der Kanalanschriften an den beteiligten Transitionen (Downlink, Uplink) nicht zu der Annahme verleiten, Daten könnten nur in eine Richtung fließen. Stattdessen sorgt die Unifikation für einen bidirektionalen Datenaustausch über synchrone Kanäle.

  Der Schaltvorgang der Abbildung verdeutlicht den bidirektionalen Parameteraustausch. Die Produktbeschreibung (Zeichenkette "chair") wird vom Netz box1 bereitgestellt. Das passende Produkt selber (Zeichenkette "a_chair") wird vom Netz assembly line bereitgestellt.

  Dabei wird ersichtlich, dass das Prinzip der Lokalität von Transitionen gewahrt bleibt. Es ist für die Unifikation nicht notwendig, dass auf beiden Seiten des synchronen Kanals die gleichen Bezeichner für zu unifizierende Variablen verwendet werden. Stattdessen gilt, dass Argumente an derselben Parameterposition auf beiden Seiten des Kanals unifizieren müssen. Die Variable prodDesc auf Seite der Transition t1 und die Variable desc auf Seite der Transition t2 müssen trotz unterschiedlicher Bezeichner unifizieren da sie auf den beiden Kanalseiten jeweils für den ersten Kanalparameter verwendet werden. Für den zweiten Kanalparameter verwenden beide Kanalseiten zufällig den gleichen Variablenbezeichner product.

- Es fällt auf, dass auf Seiten der Transition t1 kein *konkretes* Netz als Gegenseite des Kanals vorgegeben ist. Stattdessen könnten ganz unterschiedliche Netze an die Variable box gebunden werden, solange sie eine Transition mit dem passenden Uplink beinhalten. Sind für einen Schaltvorgang mehrere Bindungen möglich, so wird eine davon nichtdeterministisch ausgewählt. Außerdem ist möglich, dass ein einziges Netz mehrere

Transitionen mit passendem Uplink enthält. Auch hier würde bei mehreren möglichen Kanal-Synchronisierungen eine nichtdeterministisch ausgewählt.

Es ist besonders wichtig zu ergänzen, dass *ein* synchroner Kanal zwar immer nur zwei Transitionen für einen Schaltvorgang synchronisiert. Durch die Kombination mehrerer Kanäle ist allerdings die Synchronisierung *beliebig vieler* Transitionen in einem einzigen Schaltschritt möglich. Eine Veranschaulichung dieser Möglichkeit zeigt die Erweiterung des Fließbandbeispiels in Abbildung 6.6, bei der nun auch die Produkte selber als Netze modelliert sind.

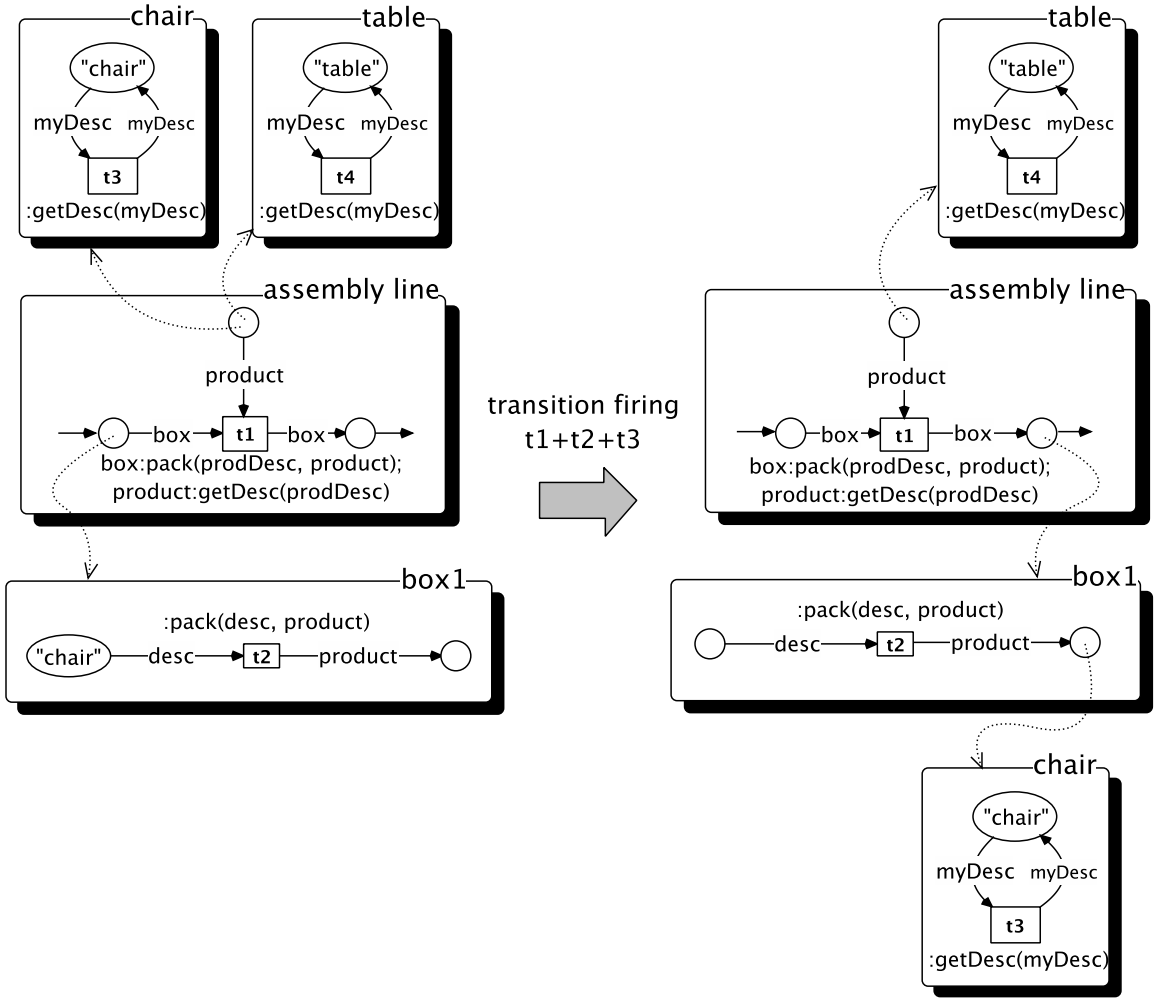

Abbildung 6.6: Bepacken einer Kiste über zwei synchrone Kanäle

Der Schaltvorgang zum Bepacken der Kiste bezieht nun zwei synchrone Kanäle (:pack() und :getDesc()) und drei Transitionen (t1, t2 und t3) auf einmal mit ein. Dabei werden Kanalparameter über drei verschiedene Systemebenen hinweg ausgetauscht. An diesem Beispiel wird auch deutlich, dass Transitionsanschriften *ungeordnet* sind. Die textuelle Darstellung als durch Semikolons getrennte, untereinander aufgeführte Zeichenketten darf nicht als sequentielle Abarbeitung missverstanden werden. Es geht um Unifikation von Variablen und Werten für

einen Schaltvorgang. So hängen für das Schalten der Transition t2 beide Transitionsanschriften wechselseitig voneinander ab.

Darüber hinaus wird eine weitere wichtige Möglichkeit der Modellierung mit Netzen in Netzen deutlich. Im Gegensatz zum vorigen Beispiel handelt es sich hier bei einem der Kanalparameter (beziehungsweise dessen Bindung) um ein Netz. Die Mobilität von Netzmarken ist also nicht nur auf Stellenwechsel innerhalb eines einzigen einbettenden Netzes beschränkt. Stattdessen können sie ihre Lokation auch über Netzgrenzen hinaus verändern. Hier ergibt sich also unmittelbar die Möglichkeit der Modellierung dynamisch rekonfigurierbarer Systemkompositionen. Insgesamt wird deutlich, dass das Konzept der Netze in Netzen mit seinem Konzept der aktiven, mobilen Marken in Kombination mit unifikationsbasierter Kommunikation über synchrone Kanäle die Modellierung hochdynamischer Systemkompositionen auf multiplen Ebenen ermöglicht.

Zusätzlich zu dem bisher vorgestellten „reinen“ Referenznetzformalismus bietet Renew den erweiterten Formalismus der *Java-Referenznetze*. Dabei dürfen Marken neben Netzen und elementaren Datentypen beliebige Objekte der Programmiersprache Java sein (da Renew selber in Java programmiert ist, handelt es sich auch bei Netzmarken eigentlich um spezielle Java-Objekte). Transitionsanschriften dürfen (mit wenigen Einschränkungen) beliebige Java-Befehle umfassen. In den Modellen der vorliegenden Arbeit wird von dieser Möglichkeit nur sehr eingeschränkt Gebrauch gemacht, daher wird hier auf eine ausführlichere Betrachtung verzichtet. Allerdings sei auf das mit Java-Referenznetzen verbundene Potential hingewiesen. Es bietet sich die Möglichkeit einer grafischen, *petrinetzbasierten Programmierung*. Gegenüber herkömmlichen Programmiersprachen werden vor allem die Konzepte der Nebenläufigkeit, Synchronisierung und des Konfliktes nativ und intuitiv unterstützt. Darüber hinaus eröffnet sich ein Ansatz der *Implementation durch Spezifikation*. Beginnend mit abstrakten Petrinetzmodellen können sukzessive Verfeinerungen, Konkretisierungen und Erweiterungen zu ausführbaren und schließlich sogar zu softwaretechnisch nutzbaren Systemen führen. Ein Beispiel, bei dem sich dieser Ansatz als überaus erfolgreich erwiesen hat ist das Multiagentensystem-Framework Mulan/Capa, welches in Abschnitt 6.2 skizziert wird.

Dieses Potential ist auch für die vorliegende Arbeit von Relevanz. Im zweiten Teil der Arbeit werden zunächst abstrakte und später detaillierte Struktur- und Verhaltensmodelle von organisationsorientierten Systemen erarbeitet. Bei den detaillierten Modellen handelt es sich tatsächlich um ausführbare Prototypen. Sie stellen vor allem einen *Proof of Concepts* hinsichtlich der zuvor abstrakt modellierten und formulieren Struktur- und Verhaltenskonzepte dar. Ihre softwaretechnische Nutzbarkeit steht dabei nicht im Fokus. Dementsprechend handelt es sich auch um reine Referenznetzmodelle ohne Rückgriff auf zusätzliche Java-Funktionalität. Wie im Falle Mulan/Capa können Erweiterungen jedoch zukünftig zu einem leistungsfähigen Softwareprodukt führen.

6.1.4 Weitere grafische Elemente in Renew

In diesem Abschnitt werden einige zusätzliche grafische Elemente für die Modellierung mit Referenznetzen behandelt. Dabei erfolgt eine Beschränkung auf Elemente, die im Rahmen der vorliegenden Arbeit verwendet werden.

Abbildung 6.7 zeigt verschiedene Typen von Kanten. Die Kanten aus Teil (a) der Abbildung entsprechen den bisher bereits verwendeten einseitig gerichteten Kanten. Beim Schalten einer der beiden Transitionen wird die Marke “access” von der Stelle abgezogen und mit Beendigung des Schaltvorganges wieder darauf gelegt. Die beiden Transitionen können also im wechselsei-

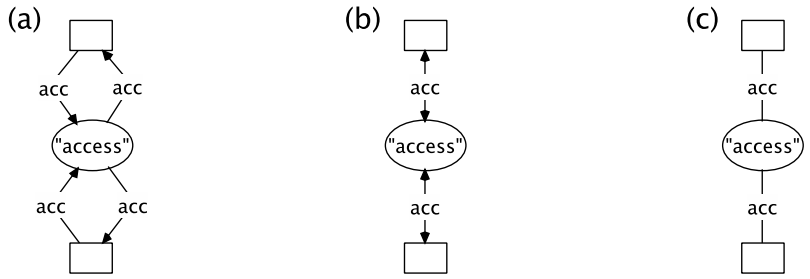

Abbildung 6.7: Verschiedene Typen von Kanten in Renew

tigen Ausschluss schalten. Teil (b) der Abbildung verwendet *Reservierungskanten*. Dieser Teil der Abbildung stellt lediglich eine Kurzschreibweise für die Darstellung aus Teil (a) dar. Auch hier wird beim Schalten einer der beiden Transitionen die Marke von der Stelle abgezogen und mit Beendigung des Schaltvorganges wieder darauf gelegt. Auch hier gilt wechselseitiger Ausschluss bezüglich des Schaltens der beiden Transitionen.

Teil (c) der Abbildung verwendet *Testkanten*. Ihre Verwendung führt zu unterschiedlichem Verhalten als in den anderen beiden Fällen. Beim Schalten einer der beiden Transitionen wird die Marke der Stelle lediglich *getestet*. Das bedeutet im Wesentlichen, dass zwar ein Zugriff auf die Marke erfolgt, sie jedoch nicht von der Stelle abgezogen wird. Weitere Transitionen können über Testkanten – und nur über Testkanten – ebenfalls Zugriff auf die Marke erhalten. Dies führt dazu, dass zwischen den beiden Transitionen kein Konflikt existiert. Beide können nebenläufig Schalten. Von besonderer Wichtigkeit ist die Verwendung von Testkanten, wenn Transitionsanschriften derart sind, dass der Schaltvorgang der Transition gewisse Zeit in Anspruch nehmen kann. Dies kann bei der Verwendung von Java-Anschriften leicht der Fall sein. Allerdings ist speziell im Formalismus der Java-Referenznetze große Vorsicht im Hinblick auf die Verwendung der Marke geboten. Unifikation über Java-Objekten als Marken greift auf Objektidentität (*Hash Code* des Objektes) zurück. Modifikationen können zur Änderung der Objektidentität führen. Greift also eine Transition über eine Testkante auf ein Java-Objekt zu und nimmt eine Modifikation vor, so kann es zu einem Fehlerfall kommen Darüber hinaus können Operationen auf einem Java-Objekt in einigen Fällen zwar nicht zur Änderung der Objektidentität führen, dabei aber dennoch Seiteneffekte auf den internen Objektzustand haben. Stehen solche Operationen als Anschrift an einer Transition, so werden sie *während* der Bindungssuche ausgeführt, unabhängig davon, ob die Transition schaltet oder nicht. Dieses kann über das Schlüsselwort action für eine Aktionsanweisung unterbunden werden. Aktionsanweisungen werden erst beim Schalten der Transition ausgeführt.

Ein weiteres grafisches Element dient vor allem der Übersichtlichkeit von Netzmodellen und ist für größere Netzmodelle häufig unerlässlich. Abbildung 6.8 zeigt die Verwendung einer *virtuellen Stelle*. Das Verhalten der beiden Netze aus den Teilen (a) und (b) der Abbildung ist äquivalent. In Teil (b) der Abbildung stellt die doppelt umrandete Stelle eine exakte Kopie der anderen Stelle dar. Die Markierung der virtuellen Stelle ist also lediglich eine Spiegelung der Markierung der Ursprungsstelle. Hier liegt also genau *eine* Marke des Wertes "access" vor. Alle vier Transitionen können nur im wechselseitigen Ausschluss zueinander schalten.

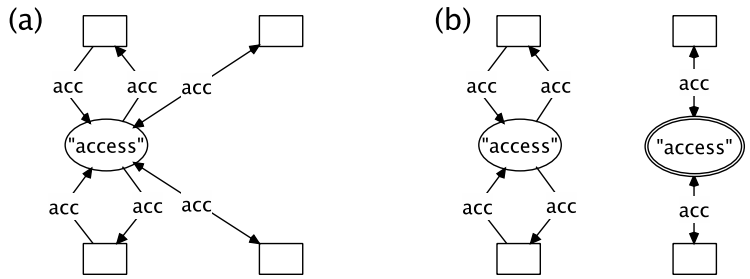

Abbildung 6.8: Verwendung einer virtuellen Stellen

6.2 Referenznetze für die Modellierung sozialer Systeme

Im zweiten Teil der Arbeit wird auf Petrinetze als Technik für die Modellierung von organisationsorientierten Softwaresystemen zurückgegriffen. Diese Herangehensweise hat sich bereits an verwandter Stelle bewährt: Im Rahmen des Sozionik-Forschungsprojektes ASKO[4] und des Nachfolgeprojektes DISPO[5] wurden Petrinetze für die Modellierung soziologischer Theorien verwendet. Ein großer Teil der Ergebnisse der Projekte ist in den Buchbänden von von Lüde et. al. [vLMV03, vLMV09] veröffentlicht.[6] Ein Ziel des Sozionik-Forschungsvorhabens ist unter anderem die Entwicklung von sozialtheoretisch fundierten Multiagentensystemen. Ein Gewinn soll sich sowohl für die Informatik als auch für die Soziologie ergeben. Die Versprechungen einer soziologisch fundierten Sichtweise auf Softwaresysteme ist im Wesentlichen deckungsgleich mit den Ausführungen aus Teilabschnitt 3.1: Die moderne Gesellschaft bietet ein reichhaltiges Reservoir an Vorbildern für die Beherrschung von verteilten, groß angelegten Systemen bestehend aus eigenständigen, zweckgerichteten Komponenten/Akteuren. Dabei kann die Informatik von der Adaptivität, Robustheit, Skalierbarkeit und Reflexivität sozialer Systemen lernen und deren Bauprinzipien in leistungsfähige Technologie umsetzen. Der Gewinn für die Soziologie auf der anderen Seite besteht zum einen in der Erweiterung ihres Methodenschatzes. Allein der Versuch der formalisierten Modellierung soziologischer Theorien kann gegebenenfalls helfen, Unklarheiten und Inkonsistenzen aufzudecken. Zum anderen besteht der Gewinn darin, mit Multiagentensystemen ein Simulationswerkzeug zur Überprüfung und Ausarbeitung der eigenen Begriffe, Modelle und Theorien an die Hand zu bekommen.

Auch wenn die Untersuchungen und Ergebnisse der ASKO/DISPO-Projekte vielfach einen universelleren Charakter haben, so standen von Informatikseite doch vor allem Multiagentensysteme und damit die Metapher des individuellen Agenten sowie umgebende soziale Strukturen im Mittelpunkt. Wie in den Kapiteln 3 und 4 expliziert, stehen im Rahmen der vorliegen Arbeit vor allem organisierte Kollektivitäten als kollektive Akteure im Zentrum der Betrachtung. Der Unterschied besteht demnach im Abstraktionsniveau der im Kern betrachteten sozialen Entitäten. Ganz offensichtlich ist das Anliegen in beiden Fällen jedoch sehr ähnlich. Es geht um den Transfer sozialer Konzepte (im Falle der vorliegenden Arbeit insbesondere von organisationstheoretischen Konzepten) auf die Ebene der Gestaltung von Softwaresystemen.

[4] **A**gieren in **s**ozialen **Ko**ntexten

[5] **Di**rigismus und **S**ymbolische **Po**litik am Beispiel universitärer Governance-Strukturen

[6] Weitere Informationen zu den Projekten sind im Internet unter http://www.informatik.uni-hamburg.de/TGI/forschung/projekte/sozionik verfügbar.

Dabei ergeben sich auch analoge Herausforderungen. Insbesondere ist eine Beschreibungstechnik notwendig, die zwischen den zunächst einmal sehr verschiedenen Domänen der textuellen soziologischen Theorien und der Softwaretechnik vermittelt. Im Falle der ASKO/DISPO-Projekte wurde die Ebene, die diese Lücke schließt, in Form von Petrinetzen gefunden. In diesem Zusammenhang wurde der Begriff der „vermittelnden Netze“ geprägt.

Petrinetze besitzen eine anschauliche grafische Darstellungsform. Diese bringt lediglich vier grafische Elemente (Stellen, Transitionen, Kanten, Marken) mit sich und bietet somit einen leichten Zugang zu Kernkonzepten wie Verteilung, Nebenläufigkeit, Konflikt und Synchronisierung. Die Modellierungselemente sind hochabstrakt und somit universell einsetzbar. Petrinetzmodelle basieren auf der Dualität von aktiven und passiven Elementen und bieten auf diese Weise eine intuitiv verständliche Semantik von Struktur und Verhalten eines modellierten Systems über das Petrinetzmarkenspiel. In den ASKO/DISPO-Projekten stellten sich diese Eigenschaften als gute Voraussetzungen heraus, die Modellierung mit Petrinetzen in den Methodenapparat der Soziologie aufzunehmen. Die zur grafischen Darstellung äquivalente mathematische Darstellung und die exakte operationale Semantik von Petrinetzen sichert jederzeit die Anschlussfähigkeit der Informatik. Werkzeuge erlauben nicht nur die Repräsentation und Analyse der statischen Struktur von Petrinetzen, sondern zusätzlich die Simulation und Animation des Markenspiels, also des Ablaufs von Prozessen. In Kombination mit einer entsprechenden Anschriftensprache handelt es sich bei solchen Prozessen tatsächlich um Softwareprogramme. Zusammenfassend bieten Petrinetze eine Vermittlungsschicht zwischen den beiden zunächst artfremden Gebieten der Informatik und Soziologie.

Im folgenden werden zwei konkrete Beispiele von Petrinetzmodellen in der Schnittmenge von Informatik und Soziologie exemplarisch vorgestellt. Die Auswahl ausgerechnet dieser beiden Beispiele erfolgt aufgrund deren inhaltlicher Nähe zur vorliegen Arbeit. Abbildung 6.9 zeigt ein Übersichtsmodell der Multiagentensystemarchitektur Mulan[7] nach Rölke [Röl04]. Mulan ist ein vierstufiges Modell einer Architektur für Multiagentensysteme basierend auf dem Konzept der Netze in Netzen. *Agenten* sind auf *Plattformen* beheimatet und nehmen an *Konversationen* mit anderen Agenten teil.[8] Konversationen können zwischen Agenten auf derselben Plattform stattfinden oder auch plattformübergreifend mit Agenten auf anderen Plattformen des gesamten *Multiagentensystems*. Die Abbildung zeigt eine abstrakte Sicht auf Mulan. Tatsächlich steckt dahinter ein ausführbares Modell und damit die Möglichkeit der petrinetzbasierten Simulation von Agentensystemen. Mulan wurde erstmals von Köhler et. al. [KMR01] vorgestellt und in der Dissertation von Rölke [Röl04] zur Vollendung gebracht. Während Mulan selbst über diverse Stufen von einem abstrakten zu einem ausführbaren Modell gereift ist, haben Duvigneau et. al. [DMR03] sogar die vollständig FIPA-konforme, verteilbare und softwaretechnisch nutzbare Erweiterung Capa[9] vorgestellt. Das Capa-Framework wurde im Rahmen verschiedener studentischer Projekte an der Universität Hamburg eingesetzt und ist mittlerweile in den umfassenden Softwareentwicklungsansatz Paose[10] eingebettet (vergleiche die Arbeiten von Cabac et. al. [Cab10, CDD+08, Cab07, Mol06]). Mulan/Capa ist also ein sehr gutes Beispiel für den in Teilabschnitt 6.1.3 benannten Ansatz der Implementation durch

[7] **Mul**ti-**A**gent **N**ets

[8] Rölke zitiert in seiner Arbeit die Arbeit von Moldt [Mol96] zur Spezifikation objektorientierter Softwaresysteme. Anhand der Gegenüberstellung des Petrinetzmodells eines Agenten nach Rölke und des Petrinetzmodells eines Objektes nach Moldt lassen sich die konzeptionellen Unterschiede der beiden Entitäten (vergleiche Teilabschnitt 5.1.2 der vorliegenden Arbeit) sehr anschaulich nachvollziehen.

[9] **C**oncurrent **A**gent **P**latform **A**rchitecture

[10] **P**etrinet–Based **A**gent–**O**riented **S**oftware **E**ngineering

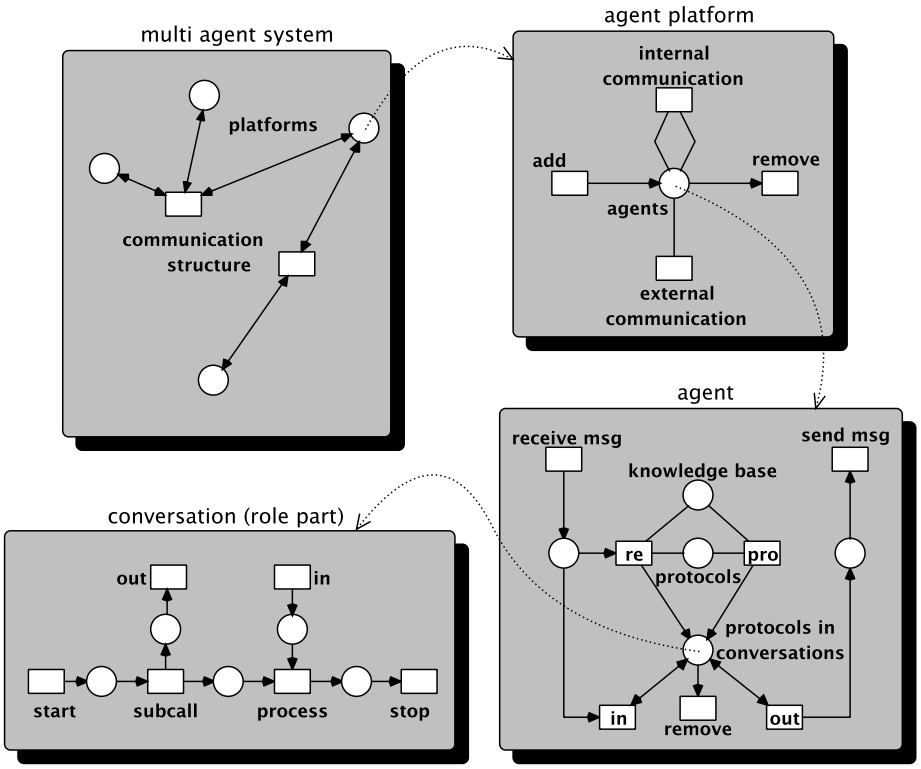

Abbildung 6.9: Multiagentensystemarchitektur MULAN nach [Röl04, S. 157]

Spezifikation mit Petrinetzen (von abstrakten Modellen über ausführbare Prototypen bis hin zu ausgereiften Softwareprodukten).

Die MULAN-Architektur ist in verschiedener Weise nutzbar. Für CAPA wird MULAN eher als ein Modell der physikalischen/technischen Umsetzung von Multiagentensystemen verwendet. Plattformen dienen vor allem der softwaretechnischen *Ausführung* von Softwareagenten. MULAN kann aber auch als abstraktes (und vor allem sozial fundiertes) *Denkmodell* dienen. Bei Plattformen kann es sich dann um soziale Kontexte (Gruppen, Institutionen), die eine Zwischenebene zum umfassenden Gesamtsystem darstellen, handeln. In diesem Fall können Agenten auch auf mehreren Plattformen beheimatet sein, es handelt sich somit eher um *logische* anstatt physikalische Plattformen. Eine solche Interpretation des MULAN-Modells wird beispielsweise im Kontext der Architektur SONAR[11] nach Köhler-Bußmeier [KB09] vorgenommen. SONAR wurde bereits in Kapitel 5 erwähnt und wird in Kapitel 10 näher beleuchtet. An dieser Stelle soll der Bezug zum MULAN-Denkmodell beziehungsweise die spezielle Interpretation von MULAN aufgezeigt werden. Diese Interpretation ist in Abbildung 6.10 illustriert. Die Multiagentensystemebene nach MULAN ist hier eine *Organisation*, im Wesentlichen verkörpert durch ein Delegationsnetzwerk zwischen Organisationspositionen. Die Plattformen nach MULAN sind hier *Positionsagenten*. Bei einem Positionsagenten handelt es sich um ein reifiziertes organisatorisches Artefakt. *Mitgliedsagenten* werden eingestellt, um Positionen auszufüllen und können auch wieder von ihrer Position entbunden werden. Analog zu realweltlichen Ge-

[11] **S**elf-**O**rganizing **N**et **Ar**chitecture

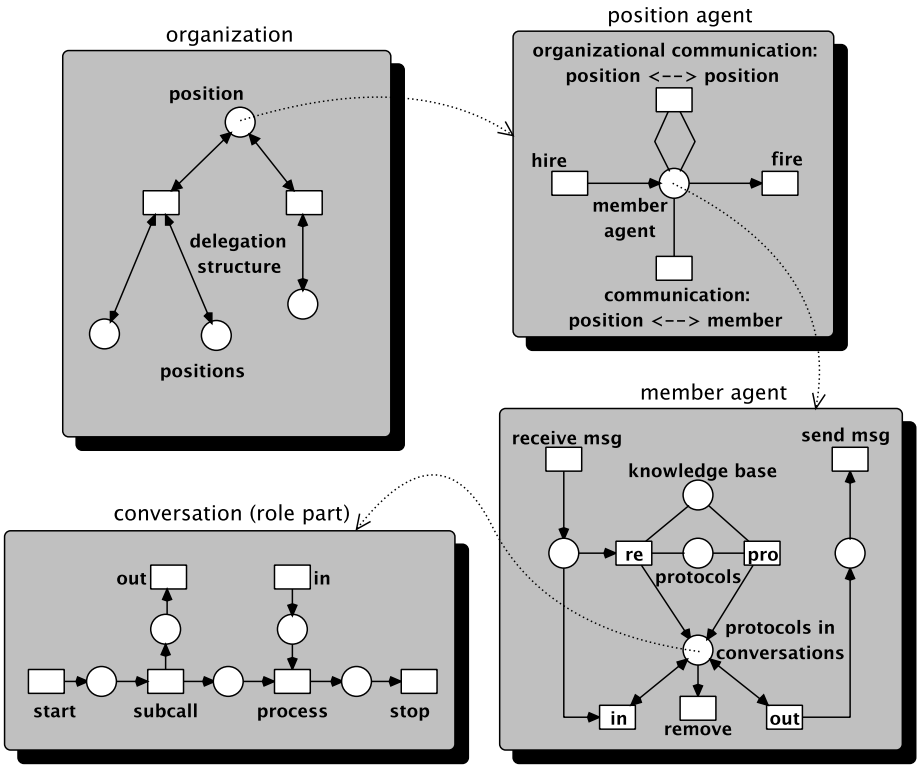

Abbildung 6.10: Sonar-Architektur als Interpretation von Mulan, nach [KB09, S. 451]

sellschaften können konkrete Agenten Positionen in mehreren Organisationen einnehmen und somit auf mehreren „Plattformen“ beheimatet sein.

Analog der Konkretisierung/Erweiterung der abstrakten Mulan-Architektur durch das Capa-Framework wurden auch für die abstrakte Sonar-Architektur aus Abbildung 6.10 Konkretisierungen/Erweiterungen vorgenommen, die schließlich in einem ausführbaren (prototypischen) Framework für Agentenorganisationen resultierte. Der erste Prototyp des Mulan4Sonar-Frameworks nach Köhler-Bußmeier und Wester-Ebbinghaus [KBWEM10, KBWE09b] basiert auf dem Prinzip, aus einem konkreten Sonar-Modell einer Organisation die oberste Ebene der Sonar-Architektur direkt zu *generieren*, so dass eine konkrete Agentenorganisation mit explizit verkörperter Formalstruktur entsteht.

Mulan und Sonar stellen damit also zwei verwandte Beispiele der sozial fundierten Modellierung von Systemen dar. Insbesondere das Modellierungskonzept der Netze in Netzen erweist sich hier als ideale Strukturierungshilfe. In der hier gezeigten Form kommen die Modelle vollständig mit der Namensgebung für Netzelemente aus. Weitere Anschriften gibt es nicht. Trotzdem ist trotz dieser Abstraktionsebene ein intuitives Verständnis der Struktur und des Verhaltens der modellierten Systeme möglich. Zusätzliche Details legen zusätzliche Aspekte, Konzepte und Mechanismen offen. Der Informatiker ist letzten Endes an ausführbaren oder zumindest implementierungsnahen Modellen interessiert. Dem Soziologen reicht aber gegebenenfalls ein Modell auf höherer Abstraktionsebene für seine Theoriebildung.

6.3 Abstraktionsstufen der Modellierung in dieser Arbeit

Die Referenznetzmodellierung kann auf verschiedenen Ebenen der Abstraktion stattfinden. Der vorhergehende Abschnitt hat diese Möglichkeit und die damit verbundenen Vorteile bereits beleuchtet. In der vorliegenden Arbeit werden in der Folge drei Abstraktionsstufen unterschieden. Abbildung 6.11 veranschaulicht die drei Stufen. Dabei erfolgt ein Rückgriff auf das Fließband-Beispiel aus Teilabschnitt 6.1.3.

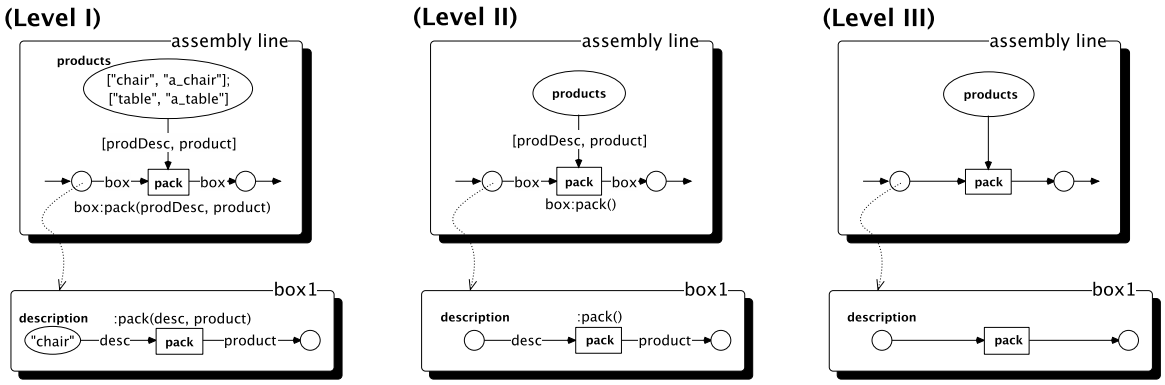

Abbildung 6.11: Drei Abstraktionsstufen der Referenznetzmodellierung in dieser Arbeit

- *Abstraktionsebene I:* Auf der ersten Abstraktionsstufe sind alle Details vorhanden. Das betrifft Marken sowie sämtliche Kanal- und Variablenanschriften. Die Modelle sind direkt ausführbar.

- *Abstraktionsebene II:* Auf der zweiten Abstraktionsstufe sind hinsichtlich der Marken nur noch die Netzmarken modelliert, um weiterhin ein Bild der Einbettungsbeziehungen und damit der Systemkomposition zu vermitteln. Kanal- und Variablenanschriften sind weiterhin aufgeführt. Allerdings wird vom Informationsfluss über Kanäle hinweg abstrahiert, Kanalanschriften tragen somit keine Parameterinformationen mehr. Auf dieser Abstraktionsebene ist es also in den meisten Fällen nicht mehr möglich, Unifikationsmechanismen für die Bindungssuche nachzuvollziehen.

- *Abstraktionsebene III:* Auf der dritten Abstraktionsebene sind keinerlei Kanal- oder Variablenanschriften mehr aufgeführt. Einbettungsbeziehungen zwischen Netzen sind weiterhin repräsentiert. Das Verständnis des Betrachters für das Synchronisationsverhalten des modellierten Systems basiert dabei allerdings nur noch auf der Namensgebung für Stellen und Transitionen (und gegebenenfalls auf weiteren Erläuterungen im begleitenden Text).

6.4 Zusammenfassung

In diesem Kapitel wurden Möglichkeiten der Modellierung mit Petrinetzen betrachtet. Ausgehend von einfachen S/T-Netzen wurden gefärbte Netze, das Konzept der Netze in Netzen und

Referenznetze beleuchtet. Beispiele wurden jeweils anhand der konkreten Modellierungsmöglichkeiten des Werkzeugs Renew betrachtet. Schließlich wurde die Eignung der Petrinetzmodellierung für die Modellierung sozialer beziehungsweise sozial fundierter Systeme anhand des Erfolgs an anderer Stelle und entsprechender Beispiele aufgezeigt.

Insbesondere das Konzept der Netze in Netzen wird für das Modellierungsvorhaben des zweiten Teils der vorliegenden Arbeit eine bedeutende Rolle spielen. Die Verwendung und Explizierung verschiedener Systemebenen und die damit einhergehende Schachtelung von Systemeinheiten stellt eine erhebliche Strukturierungshilfe dar. Anhand der bisherigen Ausführungen zu Systemen von Systemen, Mikro/Makro-Dynamik und kollektiven Akteuren wird offenbar, dass dieser Aspekt im Kontext der vorliegenden Arbeit ein besonders starkes Gewicht bekommt.

Die Idee der Netze in Netzen führt (nicht-disjunkte) Hierarchien beziehungsweise Rekursivität sowie Synchronisationsmechanismen über multiple Systemebenen hinweg als erstrangige Konzepte ein und unterstützt auf diese Weise die vertikale Strukturierung von System-von-Systemen-Modellen inhärent. Die exakte Semantik als eine große Stärke der Modellierung mit Petrinetzen bleibt dabei jederzeit gewahrt. Das Markenspiel, in dem sich das Zusammenspiel von Struktur und Verhalten eines Petrinetzes manifestiert, wird um die Dimension der aktiven, mobilen Netzmarken erweitert. Netze in Netzen erlauben die Modellierung beliebig tief geschachtelter, dynamischer und adaptiver Systemkonfigurationen. Darüber hinaus ist der Autor der vorliegenden Arbeit der Ansicht, bereits die Ähnlichkeit der Namensgebung deute die ideale Passung von *Netzen in Netzen* als Modellierungstechnik für *Systeme von (in) Systemen* an.

Teil II

Modellierungstechniken und Grundmodelle

7 Eingrenzung und Anforderungen

Der zweite Teil der Arbeit ist der Modellierung von organisationsorientierten Softwaresystemen gewidmet. In diesem Kapitel wird eine inhaltliche Eingrenzung des Modellierungsvorhabens sowie die Erarbeitung eines Anforderungskatalogs vorgenommen. Die Eingrenzung wird in Abschnitt 7.1 vorgenommen. Darüber hinaus wird die im weiteren Verlauf der Arbeit verwendete Terminologie eingeführt. In Abschnitt 7.2 wird darauf aufbauend und unter Einbezug der organisationstheoretischen Grundlagen aus den Kapiteln 3 und 4 ein Anforderungskatalog für das Modellierungsvorhaben der beiden folgenden Kapitel erstellt.

7.1 Inhaltliche Eingrenzung und verwendete Terminologie

Im ersten Teil der Arbeit wurde ausgehend von den Charakteristika großangelegter Softwaresysteme der Kategorie *Systeme von Systemen* und den damit einhergehenden Problemstellungen die Motivation für ein Organisationsparadigma der Softwareentwicklung vorgetragen. Die Kernidee ist dabei die Gestaltung von Softwaresystemen entlang von Konzepten und Mechanismen als dem Bereich sozialer Organisationssysteme. In Kapitel 5 wurde beschrieben, dass diese Art Softwaresysteme insbesondere im Bereich der Multiagentensysteme bereits auf den Weg gebracht wurden. Die Kritik am *State of the Art* der Multiagentensysteme wurde an der Vernachlässigung der Funktion von organisierten Kollektivitäten als kollektive Akteure festgemacht. Es gibt zwar Ansätze für die technische Realisierung kollektiver Akteure, inhaltliche Betrachtungen beziehen sich jedoch nahezu ausschließlich auf softwaretechnische Kollektivitäten als Kontexte und nicht auf Kollektivitäten als Akteure.

Als Grundlage für den weiteren Verlauf der vorliegenden Arbeit wird daher eine konzeptionelle Erweiterung von Multiagentensystemen betrachtet. Abbildung 7.1 zeigt eine kanonische Sichtweise auf Softwaresysteme mit Kollektivitäten, die sowohl Kontexte als auch Akteure sein können.[1] Auf der untersten Ebene befindet sich die softwaretechnische Laufzeitumgebung in Form von Plattformen, auf denen alle Komponenten des betrachteten Softwaresystems ausgeführt werden. Die Plattformen werden hier als *softwaretechnische Deployment-Plattformen* bezeichnet. Multiple Plattformen sind mittels einer Kommunikations- und Mobilitätsinfrastruktur zu einem vernetzten Gesamtsystem verbunden. Auf der zweiten Ebene finden sich *„klassische“ Agenten als individuelle Akteure* wieder. Sie werden auf den Deployment-Plattformen ausgeführt und können über die plattformübergreifende Infrastruktur zwischen Plattformen migrieren und miteinander kommunizieren (vergleiche Teilabschnitt 5.1.1). Der Einfachheit halber wird hier angenommen, jeder Agent werde auf genau einer Deployment-Plattform ausgeführt. Auf der dritten Ebene schließlich finden sich *Kollektivitäten als logische Plattformen* wieder. Der Begriff der logischen Plattform wird in Abgrenzung zu den Deployment-Plattformen verwendet. Ebenso wie Deployment-Plattformen stellen Kollektivitäten eine Ausführungsumgebung dar. Allerdings geht es hier nicht um eine softwaretechnische Ausführung

[1] Die Abbildung zeigt ein Schema für Softwaresysteme. Verbindungen zu physikalischen Systemen oder sozialen Akteuren für den Fall von softwareintensiven Gesamtsystemen sind hier nicht berücksichtigt, können aber bei Bedarf einbezogen werden.

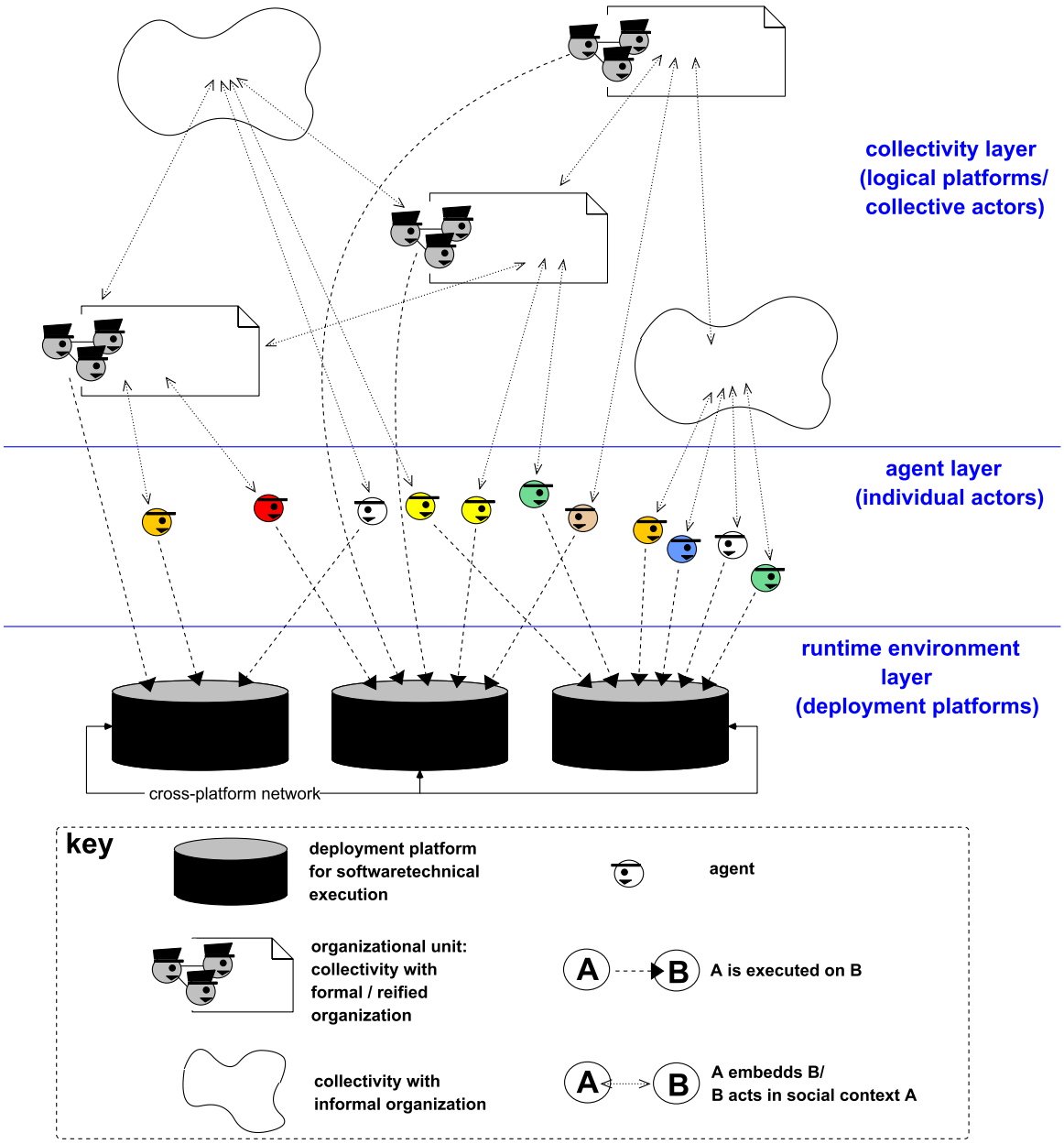

Abbildung 7.1: Kanonische Sichtweise auf Multiorganisationssysteme

von Komponenten, sondern um anwendungsbezogene (künstlich) soziale Kontexte, in denen die Komponenten ihrer jeweiligen Funktion gerecht werden können.

Zwei Formen von Kollektivitäten werden hier unterschieden. In beiden Fällen handelt es sich um eine Manifestation organisatorischer Beziehungen zwischen Akteuren. Die Unterscheidung geschieht analog der Unterscheidung zwischen Agentenorganisationen in den Quadranten (b) und (d) der Klassifikation aus Abbildung 5.2 in Kapitel 5. Im Falle der Kollektivitäten mit informeller Organisation manifestieren sich die organisatorischen Beziehungen lediglich in Form

interner Repräsentationen der beteiligten Akteure. Sie erhalten damit keine eigenständige, von ihren Mitgliedern unabhängige softwaretechnische Verkörperung. Im Falle der Kollektivitäten mit formaler Organisation werden Organisationsspezifikationen und Mechanismen für ihre Administration eigenständig softwaretechnisch verkörpert. Es findet also eine explizite Trennung der softwaretechnischen Realisierung von Formalorganisation und Mitgliedern statt. Ebenso wie Agenten werden die entsprechenden Strukturen also auf Deployment-Plattformen ausgeführt. Der Einfachheit halber wird hier auch angenommen, die Formalorganisation einer Kollektivität werde auf genau einer Deployment-Plattform ausgeführt.[2] Für beide Formen von Kollektivitäten wird grundsätzlich angenommen, sie könnten als kollektive Akteure im Kontext weiterer Kollektivitäten auftreten. Dadurch entstehen inhaltliche/logische Einbettungsbeziehungen zwischen den Komponenten eines Multiorganisationssystems zusätzlich zu den softwaretechnischen/„physikalischen“ Einbettungsbeziehungen von Komponenten zu Deployment-Plattformen.

Anhand der Ausführungen aus Kapitel 5 wird deutlich, dass Multiagentensysteme eine *technische Realisierung* von Systemen nach Abbildung 7.1 bereits weitestgehend unterstützen. Es existieren sowohl Frameworks für die softwaretechnische Integration von Agenten und Organisationsinfrastrukturen als auch Möglichkeiten, Kollektivitäten als Akteure auftreten zu lassen. Ebenso machen die Ausführungen aus Kapitel 5 allerdings den erheblichen Forschungsbedarf hinsichtlich einer *inhaltlichen Realisierung* von Systemen nach Abbildung 7.1 deutlich. Nicht umsonst ist gerade im Bereich der elaborierten Middleware-Unterstützung für Agentenorganisationen bislang kein kollektives Akteurswesen vorgesehen. Aus diesem Missstand leitet der Autor der vorliegenden Arbeit die Rechtfertigung ab, Softwaresysteme nach Abbildung 7.1 als eine konzeptionelle Erweiterung der Multiagentensysteme aufzufassen. Um die Fokusverschiebung vom individuellen zum kollektiven Akteur zu verdeutlichen, werden sie in der Folge als *Multiorganisationssysteme* bezeichnet. Dabei stellen Kollektivitäten mit *formaler* Organisation die gegenüber Multiagentensystemen „neuartigen“ Softwarebausteine dar. Sie kombinieren die bisher separaten Konzepte des softwaretechnischen Akteurs und des softwaretechnischen Organisationskontextes. Sie werden in der Folge als *Organisationseinheiten* bezeichnet. Im Gegensatz zu Multiagentensystemen weisen Multiorganisationssysteme eine inhärent rekursive Struktur auf.

Das zweite übergeordnete Ziel aus der Einleitung der Arbeit (Vermittlung eines klaren Verständnisses für organisationsorientierte Softwaresysteme anhand der Modellierung von Multiorganisationssystemen als konkreter Ausprägung) kann nun wieder aufgegriffen und inhaltlich genauer beschrieben werden. Das konkrete Ziel des zweiten Teils der Arbeit besteht in der Bereitstellung von Modellierungstechniken für Organisationseinheiten und Multiorganisationssysteme und der Erstellung von intuitiv zugänglichen Grundmodellen. Dabei wird eine durchgängige Fundierung auf operationaler Petrinetzsemantik verlangt. Darauf aufbauend werden im dritten Teil der Arbeit konkrete Beispiele behandelt und eine Referenzarchitektur für Multiorganisationssysteme abgeleitet. In der Folge werden einige inhaltliche Eingrenzungen für das Modellierungsvorhaben im zweiten Teil der Arbeit vorgenommen. Es gilt, ein handhabbares und in sich weitestgehend abgeschlossenes Paket zu schnüren.

1. ***Die Kollektivitätsmodellierung ist auf Organisationseinheiten beschränkt.*** Die in Abbildung 7.1 ebenfalls aufgeführten Kollektivitäten mit informeller Organisation werden nicht betrachtet. Dies manifestiert sich bereits in der Wahl der aus der Organi-

[2] Diese Einschränkung kann für konkrete Anwendungsfälle aufgehoben.

sationsliteratur zitierten Grundlagen für die Gestaltung organisationsorientierter Softwaresysteme in Kapitel 4. Die dort betrachteten Prinzipien, Konzepte und Mechanismen beziehen sich auf organisierte Kollektivitäten, deren herausstehende Charakteristika Zielspezifität und formalisierte Steuerungs- und Kontrollstrukturen sind. Gegenüber informellen Kollektivitäten sorgen diese beiden Charakteristika für eine höhere „Sichtbarkeit“ organisierter Kollektivitäten sowie für einen einfacheren Zugang zu Steuerung, Kontrolle und Nachvollziehbarkeit ihrer Handlungen. Insbesondere wird dadurch eine softwaretechnische „Nachbildung“ unterstützt.

Das Modellierungsvorhaben ist daher auf Organisationseinheiten beschränkt. Weiter oben wurde auf die inhärent rekursive Struktur von Multiorganisationssystemen hingewiesen. Die Modellierung von Organisationseinheiten führt somit automatisch zur Modellierung von Multiorganisationssystemen, so dass letztere zumindest für die allgemeine Modellbildung nicht separat betrachtet werden. Die Entwicklung *konkreter* Multiorganisationssysteme erfordert allerdings zusätzliche Architekturprinzipien dafür, welche Typen von Organisationseinheiten auf welche Art und Weise kombiniert werden.

Die Begriffe *Organisationseinheit* und *Multiorganisationssystem* wurden als Bezeichnungen für die softwaretechnischen Entsprechungen der realweltlichen/soziologischen Konzepte *organisierte Kollektivität* und *Organisationssystem* eingeführt. Diese terminologische Trennung sei hier explizit hervorgehoben. Zwar werden Organisationseinheiten und Multiorganisationssysteme entlang realweltlicher *Inspirationen* modelliert, punktuell können allerdings vereinfachende Annahmen getroffen werden (analog zum Verhältnis des softwaretechnischen Agenten zum Vorbild des menschlichen Akteurs). Inwieweit die in dieser Arbeit vorgestellten Modellierungstechniken und Grundmodelle für die Beschreibung natürlicher/sozialer Organisationssysteme dienen können, ist zukünftiger Forschung vorbehalten. Besonders interessant wird dabei die Frage sein, wie softwareintensive Gesamtsysteme als Kombination aus Softwaresystemen, sozialen Systemen und gegebenenfalls physikalischen Systemen integriert modelliert werden können.

2. ***Das Zusammenspiel von formalen und informellen Strukturen wird nicht explizit thematisiert.*** Auch im Falle von softwaretechnischen Organisationskontexten kann es zu dem in den Kapiteln 3 und 4 für soziale Organisationen beschriebenen Wechselspiel zwischen formalen und informellen Strukturen kommen. Dies wurde in Kapitel 5 im Zusammenhang mit Agentenorganisationen beschrieben. Insofern stellt Abbildung 7.1 eigentlich eine Vereinfachung dar. Im Bereich jeder Organisationseinheit müsste strenggenommen auch eine informelle Kollektivität parallel dazu aufgeführt sein.

   Mechanismen des wechselseitigen Einflusses von formalen und informellen Strukturen sind jedoch sehr diffizil und werden in der Organisationstheorie kontrovers diskutiert. Eine Übertragung von konkreten Theorien zu diesem Thema auf Softwareebene stellt ein eigenständiges Forschungsthema dar. In dieser Arbeit wird schlicht davon ausgegangen, dass informelle Strukturen neben den formalen Strukturen existieren können und gegebenenfalls einen Einfluss auf die Funktionsweise einer Organisationseinheit haben. Explizite Berücksichtigung in den Modellen erhalten jedoch nur formale Strukturen sowie das durch diese Strukturen reglementierte Verhalten.

   Insgesamt gilt aufgrund der expliziten Trennung des Formalanteils einer Organisationseinheit und den Mitgliedern in Kombination mit der Entstehung informeller Strukturen, dass Handlungen unter Umständen „an der Formalstruktur vorbei“ stattfinden (analog

zu sozialen Gesellschaften). Es kann nicht davon ausgegangen werden, dass mit Hilfe von Organisationseinheiten unerwünschtes Systemverhalten kategorisch ausgeschlossen werden könnte. Es ist aber möglich, bestimmte Bereiche eines Systems als Organisationseinheiten zu realisieren und Zugang zu kritischen Ressourcen ausschließlich über formalstrukturell reglementierte Handlungen zuzulassen.

3. ***Von individuellen Akteuren in Form von Agenten wird abstrahiert.*** In der vorliegenden Arbeit wird der Entwicklung im Bereich der organisationsorientierten Programmierung von Multiagentensystemen (Quadrant (d) aus Abbildung 5.2 in Kapitel 5) gefolgt: Von „klassischen" Agenten als individuellen Akteuren wird bei der Modellierung weitestgehend abstrahiert. Natürlich sind es die individuellen Akteure, die formale Spezifikationen überhaupt erst mit Leben füllen. In Multiorganisationssystemen mit kollektiven Akteuren gilt dies über multiple Einbettungsebenen hinweg. Zwar mögen Organisationseinheiten als eigenständige Akteure auftreten, doch ohne individuelle Akteure auf der Agentenebene eines Multiorganisationssystems handelt es sich bei allen Organisationseinheiten darüber lediglich um „leere Hüllen".

   Nichtsdestotrotz vertritt der Autor der vorliegenden Arbeit die Auffassung, die systematische Betrachtung der Einbettung individueller Agenten in einen organisatorischen Kontext sei dem Agentenparadigma zuzuordnen. Hier wird der Wechsel zum Organisationsparadigma vollzogen, so dass die Einbettung von Organisationseinheiten in weitere Organisationseinheiten im Fokus steht. Gelegentlich werden auch hier Interna von Agenten modelliert. Dies bezieht sich aber lediglich auf Schnittstellen und geschieht zu dem Zweck, bestimmte (operationale/technische) Mechanismen zu verdeutlichen.

   Um Agentenorientierung und Organisationsorientierung zunächst analytisch zu trennen, wird die Verwendung des Begriffes *Agent* in der Folge vermieden. Stattdessen wird der Begriff des *Akteurs* verwendet und bei Bedarf zwischen individuellen und kollektiven Akteuren differenziert.

4. ***Artefakte in Form von Materialien, Ressourcen und Werkzeugen werden in den Modellen nicht explizit berücksichtigt.*** In den Kapiteln 3 und 4 wurde an verschiedenen Stellen darauf hingewiesen, der überindividuelle Charakter von Organisationen würde sich nicht nur in Form einer Formalstruktur, sondern ebenfalls im Besitz von Ressourcen sowie physikalischen Grenzen manifestieren (analog dazu wurde in Kapitel 5 der auf Agenten und Artefakten als gleichberechtigten Konzepten beruhende Ansatz ORA4MAS nach Hübner et. al. [HBKR10] referenziert). Für Multiorganisationssysteme nach der obigen Abbildung 7.1 ergeben sich verschiedene Möglichkeiten, dingliche Artefakte neben den Akteuren zu berücksichtigen. Beispielsweise könnte die kanonische Sichtweise auf Multiorganisationssysteme um Artefakte erweitert werden. Alternativ dazu könnten Artefakte Anteile der softwaretechnischen Verkörperung der Formalanteile von Organisationseinheiten sein. Für die vorliegende Arbeit wird eine dritte, einfachere Alternative gewählt. Artefakte werden nicht explizit modelliert und berücksichtigt. Stattdessen können sie als „dumme Agenten" realisiert sein. Diese Herangehensweise ist auch in der Agententechnik üblich. Artefaktagenten bieten die Funktionen/Operationen des durch sie repräsentierten Artefakts nach außen hin an und weisen im Gegensatz zu anderen Agenten keine Autonomie bezüglich ihrer Handlungen und Entscheidungsfindung auf (damit sind sie konzeptionell Objekte).

5. ***Das Deployment von Organisationseinheiten/Multiorganisationssystemen in verteilten Softwareumgebungen und die Entwicklung konkreter Anwendungen werden nicht vertieft diskutiert.*** Bei diesem Punkt geht es um den Zweck des Modellierungsvorhabens im zweiten Teil der Arbeit. Die erarbeiteten Grundmodelle sowie die vorgestellten Modellierungstechniken dienen dem Zweck, Organisationseinheiten und Multiorganisationssysteme zu modellieren. Dabei liegt der Fokus auf der Modellierung von Strukturen und Abläufen auf der logischen Ebene eines Multiorganisationssystems. Von der softwaretechnischen Ausführungsebene wird weitestgehend abstrahiert.

   Einerseits werden hier abstrakte Perspektiven für die Erstellung von *Denkmodellen* unterstützt. Andererseits werden aber auch detaillierte, feingranulare Perspektiven für die Erstellung von *Programmiermodellen* unterstützt. Letztere werden auf Basis der in Kapitel 6 eingeführten Referenznetze erstellt. Sie sind im Kontext des Werkzeugs Renew direkt ausführbar. Diese direkte Ausführbarkeit stellt vor allem einen komfortablen *Proof of Concepts* dar. Man kann Renew (gegebenenfalls sogar in mehreren Instanzen über ein Rechnernetzwerk verteilt) als Deployment-Plattform und damit als softwaretechnische Ausführungsumgebung auffassen kann. Die Programmiermodelle sind damit automatisch auch softwaretechnische Prototypen. Auf diese Weise wird eine exemplarische Möglichkeit des konkreten Übergangs von der logischen auf die Deployment-Ebene eines Multiorganisationssystems aufgezeigt.

   Von größerer Bedeutung als die direkte Ausführbarkeit mittels eines Petrinetzsimulators ist allerdings, dass die Programmiermodelle ein exaktes technisches Verständnis der Funktionsweise von Organisationseinheiten und der Einbettungsmechanismen zwischen Organisationseinheiten vermitteln. Von einer allgemeineren Warte aus handelt es sich hier vor allem um *konzeptionelle Modelle*, deren Konzepte und Mechanismen auf verschiedene Möglichkeiten des Deployment übertragen werden können. Weiter oben wurden Multiagentensysteme als mögliches Vehikel vorgeschlagen. Wie eine solche Übertragung im allgemeinen Fall aussehen kann, wird in dieser Arbeit nur skizziert.

   Hier ergibt sich eine Analogie zu verwandten Forschungsbestrebungen der Arbeitsgruppe TGI[3], der der Autor der vorliegenden Arbeit angehört. Diese Bestrebungen wurden bereits in Kapitel 6 anhand der Erläuterungen zu Mulan/Capa behandelt. Rölke [Röl04] verfolgt in seiner Dissertation ein Forschungsvorhaben, das ähnlich gelagert ist wie das der vorliegenden Arbeit. Allerdings bezieht sich Rölke auf Agenten- statt Organisationsorientierung. Rölke stellt petrinetzbasierte Modellierungstechniken und Grundmodelle für Agenten und Multiagentensysteme vor, behandelt konkrete Beispiele und leitet eine Referenzarchitektur ab. Die Einbettung beziehungsweise Überführung der Ergebnisse von Rölke in ein verteiltes Framework, die Unterstützung durch Entwicklungswerkzeuge und ein konkretes Entwicklungsvorgehen (Methodologie) stellte ein mehrjähriges, eigenständiges Forschungsvorhaben dar und resultierte in der Dissertation von Cabac [Cab10]. Analog sieht der Autor der vorliegenden Arbeit die Ergebnisse der Arbeit als Grundlage für anschließende Forschung, die sich mit einer Überführung der Ergebnisse in konkrete softwaretechnische Kontexte beschäftigt.

[3] *Theoretische Grundlagen der Informatik* am Fachbereich Informatik der Universität Hamburg

7.2 Anforderungen

Eine grundlegende Anforderung wurde bereits in der Einleitung der Arbeit sowie im vorigen Abschnitt formuliert: Für die Modellierung von Organisationseinheiten und Multiorganisationssystemen werden verschiedene Grade der Modellabstraktion unterstützt. Besonders abstrakte Modelle können als *Denkmodelle* in Phasen des Grobentwurfs dienen. Hier geht es um die Identifizierung analytischer Komponenten des Modellgegenstandes und um die funktionalen Zusammenhänge zwischen ihnen. Besonders detaillierte Modelle können als *Programmiermodelle* für die softwaretechnische Realisierung dienen. Hier geht es um ein exaktes technisches Verständnis der Funktionsweise und der operationalen Zusammenhänge des Modellgegenstandes. Es folgt demnach die Forderung nach Modellierungstechniken, die eine systematische Modulation der Modellabstraktion unterstützen. An dieser Stelle werden die Sichtweise und Terminologie von Cabac [Cab10, S. 9+10] übernommen. *Modelle* sind Abbilder von Systemen beziehungsweise von Systemanteilen und stellen dabei in der Regel Abstraktionen dar. Sie werden häufig in Form von grafischen *Diagrammen* expliziert. Diagramme werden auf Basis von *Modellierungstechniken* erstellt. Durch die Gruppierung mehrerer einem gemeinsamen Zweck dienlichen Modellierungstechniken ergibt sich eine *Modellierungssprache*. Im zweiten Teil der Arbeit werden verschiedene Techniken für die Modellierung von Organisationseinheiten verwendet, so dass sich insgesamt eine Modellierungssprache ergibt.

Zusätzlich zu der grundlegenden Forderung nach Unterstützung von unterschiedlich abstrakten Perspektiven werden im Folgenden konkrete konzeptionelle beziehungsweise inhaltliche Anforderungen an das softwaretechnische Konzept einer Organisationseinheit formuliert. Diese Anforderungen ergeben sich im Wesentlichen aus den Ausführungen in den Kapiteln 3 und 4. Zunächst wurde das Plädoyer für ein Organisationsparadigma der Softwareentwicklung in Kapitel 3 entlang von mehreren Kernmerkmalen realweltlicher Organisationen geführt: (1) Organisationen stellen ordnende Kontexte dar, (2) nehmen eine sozio-technische Integration vor und sind (3) offene Systeme, die als kollektive Akteure auftreten können und damit die (4) Möglichkeit einer systematischen sozial fundierten Meso-Strukturierung von Softwaresystemen bieten. Während die ersten beiden Merkmale im Bereich der Multiagentensystemforschung bereits verstärkt thematisiert werden, bestehen Defizite bezüglich der Aspekte der Offenheit und Meso-Strukturierung. Für ein Organisationseinheitsmodell gilt es, eine Integration aller Aspekte vorzunehmen.

Als Startpunkt wird dem Modellierungsvorhaben das in Abbildung 7.2 als UML-Konzeptdiagramm illustrierte Verständnis von Organisationseinheiten zugrunde gelegt. Die

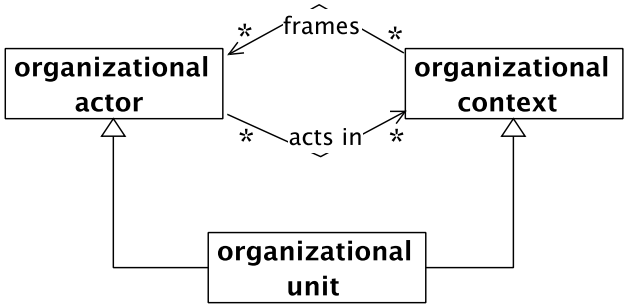

Abbildung 7.2: Grundlegendes Verständnis von Organisationseinheiten

Abbildung unterstreicht den *janusgesichtigen* Charakter von Organisationseinheiten. Eine Organisationseinheit stellt auf der einen Seite einen formalisierten organisatorischen Kontext für Akteure dar. Auf der anderen Seite ist dieselbe Organisationseinheit wiederum ein zielgerichteter Akteur, der im Kontext umschließender Organisationseinheiten agiert. Sie ist ein kollektiver Akteur, konstituiert durch ihre internen Akteure. Genau dieses Verständnis erlaubt einen rekursiven Aufbau von Multiorganisationssystemen.

Dabei besteht das Ziel einerseits in der Bereitstellung geeigneter Modellierungstechniken für Organisationseinheiten, andererseits in der Erarbeitung eines möglichst *schlanken und universellen Grundmodells* einer Organisationseinheit. Die Ausdrucksmächtigkeit des Grundmodells stützt sich darauf, ein gemeinsames Schema für ein breites Spektrum von konkreten Organisationseinheiten bereitzustellen. Insbesondere kann das Grundmodell eine Konkretisierung/Ausgestaltung für Organisationseinheiten auf beliebigen Ebenen eines Multiorganisationssystems erfahren. Während weiter oben also die Unterstützung für die Erstellung verschieden abstrakter *Modelle* gefordert wurde, wird hier eine gemeinsame Basis für die Modellierung verschieden abstrakter *Modellgegenstände* gefordert. Die Konzeptionalisierung von Multiorganisationssystemen ergibt sich in demnach automatisch aus der Konzeptionalisierung von Organisationseinheiten.

Aus den bisherigen Ausführungen folgen unmittelbar drei verschiedene *konzeptionelle Richtungen*, die für jede Organisationseinheit betrachtet werden müssen. Es handelt sich um Richtungen, aus denen die Organisationseinheit beeinflusst wird beziehungsweise in die die Organisationseinheit selbst Einfluss nimmt. Abbildung 7.3 bietet eine Illustration. Aus der

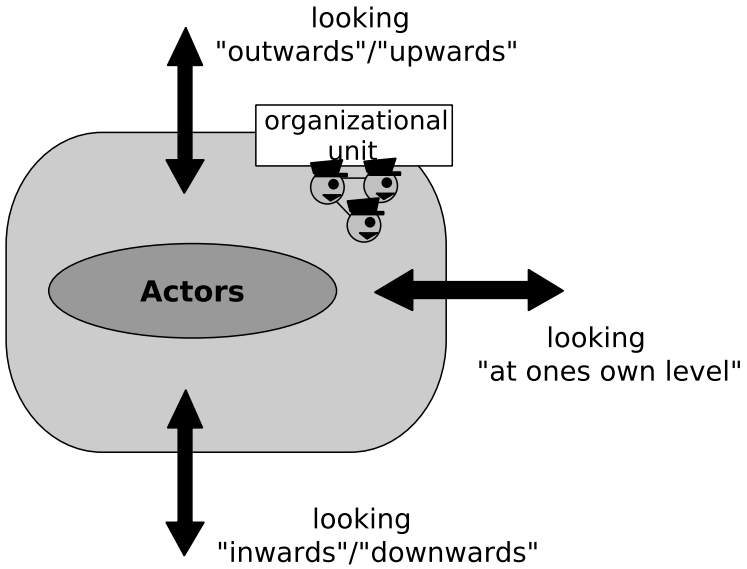

Abbildung 7.3: Konzeptionelle Richtungen bei der Betrachtung von Organisationseinheiten

Perspektive einer fokussierten Organisationseinheit werden die Richtungen des „nach unten (beziehungsweise nach innen) Schauens“, des „auf die eigene Ebene Schauens“ sowie des „nach oben (beziehungsweise nach außen) Schauens“ unterschieden.

Die nach unten gerichtete Perspektive beschäftigt sich damit, wie und entlang welcher Systemregeln interne Akteure in die Organisationseinheit eingebettet werden. Die Bezeichnung

des „nach unten Schauens“ rührt daher, dass es hierbei um die Integration von Akteuren einer tieferen Systemebene geht. Bei der auf die eigene Ebene gerichteten Perspektive stehen die Systemregeln selber im Mittelpunkt. Aktuelle Regeln müssen kontinuierlich evaluiert und gegebenenfalls angepasst werden. Dementsprechend steht die Konfiguration der fokussierten Organisationseinheit als Ganzem im Fokus, wodurch sich die Bezeichnung des „auf die eigene Ebene Schauens“ erklärt. Die nach oben gerichtete Perspektive bezieht sich darauf, wie sich die Organisationseinheit als kollektiver Akteur in weitere, umschließende Organisationseinheiten einbettet. Diese Perspektive muss konsistent mit den anderen beiden Perspektiven sein, um einen kohärenten Zusammenhang zwischen internem Zusammenhalt und dem kollektiven Akteursverhalten der Organisationseinheit herzustellen. Bei den umschließenden Organisationseinheiten handelt es sich um solche einer höheren Systemebene, dementsprechend folgt die Charakterisierung des „nach oben Schauens“ aus Sicht der fokussierten Organisationseinheit.

Die analytische Differenzierung entlang der drei konzeptionellen Richtungen liefert einen groben Rahmen für die Berücksichtigung inhaltlicher Aspekte bei der Modellierung von Organisationseinheiten. Im Folgenden werden konkrete Fragestellungen bezüglich des Zusammenspiels zwischen internem Zusammenhalt und kollektivem Akteurswesen einer Organisationseinheit aufgeworfen. Diese Fragestellungen leiten sich unmittelbar aus den Betrachtungen in Kapitel 4 zu realweltlichen organisierten Kollektivitäten ab. Sie müssen nun bei der Modellierung von softwaretechnischen Organisationseinheiten berücksichtigt werden. Obwohl der zweite Teil der vorliegenden Arbeit der Erarbeitung eines möglichst universellen Organisationseinheitsmodells gewidmet ist und konkrete Ausgestaltungsmöglichkeiten auf den dritten Teil der Arbeit verschoben werden, so muss das Grundmodell einer Organisationseinheit die systematische Beantwortung der folgenden Fragen dennoch zumindest unterstützen und leiten.

- *Domänendefinition:* Was ist die generelle Arena der Aktivität, in der sich die Organisationseinheit bewegt?

- *Bestimmung von Zielen/Strategien* Welche grundlegenden/langfristigen Ziele beziehungsweise Strategien gibt es auf Ebene der Organisationseinheit? Welchem allgemeinen Systemzweck dienen sie?

- *Steuerung und Kontrolle:* Welche Systemregeln in Form von Strukturen, Standards, Teilzielen, Plänen und Verhaltensvorschriften ergeben sich aus den grundlegenden Zielen/Strategien? Wie werden diese vermittelt? Welche Steuerungs- und Kontrollsysteme werden zur Ziel-/Strategieverfolgung und Einhaltung der Systemregeln etabliert?

- *Machtverteilung:* Welche Akteure legen Domäne, Ziele/Strategien und Regeln fest? Welche Akteure werden im Zuge der Etablierung von Steuerungs- und Kontrollsystemen darüber hinaus in Machtpositionen gehoben? Inwieweit ist Macht vor allem in Form von Organisationsstrukturen formalisiert oder individuellen Akteuren zu Eigen?

- *Organisationsgrenzen:* Wie definieren sich die Grenzen der Organisationseinheit gegenüber ihrer Umgebung? Welche Kriterien der Grenzdefinition (Akteure/Mitgliedschaft, soziale Relationen, Aktivitäten) dominieren?

- *Umwelt und Umgebungen:* Aus welchen verschiedenen Umgebungen setzt sich die gesamte Umwelt der Organisationseinheit zusammen. Wie hängen sie mit der Domänenwahl zusammen? Welche Positionen nimmt die Organisationseinheit entsprechend ihrer Ziele

und Strategien in ihren jeweiligen Umgebungen ein? Welche speziellen Funktionen erfüllt sie auf höheren Systemebenen?

- *Grenzmanagement:* Wie sieht die Informations- und Zuwendungsstruktur der Organisationseinheit gegenüber einer Umgebung aus? Wie gestaltet sich die Einbettung der Organisationseinheit in eine Umgebung unter Berücksichtigung der dualen Funktion der Umgebung als Aufgabenumgebung und als Institution (Pufferungs-/Brückenschlagsmechanismen)?

- *Kollektives Akteurswesen:* Welche konkreten Interaktionen sind mit einer Umgebung (beziehungsweise mit ihren eingebetteten Akteuren) möglich und wie werden sie auf die Interna der fokussierten Organisationseinheit abgebildet? Auf welche Weise sorgen interne Steuerungs- und Kontrollmechanismen für eine geeignete Aggregation des Verhaltens der individuellen Akteure, so dass die Organisationseinheit „als Ganzes“ ihre Position in der Umgebung einnehmen und ihrer Funktionen gerecht werden kann?

- *Autonomie:* Inwieweit ist die Organisationseinheit durch ihre Umgebung geprägt? Hat sie die Entscheidungshoheit über ihre Handlungen oder sind ihre Handlungen von außen gesteuert? Wie autonom ist die Organisationseinheit also?

- *Autarkie:* Welche charakteristischen Leistungen bietet die Organisationseinheit an? Von welchen Leistungen anderer und von welchen Umgebungsbedingungen hängt die Handlungsfähigkeit der Organisationseinheit selber ab? Wie autark ist die Organisationseinheit also?

Die genannten Fragestellungen sind natürlich eng miteinander verwoben und müssen *gemeinsam* berücksichtigt werden. Das zu erarbeitende universelle Grundmodell einer Organisationseinheit übernimmt dabei die Funktion eines integrativen Schemas. An diesem Schema wird die Ausprägung von konkreten Organisationseinheiten ausgerichtet und damit auf eine einheitliche Basis gestellt.

Die Kapitelaufteilung des zweiten Teils der Arbeit spiegelt vor allem die oben formulierte Forderung nach unterschiedlich abstrakten Perspektiven für die Modellierung von Organisationseinheiten wider. In Kapitel 8 werden verschiedene Techniken der abstrakten Modellierung von Organisationseinheiten vorgestellt sowie ein abstraktes Grundmodell einer Organisationseinheit erarbeitet. In Kapitel 9 erfolgt der Übergang zur Erstellung detaillierter, ausführbarer Modelle von Organisationseinheiten. In beiden Kapiteln wird eine zweistufige Vorgehensweise verfolgt. Auf der ersten Stufe werden minimalistische Modelle von *Kollektivitätseinheiten* basierend auf den Kernkonzepten *Akteure*, *Aktivitäten* und *Akteursbeziehungen* präsentiert. Die Modellierung von Organisationseinheiten auf der zweiten Stufe geschieht durch eine konkrete Verwendung der zuvor für Kollektivitätseinheiten eingeführten Techniken.

8 Abstrakte Modelle von Organisationseinheiten

Große Multiorganisationssysteme umfassen multiple Systemebenen und eine Vielzahl von vertikalen sowie horizontalen Verbindungen zwischen den individuellen und kollektiven Akteuren. Es besteht die Notwendigkeit, Mittel und Wege für eine systematische Modulation des Abstraktionsgrades bei der Betrachtung der Systeme zu finden. Die gezielte Ausblendung von Details auf mehreren Stufen unterstützt Planung, Entwurf, Analyse und Diskussion von Multiorganisationssystemen.

In diesem Kapitel werden Modellierungstechniken und Grundmodelle für verschiedene Ebenen der Modellabstraktion vorgestellt. Den Ausgangspunkt stellen dabei Referenznetzmodelle auf Abstraktionsstufe III dar (entsprechend der Charakterisierung in Abschnitt 6.3). Darauf aufbauend erlauben weitere Modellierungstechniken die Anhebung des Abstraktionsgrades über die exakte Petrinetzsemantik hinaus.

Alle Modelle des Kapitels sind abstrakt in dem Sinne, dass von einigen diffizilen Aspekten bei der Betrachtung von Organisationseinheiten abstrahiert wird. Dies betrifft insbesondere die Handhabung elaborierter Systemregeln und die daraus resultierenden Feinheiten von Akteursbeziehungen sowie deren Evolution. Diesen Aspekten wird im Kontext der operationalen Modelle des folgenden Kapitels 9 explizit Rechnung getragen. In diesem Kapitel werden verschiedene Stufen der Modellabstraktion unterschieden und damit einhergehend verschiedene Modellierungstechniken vorgestellt. Insgesamt erlauben die Modelle dieses Kapitels die Darstellung von Situationen und Abläufen eines Multiorganisationssystems auf anschauliche Weise. Je höher der Abstraktionsgrad gewählt wird, desto weniger stehen konkrete Ablaufdetails im Vordergrund. Stattdessen werden vor allem analytische Systemanteile und funktionale Beziehungen zwischen ihnen modelliert.

Bereits im Kontext der abstrakten Modelle dieses Kapitels soll ein klares Verständnis der Funktionsweise von Organisationseinheiten und Multiorganisationssystemen auf Basis von Petrinetzsemantik vermittelt werden. Daher werden einige Sachverhalte und Mechanismen punktuell mit detaillierten Referenznetzmodellen beschrieben. Auf diese Weise werden erste Anhaltspunkte für die Erstellung von voll funktionsfähigen operationalen Modellen im folgenden Kapitel geliefert.

Die Modellierung von Organisationseinheiten wird in diesem Kapitel inkrementell vorangetrieben. Abschnitt 8.1 widmet sich zunächst grundlegenden Techniken für die Modellierung sogenannter *Kollektivitätseinheiten* basierend auf den Kernkonzepten *Akteure*, *Aktivitäten* und *Akteursbeziehungen*. Die Modellierung von Organisationseinheiten in Abschnitt 8.2 erfolgt durch konkrete Verwendung der zuvor vorgestellten Modellierungstechniken. In diesem Zusammenhang wird ein abstraktes Grundmodell einer Organisationseinheit als universelles Schema für konkrete Ausgestaltungen vorgestellt.

Die mit diesem Kapitel verbundenen Forschungsaktivitäten des Autors haben zu mehreren Publikationen geführt. Die Publikationen betreffen die in Abschnitt 8.1 verwendeten Modellierungstechniken [WEM10] sowie das Grundmodell einer Organisationseinheit aus Abschnitt 8.2

in verschiedenen Evolutionsstadien [WEM08c, WEKBM09, WEM08b, WEM09]. Als Mitautoren waren Daniel Moldt und Michael Köhler-Bußmeier (in einem Fall) beteiligt.

8.1 Akteure, Aktivitäten und Kollektivitätseinheiten

Konzeptionelle Grundlage für die Modellierung von Kollektivitätseinheiten ist das soziologische Konzept einer Kollektivität. In Kapitel 3 wurde die Charakterisierung von Scott [Sco03, S. 20+186] eingeführt. Bei einer Kollektivität handelt es sich um eine spezifische Instanz sozialer Ordnung. Eine Kollektivität weist somit eine soziale Struktur in Form eines begrenzten Netzwerkes sozialer Relationen zwischen Akteuren auf. Die soziale Struktur besteht aus generalisierten Verhaltensregeln (normative Struktur), gemeinsamen Werten und Begriffsinhalten (kulturell-kognitive Struktur) und tatsächlich auftretenden Verhaltensregelmäßigkeiten (Verhaltensstruktur).

Dieses Kollektivitätsverständnis ist recht elaboriert und damit schwer zugänglich, wenn es um erste Modellierungsschritte geht. Daher wird hier zunächst ein reduziertes Kollektivitätsverständnis zugrunde gelegt. Als Kernkonzepte sieht der Autor der vorliegenden Arbeit *soziale Akteure*, die an *sozialen Aktivitäten* teilnehmen und zwischen denen *soziale Beziehungen* existieren. Diese drei Kernkonzepte dienen in der Folge als Grundlage für die Modellierung von *Kollektivitätseinheiten*, die eine Vorstufe zu den später behandelten Organisationseinheiten darstellen.

8.1.1 Kollektivitätseinheiten als Kontexte

Zunächst werden Kollektivitätseinheiten als reine Kontexte modelliert. Der Übergang zu Kollektivitätseinheiten als kollektive Akteure erfolgt im nachfolgenden Teilabschnitt.

8.1.1.1 Einführendes Beispiel

Abbildung 8.1 zeigt ein erstes, sehr einfaches Petrinetzmodell einer exemplarischen Kollektivitätseinheit. Es handelt sich um ein Referenznetzmodell der Abstraktionsstufe III, dementsprechend ist ein Großteil der technischen Operationalisierungsdetails ausgeblendet.

Ein Kollektivitätseinheitsnetz ist im Wesentlichen aus Teilnetzen für die Spezifikation von Aktivitäten und aus Stellen für die Einbettung von Akteuren zusammengesetzt. Das Kollektivitätseinheitsnetz der Abbildung umfasst lediglich ein einziges Aktivitätsnetz (transaction) sowie zwei Stellen für die Einbettung von Akteuren (senders, receivers). Dem Modellierungskonzept der Netze in Netzen folgend sind Akteure als Referenznetzmarken auf diesen beiden Stellen eingebettet. Allerdings wird in der Abbildung von der Netzdarstellung der Akteure abstrahiert. Das Aktivitätsnetz modelliert eine Transaktion zwischen zwei Parteien. Die eine Seite versendet einen Gegenstand und die andere Seite quittiert den Empfang nach Erhalt des Gegenstandes. Transitionen des Aktivitätsnetzes repräsentieren *Aktionen*. Aktionen werden von Akteuren ausgeführt und dementsprechend ist jede Aktivitätstransition mit einer Akteursstelle verbunden. Weitere Verbindungen zwischen Transitionen sorgen für die Kombination von Einzelaktionen zu einer Gesamtaktivität.

Für eine Aktivität ist typischerweise nicht nur wichtig, *dass* ein Akteur eine bestimmte Aktion ausführt, sondern auch, *welcher* Akteur es ist. Insbesondere wird eine Menge von Aktionen meist von ein und demselben Akteur ausgeführt. In den Aktivitätsmodellen dieser Arbeit werden in der Regel grafische Konventionen ähnlich denen aus AUML-Interaktionsdiagrammen

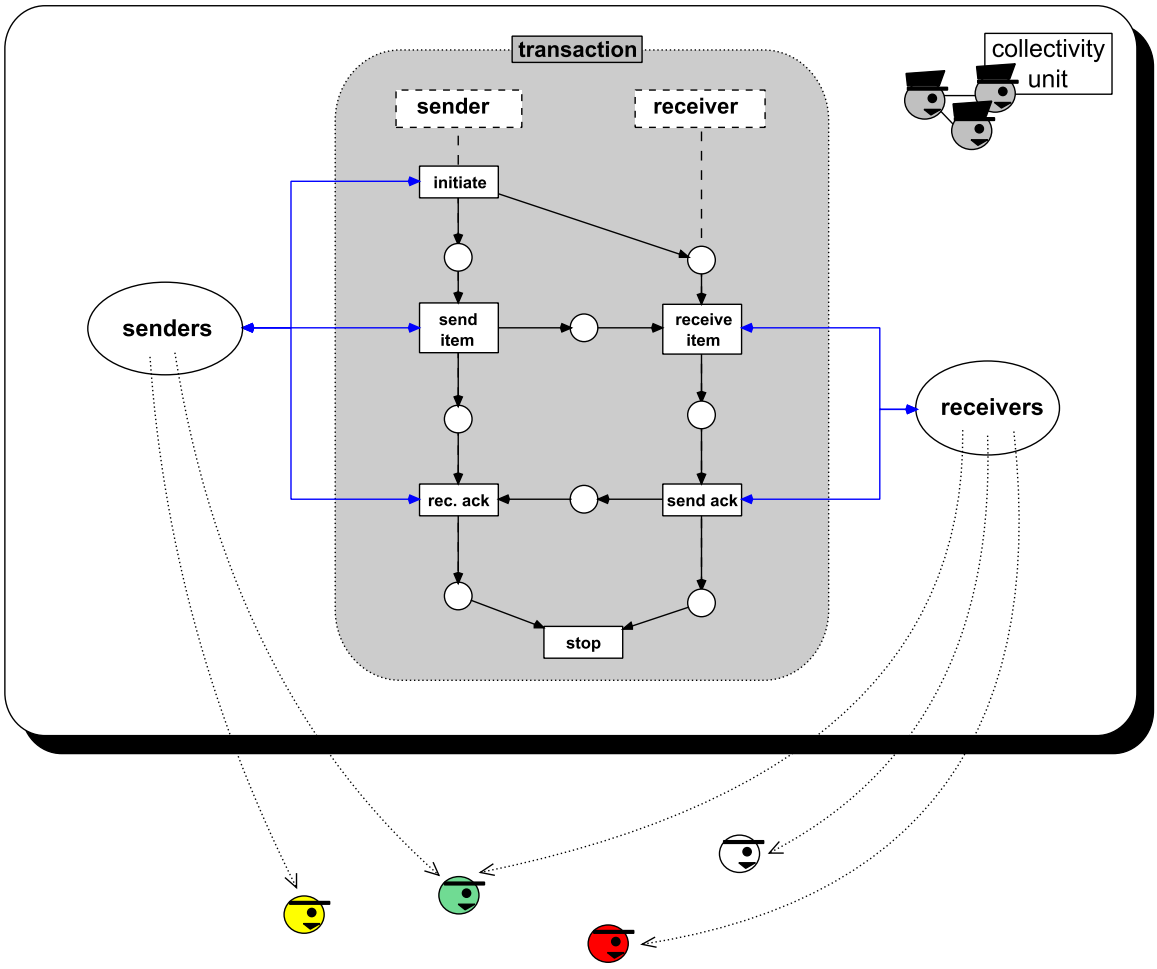

Abbildung 8.1: Einfache Kollektivitätseinheit und exemplarische Einbettung von Akteuren

(vergleiche Cabac et. al. [CMR03]) verwendet. Transitionen, die vertikal untereinander aufgeführt sind, sind vom selben Akteur auszuführen. Durch die vertikale Verbindung der Transitionen mittels Stellen wird die sogenannte *Lebenslinie* einer *Aktivitätsrolle* gebildet. Genau eine dieser Rollen ist dabei für die Initiierung der Aktivität vorgesehen. Für das Aktivitätsnetz transaction sind also die beiden Aktivitätsrollen sender und receiver spezifiziert, wobei die Rolle sender als Initiator vorgesehen ist. Die gesammelten Lebenslinien eines Aktivitätsnetzes repräsentieren den Steuerfluss. Stellen, die Transitionen unterschiedlicher Lebenslinien verbinden, modellieren einen *Nachrichtenversand* und repräsentieren den Datenfluss. Für die meisten in der vorliegenden Arbeit betrachteten Aktivitätsnetze werden diese grafischen Konventionen eingehalten. Lediglich in Kapitel 10 des dritten Teils der Arbeit wird an einer Stelle davon abgewichen.

Aktivitätsrollen sind Abstraktionen von konkreten Akteuren, die an einer Aktivität teilnehmen. Insofern repräsentiert das Aktivitätsnetz transaction aus Abbildung 8.1 auch keine *einzelne* Aktivität, sondern eine *Aktivitätsschablone* beziehungsweise ein *Muster*. Jede Nutzung dieser Schablone ist eine eigene Aktivität. Konkrete Nutzungen/Aktivitäten unterscheiden sich in

ihren Parametern, also insbesondere in der Besetzung der Aktivitätsrollen sender und receiver mit verschiedenen Akteuren sowie bezüglich des versandten Gegenstandes. Dementsprechend kann man Aktivitätsrollen als Schnittstellen für die Einbindung von Akteuren auffassen. Ein Akteur, der eine bestimmte Rolle einnimmt, muss sich an die Schnittstellenspezifikation halten. Welche internen Abläufe notwendig sind, damit ein Akteur eine Rolle tatsächlich einnehmen kann, kann von Akteur zu Akteur variieren.

In der Abbildung sind zwei Akteursstellen enthalten. Da jede Stelle mit mehreren Akteursmarken belegt sein kann, ergeben sich hier zwei *Akteursmengen*. Eine Akteursstelle ist immer mit allen oder keiner der Transitionen einer Aktivitätsrolle über Kanten verbunden. Damit wird der Umstand modelliert, dass Akteure dieser Stelle die grundsätzliche *Berechtigung* haben, diese Rolle in konkreten Aktivitäten einzunehmen (dabei kann je nach Aktivität gegebenenfalls eine weitere Einschränkung auf ganz bestimmte Akteure der Menge vorgenommen werden). Gleichzeitig haben sie die *Verpflichtung*, diese Rolle in einer Aktivität vollständig auszufüllen. Akteursmengen müssen nicht disjunkt sein. Konkrete Akteure können als Marken auf mehreren Akteursstellen enthalten sein. Diese Sichtweise entspricht der Referenzsemantik der hier verwendeten Modellierungstechnik der Referenznetze.

8.1.1.2 Semantik der Akteureinbettung

In diesem Kapitel werden abstrakte Modelle betrachtet. Für eine Operationalisierung der bisher beschriebenen Mechanismen (Netze-in-Netzen-Einbettung von Akteuren, Einbindung ganz bestimmter Akteure einer Aktivitätsmenge, Sicherstellung der Ausführung aller Aktionen einer Aktivitätsrolle durch denselben Akteur, Ermöglichung der multiplen Nutzung eines Aktivitätsnetzes für verschiedene konkrete Aktivitäten) sind feinere Spezifikationen der Netzmodelle und insbesondere konkrete Anschriften an Netzelementen notwendig. Auf eine ausführliche Diskussion der technischen Details wird hier verzichtet, da das Kapitel 9 dieser Thematik gesondert gewidmet ist. Allerdings hat gerade der Aspekt der Einbettung von Akteuren in Kollektivitäts- und Organisationseinheiten eine zentrale Bedeutung für das gesamte Modellierungsvorhaben des zweiten Teils der Arbeit. Daher ist eine exakte Semantik der Einbettung von Akteuren bereits für das tiefere Verständnis der abstrakten Modelle notwendig. Als Vorbereitung für die Betrachtung der Einbettung kollektiver Akteure im folgenden Teilabschnitt zeigt Abbildung 8.2 die exemplarische Einbettung eines individuellen Akteurs der Akteursmenge receivers. Für dieses Referenznetzmodell wurde die Abstraktionsstufe I gewählt.[1]

Auch wenn in Kapitel 9 eine kompaktere Herangehensweise an die Operationalisierung/Programmierung von Kollektivitäts- und Organisationseinheit gewählt wird, so treten die grundsätzlichen Mechanismen bereits hier zutage. Akteure sind mit Identifikatoren assoziiert und können mittels dieser aus einem Aktivitätsnetz heraus direkt als Verantwortliche für die Ausführung von Aktionen benannt werden. So kann beispielsweise ein ganz bestimmter Akteur der Akteursmenge receivers für die Besetzung der Akteursrolle receiver in einer konkreten Aktivität des Aktivitätsnetzes transaction festgelegt werden.

Die Ausführung von Aktionen geschieht mittels der Fusion von Transitionen auf verschiedenen Netzebenen auf Basis der in Kapitel 6 ausführlich erläuterten synchronen Kanäle. Auf

[1] Auch wenn die Abbildung die meisten relevanten Operationalisierungsdetails enthält, so fehlen zumindest Details bezüglich der Unterscheidung verschiedener konkreter Nutzungen/Aktivitäten des Aktivitätsnetzes. Eine mögliche Herangehensweise ist, jede Aktivität mit einem eigenen Bezeichner zu versehen und diesen an die zur Aktivität gehörenden Marken anzuhängen. Da in Kapitel 9 ohnehin eine Herangehensweise mit separaten Netzinstanzen für Aktivitäten gewählt wird, wird die Thematik hier nicht vertieft.

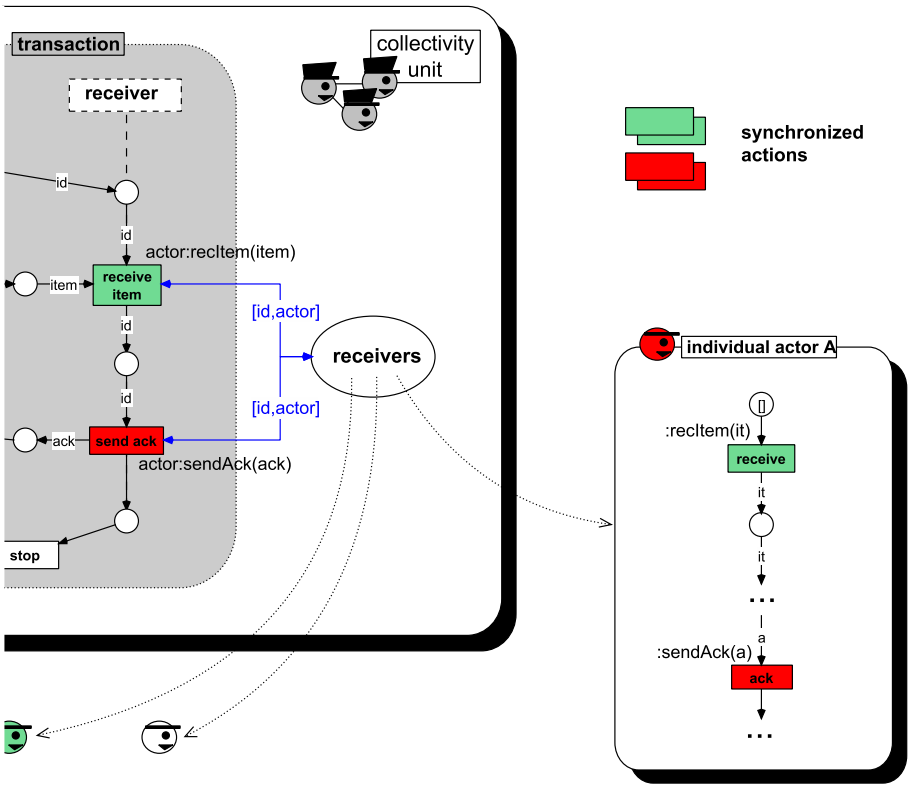

Abbildung 8.2: Exemplarische Einbettung eines individuellen Akteurs

Seite des Aktivitätsnetzes werden die Downlinks der Kanäle spezifiziert. Die zu einer Aktivitätsrolle gehörenden Downlinks in Kombination mit deren Anordnung ergibt gerade die weiter oben beschriebene Schnittstelle für die Akteurseinbindung. Akteure müssen eine zu dieser Schnittstelle passende Menge und Anordnung von Uplinks bereitstellen.[2]

8.1.1.3 Weiterführende Aspekte

Das bisher betrachtete Beispiel einer Kollektivitätseinheit ist sehr einfach. Natürlich kann eine Kollektivitätseinheit im allgemeinen Fall sehr viele Akteursmengen und Aktivitätsspezifikationen umfassen. Abbildung 8.3 zeigt ein weiteres Beispiel einer Kollektivitätseinheit, anhand dessen einige weiterführende Aspekte betrachtet werden können. Die Einbettung von Akteursmarken ist hier ausgeblendet. Zunächst fällt auf, dass Kollektivitätseinheitsmodelle auch auf Abstraktionsstufe III sehr schnell recht komplex und ausladend werden. In Abschnitt 8.1.3 werden daher Techniken für eine abstraktere/kompaktere Modellierung präsentiert. Diese erlauben eine nahezu beliebig grobgranulare Abstraktion von Anteilen einer Kollektivitätseinheit, so dass lediglich für punktuell ausgewählte Anteile Detailbetrachtungen vorgenommen werden

[2] Die Frage, wann eine Downlink-Menge und -Anordnung zu einer Uplink-Menge und -Anordnung „passt“, wird hier für den allgemeinen Fall nicht betrachtet. Als ganz allgemeine Forderung lässt sich formulieren, es dürfe nicht zu Verklemmungen kommen, sobald ein Akteur eine Aktivitätsrolle besetzt hat. In Kapitel 10 werden einige konkrete Ausprägungen/Typen von Organisationseinheiten behandelt. In deren Kontext können konkrete Kriterien für eine Passung besser definiert werden.

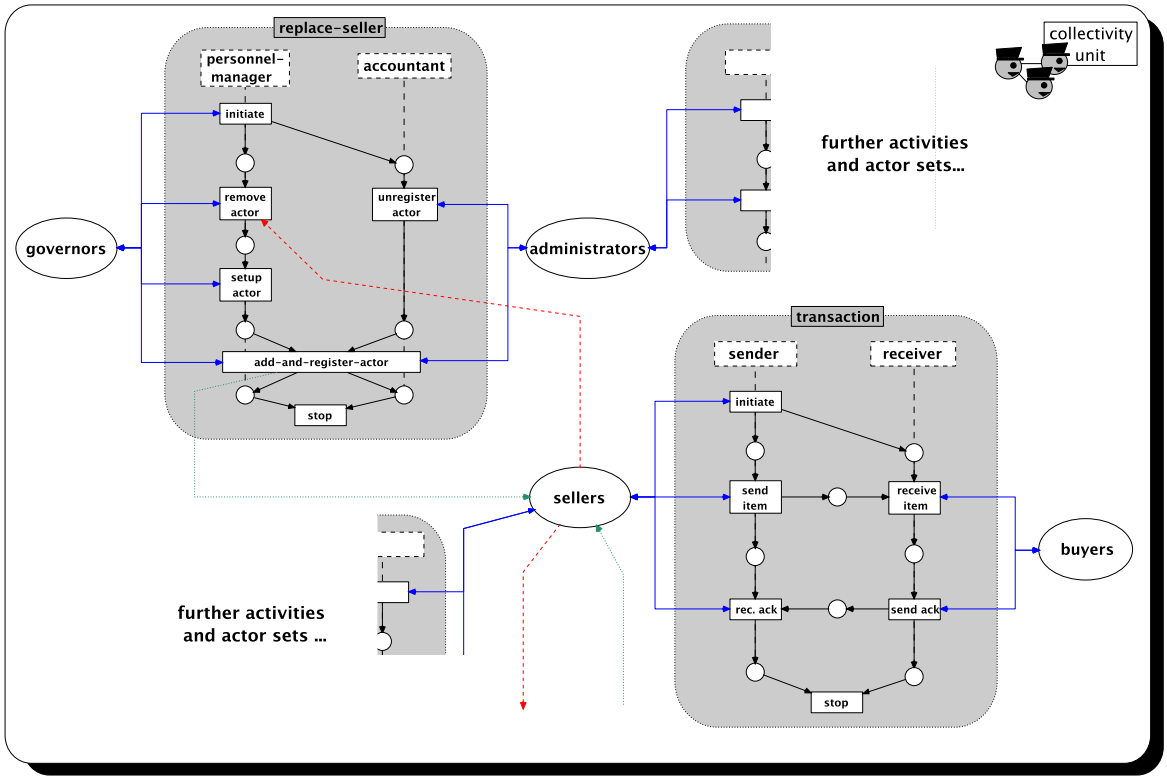

Abbildung 8.3: Ausschnitt einer komplexeren Kollektivitätseinheit

müssen.

Gegenüber den vorherigen einfachen Kollektivitätseinheitsmodell wird hier deutlich, dass eine Akteursmenge mit mehr als nur einer Aktivitätsrolle assoziiert sein kann. Beispielsweise ist angedeutet, dass Akteure der Menge sellers sowohl in der Rolle sender im Kontext einer Transaktionsaktivität auftreten können als auch in anderen Rollen im Kontext weiterer Aktivitäten. Akteursmengen sind dadurch mit bestimmten *Positionen* im Kontext einer Kollektivitätseinheit assoziiert. Mit jeder Position sind dabei multiple Aktivitätsberechtigungen verknüpft.

Ein weiterer zusätzlicher Aspekt gegenüber dem vorigen einfachen Modell besteht in der „Verwendung“ der Akteure für die Aktivitätsabwicklung. Im Aktivitätsnetz replace-seller ist nicht nur die Ausführung von Aktionen durch Akteure, sondern darüber hinaus das Entfernen und das Hinzufügen von Akteuren spezifiziert. Zusätzlich wird das Auftreten einer *gemeinsamen* oder *synchronen Aktion* zwischen mehreren Akteuren spezifiziert. Das Aktivitätsnetz modelliert einen Akteursaustausch und führt zu diesem Zweck die Aktivitätsrollen eines Managers (personnel-manager) und eines Buchführers (accountant) ein. Während der Manager einen Akteur aus der Menge sellers entfernt (remove-actor) und einen neuen Akteur vorbereitet (setup-actor), meldet der Buchführer den alten Akteur ab (unregister-actor). Um sicherzustellen, dass der neue Akteur zeitgleich mit seinem Auftreten in der Kollektivitätseinheit auch registriert ist, ist die entsprechende Aktion add-and-register-actor eine gemeinsame, synchrone Aktion von Manager und Buchführer.

Demnach modelliert eine Aktion zwar atomares Verhalten, kann dabei allerdings auf multiple Art und Weise mit Akteursmengen in Zusammenhang stehen. Dabei werden grundsätzlich drei *Basisoperationen* unterschieden. Jede Aktion setzt sich aus mindestens einer solchen Basisoperation zusammen. Umfasst die Aktion mehrere Basisoperationen, so werden diese synchron ausgeführt.

1. **Verwendung/Modifikation** *(modelliert durch eine durchgezogene Reservierungskante):* Akteure werden für die Ausführung einer Aktion *verwendet*, wenn sie zur Ausführung der Aktion beitragen.

2. **Hinzufügen** *(modelliert durch eine gepunktete Kante mit Pfeilende an der Akteursstelle):* Akteure können im Zuge einer Aktion zur Kollektivitätseinheit *hinzugefügt* werden. In geschlossenen Kollektivitätseinheiten (wie bisher betrachtet) ist es notwendig, dass dieses Hinzufügen von bereits vorhandenen Akteuren vorgenommen wird. In offenen Kollektivitätseinheiten (wie später im Kapitel betrachtet) kann es dagegen auch vorkommen, dass Akteure die Systemeinheit von außen *betreten*.

3. **Entfernen** *(modelliert durch eine gestrichelte Kante mit Pfeilende an der Aktionstransition):* Akteure können im Zuge einer Aktion aus der Kollektivitätseinheit *entfernt* werden. Dies kann von den betroffenen Akteuren selber vollzogen oder aber von anderen Akteuren erzwungen werden. Im Falle offener Kollektivitätseinheiten kann das Entfernen eines Akteurs mit dem Hinzufügen desselben Akteurs in eine andere Kollektivitätseinheit kombiniert werden, so dass sich eigentlich um eine *Migration* handelt.

Mit dieser Unterscheidung von Basisoperationen und dem zuvor erläuterten Konzept der Aktivitätsrollen ist die Charakterisierung des Aktivitätsverständnisses für die vorliegende Arbeit abgeschlossen. Natürlich gibt es gerade im Bereich der Petrinetzmodellierung einen großen Fundus an Ansätzen der Verhaltensmodellierung. Dabei bieten sich im Kontext der Arbeit beispielsweise inter-organisatorische Workflownetze [KBMWE09, vdA99] an, da sie die Vorteile der klassischen Workflowmodellierung mit der expliziten Unterscheidung mehrerer Parteien kombinieren. Für das universelle Verständnis von Kollektivitäts- und Organisationseinheiten soll jedoch keine spezielle Form der Aktivitätsmodellierung über die bisherige Charakterisierung hinaus vorgeschrieben werden.

8.1.1.4 Akteursbeziehungen

Insgesamt lässt sich an dieser Stelle bereits festhalten, dass alle zu Beginn des Kapitels geforderten Kernkonzepte *Akteure, Aktivitäten* und *Akteursbeziehungen* für die Kollektivitätseinheitsmodellierung berücksichtigt werden. Akteursmengen und Aktivitätsspezifikationen bilden die Basisbausteine einer Kollektivitätseinheit. Akteursbeziehungen ergeben sich implizit aus den Aktivitätsberechtigungen der einzelnen Akteursmengen sowie den Aktivitätsinhalten. Allerdings sind die Möglichkeiten der Modellierung von Akteursbeziehungen damit recht eingeschränkt. Viele Formen von sozialen Beziehungen lassen sich auf diese Weise nicht oder nur schwer ausdrücken. Bereits einfache Dinge wie eine Autoritätsbeziehung zwischen Akteursmengen oder kausale Zusammenhänge zwischen verschiedenen Aktivitäten können mit den vorgestellten Mitteln zumindest nicht unmittelbar modelliert werden.

Damit wird die zu Beginn des Kapitels getroffene Ankündigung belegt, dass elaborierte Systemregeln und daraus resultierende Feinheiten von Akteursbeziehungen nicht zu den Kernkonzepten der hier betrachteten Modelle gehören. Für konkrete Typen von Kollektivitätseinheiten

können gegebenenfalls proprietäre Erweiterungen der hier vorgestellten Modellierungstechniken vorgenommen werden. In einigen Fällen kann bereits die Annotation von natürlichsprachlichen Kommentaren als zufriedenstellende semantische Anreicherung dienen.

Für das weitere Vorhaben dieses Kapitels wird allerdings bewusst darauf verzichtet, weitere Modellierungsprimitive einzuführen. Der Autor der vorliegenden Arbeit sieht es vielmehr als Vorteil, alle Modelle im Kern auf der bisher vorgestellten reduzierten Menge an Konzepten und der damit einhergehenden einfach zugänglichen Petrinetzsemantik aufzubauen. Für die Erstellung ausführbarer Modelle im folgenden Kapitel werden der Definition und Handhabung elaborierter Akteursbeziehungen dagegen sehr viel mehr Aufmerksamkeit zuteil.

8.1.1.5 Evolution

Neben den fehlenden Möglichkeiten, Akteursbeziehungen in Form elaborierter Systemregeln beschreiben zu können, manifestiert sich auch bereits der zweite zu Beginn des Kapitels angekündigte Missstand. Die hier vorgestellten Modelle eigenen sich nur sehr bedingt für die Illustration der Evolution einer Kollektivitätseinheit. Zwar wird das Hinzufügen und Entfernen von Akteuren adressiert, für die Entstehung neuer Akteursmengen, neuer Aktivitätsmuster oder neuer Akteursbeziehungen müssen allerdings Modifikationen an den Modellen selber vorgenommen werden. Als Ausweg bietet sich gegebenenfalls die Spezifikation sehr generischer Aktivitätsmuster beziehungsweise einer sehr großen Anzahl von Aktivitätsmustern an, so dass deren unterschiedliche konkrete Nutzung über die Zeit eine Evolution repräsentiert. Allerdings ist hier schlicht festzuhalten, dass dieser Aspekt keine tragende Rolle im Kontext der abstrakten Modelle spielt. Hier geht es darum, bestimmte Situationen und Konstellationen eines Multiorganisationssystems sowie mögliche Abläufe abstrakt zu modellieren. Im folgenden Kapitel 9 werden dagegen Modelle vorgestellt, bei denen die Operationalisierung von Evolutionsaspekten eine prominente Rolle einnimmt.

8.1.2 Kollektivitätseinheiten als kollektive Akteure

Die Auffassung von Kollektivitätseinheiten als Kontexte stellt lediglich eine Seite der Medaille dar. Abbildung 8.4 zeigt die exemplarische Einbettung einer Kollektivitätseinheit als kollektiven Akteur im Kontext einer umschließenden Kollektivitätseinheit. Um die Semantik hinter der Einbettung kollektiver Akteure als Grundlage für den weiteren Verlauf des Kapitels deutlich herauszuarbeiten, ist für das Modell die Abstraktionsstufe II mit Kanal- und Kantenanschriften gewählt. Allerdings sind nur einige der Anschriften aufgeführt.

Wie für individuelle Akteure gilt, dass ein kollektiver Akteur die Schnittstellenspezifikationen der Aktivitätsrollen, zu deren Besetzung der Akteur die Berechtigung hat, erfüllen muss. Ein kollektiver Akteur handelt vermittels seiner eingebetteten Akteure. Die Handlungen der eingebetteten Akteure müssen also derart koordiniert werden, dass der kollektive Akteur „als Ganzes“ auf der darüber liegenden Kollektivitätsebene die ihm zugewiesenen Aktivitätsrollen einnehmen kann. Dies geschieht mittels der Definition geeigneter *Peripherieaktivitäten*. Eine Peripherieaktivität unterscheidet sich von den bislang betrachteten Aktivitäten durch den Einbezug von *Peripherieaktionen*. Eine Peripherieaktion hat nicht nur auf Ebene der Kollektivitätseinheit, im Kontext derer sie spezifiziert ist, sondern außerdem auf der darüber liegenden Kollektivitätsebene Effekte. Aus Sicht einer fokussierten Kollektivitätseinheit wird eine Peripherieaktion durch einen eingebetteten Akteur ausgeführt. Auf der darüber liegenden Kollektivitätsebene wird die Aktion als Aktion der fokussierten Kollektivitätseinheit selbst

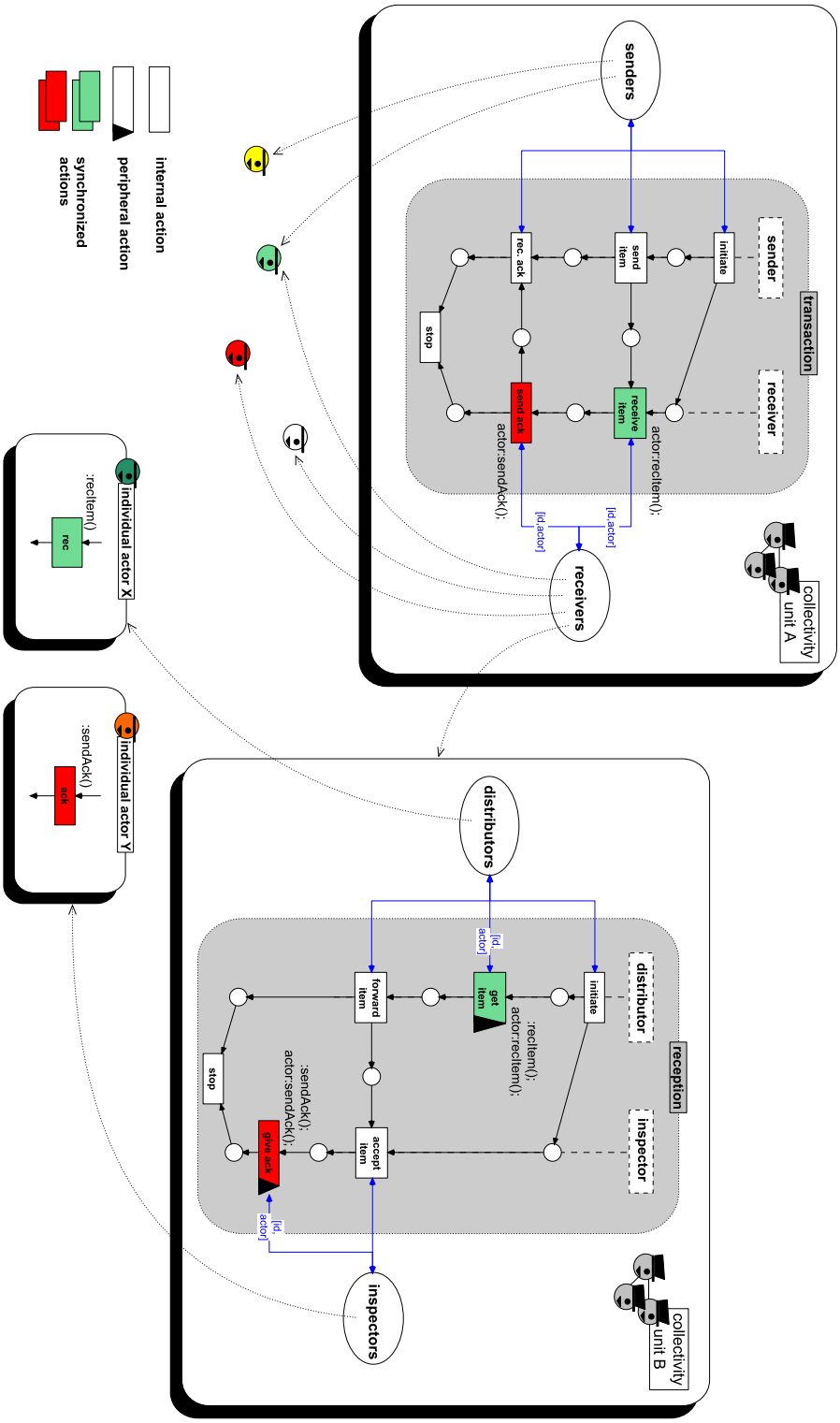

Abbildung 8.4: Exemplarische Einbettung einer Kollektivität als Akteur

wahrgenommen.

Im Beispiel aus Abbildung 8.4 bettet die Kollektivitätseinheit A die Kollektivitätseinheit B als kollektiven Akteur ein. Aus Sicht von A handelt es sich bei B um einen Akteur wie alle anderen Akteure der Akteursmenge receivers. Er wird über die gleiche Schnittstelle/Aktivitätsrolle receiver angesprochen wie eingebettete individuelle Akteure (vergleiche die Abbildung 8.2 des vorherigen Teilabschnittes). Aus Sicht von B allerdings gestaltet sich die Besetzung dieser Aktivitätsrolle als Verteilung über mehrere eigens eingebettete Akteure. Eingebettete Akteure nehmen Aktivitätsrollen im Kontext der Peripherieaktivität reception ein. Die beiden Peripherieaktionen get item und give ack werden von unterschiedlichen Akteuren ausgeführt (hier von den beiden individuellen Akteuren X und Y), werden aber auf Ebene von A als Ausführung der Aktionen receive item und send ack durch die Kollektivitätseinheit B selbst wahrgenommen. Zusammenfassend lässt sich also festhalten, dass eine Kollektivitätseinheit eine Akteursrolle besetzt, indem eingebettete Akteure Aktivitätsrollen einer passenden Peripherieaktivität besetzen. Man kann von einer *Rollenimplementierung (auf Ebene n) durch Rollenimplementierung (auf Ebene $n-1$)* sprechen.

Auf diese Weise ergibt sich zunächst ein *technisches Verständnis kollektiven Akteurswesens*. Die Beschriftungen mit synchronen Kanälen in Abbildung 8.4 sorgt dafür, dass die farblich einheitlich markierten Transitionen synchron schalten und sich dadurch jeweils *eine einzige Aktionsausführung* ergibt. Diese tritt aber auf mehreren Kollektivitätsebenen zugleich auf und hat auf allen Ebenen spezifische Effekte. Nach Ansicht des Autors liegt hier einer der bedeutenden Vorteile des Modellierens mit Netzen in Netzen. In Kapitel 4 wird an verschiedenen Stellen gefordert, die fälschliche Assoziation kollektiver Akteure mit antropomorphen Eigenschaften zu vermeiden. Für die softwaretechnische Umsetzung von Kollektivitäten als kollektive Akteure ist dabei also wichtig, dass ihre Aktionen nicht „aus dem Nichts“ heraus ausgeführt werden. Zwar besitzt eine Kollektivitätseinheit eine Existenz unabhängig von konkreten eingebetteten Akteuren (sie ist durch ein eigenes Petrinetz verkörpert), doch ohne solche eingebetteten Akteure ist sie lediglich eine leere Hülle und nicht in der Lage zu handeln. Das Ausführen einer Aktion durch eine Kollektivitätseinheit ist an das Ausführen dieser Aktion durch einen eingebetteten Akteur „im Auftrag/Namen“ der Kollektivitätseinheit geknüpft. Gerade dieser fundamentale Umstand lässt sich auf direkte Weise mittels der Semantik des Konzeptes der Netze in Netzen ausdrücken. Im Modell aus Abbildung 8.4 erzwingen die Kanalanschriften, dass jede eigene Aktion der Kollektivitätseinheit B durch eine Peripherieaktion eines eingebetteten Akteurs realisiert wird. Auf diese Weise handelt die Kollektivitätseinheit als Kollektiv.

In Abbildung 8.4 sind lediglich drei Systemebenen illustriert, nämlich zwei Kollektivitätsebenen und die Ebene individueller Akteure. Es ist natürlich möglich, die Einbettung von Kollektivitätseinheiten in Kollektivitätseinheiten bis zu beliebiger Tiefe fortzusetzen. Die in der Kollektivitätseinheit B eingebetteten Akteure könnten also wiederum Kollektivitätseinheiten sein und selber wiederum Kollektivitätseinheiten einbetten. Jede Peripherieaktion ist immer mit nur zwei synchronen Kanälen assoziiert, nämlich einem in Richtung umschließender Kollektivitätseinheiten (Uplink) und einem in Richtung eingebetteter Akteure (Downlink). Die Handhabung kollektiven Akteurswesens kann also auf jeder Kollektivitätsebene separat vorgenommen werden. Erst wenn es zur tatsächlichen Ausführung einer Aktion kommt, führen Kanalketten zur Fusion von unter Umständen sehr vielen Transitionen für einen Schaltvorgang. Individuelle Akteure bilden den Abschluss einer solchen Rekursion. Sie sind diejenigen, auf die *alle* Aktionen eines Multiorganisationssystems zurückzuführen sind. Für ihre Aktionen ergibt sich also ein mehrschichtiger Hintergrund an einbettenden Kollektivitätseinheiten. Abbildung 8.5 verdeutlicht dieses fundamentale Prinzip der Aktionsrekursion in Multiorganisationssystemen

noch einmal kompakt ohne den Ballast von kompletten Kollektivitätseinheitsmodellen.

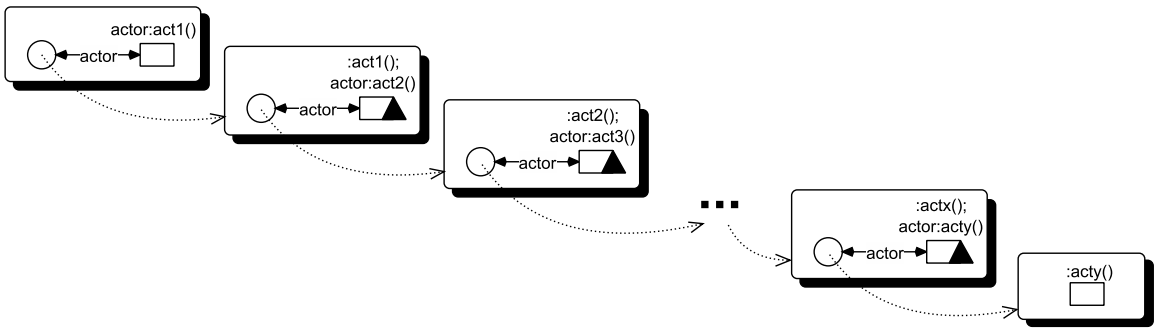

Abbildung 8.5: Grundprinzip der Aktionsrekursion in Multiorganisationssystemen

Abbildung 8.6 zeigt ein weiteres Beispiel der Einbettung einer Kollektivitätseinheit. Dabei ist die Kollektivitätseinheit C als Akteur in zwei weiteren Kollektivitätseinheiten A und B eingebettet. Das Wesen realweltlicher sozialer Systeme als nichtdisjunkte Hierarchien von Kollektivitäten wurde in den Kapitel 3 und 4 als wichtige Charakteristik herausgestellt. Die Referenzsemantik von Referenznetzmodellen erlaubt die unmittelbare Übertragung dieser Charakteristik auf Modelle von Multiorganisationssystemen. Individuelle Akteure sowie Kollektivitätseinheiten können mehrfach als Referenznetzmarken eingebettet werden. Daraus ergeben sich Möglichkeiten für die Modellierung elaborierter Aktivitätsverzahnungen über Kollektivitätsgrenzen und -ebenen hinweg. In Abbildung 8.6 ist modelliert, dass die Kollektivitätseinheit C ihre reception-Aktivitäten benutzt, um die Besetzung von zwei verschiedenen Aktivitätsrollen in verschiedenen umschließenden Kollektivitätseinheiten zu kombinieren. Auf diese Weise entsteht eine implizite Akteursbeziehung zwischen den Kollektivitätseinheiten A und B, ohne dass es auf Ebene dieser beiden Kollektivitätseinheiten notwendigerweise wahrgenommen wird.

Aufbauend auf den hier beschriebenen Mechanismen der Einbettung kollektiver Akteure lassen sich unterschiedliche Szenarien konstruieren. Im folgenden Teilabschnitt wird eine Klassifikation typischer Szenarien vorgenommen.

8.1.3 Abstraktionsstufen der Kollektivitätseinheitsmodellierung

Die bisher betrachteten Modelle werden in der Folge als *Akteur-Aktivitätsablauf-Modelle* von Kollektivitätseinheiten bezeichnet. Dabei wurden bislang nur Kollektivitätseinheiten mit wenigen und einfachen Aktivitätsnetzen betrachtet. Bei steigender Anzahl von Aktivitätsnetzen und bei der Betrachtung umfangreicherer Aktivitätsnetze werden Akteur-Aktivitätsablauf-Modelle schnell unübersichtlich. Die verwendete Petrinetzdarstellung von Kollektivitätseinheiten ist eigentlich nur dann relevant, wenn die Ablaufdetails von Aktivitäten von Interesse sind. Unter Umständen ist aber eine abstraktere Perspektive ausreichend. Wenn eher analytische Komponenten und funktionale Zusammenhänge interessant sind, können Details bezüglich des konkreten Kontroll- und Datenflusses zunächst vernachlässigt werden. Daher werden in diesem Teilabschnitt Modellierungstechniken für das Anheben der Modellabstraktion über die Petrinetzdarstellung hinaus vorgestellt. Obwohl die Petrinetzsemantik nicht mehr direkt sichtbar ist, bildet sie weiterhin die Grundlage für das Verständnis der Modelle.

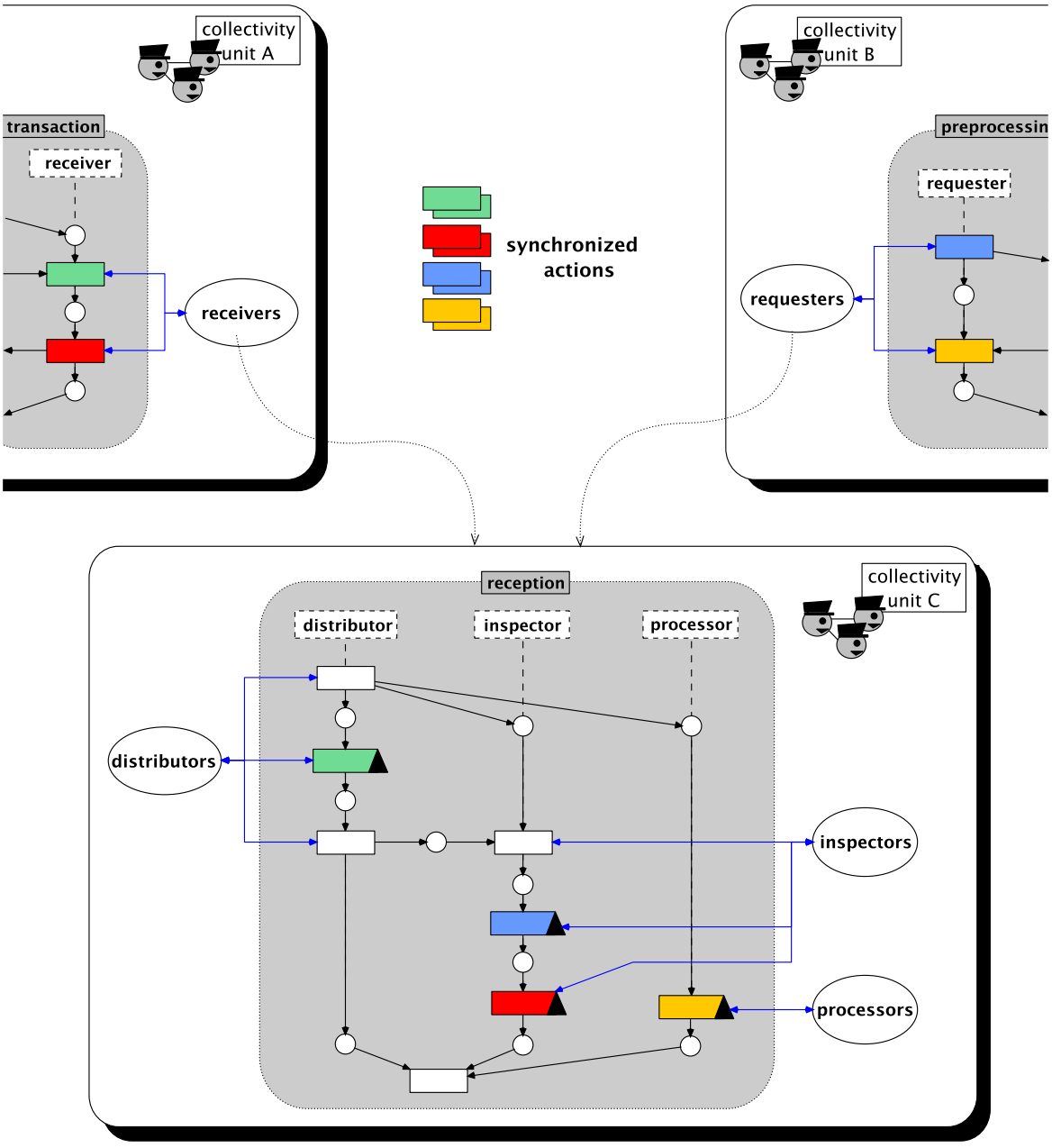

Abbildung 8.6: Mehrfache Einbettung einer Kollektivität

Dabei gilt für alle in diesem Kapitel vorgestellten Modellierungstechniken, dass sie für die Darstellung von bestimmten Situationen und damit von Ausschnitten eines Multiorganisationssystems verwendet werden können. Der steigende Abstraktionsgrad der Modelle erlaubt dabei kompakte Darstellungen immer umfassenderer Ausschnitte.

8.1.3.1 Abstraktion von Aktivitätsabläufen

Der erste Abstraktionsschritt besteht in der Verbergung von Details des Ablaufes von Aktivitäten. Abbildung 8.7 zeigt ein abstrakteres Modell der Kollektivitätseinheit aus Abbildung 8.3, eingeschränkt auf die beiden Aktivitätsmuster transaction und replace-actor. Die Spezifikation

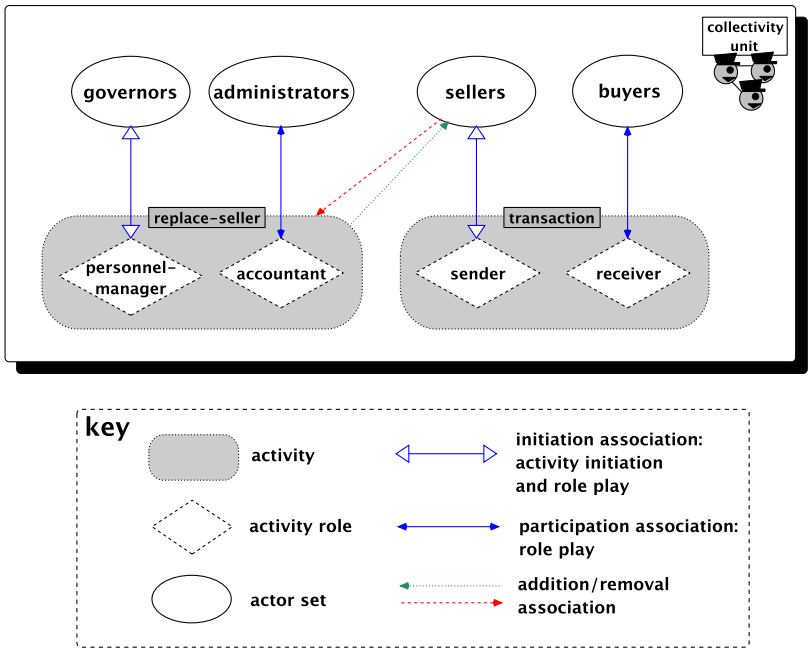

Abbildung 8.7: Abstraktion von den Aktivitätsabläufen aus Abbildung 8.3

von Aktivitätsmustern ist nun auf die bloße Benennung von Aktivitätsrollen beschränkt. Daher werden die in diesem Unterabschnitt betrachteten Modelle von Kollektivitätseinheiten als *Akteur-Aktivitätsrollen-Modelle* bezeichnet. Die bisher betrachteten Akteursmengen tauchen unverändert auf und aufbauend auf den bisherigen Ausführungen bestehen vier Möglichkeiten, sie mit Aktivitätsmustern zu assoziieren.

1. *Teilnahmeassoziation:* Akteure einer Menge können die *Berechtigung zur Besetzung/Implementierung* bestimmter Akteursrollen haben.

2. *Initiierungassoziation:* Wie bisher ist für jedes Aktivitätsmuster genau eine Aktivitätsrolle für die Initiierung konkreter Aktivitäten vorgesehen. Dies wird mittels einer *Berechtigung zur initiierenden Besetzung/Implementierung einer Aktivitätsrolle* für eine Akteursmenge modelliert.[3]

3.+4. *Hinzufügen-/Entfernen-Assoziation:* Akteure einer Akteursmenge können im Laufe von Aktivitäten hinzugefügt und entfernt werden.

[3] Der Grund dafür, Aktivitätsinitiierung als Eigenschaft der Relation zwischen Akteursmenge und Aktivitätsrolle und nicht direkt als Eigenschaft der Aktivitätsrolle zu modellieren, wird weiter unten beim Übergang zur Modellierung mit Aktivitätsmengen deutlich.

Akteursmengen werden also mit Aktivitätsmustern und Aktivitätsrollen assoziiert, ohne dass konkrete Aktionen betrachtet werden. Diese Art der Modellierung ist mit der Verwendung von *Use-Case*-Diagrammen in der agentenorientierten Softwareentwicklung verwandt (vergleiche Cabac [Cab10, Kapitel 10]).

Gegenüber der herkömmlichen Verwendung von Use-Case-Modellen ist für die Akteur-Aktivitätsrollen-Modelle der vorliegenden Arbeit allerdings eine inhaltliche Erweiterung notwendig, damit kollektive Akteure berücksichtigt werden können. Weiter oben wurde die Besetzung von Akteursrollen mit Kollektivitäten als *Rollenimplementierung (auf Ebene n) durch Rollenimplementierung (auf Ebene n−1)* charakterisiert. Soll also die Einbettung eines kollektiven Akteurs in ein Modell aufgenommen werden, so müssen ebenenübergreifende Rollenassoziationen expliziert werden. Abbildung 8.8 zeigt dies anhand des Beispiels aus Abbildung 8.6. Hier wird der Umstand modelliert, dass die Implementierung der Aktivitätsrollen distributor und inspector im Kontext einer reception-Aktivität auf Ebene der Kollektivitätseinheit B zur Implementierung der Aktivitätsrolle receiver einer transaction-Aktivität auf Ebene der Kollektivitätseinheit A führt. Damit wird deutlich, dass es sich bei der Kollektivitätseinheit B um einen Akteur der Akteursmenge buyers auf Ebene der Kollektivitätseinheit A handelt. Einbettungspfeile wie in den petrinetzbasierten Akteur-Aktivitätsablauf-Modellen sind nicht notwendig. Der Veranschaulichung wegen werden Peripherieaktivitäten auf den Rand einer Kollektivitätseinheit gezeichnet.

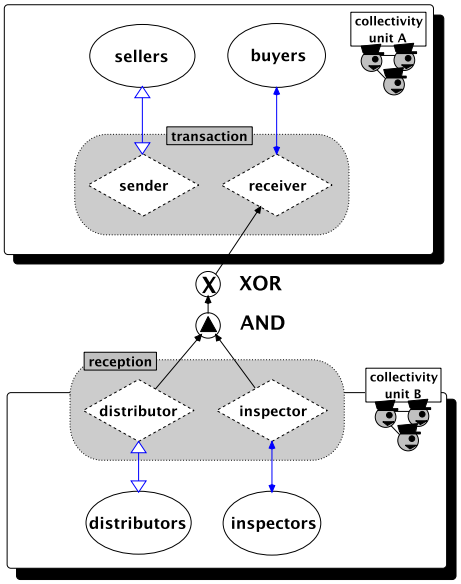

Abbildung 8.8: Einbettung kollektiver Akteure am Beispiel (vergleiche Abbildung 8.4)

Der allgemeine Fall ist in Abbildung 8.9 schematisch veranschaulicht. Ausgangspunkt ist die Betrachtung einer Kollektivitätseinheit A. Hier wird eine Teilmenge $\{a_i, ..., a_j\}$ von Aktivitätsrollen eines Aktivitätsmusters $activity_a$ betrachtet. Die kombinierte Implementierung dieser Rollen in einer konkreten Aktivität führt zur Implementierung *genau einer* Aktivitätsrolle der Menge $\{b_{1i}, ..., b_{nj}\}$ im Kontext einer der Kollektivitätseinheiten $B_1, ..., B_n$, in die A eingebettet ist. Die implementierenden Rollen werden durch einen AND-Knoten (logisches *Und*) verknüpft, während die implementierten Rollen durch einen XOR-Knoten (logisches *Exklusiv-*

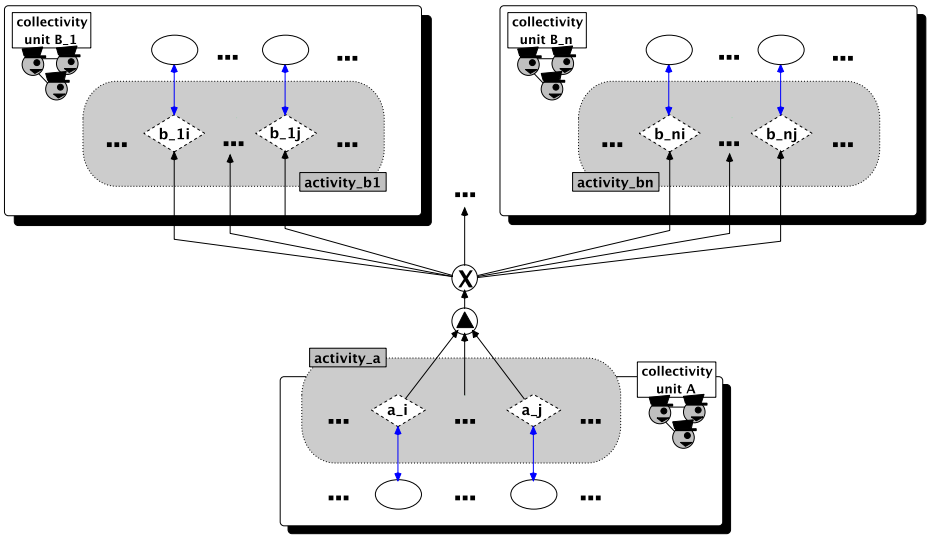

Abbildung 8.9: Einbettung kollektiver Akteure: allgemeines Schema

Oder) verknüpft werden. In verschiedenen Aktivitäten des Aktivitätsmusters $\mathsf{activity_a}$ kann die Implementierung der Aktivitätsrollen $\{\mathsf{a_i}, ..., \mathsf{a_j}\}$ also zur Implementierung verschiedener (aber jeweils nur genau einer) der Aktivitätsrollen $\{\mathsf{b_{1i}}, ..., \mathsf{b_{nj}}\}$ führen.

Es ist möglich, ausgehend von einem Aktivitätsmuster mehrere AND/XOR-Konfigurationen zu kombinieren. Dies ist beispielsweise notwendig, wenn die Situation aus Abbildung 8.6 abstrakt modelliert werden soll. Das entsprechende Modell ist in Abbildung 8.10 zu sehen. Hier gilt, dass die von einem Aktivitätsmuster ausgehenden AND/XOR-Konfigurationen beim Auftreten konkreter Aktivitäten des Musters immer *gemeinsam* in Kraft treten. In diesem Sinne handelt es sich bei der Kombination mehrerer AND/XOR-Konfigurationen um eine übergreifende logische Und-Verknüpfung. Dabei wird gefordert, dass die Mengen implementierter (durch XOR-Knoten verknüpften) Aktivitätsrollen von verschiedenen AND/XOR-Konfigurationen paarweise disjunkt sind. Auf diese Weise ist jede AND/XOR-Konfiguration einem eigenen „Implementierungsauftrag“ gewidmet. Die Mengen implementierender (durch AND-Knoten verknüpften) Aktivitätsrollen von verschiedenen AND/XOR-Konfigurationen können hingegen in beliebiger Mengenbeziehung zueinander stehen.

Aufgrund dieser Erklärungen und der logischen Semantik hinter den Verknüpfungsknoten wird klar, dass eine bestimmte Einbettungssituation mittels verschiedener AND/XOR-Konfigurationen modelliert werden kann. Alternativ zu dem Modell in Abbildung 8.10 könnten beide AND-Knoten jeweils mit *allen* Aktivitätsrollen des reception-Aktivitätsmusters verbunden sein. Das entstehende Modell würde die Einbettungssituation des Petrinetzmodells aus Abbildung 8.6 ebenfalls korrekt wiedergeben, allerdings auf eine weniger exakte Weise als das Modell aus Abbildung 8.10.

Die Notation der Verknüpfung von Rollen (und damit von Aktivitätsablaufanteilen) über logische Junktoren weist eine Ähnlichkeit zur Notation der *ereignisgesteuerten Prozessketten* auf (*EPK*, vergleiche Keller et. al. [KNS92] oder Scheer [Sch02, Teil B]). In ereignisgesteuerten Prozessketten kann der Kontrollfluss eines Prozesses über die logischen Junktoren *Und, Oder* sowie *Exklusiv-Oder* vereinigt oder aufgespalten werden. Allerdings geht es im Fall der hier

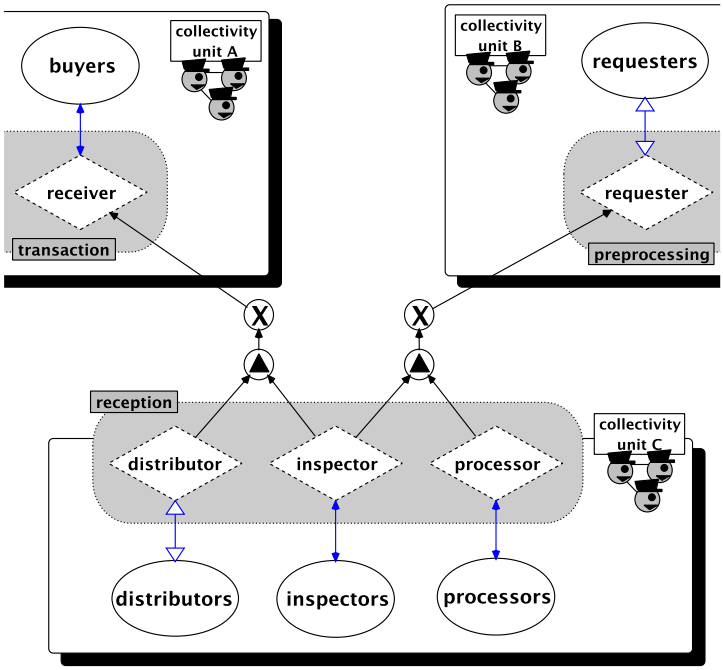

Abbildung 8.10: Kombination von AND/XOR-Konfigurationen (vergleiche Abbildung 8.6)

behandelten Akteur-Aktivitätsrollen-Modelle vor allem um die Verzahnung von Abläufen über Systemebenen hinweg. Nichtsdestotrotz bietet es sich auch hier an, gegebenenfalls weitere logische Junktoren oder sogar geschachtelte logische Ausdrücke für die Verknüpfung zuzulassen. Nach Ansicht des Autors kann dies je nach Bedarf zugelassen werden. Doch zum einen spielt die Modellierungstechnik zum Erstellen von Akteur-Aktivitätsrollen-Modellen im Folgenden der Arbeit eine eher untergeordnete Rolle und stellt hier vor allem einen Zwischenschritt dar. Zum anderen haben AND/XOR-Konfigurationen eine recht einfache und leicht zugängliche Semantik. Sie korrespondieren in der Regel zu einfachen und klar strukturierten Spezifikationen von Aktivitätsrollen auf der darunter liegenden Abstraktionsebene der petrinetzbasierten Akteur-Aktivitätsablauf-Modelle.

Zur Verdeutlichung der Ausdrucksmächtigkeit, die sich bereits durch die Verwendung von AND/XOR-Konfigurationen ergibt, zeigt Abbildung 8.11 einige Einbettungssituationen in Multiorganisationssystemen. Dabei handelt es sich eigentlich um Situationstypen, von denen hier einige Ausprägungen illustriert sind. Die Situationstypen sind orthogonal zueinander und können kombiniert werden. Nach Ansicht des Autors ist damit ein großer Teil der relevanten Einbettungssituationen in Multiorganisationssystemen durch die Verwendung von AND/XOR-Konfigurationen abgedeckt.

8.1.3.2 Akteursmengen und Aktivitätsmengen

Trotz des Verzichts auf die Betrachtung von Aktivitätsablaufdetails können Akteur-Aktivitätsrollen-Modelle bei einer großen Anzahl von Akteursmengen und Aktivitätsmustern weiterhin recht ausladend und unübersichtlich werden. Daher wird in diesem Unterabschnitt eine weitere Modellierungstechnik vorgestellt. Diese erlaubt ein beliebiges Anheben der Mo-

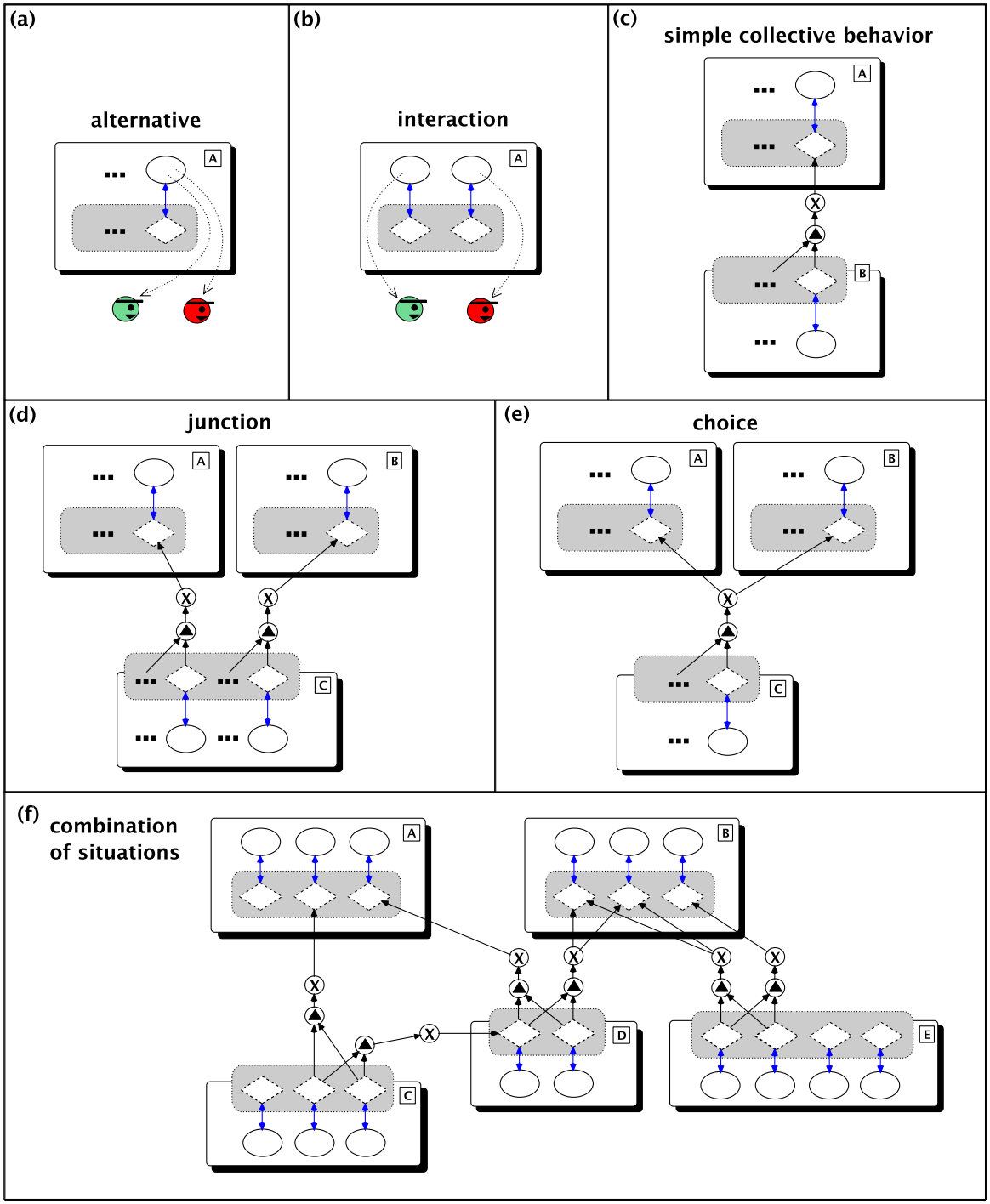

Abbildung 8.11: Einbettungssituationen in Multiorganisationssystemen

dellabstraktion durch die Betrachtung beliebig umfassender Akteurs- und Aktivitätsmengen. Die entstehenden Modelle werden dementsprechend als Akteurs-/Aktivititätsmengen-Modelle bezeichnet. Abbildung 8.12 zeigt ein einführendes Beispiel.

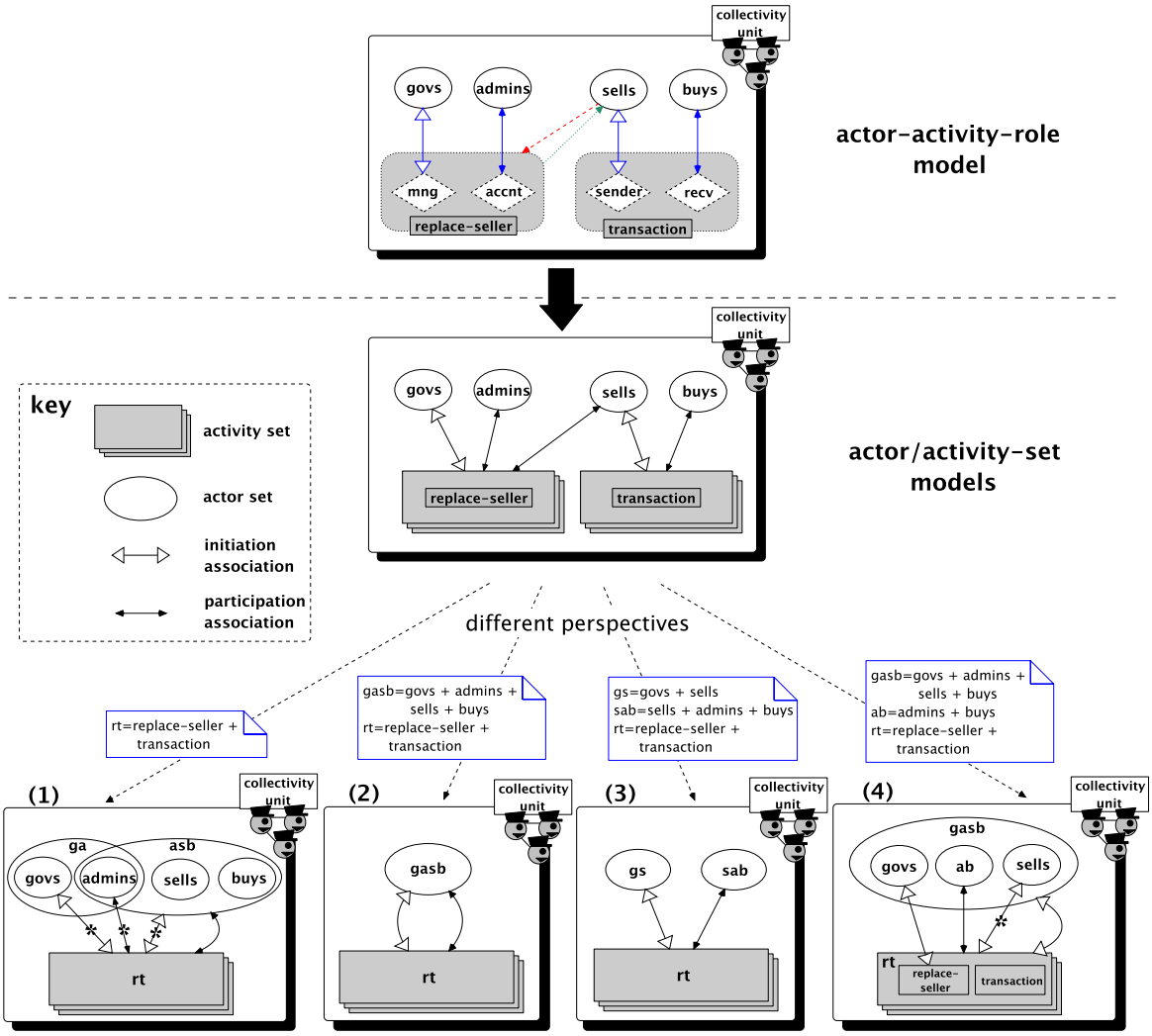

Abbildung 8.12: Exemplarischer Übergang zu abstrakter Mengendarstellung

Ausgangspunkt ist die Modellabstraktion des vorigen Unterabschnittes. Ganz oben in der Abbildung ist das Akteur-Aktivitätsrollen-Modell aus Abbildung 8.7 zu sehen. Der nächste (vorbereitende) Abstraktionsschritt besteht darin, gänzlich auf die Betrachtung von Aktivitätsrollen zu verzichten. Stattdessen werden Akteursmengen direkt mit Aktivitätsmustern assoziiert. Dabei werden nur noch zwei Assoziationen betrachtet. Die *Initiierungsassoziation* repräsentiert weiterhin die Berechtigung zur Initiierung von Aktivitäten eines Musters. Die *Teilnahmeassoziation* repräsentiert weiterhin die Berechtigung zur Teilnahme an Aktivitäten und subsumiert nun zusätzlich die Hinzufügen- und Entfernen-Assoziation. Es ergibt sich das Modell in der Mitte der Abbildung.

Bislang wurden nur diejenigen Akteursmengen betrachtet, die bereits als Stellen in den petrinetzbasierten Akteur-Aktivitätsablauf-Modellen repräsentiert waren. Auf Seite der Aktivitäten wurden Mengen bislang nur insofern betrachtet, als dass jedes Aktivitätsmuster eine Menge konkreter Aktivitäten impliziert. In der Folge ist hier von *Akteursbasismengen* beziehungsweise

Aktivitätsbasismengen die Rede. Somit stellt das Modell in der Mitte von Abbildung 8.12 das feingranularste Akteurs-/Aktivitätsmengen-Modell der betrachteten Kollektivitätseinheit dar. Der nächste Abstraktionsschritt besteht im Übergang zur Betrachtung beliebig umfassender Akteurs- und Aktivitätsmengen. Im allgemeinen Fall ergeben sich Assoziationen zwischen (nicht-leeren) Akteurs- und Aktivitätsmengen wie folgt.

- Zwischen einer Akteursmenge A und einer Aktivitätsmenge B existiert eine Assoziation vom Typ X (Initiierung, Teilnahme), wenn *für jede* Teilmenge $B_{sub} \subseteq B$ gilt, dass eine Assoziation vom Typ X zwischen B_{sub} und mindestens einer Akteursmenge $A_{sub} \subseteq A$ existiert.

- Zwischen einer Akteursmenge A und einer Aktivitätsmenge B existiert eine *bedingte* Assoziation vom Typ X (Initiierung, Teilnahme), wenn *für mindestens eine, aber nicht für jede* Teilmenge $B_{sub} \subseteq B$ gilt, dass eine Assoziation vom Typ X zwischen B_{sub} und mindestens einer Akteursmenge $A_{sub} \subseteq A$ existiert. Bedingte Assoziationen werden in Modellen durch einen Stern * gekennzeichnet.

Es gibt verschiedene Möglichkeiten, Akteurs-/Aktivitätsmengen-Modelle zu einer gegebenen Kollektivitätseinheit zu erstellen. Bei n Akteursbasismengen und m Aktivitätsbasismengen können bis zu $2^n - 1$ Akteursmengen und bis zu $2^m - 1$ Aktivitätsmengen explizit in ein konkretes Modell aufgenommen werden (die leeren Mengen jeweils ausgenommen). Eine konkrete Auswahl aus diesen Möglichkeiten impliziert eine konkrete *Perspektive* auf eine Kollektivitätseinheit. Auf der untersten Ebene der Abbildung 8.12 ist eine Auswahl von vier Perspektiven ausgehend vom Kollektivitätseinheitsmodell in der Mitte der Abbildung aufgeführt. Jede dieser Perspektiven verbirgt unterschiedlichen und unterschiedlich viel Informationsgehalt. Wäre nicht explizit angegeben, welche Mengeninklusionen ausgehend vom Modell in der Mitte der Abbildung vorgenommen werden, so könnte es sich um Modelle gänzlich verschiedener Kollektivitätseinheiten handeln. Nimmt man eine der vier Perspektiven als Ausgangspunkt der Modellierung anstatt als Resultat einer Abstraktion, so können Verfeinerungsschritte zu anderen Akteurs-/Aktivitätsbasismengen und Assoziationen als den in der Mitte der Abbildung modellierten führen.

Akteurs-/Aktivitätsmengen-Modelle können also sowohl für eine *Bottom-Up*-Entwicklung immer abstrakterer Modelle – ausgehend von einer konkreten Kollektivitätseinheit – als auch für einen *Top-Down*-Entwurf – ausgehend von einem groben Ausgangsmodell – verwendet werden. Im Kontext von Reengineering-Bemühungen kann beides kombiniert werden. Während die Akteur-Aktivitätsrollen-Modellen des vorigen Unterabschnittes genau *einen* Abstraktionsschritt gegenüber den petrinetzbasierten Akteur-Aktivitätsablauf-Modellen repräsentieren, bietet sich mit Akteurs-/Aktivitätsmengen-Modellen die Möglichkeit einer beliebigen Modulation des Abstraktionsgrades. Von besonderer Bedeutung ist dabei, ob ein Modell eine *kompatible Vergröberung* beziehungsweise *Verfeinerung* eines anderen Modells ist. Dies lässt sich wie folgt definieren:

> M_1 und M_2 seien zwei Akteurs-/Aktivitätsmengen-Modelle von Kollektivitätseinheiten. Darüber hinaus sei festgelegt, welche Mengenbeziehungen zwischen den Akteurs- und Aktivitätsmengen der beiden Modelle existieren. Dies ist beispielsweise in Abbildung 8.12 beim Übergang vom Modell in der Mitte der Abbildung zu den verschiedenen Modelle im unteren Teil der Abbildung festgehalten. M_2 ist

genau dann eine *kompatible Vergröberung* von M_1 (beziehungsweise M_1 ist eine *kompatible Verfeinerung* von M_2), wenn die folgenden Bedingungen erfüllt sind. [4]

1. Für jede Akteursmenge A in M_1 gilt, dass in M_2 eine Akteursmenge A_{super} mit $A \subseteq A_{super}$ enthalten ist. Akteursmengen aus M_1 sind also Akteursmengen aus M_2 oder sind Teilmengen von Akteursmengen aus M_2.

   Für jede Akteursmenge A in M_2 gilt, dass in M_1 Akteursmengen $A_i, ..., A_j$ mit $A = \bigcup_{i \leq z \leq j} A_z$ enthalten sind. Akteursmengen aus M_2 sind also Akteursmengen aus M_1 oder Vereinigungen über Akteursmengen aus M_1.

   Darüber hinaus gilt für *jedes* Akteurs-/Aktivitätsmengen-Modell M, dass zu einer enthaltenen Akteursmenge A entweder gar keine Akteursmenge A_{sub} mit $A_{sub} \subset A$ oder aber Akteursmengen $A_i, ..., A_j$ mit $A_z \subset A$ $(i \leq z \leq j)$ und $A = \bigcup_{i \leq z \leq j} A_z$ in M enthalten sind. Zu einer betrachteten Akteursmenge sind also entweder keine echten Teilmengen oder aber echte Teilmengen, die die betrachtete Menge aufspannen, im Modell enthalten.

   Analoge Bedingungen gelten bezüglich der Aktivitätsmengen in M_1 und M_2.

2. Es gibt in M_1 *genau dann* eine Akteursmenge A_1, eine Aktivitätsmenge B_1 und eine Assoziation vom Typ X (Initiierung, Teilnahme) zwischen ihnen, wenn es in M_2 eine Akteursmenge A_2 mit $A_1 \subseteq A_2$, eine Aktivitätsmenge B_2 mit $B_1 \subseteq B_2$ und eine Assoziation vom Typ X zwischen ihnen gibt.

   Es gilt also, dass keine Assoziation aus M_1 „verloren geht“ beziehungsweise dass jede Assoziation aus M_2 „ihren Ursprung“ in Assoziationen aus M_1 hat.

3. Enthält M_2 eine Akteursmenge A, eine Aktivitätsmenge B und eine Assoziation vom Typ X zwischen ihnen, dann ist diese Assoziation *genau dann* unbedingt, wenn Folgendes erfüllt ist: M_1 enthält Aktivitätsmengen $B_i, ..., B_j$, so dass
   a) $B = \bigcup_{i \leq z \leq j} B_z$,
   b) für jede Aktivitätsmenge B_z $(i \leq z \leq j)$ gilt, dass in M_1 eine Akteursmenge A_z mit $A_z \subseteq A$ und eine unbedingte Assoziation vom Typ X zwischen A_z und B_z enthalten ist.

Die Ausführungen machen deutlich, dass jedes Akteurs-/Aktivitätsmengen-Modell einer Kollektivitätseinheit eine kompatible Vergröberung und Verfeinerung von sich selbst ist. Desweiteren gilt, dass die Akteurs-/Aktivitätsmengen-Modelle im unteren Teil von Abbildung 8.12 kompatible Vergröberungen des Modells in der Mitte der Abbildung sind. Dabei werden Mengeninklusionen auch grafisch durch Inklusion grafischer Elemente in anderen Elementen dargestellt. Dies ist in den Vergröberungen (1) und (4) auf der unteren Hälfte von Abbildung 8.12 zu sehen. Zusätzlich zu solchen in einem Modell explizierten Mengenüberlagerungen können weitere Schnittmengen existieren, die zumindest in der durch das Modell repräsentierten Perspektive nicht sichtbar sind. Dies tritt bei der Vergröberung (3) auf. Die ursprüngliche Akteursmenge sells ist in beiden Akteursmengen gs und sab der vergröberten Perspektive enthalten. Da die Menge sells selber im vergröberten Modell aber nicht enthalten ist, ist auch

[4] Die Ausführungen zu kompatibler Verfeinerung/Vergröberung beziehen sich auf konkrete, grafisch explizierte Akteurs-Aktivitätsmengen-Modelle (Diagramme) von Kollektivitätseinheiten. Wenn hier also vom „Enthaltensein“ von Akteurs-/Aktivitätsmengen und Assoziationen in Modellen die Rede ist, so ist gemeint, diese Mengen seien in den jeweiligen Diagrammen explizit grafisch eingezeichnet.

die Schnittmenge in dieser Perspektive nicht expliziert. In der Gegenrichtung gilt dementsprechend, dass Verfeinerungen von Kollektivitätseinheitsmodellen Schnittmengen von zuvor analytisch getrennten Akteurs-/Aktivitätsmengen offen legen können.

Anhand dieser Beispielvergröberung (3) lässt sich ein weiterer Aspekt beobachten. Im Modell in der Mitte der Abbildung ist eine Initiierungsassoziation zwischen der Akteursmenge sells und der Aktivitätsmenge transaction enthalten. Daher wäre es *möglich*, Vergröberung (3) um eine (bedingte) Initiierungsassoziation zwischen der Akteursmenge sab und der Aktivitätsmenge rt zu erweitern. Es ist aber nicht *notwendig*, diese Assoziation einzuzeichnen da in der Vergröberung bereits eine Initiierungsassoziation zwischen der Akteursmenge gs und der Aktivitätsmenge rt existiert. Die entsprechende Assoziation zwischen der Akteursmenge sells und der Aktivitätsmenge transaction geht also bei der Vergröberung nicht „verloren".

Die Gegenrichtung zu den in Abbildung 8.12 betrachteten Vergröberungen von Kollektivitätseinheitsmodellen ist in Abbildung 8.13 gezeigt. Hier wird ausgehend von Vergröbe-

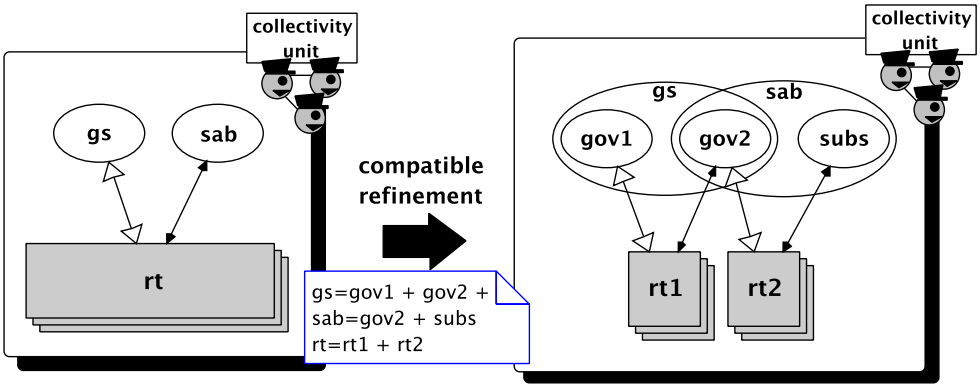

Abbildung 8.13: Alternative Verfeinerung von Perspektive (3) aus Abbildung 8.12

rung (3) aus Abbildung 8.12 eine alternative Verfeinerung gegenüber dem Modell in der Mitte der Abbildung vorgenommen.

Abschließend zeigt Abbildung 8.14 den Übergang von der Betrachtung einzelner Kollektivitätseinheiten zu Multiorganisationssystemen auf Basis von Akteurs-/Aktivitätsmengen-Modellen. Gegenüber den zuvor behandelten Akteur-Aktivitätsrollen-Modellen ist der Informationsgehalt erwartungsgemäß erheblich eingeschränkt. Betrachtet werden Spezifikationen bezüglich der Frage, ob ausgehend von einer Menge Peripherieaktivitäten einer Kollektivitätseinheit Verzahnungen/Überlappungen zu Aktivitätsmengen anderer Kollektivitätseinheiten existieren. Dabei ist zu beachten, dass jede Aktivitätsmenge eigentlich auf eine Menge von Aktivitätsmustern zurückgeht (aus denen sich die möglichen Aktivitäten der Menge ergeben). Es wird spezifiziert, ob für alle oder für einige der Aktivitätsmuster einer Peripherieaktivitätsmenge eine potentielle Verzahnung mit Aktivitätsmustern anderer Aktivitätsmengen existiert.[5] Dabei können für eine Menge von Peripherieaktivitäten natürlich mehrere solcher Spezifikationen vorgenommen werden (analog zur Spezifikation beliebig vieler AND/XOR-Konfiguration ausgehend von einer Peripherieaktivität in Akteur-Aktivitätsrollen-Modellen).

[5] Der Zusatz *potentiell* rührt daher, dass zwischen zwei Aktivitätsmustern eine Verzahnungsmöglichkeit spezifiziert sein kann, die nicht in allen Aktivitätsinstanzen der beiden Muster auftritt. Dies wird anhand der Erläuterungen zu Akteur-Aktivitätsrollen-Modellen des vorigen Unterabschnittes deutlich.

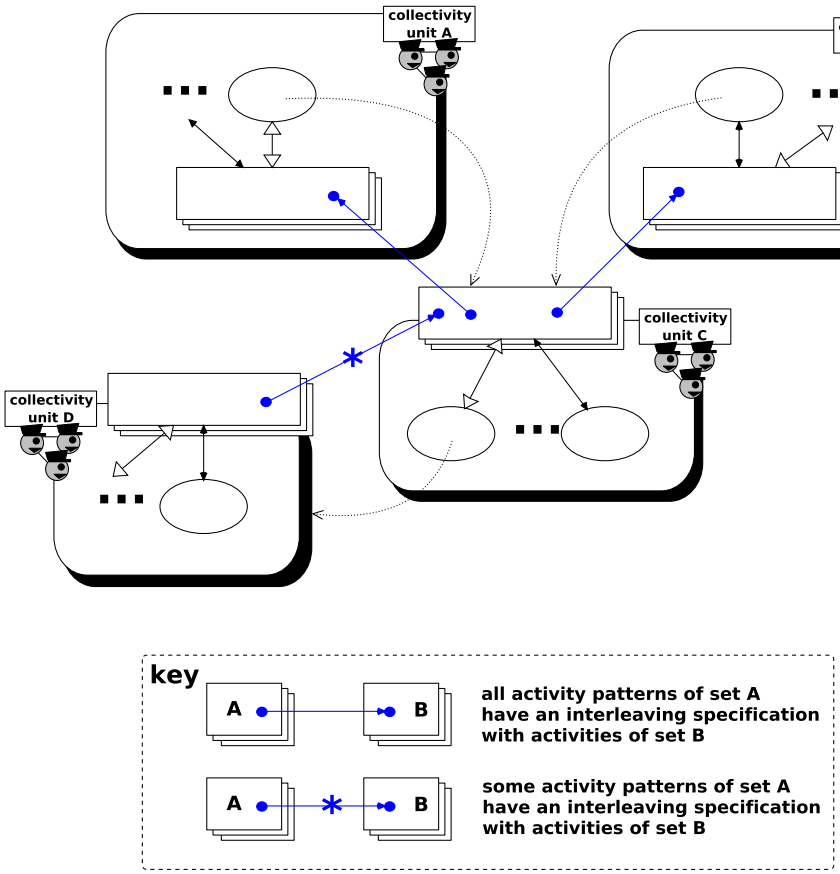

Abbildung 8.14: Multiorganisationssystem basierend auf Akteurs-/Aktivitätsmengen-Modellen

8.2 Organisationseinheiten

Bisher wurden Modellierungstechniken für die Erstellung von Kollektivitätseinheitsmodellen sowie einige einfache Beispiele von Kollektivitätseinheiten präsentiert. In der Folge werden Organisationseinheiten als spezifische Ausprägungen von Kollektivitätseinheiten vorgestellt. Im Fokus steht dabei, die drei konzeptionellen Richtungen des „nach unten/innen Schauens“, des „auf die eigene Ebene Schauens“ und des „nach oben/außen Schauens“ herauszuarbeiten. Diese drei Richtungen wurden in Abschnitt 7.2 als abstrakter und allgemeingültiger Rahmen für die Betrachtung von formalisierten Steuerungs-/Kontrollstrukturen und Zielorientierung einer Organisationseinheit propagiert. Hier wird ein *universelles Grundmodell* einer Organisationseinheit entlang der drei konzeptionellen Richtungen abgeleitet. Dessen Ausprägung für konkrete Organisationseinheiten in spezifischen Anwendungskontexten wird in diesem Kapitel nicht thematisiert und wird erst in Teil III der vorliegenden Arbeit behandelt.

Die nach innen und auf die eigene Ebene gerichteten Perspektiven werden in Teilabschnitt 8.2.1 betrachtet, bevor die nach außen gerichtete Perspektive in Teilabschnitt 8.2.2 beleuchtet wird. Die Modelle dieses Abschnittes werden als Akteurs-/Aktivitätsmengen-Modelle präsentiert, da es hier vornehmlich um ein abstraktes „Denkmodell“ für die Betrachtung

und das Studium von Organisationseinheiten geht. Der im vorigen Abschnitt vorgenommene schrittweise Übergang zwischen verschiedenen Graden der Modellabstraktion stellt zum einen die Fundierung der Modelle dieses Abschnittes auf (höherer) Petrinetzsemantik sicher. Zum anderen dient er als Anleitung, wie das abstrakte und universelle Grundmodell einer Organisationseinheit schrittweise konkretisiert werden kann.

8.2.1 Steuerung und Kontrolle

Entsprechend der Erläuterungen in Abschnitt 7.2 beschäftigt sich die nach innen gerichtete Perspektive mit der organisatorischen Einbettung von Akteuren gemäß geltender Systemregeln. Dabei wird vor allem der Frage nachgegangen, wie aus den einzelnen Akteuren ein gemeinsames System geformt wird. Die auf die eigene Ebene gerichtete Perspektive beschäftigt sich mit den Systemregeln selber. Regeln müssen kontinuierlich evaluiert und gegebenenfalls angepasst werden. Demnach gibt die auf die eigene Ebene gerichtete Perspektive einen strategischen Rahmen für die nach innen gerichtete Perspektive vor. Diese wiederum gibt ihrerseits einen operationalen Rahmen für regelkonformes Systemverhalten vor.

Hier stellt sich die Frage, was mit *Systemregeln* eigentlich gemeint ist. Die Ausführungen in Kapitel 4 der vorliegenden Arbeit machen deutlich, dass es in sozialen Organisationen ein breites Spektrum an verschiedenen Arten von Regeln gibt. In Kapitel 7.1 wurde die inhaltliche Eingrenzung vorgenommen, dass hier ausschließlich formale Strukturen einer Organisationseinheit betrachtet werden. Regeln bezeichnen hier somit formalisierte Verhaltensvorschriften und Strukturspezifikationen Diese Sichtweise ist weiterhin recht vage denn entsprechend der Erklärungen in Kapitel 4 ist auch das Spektrum formaler Systemregeln breit gefächert. Allerdings wird in diesem Kapitel keine systematische Differenzierung verschiedener Regeln und den daraus resultierenden Formalstrukturen vorgenommen. Dieser Thematik ist der dritte Teil der vorliegenden Arbeit ausführlich gewidmet. Für den Moment genügt es, Systemregeln ganz allgemein als die Formalisierung der organisatorischen Prinzipien einer Organisationseinheit zu begreifen.

Abbildung 8.15 zeigt eine erste Version eines Organisationseinheitsmodells. Die beiden konzeptionellen Richtungen des „nach unten Schauens“ und des „auf die eigene Ebene Schauens“ finden sich als Mengen von *Integrationsaktivitäten* (integration activities) beziehungsweise *Governance-Aktivitäten* (governance activities) wieder. Mit diesen beiden grundsätzlichen Mengen von Systemaktivitäten gehen verschiedene Verantwortlichkeiten seitens der Akteure einher. Die Abbildung zeigt dementsprechend eine Differenzierung von Akteuren in *Governance-Akteure* (governance actors), *Integrationsakteure* (integration actors) und *Operationsakteure* (operational actors) zusammen mit einer Differenzierung bezüglich ihrer jeweiligen Assoziationen mit den Aktivitätsmengen.

Zunächst bedarf es an dieser Stelle eines Einschubs für die Klärung des Begriffs *Governance*. Grundsätzlich unterliegt der Begriff heterogenen Verwendungsweisen. Unter einer wirtschaftswissenschaftlichen Perspektive bezeichnen Holtschke et. al. [HHH09, S. 161] Governance beispielsweise schlicht als die „Schaffung verbindlicher Rahmenbedingungen für eine gute und richtige Unternehmensführung“. Langer und Hüther [LH09] nehmen hingegen eine sehr differenzierte Betrachtung der Begriffsgeschichte aus soziologischer und politikwissenschaftlicher Perspektive vor. Als Quintessenz ergibt sich bei ihnen, dass Governance sich mit zweckmäßigem beziehungsweise zielgerichtetem Verhalten und Regeln für die Steuerung solchen Verhaltens auseinandersetzt. Allerdings wird gegenüber dem klassischen Steuerungsbegriff (klare Trennung von Steuerungssubjekt und -objekt, vergleiche auch Abschnitt 4.3.4 zur Kyberne-

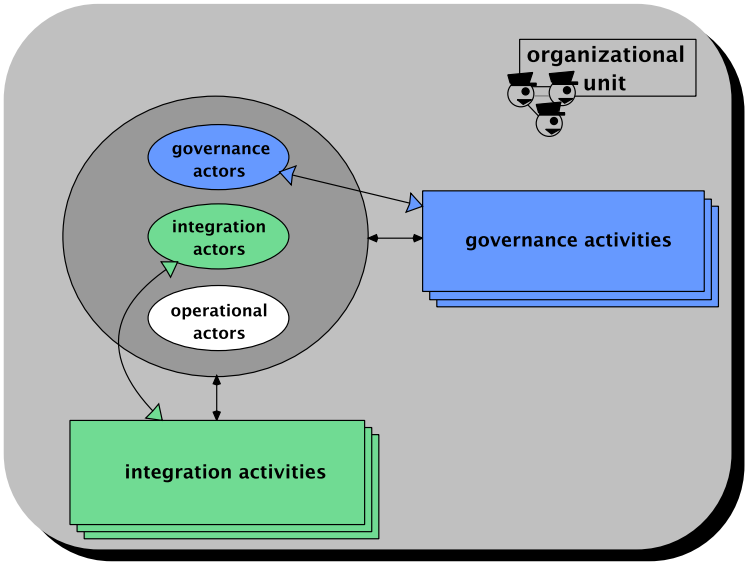

Abbildung 8.15: Organisationseinheit intern: Governance, Integration, Operation

tik) betont, dass es mehrere Akteure mit Steuerungsbefugnis gibt, diese aufeinander einwirken und ihre Beziehungen damit untereinander regulieren. Darüber hinaus überschreitet Governance „herkömmliche“ Grenzen. Steuerungsakteure verschiedener konkreter Systeme können somit nicht als souverän und autonom betrachtet werden, sondern unterliegen wechselseitigen Abhängigkeitsbeziehungen. Damit ergibt sich eine enge inhaltliche Korrespondenz des Governance-Begriffs zu dem in Kapitel 4.3.1 betrachteten Konzept der dominanten Koalition in Organisationen. Da die dominante Koalition allerdings traditionell mit Unternehmen/Firmen assoziiert ist, wird der Governance-Begriff in der Folge als der allgemeinere für die Bezeichnung der strategischen Steuerungs-/Kontrollinstanz im universellen Grundmodell einer Organisationseinheit verwendet.

Die Funktion der einzelnen Akteursmengen einer Organisationseinheit sowie der inhaltliche Zweck der Aktivitätsmengen werden im Folgenden beschrieben. Dabei stellt der hohe Abstraktionsgrad der Akteurs-/Aktivitätsmengen-Modelle von Organisationseinheiten dieses Abschnittes ein gewisses Problem dar. Verfeinerungen würden allerdings Konkretisierungen mit sich bringen, während hier das universelle Grundmodell im Fokus steht. Daher wird hier vor allem auf eine Charakterisierung in Form textueller Erläuterungen und eines UML-Konzeptdiagramms zurückgegriffen. Für das abstrakte Denkmodell einer Organisationseinheit erachtet der Autor der vorliegenden Arbeit diese Herangehensweise als passend. Eine detaillierte petrinetzbasierte Operationalisierung (/Interpretation) des hier beschriebenen Zusammenspiels von Governance, Integration und Operation erfolgt in Kapitel 9 im Rahmen der Vorstellung des operationalen „Programmiermodells“ von Organisationseinheiten.

Bei Operationsakteuren handelt es sich sozusagen um die „intrinsischen“ Akteure der Organisationseinheit. Sie führen die primären Systemoperationen aus und repräsentieren auf diese Weise, in welchen Domänen sich die Organisationseinheit als Ganzes bewegt. Nichtsdestotrotz sind Operationsakteure von den anderen beiden Akteursmengen abhängig, um tatsächlich im Kontext der Organisationseinheit agieren zu können. Die Integrations- und Governance-

Akteure sind jeweils für die Integrations- beziehungsweise Governance-Aktivitäten verantwortlich. Sie initiieren und ermöglichen die Aktivitäten überhaupt erst. Außerdem nehmen sie mittels Aktivitätsparametern und gegebenenfalls eigener Teilnahme eine inhaltliche Steuerung und Kontrolle der Aktivitäten vor. Die Namensgebung der Aktivitätsmengen deutet bereits ihren Zweck an. Bei Integrationsaktivitäten handelt es sich um „alltägliche“ Aktivitäten. Sie repräsentieren das Systemverhalten entlang geltender Systemregeln und dienen der Integration der Einzelakteure in ein kohärentes Ganzes. Bei Governance-Aktivitäten handelt es sich um „außerordentliche“ Aktivitäten, im Zuge derer die Organisationsprinzipien der Organisationseinheit in Form ihrer Systemregeln in Frage gestellt und neuausgerichtet werden können. Governance- und Integrationsakteure haben jeweils die Hoheit über diese Aktivitäten und bilden damit ein zweistufiges Steuerungs- und Kontrollsystem, wie es bereits in Kapitel 4 als grundlegend für soziale Organisationen (und auch für kybernetische Systeme) identifiziert wurde. Als Grundlage für den Fortgang dieses Teilabschnittes dient das UML-Konzeptdiagramm aus Abbildung 8.16. Es illustriert das Beziehungsgeflecht zwischen Governance, Integration und Operation anschaulich. Governance und Integration bilden gemeinsam

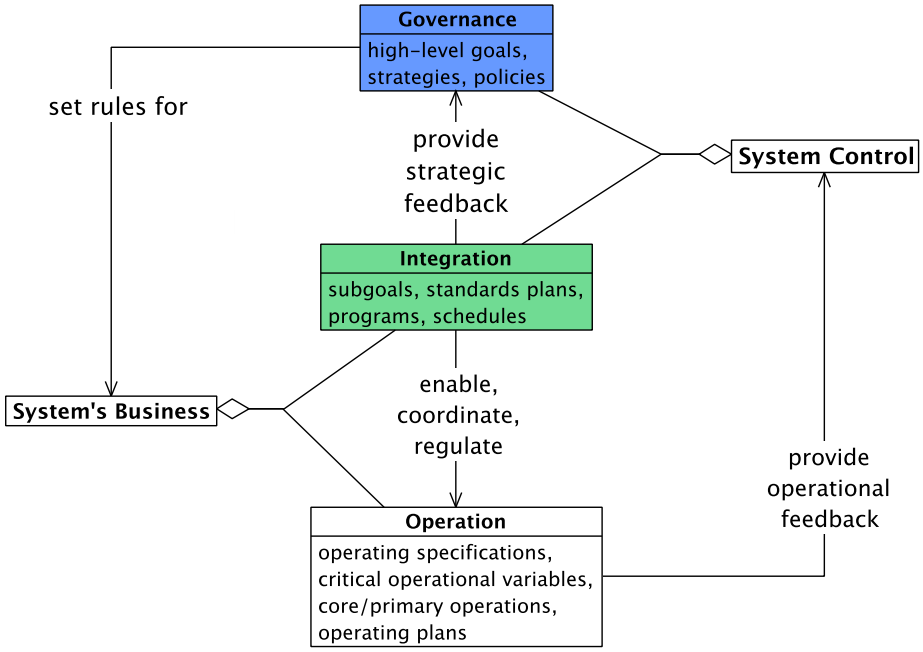

Abbildung 8.16: Konzeptdiagramm für das Zusammenspiel von Operation, Integration und Governance

die Systemsteuerung/-kontrolle der Organisationseinheit. Integration und Operation bilden gemeinsam den „täglichen Betrieb“ der Organisationseinheit.

Governance-Akteure entwickeln globale Ziele/Strategien der Organisationseinheit. Daraus werden Systemregeln für den täglichen Betrieb der Organisationseinheit abgeleitet. Die Integrationsakteure sind dafür verantwortlich, diese Regeln auf den aktuellen operationalen Kontext der Organisationseinheit anzuwenden und sie entsprechend zu konkretisieren. Sie entwickeln Teilziele und Standards auf Basis der Systemregeln und generieren darauf aufbauend konkrete Pläne, Programme und Abläufe, anhand derer sie die Operation des Systems

ermöglichen, koordinieren und regulieren. In diesem Kontext führen die Operationsakteure schließlich die Primär-/Kernoperationen und -aufgaben der Organisationseinheit aus. Dabei müssen sie ihre lokalen Operationspläne und -spezifikationen den Vorgaben von Seiten der Integration anpassen. Je nachdem, wie reibungslos sich die Arbeitsvorgänge aus der Perspektive der Operationsakteure gestalten, liefern sie operationales (niedrigrangiges) Feedback an die Systemsteuerung/-kontrolle. Als steuernde und kontrollierende Instanz haben die Integrationsakteure ein globaleres (und gegebenenfalls von irrelevanten technischen Detail bereinigtes) Bild davon, wie sich die Operation der Organisationseinheit entlang geltender Systemregeln betreiben lässt. Sie liefern dementsprechend strategisches (hochrangiges) Feedback an die Governance-Akteure.

Anhand dieser Erläuterungen ergibt sich ein klareres Bild von Integrations- und Governance-Aktivitäten als Inkarnationen der Richtungen des „nach unten/innen“ beziehungsweise des „auf die eigene Ebene“ Schauens. Im Zuge von Governance-Aktivitäten verhandeln Governance-Akteure die Systemziele und -strategien untereinander, vermitteln die daraus abgeleiteten Systemregeln an die übrigen Akteure und sammeln Feedback darüber, wie sich diese Systemregeln bewähren. Dementsprechend steht die Konfiguration der Organisationseinheit in Form ihrer organisatorischen Prinzipien im Fokus. Im Zuge von Integrationsaktivitäten koordinieren sich die Integrationsakteure, um eine effektive und konsistente Anwendung der Systemregeln auf den aktuellen operationalen Kontext der Organisationseinheiten zu erzielen. Dazu gehört die situationsbezogene Interpretation, Konkretisierung und Verfeinerung von Systemregeln. Darüber hinaus initiieren Integrationsakteure regelkonforme Aktivitäten, im Zuge derer Operationsakteure ihre Aufgaben ausführen. Damit finden sich die Integrationsakteure „zwischen“ den Governance-Akteuren und den Operationsakteuren wieder. Während Governance-Akteure *strategische Steuerungsentscheidungen* (Bestimmung von Systemregeln) treffen, ist die Umsetzung dieser Entscheidungen den Integrationsakteuren vorbehalten. Zu diesem Zweck treffen die Integrationsakteure *operationale Steuerungsentscheidungen* (Implementierung von Systemregeln) innerhalb des von den Governance-Akteuren vorgegebenen strategischen Rahmens. Diese Trennung von Zuständigkeiten zielt darauf ab, Oszillation zwischen hochrangigen Entscheidungen der Systemsteuerung und deren Auswirkungen auf die feingranularen und technischen Details der Systemoperation abzuschwächen. Zwar ist nicht ausgeschlossen, dass Operationsakteure und Governance-Akteure unmittelbar miteinander interagieren, typischerweise wirken die Integrationsakteure jedoch als Puffer zwischen ihnen und haben einen „beruhigenden Effekt“ auf die Organisationseinheit. Einer der wesentlichen Unterschiede zwischen strategischen und operationalen Steuerungs-/Kontrollentscheidungen ist der Zeitraum, für den sie bestimmt sind. Strategische Entscheidungen gelten typischerweise für einen langen Zeitraum da sie stabile und verlässliche Rahmenbedingungen vorgeben. Operationale Entscheidungen hingegen sind vom aktuellen operationalen Kontext der Organisationseinheit abhängig und werden mit höherer Frequenz neu getroffen.

Operation, Integration und Governance sind vor allem *analytische Aspekte* einer Organisationseinheit. Die Trennung in die entsprechenden Mengen von Akteuren ist dementsprechend eine *analytische Trennung*. Die drei Aspekte sind inhärent miteinander verwoben. Diesem Umstand wurde im Laufe des Teilabschnittes bereits Rechnung getragen, indem das Zusammenspiel der drei Akteursmengen beschrieben und charakterisiert wurde. Ebenso wichtig für das Verständnis einer Organisationseinheit ist allerdings die Tatsache, dass die drei Akteursmengen auf der Abstraktionsebene des universellen Grundmodells einer Organisationseinheit getrennt aufgeführt sind, die Mengen jedoch typischerweise *nicht* disjunkt sind. Bereits bei der Vorstellung der Modellierungstechnik für die Erstellung von Akteurs-/Aktivitätsmengen-Modellen in

Unterabschnitt 8.1.3.2 wurde explizit darauf verwiesen, dass das Absenken des Abstraktionsgrades Schnittmengen zwischen zuvor analytisch getrennten Akteurs- und Aktivitätsmengen offenbaren kann. Darüber hinaus kann multiple Akteurseinbettung auftreten und sogar von vorneherein vorgesehen sein. Beispielsweise mag es zunächst etwas seltsam anmuten, dass die Operationsakteure vollständig abhängig von den anderen beiden Akteursmengen sind und ihrerseits keinerlei Aktivitäten initiieren können (zumindest nicht entlang der Formalstruktur einer Organisationseinheit). Ihnen ist somit nicht einmal möglich, kritische Fehlfunktionen proaktiv zu melden. Als Abhilfe ist beispielsweise möglich, dass von Seiten der Integrations- oder Governance-Akteure einige dauerhaft aktive Aktivitäten bereitgestellt werden, an denen die Operationakteure bei Bedarf teilnehmen können. Intuitiver ist jedoch die Sichtweise, dass Akteure je nach Kontext in unterschiedlichen analytischen Rollen (Operation, Integration, Governance) auftreten können. So könnte beispielsweise vorgesehen sein, dass hinsichtlich essentieller niedrigrangiger Aktivitäten wie dem Absetzen von Alarmsignalen *jeder* Akteur einer Organisationseinheit als Integrationsakteur auftritt. Speziell hinsichtlich der Mengen von Integrations- und Operationsakteuren ist ein hoher Grad an Überlappung zu erwarten.

Abschließend sei noch einmal auf den Charakter der hier vorgestellten Modelle als universelle Grundmodelle von Organisationseinheiten hingewiesen. Die weitläufige Charakterisierung von Steuerungs-/Kontrollaspekten in Form von Governance und Integration ist hochgenerisch. Für spezifische Ausprägungen von Organisationseinheiten können sich diese Aspekte innerhalb eines breiten Spektrums manifestieren. Für den Aspekt der Governance (und das verwandte Konzept der dominanten Koalition) wurde bereits betont, dass hier komplexe und veränderliche Koalitionen von Akteuren mit unterschiedlichen Verantwortlichkeiten und Einflüssen hinsichtlich der Steuerungsentscheidungen auftreten können. Auch auf Seiten der Integration gilt, dass komplexe Beziehungen inklusive verschiedener Verantwortlichkeiten und Hierarchieebenen der Autorität zwischen Integrationsakteuren auftreten können. Für konkrete Organisationseinheiten werden solche Zusammenhänge erst deutlich, wenn der Abstraktionsgrad ausgehend von den hier vorgestellten Grundmodellen gesenkt wird.

8.2.2 Organisationseinheit: Offenheit und Grenzmanagement

Bislang wurde lediglich die Funktion einer Organisationseinheit als organisatorischer *Kontext* thematisiert. Um dem geforderten janusgesichtigen Charakter (vergleiche Abbildung 7.2) gerecht zu werden, gilt es nun, auch die Funktion als kollektiver Akteur zu adressieren. Kollektives Akteurswesen wurde in Teilabschnitt 8.1.2 bereits für den allgemeineren Fall einer Kollektivitätseinheit behandelt. Für Organisationseinheiten werden spezifische Betrachtungen notwendig. Damit wird das abstrakte, universelle Grundmodell einer Organisationseinheit zum Abschluss gebracht. Gleichzeitig erfolgt der Übergang von der Betrachtung einer Organisationseinheit zu Multiorganisationssystemen. Der janusgesichtige Charakter einer Organisationseinheit erlaubt einen rekursiven Aufbau von Multiorganisationssystemen, indem das universelle Modell einer Organisationseinheit auf beliebigen Ebenen der Systemabstraktion angewendet wird.

Abbildung 8.17 illustriert die „Öffnung“ des bisherigen Grundmodells einer Organisationseinheit. Aus der Perspektive einer Organisationseinheit entspricht ihr kollektives Akteurswesen gerade der konzeptionellen Richtung des „nach oben/außen Schauens“. Im vorhergehenden Teilabschnitt wurden bereits die beiden Richtungen des „nach unten/innen Schauens“ und des „auf die eigene Ebene Schauens“ jeweils mit eigenen Mengen von Systemaktivitäten identifiziert. Dementsprechend wird auch die nach außen gerichtete Perspektive durch eine eigene

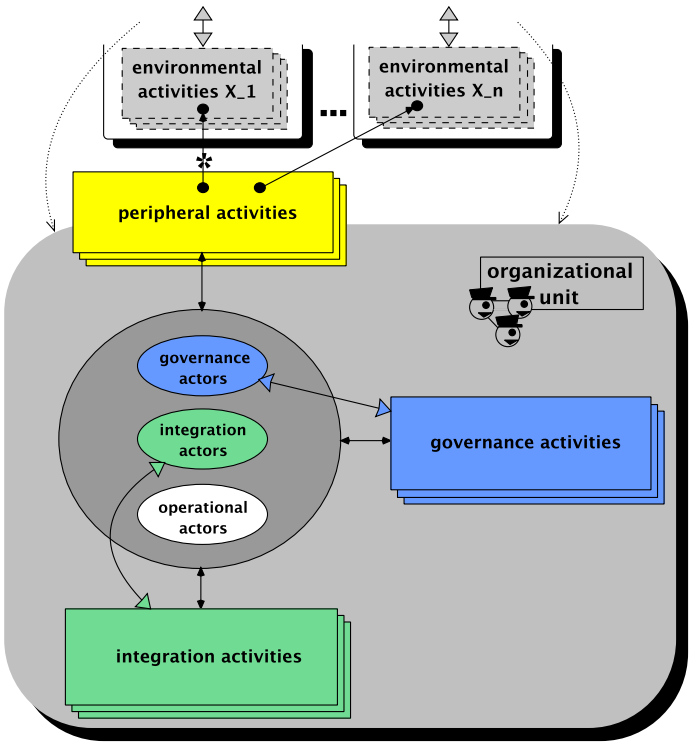

Abbildung 8.17: Organisationseinheit: Peripherieaktivitäten

Aktivitätsmenge repräsentiert. Mittels Peripherieaktivitäten (peripheral activities) nimmt eine Organisationseinheit an Umgebungsaktivitäten (environmental activities) teil. Bei diesen Umgebungsaktivitäten handelt es sich um Aktivitäten auf Ebene von umschließenden Organisationseinheiten. Dabei gilt ausdrücklich, dass eine Organisationseinheit mehrere konzeptionelle Umgebungen haben kann und demnach im Kontext mehrerer umfassender Organisationseinheiten agieren kann. Wie Umgebungsaktivitäten aussehen, wird auf Ebene der umschließenden Organisationseinheiten bestimmt. Eine teilnehmende Organisationseinheit muss ihre Peripherieaktivitäten entsprechend gestalten, so dass sie zu den Umgebungsaktivitäten passen. Dieser Aspekt der Verzahnung von Aktivitäten über Systemebenen hinweg wurde in Teilabschnitt 8.1.2 unter Bezug auf Petrinetzsemantik und insbesondere auf das Netze-in-Netzen-Konzept behandelt.

Das Organisationseinheitsmodell aus Abbildung 8.17 illustriert den Zusammenhang zwischen den drei geforderten konzeptionellen Richtungen und den Mengen von Aktivitäten zwar sehr anschaulich, stellt aber eigentlich eine Vereinfachung beziehungsweise eine Unterspezifikation dar. Tatsächlich gestalten sich die Dinge an den Grenzen einer Organisationseinheit komplizierter und differenzierter als es in dem Modell wiedergegeben ist. Während die Abbildung bereits illustriert, dass Umgebungsaktivitäten von umschließenden Organisationseinheiten bereitgestellt und gesteuert/kontrolliert werden, so bleibt die Frage nach Initiierung und Steuerung/Kontrolle der Peripherieaktivitäten zur Teilnahme an den Umgebungsaktivitäten offen. Abbildung 8.18 expliziert die beiden grundsätzlichen Möglichkeiten für die Beantwortung der Frage. Wenn eine sehr starke Abhängigkeit einer Organisationseinheit von einer umschließen-

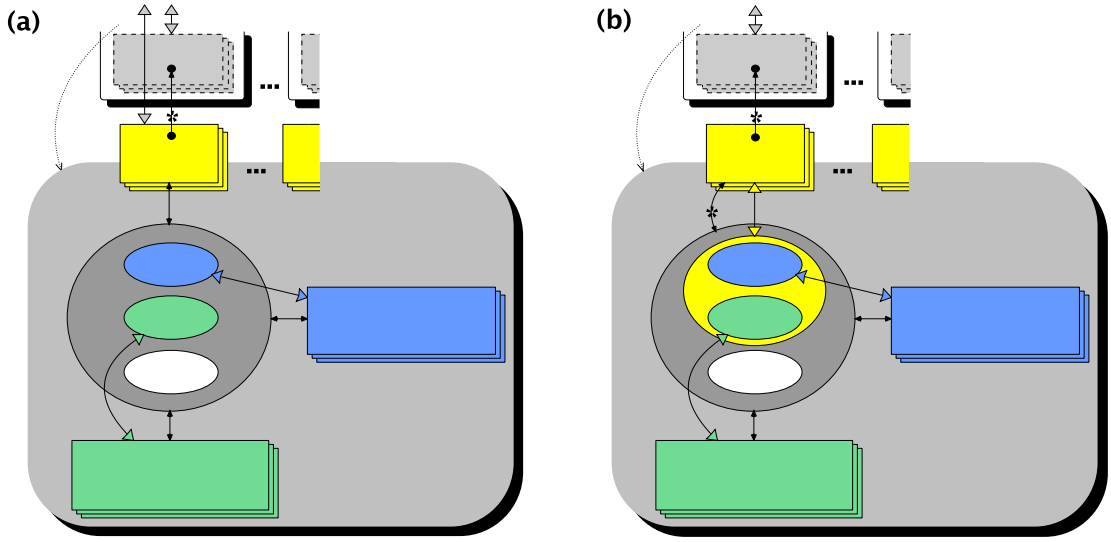

Abbildung 8.18: (a) Direkter Durchgriff, (b) Pufferung

den Organisationseinheit besteht, kann der in Abbildung 8.18 (a) modellierte *direkte Durchgriff* für die Initiierung und Kontrolle von bestimmten Peripherieaktivitäten durch umschließende Organisationseinheiten auftreten. Der Normalfall ist hingegen die in Abbildung 8.18 (b) modellierte Pufferung, bei der die Systemsteuerung-/kontrolle einer Organisationseinheit die Hoheit über Peripherieaktivitäten hat. Es hängt letztlich von der konkreten Einbettungsbeziehung zur jeweiligen Umgebung ab, welche der beiden Möglichkeiten auftritt.

Abbildung 8.18 (b) führt unmittelbar zu einer weiteren Differenzierung, die zuvor bereits für die internen Aktivitäten einer Organisationseinheit getätigt wurde. In Abbildung 8.19 ist veranschaulicht, dass auch für Peripherieaktivitäten unterschieden wird, ob sie einen Governance- oder einen Integrations-Charakter haben. Es handelt sich wieder um die Unterscheidung, ob die Aktivitäten dem Festlegen allumfassender Systemregeln oder der Anwendung dieser Systemregeln auf den operationalen Kontext der Organisationseinheit dienen. Allerdings geht es nunmehr darum, dass die entsprechenden Aktivitäten die Grenzen der Organisationseinheit überschreiten und somit in Abstimmung mit der Umgebung der Organisationseinheit stattfinden. Systemregeln werden in Übereinstimmung mit anderen Organisationseinheiten aus der Umgebung der fokussierten Organisationseinheit festgelegt. Dementsprechend laufen auch einige der Aktivitäten, die diesen Regeln folgen, Organisationseinheit-übergreifend ab. Zusammenfassend spiegelt das Grenzmanagement einer Organisationseinheit ihre interne Steuerungs- und Kontrollkonfiguration wieder.

Orthogonal dazu ergibt sich ein weiterer wichtiger Aspekt. Bislang wurden die Umgebungsaktivitäten der im Modell fokussierten Organisationseinheit nicht weiter differenziert. Aufgrund des rekursiven Aufbaus von Multiorganisationssystemen handelt es sich bei diesen Aktivitäten um Integrations-, Governance- oder Peripherieaktivitäten umschließender Organisationseinheiten. Dabei kann es zu Unterschieden bezüglich der analytischen Interpretation von Aktivitäten auf verschiedenen Systemebenen kommen. Abbildung 8.20 zeigt ein Beispiel dafür. In diesem Marktszenario wird auf Ebene der Markt-Organisationseinheit eine konkrete Aktivitätsmenge für die Formation von strategischen Allianz zwischen Teilnehmern des

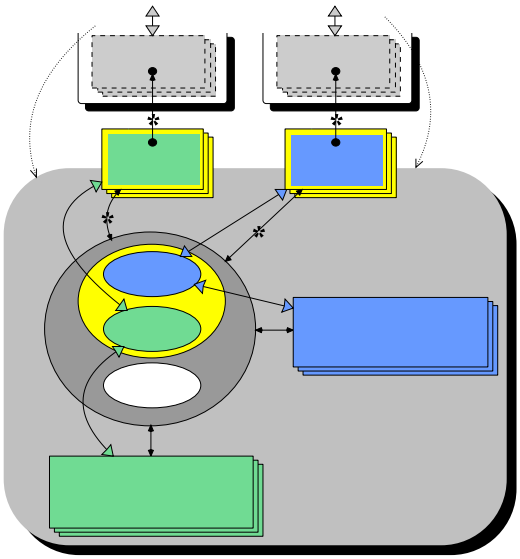

Abbildung 8.19: Peripherieaktivitäten mit Integrations- oder Governance-Charakter

Marktes betrachtet. Diese Teilnehmer sind selber Organisationseinheiten in Form eingebetteter Unternehmen. Hier sind zwei der Unternehmens-Organisationseinheiten exemplarisch aufgeführt. Sie nehmen an den Marktaktivitäten für die Formation strategischer Allianzen jeweils mittels Peripherieaktivitäten mit Governance-Charakter teil, da eine solche Formation für sie jeweils zu neuen oder modifizierten Geschäftsregeln führt. Aus Perspektive des Marktes gehören die Unternehmens-Organisationseinheiten zur Menge der Operationsakteure und die Formationsaktivitäten sind Integrationsaktivitäten: Allianzformation zwischen operationalen Teilnehmern des Feldes gehört hier zum täglichen Geschäft und findet entlang geltender (institutioneller) Marktregeln statt. Dementsprechend werden diese Aktivitäten durch eine Regulierungsbehörde initiiert und überwacht. Für diese Behörde bedarf es dafür Peripherieaktivitäten mit Integrationscharakter, da die Regulierung von Marktaktivitäten entlang der institutionellen Logik zu ihren täglichen Aufgaben zählt.

Das Beispiel illustriert eine Seite des Prinzips der *wechselseitigen Konstitution* zwischen Systemebenen. Im Beispiel hat das Agieren auf einer höheren Systemebene (dem Markt) und die Konformität zu den dort herrschenden Strukturen eine *Prägung* der Strukturen eingebetteter Organisationseinheiten (den Unternehmen) zur Folge. In der Gegenrichtung tragen eingebettete Akteure zur *Konstruktion* der Strukturen auf umschließenden Systemebenen bei. Beispielsweise könnte die Regulierungsbehörde neben ihrer Funktion als Integrationsakteur auch als Governance-Akteur auftreten und gemeinsam mit anderen einflussreichen Organisationseinheiten im Zuge von Governance-Aktivitäten für eine modifizierte institutionelle Logik des Marktes sorgen.

8.3 Zusammenfassung und Diskussion

Im Wesentlichen lassen sich zwei Ergebnisse dieses Kapitels festhalten. Das erste Ergebnis besteht in der Bereitstellung von verschiedenen Techniken für die Erstellung abstrakter Modelle

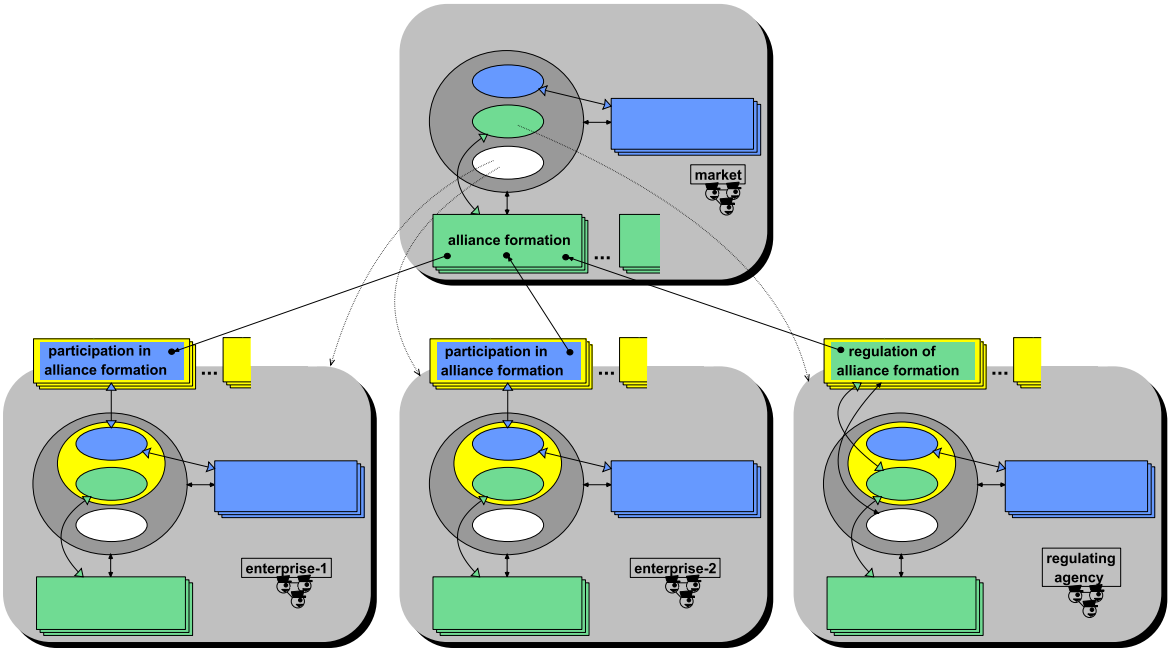

Abbildung 8.20: Unterschiedliche analytische Interpretationen von Aktivitäten am Beispiel

von Kollektivitätseinheiten, Organisationseinheiten und darauf aufbauend von Multiorganisationssystemen. Sie dienen vor allem der Planung, dem Entwurf und der Diskussion von Organisationseinheiten und Multiorganisationssystem-Ausschnitten auf einem relativ hohen Abstraktionsniveau. Die vorgestellten Modellierungstechniken erlauben dabei die Darstellung von konkreten Situationen eines Multiorganisationssystems und von Abläufe, die in diesen Situationen möglich sind. Ausgehend von petrinetzbasierten Akteur-Aktivitätsablauf-Modellen über Akteur-Aktivitätsrollen-Modelle bis hin zu Akteurs-/Aktivitätsmengen-Modelle wurden Techniken vorgestellt, die der Erstellung von Modellen dienen, mit denen immer umfassendere Ausschnitte immer kompakter dargestellt werden können. Je abstrakter die erstellten Modelle werden, desto weniger stehen konkrete Ablaufdetails im Vordergrund. Stattdessen erfolgt eine Verlagerung des Fokus auf die Identifizierung analytischer Komponenten und funktionaler Beziehung zwischen ihnen. Hier ergibt sich eine Verwandtschaft zur Vorgehensweise und dem inhaltlichen Vorhaben von Oberquelle [Obe87] bei seiner Vorstellung der Netzklasse der *Rollen-/Funktions-/Aktions-Netze*. Die Netze erlauben die Modellierung von Arbeitsumgebungen mit verschiedenen Präzisionsgraden. Aktionsnetze repräsentieren dabei den „semantischen Kern“ und alle weiteren Konzepte können als Abstraktionen davon aufgefasst werden. Analog können Akteur-Aktivitätsablauf-Modelle in der vorliegenden Arbeit als semantischer Kern aller weiteren abstrakten Modelle aufgefasst werden.

Das zweite Ergebnis des Kapitels besteht in der Bereitstellung eines universellen Grundmodells einer Organisationseinheit. Organisationseinheiten werden als spezielle Kollektivitätseinheiten modelliert und in diesem Zusammenhang wurde das Grundmodell einer Organisationseinheit als Akteurs-/Aktivitätsmengen-Modell erarbeitet. Es handelt sich somit um ein Modell auf besonders hoher Abstraktionsstufe. Dies ist angemessen, da das Modell einen universellen Charakter hat und für verschiedene Typen von Organisationseinheiten auf beliebigen

Ebenen eines Multiorganisationssystems anwendbar ist. Konkrete Organisationseinheiten ergeben sich dementsprechend durch Konkretisierung des universellen Schemas und somit durch sukzessives Absenken der Modellabstraktion.

Insgesamt wurde in diesem Kapitel das grundlegende Verständnis für die im ersten Teil der Arbeit motivierten organisationsorientierten Softwaresysteme durch die Vorstellung konkreter Modellierungstechniken und Grundmodelle geschärft. Die Fundierung aller vorgestellten Techniken auf operationaler Petrinetzsemantik sorgt explizit für die Anschlussfähigkeit softwaretechnischer Überlegungen. Im folgenden Kapitel wird dieser Aspekt noch stärker betont. Dort erfolgt der Übergang von den bisher betrachteten abstrakten, recht anschaulichen Modellen zu technischeren, schwerer zu durchdringenden Modellen. Dafür erlauben diese Modelle die Adressierung und direkte Operationalisierung elaborierter Aspekte der Funktionsweise von Organisationseinheiten.

Im Hinblick auf die in Abschnitt 7.2 formulierten Anforderungen an das Modellierungsvorhaben des zweiten Teils der Arbeit lässt sich zuerst feststellen, dass der geforderte janusgesichtige Charakter von Kollektivitäts- und Organisationseinheiten und der darauf aufbauende rekursive Charakter von Multiorganisationssystemen besonders prominent behandelt wurden. Noch konkreter ist festzuhalten, dass bei der Erarbeitung des Grundmodells einer Organisationseinheit ein besonderes Augenmerk darauf lag, die als grundlegend identifizierten drei konzeptionellen Richtungen („nach unten Schauen“, „auf die eigene Ebene Schauen“, „nach oben Schauen“) explizit im Modell zu repräsentieren. Die damit einhergehende analytische Aufteilung der Struktur und des Verhaltens einer Organisationseinheit nach Operation, Integration und Governance liefert die konzeptionelle Grundlage für die Adressierung der differenzierten Fragestellungen aus Abschnitt 7.2. Allerdings wird eine ausführliche Diskussion diesbezüglich auf das Ende des folgenden Kapitels verschoben. Die dort behandelte Operationalisierung von den in diesem Kapitel abstrakt und unter Zuhilfenahme natürlichsprachlicher Erklärungen behandelten Konzepte und Mechanismen erlaubt eine schärfere und detailliertere Diskussion.

Abschließend sei noch auf einen weiterführenden Aspekt hingewiesen. Durch die Fundierung aller betrachteten Modellarten auf einem gemeinsamen semantischen Kern ist es möglich, die Modellarten zu kombinieren und damit eine Systemmodellierung vorzunehmen, bei der punktuell Details hervorgehoben werden. Abbildung 8.21 zeigt eine Veranschaulichung.

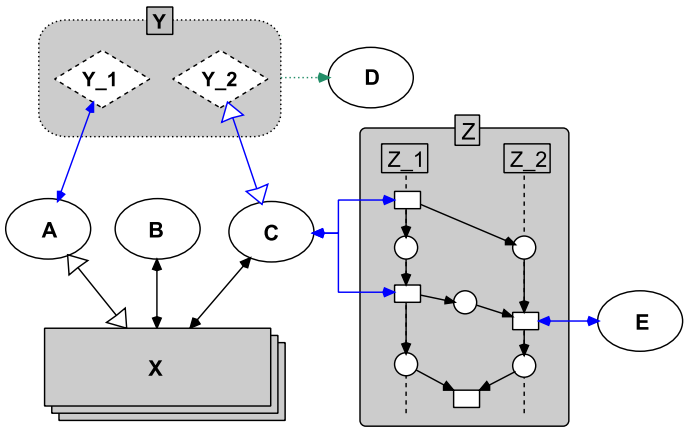

Abbildung 8.21: Exemplarische Kombination der abstrakten Modellarten

9 Operationale Modelle von Organisationseinheiten

Bereits den abstrakten Modellen des vorigen Kapitels wohnt eine operationale Semantik inne, die ein konkretes Verständnis für die Funktionsweise der Modellgegenstände sicherstellt. In diesem Kapitel steht die Nutzung dieser operationalen Semantik für die Erstellung ausführbarer Modelle jedoch prominent im Vordergrund. Die Referenznetzmodelle von Organisationseinheiten dieses Kapitels sind im Petrinetzsimulator Renew direkt ausführbar. Sie stellen vollständig funktionsfähige softwaretechnische Prototypen dar, auf Basis derer Multiorganisationssysteme programmiert werden können. Bereits die petrinetzbasierten Akteur-Aktivitätsablauf-Modelle des vorigen Kapitels können insoweit mit Anschriften angereichert werden, dass sie ausführbar werden. Allerdings wurde bereits darauf hingewiesen, dass diese Art von Modellen schnell sehr ausladend und unübersichtlich werden. Darüber hinaus wurde im vorigen Kapitel bewusst eine Abstraktion von diffizilen Aspekten wie elaborierten Systemregeln (und den daraus resultierenden Feinheiten von sozialen Beziehungen/Strukturen) und der Evolution von Organisationseinheiten vorgenommen. Für die operationalen Modelle dieses Kapitels haben diese Aspekte jedoch eine zentrale Bedeutung. Daher wird gegenüber den Petrinetzmodellen des vorigen Kapitels eine kompaktere, dynamischere und ausdrucksstärkere Herangehensweise gewählt.

Im Gegenzug wird hingenommen, dass die Modelle dieses Kapitels zumindest *im Detail* schwerer zu durchdringen sind. Ein intuitiver Zugang zu den *grundsätzlichen* Modellzusammenhängen bleibt dennoch gewahrt. Noch stärker als für die Modelle des vorigen Kapitels gilt, dass mit den operationalen Modellen dieses Kapitels ein eigenständiger Beitrag für die Petrinetzmodellierung und dabei vor allem für den Bereich der Modellierung mit Netzen in Netzen geliefert wird. Man kann hier von einer spezifischen Form der *organisationsorientierten Petrinetze* sprechen.

Es ist festzuhalten, dass die Möglichkeit der direkten Ausführung mittels eines Petrinetzsimulators nicht das alleinige Ziel der vorgestellten Modelle ist. Der Zweck der Modelle besteht nicht zuletzt darin, ein exaktes *technisches* Verständnis für die *inhaltliche* Funktionsweise einer Organisationseinheit und der Kopplungsmechanismen zwischen Organisationseinheiten zu vermitteln. Damit ergeben sich also Modelle für die Programmierung/Implementierung, die auf alternative Deployment-Umgebungen übertragen werden können.

Analog zum vorigen Kapitel beginnt auch dieses Kapitel mit der Modellierung von Kollektivitätseinheiten als Vorstufe von Organisationseinheiten in Abschnitt 9.1. Abschnitt 9.2 behandelt darauf aufbauend die Erweiterung zu Organisationseinheiten in Form des Zusammenspiels von Operation, Integration und Governance. In beiden Abschnitten wird zunächst ein erstes Grundmodell erarbeitet und anschließend auf Varianten und potentielle Erweiterungen hingewiesen. Ein wesentliches Augenmerk dieses Kapitels liegt darauf, generische Mechanismen der technischen Operationalisierung von Organisationseinheiten bereitzustellen. Dadurch ergibt sich ein großer Rahmen von möglichen Ausprägungen der hier vorgestellten Grundmodelle. Solche Ausprägungen werden im dritten Teil der Arbeit behandelt.

Ergebnisse des Kapitels wurden im Artikel [WEM09] für den *International Workshop on Organizational Modelling* (ORGMOD'2009) sowie in der erweiterten Fassung [WEMKB11] für die Zeitschrift *Transactions on Petri Nets and other Models of Concurrency* (ToPNoC) veröffentlicht (ein Vorläufer ist der Artikel [WEM08a] für den Workshop *Anwendungen und Werkzeuge für Petrinetze*, AWPN'2008).

9.1 Kollektivitätseinheiten und rekursive Akteurssysteme

Hier erfolgt zunächst die Betrachtung von Kollektivitätseinheiten als Vorstufe von Organisationseinheiten. Die analytischen Aspekte der Operation, Integration und Governance werden dabei ausgeblendet. Stattdessen richtet sich der Fokus auf die drei grundlegenden Konzepte der Akteure, Aktivitäten und Aktivitätsbeziehungen. Das Konzept des kollektiven Akteurswesens spielt dabei von Beginn an die zentrale Rolle. Am Ende des Abschnittes steht somit ein operationales Modell rekursiver Akteursysteme. Für die Referenznetzmodelle dieses Abschnittes wird auf die Abstraktionsstufen I und II zurückgegriffen.

9.1.1 Grundlegende Mechanismen

In Abschnitt 8.2 wurde die Herangehensweise gewählt, Aktivitäten unmittelbar auf Ebene einer Kollektivitätseinheit zu modellieren. Für eine abstrakte Darstellung ist dies sehr anschaulich, für eine Operationalisierung ergeben sich jedoch Nachteile. Zum einen ist eine solche Darstellung sehr unhandlich, wenn eine größere Anzahl von Aktivitäten möglich ist. Zum anderen ist es nicht möglich, das dynamische Auftreten neuartiger Typen von Aktivitäten zu modellieren. Dementsprechend wird an dieser Stelle eine andere Herangehensweise gewählt, die sich durch den Referenznetzformalismus unmittelbar anbietet. Aktivitäten werden ebenso wie Akteure als Objektnetze auf Ebene der Kollektivitätseinheit eingebettet. Die Ausführung von Aktionen einer Aktivität durch Akteure erfolgt nun mittels generischer Synchronisationsmechanismen. Abbildung 9.1 zeigt die grundlegende Idee für die Operationalisierung von Kollektivitätseinheiten als Referenznetzmodell der Abstraktionsstufe II. Dementsprechend sind noch einige Details der Operationalisierung ausgeblendet. Diese werden im folgenden Teilabschnitt behandelt. Unter Rückbezug auf die Erläuterungen aus Unterabschnitt 8.1.1 wird deutlich, dass von den drei Arten von Basisoperationen hier lediglich der Fall der Verwendung/Modifikation von Akteuren behandelt ist. Das Hinzufügen und das Entfernen von Akteuren werden in Teilabschnitt 9.1.4 gesondert behandelt.

Die Synchronisation von Akteuren und Aktivitäten für die Ausführung von Aktionen erfolgt über generische synchrone Kanäle. In Übereinstimmung mit den abstrakten Modellen geschachtelter Kollektivitätseinheitssysteme im vorigen Kapitel (speziell Teilabschnitt 8.1.2) erfolgt eine explizite Unterscheidung von internen und peripheren Aktionen. Entsprechend ergibt sich die Unterscheidung von internen Aktivitäten, die nur interne Aktionen umfassen, und Peripherieaktivitäten, die mindestens eine Peripherieaktion umfassen. Für alle Aktionstypen gilt, dass Akteure und Aktivitäten über Testkanten (vergleiche Teilabschnitt 6.1.4 aus Kapitel 6) eingebunden werden. Die Intention dahinter ist, dass Akteure (insbesondere kollektive Akteure) mehrere Aktionen nebenläufig ausführen können und auf Seiten einer Aktivität mehrere Aktivitätsschritte nebenläufig vollzogen werden können.

Im Falle einer internen Aktion findet auf Ebene der Kollektivitätseinheit eine Synchronisation eines Akteurs und einer Aktivität statt. Diese Synchronisation wird mittels der Kombination von zwei synchronen Kanälen realisiert. Der Akteur wird über den Kanal :act() und die

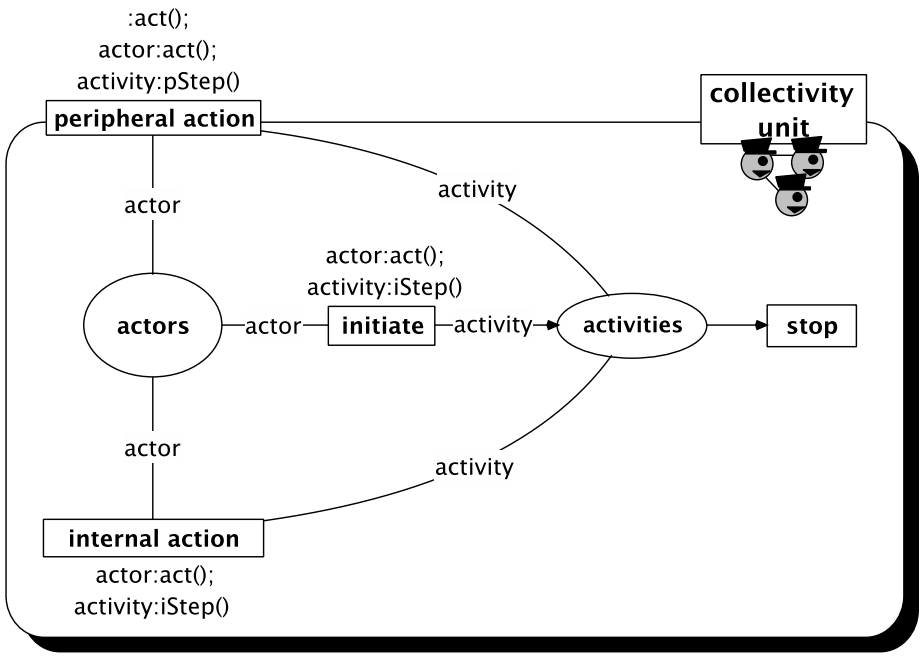

Abbildung 9.1: Operationales Grundmodell einer Kollektivitätseinheit

Aktivität über den Kanal :iStep() eingebunden. Die Bedeutung ist, dass der Akteur eine Aktion ausführt, um einen internen Aktivitätsschritt zu vollziehen. Peripherieaktionen auf der anderen Seite erlauben einer Kollektivitätseinheit, als kollektiver Akteur im Kontext umfassender Kollektivitätseinheiten zu agieren. Aus der Sicht einer fokussierten Kollektivitätseinheit wird eine Peripherieaktion ebenso wie eine interne Aktion von einem eingebetteten Akteur ausgeführt. Dementsprechend werden wieder ein Akteur und eine Aktivität synchronisiert. Der Akteur wird wie zuvor über einen :act()-Kanal eingebunden. Die Aktivität wird nun über einen :pStep() Kanal für einen peripheren Aktivitätsschritt eingebunden. Auf der nächst höheren Systemebene jedoch wird die Aktion als von der Kollektivitätseinheit selber („als Ganzem“) ausgeführt wahrgenommen. Demzufolge werden für eine Peripherieaktion nicht nur ein Akteur und eine Aktivität synchronisiert. Zusätzlich wird die Kollektivitätseinheit selber über einen :act()-Kanal auf der nächst höheren Kollektivitätseinheitsebene zum Zweck der Aktionsausführung mit einer Aktivität synchronisiert. Auf diese Weise kann die Ausführung einer Aktion eine Synchronisation über multiple Systemebenen hinweg mit sich bringen, so dass die Aktion auf mehreren Systemebenen auftritt und jeweils spezifische Effekte haben kann.

Die Initiierung von Aktivitäten folgt demselben Prinzip. Ein Akteur führt eine spezielle Initiierungsaktion aus, die gleichzeitig den ersten Schritt der neuen Aktivität darstellt. Hat die Aktivität einen Zustand erreicht, in dem eine Terminierung möglich ist, so kann dies ohne Akteursbeteiligung stattfinden.

Aufbauend auf den beschriebenen Mechanismen für interne und periphere Aktionen modelliert Abbildung 9.2 exemplarisch die Ausführung einer Aktion, die Effekte auf mehreren Systemebenen hat. Die Abbildung zeigt die entsprechenden Variablenbindungen für das Auftreten dieser Aktion. Dabei sind zwei Kollektivitätseinheiten mit ihren jeweiligen Aktivitäten

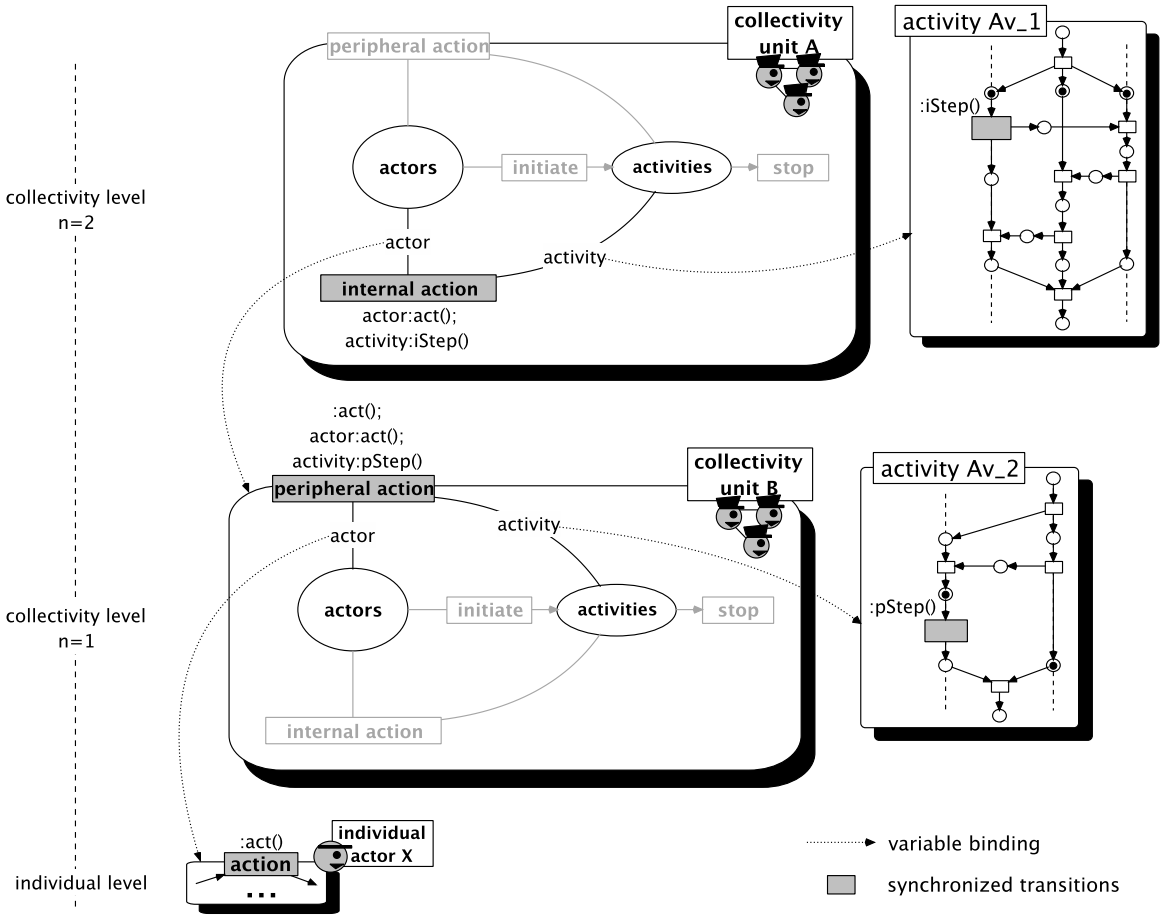

Abbildung 9.2: Ausführung einer Aktion mit Effekten auf mehreren Systemebenen

beteiligt. Für die Modellierung von Aktivitäten gelten weiterhin die gleichen Bestimmungen wie im vorigen Kapitel (Teilabschnitt 8.1.1). Es wird keine bestimmte Form der Modellierung vorgeschrieben, obwohl beispielsweise Modelle basierend auf Workflownetzen in vielen Fällen eine gute Option darstellen. Stattdessen wird weiterhin allgemein angenommen, der Kontrollfluss einer Aktivität sei durch die gesammelten *Lebenslinien* der *Aktivitätsrollen* (vertikal verbundene Transitionen) repräsentiert und der Datenfluss sei durch *Nachrichtenversand* zwischen Rollen (horizontal verbundene Transitionen) repräsentiert.

Das Auftreten der Aktion aus Abbildung 9.2 stellt sich für die Betroffenen unterschiedlich dar. Aus der Perspektive der Kollektivitätseinheit auf der zweiten Kollektivitätsebene handelt es sich um eine interne Aktion, gekennzeichnet durch einen :iStep()-Kanal der beteiligten Aktivität. Die Aktion wird durch einen eingebetteten Akteur ausgeführt. Dieser Akteur ist allerdings wiederum eine Kollektivitätseinheit, in diesem Fall eine Kollektivitätseinheit der ersten Kollektivitätsebene. Aus Perspektive dieser Kollektivitätseinheit handelt es sich bei der Aktion um eine Peripherieaktion, gekennzeichnet durch einen :pStep()-Kanal der beteiligten Aktivität. Auch in diesem Fall wird die Aktion durch einen eingebetteten Akteur ausgeführt. Gleichzeitig aber wird die Aktion als eine Aktion der Kollektivitätseinheit „als Ganzem“ auf der nächst

höheren Systemebene „ausgegeben“. Der eingebettete Akteur handelt „im Namen/Auftrag“ der gesamten Kollektivitätseinheit.

An dieser Stelle gelten die gleichen Feststellungen, die bereits in Teilabschnitt 8.1.2 im Zusammenhang mit den technischen Details der Einbettung von Kollektivitätseinheiten als Akteure getroffen wurden. Die hier modellierten Mechanismen sorgen für ein technisches Verständnis für den Übergang von individuellem zu kollektivem Akteurswesen. Vor allem bleibt die soziologische Korrektheit des Konzeptes eines kollektiven Akteurs durch die Konstitutionsbeziehung zu seinen eingebetteten Akteuren gewahrt. Die Eignung der Modellierungstechnik der Referenznetze tritt deutlich zu Tage.

Der Übergang von individuellem zu kollektivem Akteurswesen stellt sich immer aus der Perspektive einer bestimmten Kollektivitätseinheit dar. Ein Akteur, der aus Sicht einer umschließenden Kollektivitätseinheit als holistische Einheit erscheint, kann jederzeit selber eine Kollektivitätseinheit und damit ein kollektiver Akteur sein. Der in Abbildung 9.2 modellierte Mechanismus erlaubt beliebige Rekursionstiefe. Allerdings erlaubt er zugleich die Modellierung und Entwicklung jeder Kollektivitätseinheit für sich gesehen in ihrem lokalen Kontext. Bereits für die abstrakten Modelle des vorigen Kapitels galt ein Augenmerk der Berücksichtigung lokaler Sichten in einem Multiorganisationssystem. Diese Herangehensweise wird auch für das operationale Modell dieses Abschnittes gewahrt und spiegelt sich in den Abbildungen 9.1 und 9.2 durch die Verwendung des :act()-Kanals als Akteursschnittstelle für sowohl interne als auch periphere Aktionen wieder. Für Akteure spielt es also zumindest auf *technischer* Ebene keine Rolle, ob ihre Aktionen lediglich Effekte auf unmittelbar umschließende Kollektivitätseinheiten oder aber noch darüber hinaus haben. Auf der Gegenseite spielt es für eine Kollektivitätseinheit auf technischer Ebene keine Rolle, ob ihre eingebetteten Akteure individueller oder kollektiver Natur sind. Es kann in beiden Fällen auf *inhaltlicher* Ebene notwendig sein, dass solche Informationen vorliegen und kommuniziert werden. Auf der technischen Ebene hingegen zielen die Mechanismen des hier vorgestellten operationalen Modells darauf ab, Kollektivitäts-/Organisationseinheiten lediglich in ihrem unmittelbaren Kontext betrachten zu müssen. Die in den beiden folgenden Teilabschnitten vorgestellten technischen Details manifestieren diese Herangehensweise.

Wie bereits in Teilabschnitt 8.1.2 erläutert, muss es trotz der beliebigen Rekursionstiefe einen unteren Abschluss geben. An irgendeinem Punkt hat eine Aktion, die auf verschiedenen Kollektivitätsebenen Effekte hat, tatsächlich ihren Ursprung. Dementsprechend ist in der Abbildung auch ein individueller Akteur („klassischer Agent“) angedeutet. Bezüglich solcher individuellen Akteure werden ebenso wenig wie in den Ausführungen des vorigen Kapitels konkrete inhaltliche Annahmen getroffen. Auf der technischen Seite wird angenommen, sie würden mittels :act()-Kanälen eine Einbettung erlauben. Der Übergang von Kollektivitäts- und später Organisationseinheiten als kollektiven Akteuren zu individuellen Akteuren wird wie in Kapitel 7 beschrieben als Paradigmenwechsel interpretiert. Die Agentenorientierung bietet einen reichhaltigen Fundus an Ansätzen für die Einbettung von Agenten in organisatorische Kontexte.

9.1.2 Technische Details

Der vorige Teilabschnitt behandelt das grundsätzliche Konzept einer technischen Realisierung rekursiver Akteursysteme. Dabei wurden bislang allerdings viele Details der Operationalisierung ausgeblendet. Um ein exakteres Verständnis zu vermitteln, werden die Details in diesem Teilabschnitt nachgeholt. Der darauf folgende Teilabschnitt enthält als Ergänzung einige il-

lustrierende Beispiele. Diese Beispiele zeigen auf, welch flexible Dynamik sich aus den hier präsentierten Modellen in Kombination mit dem Unifikationsmechanismus des Referenznetzformalismus für die Synchronisation von Akteuren und Aktivitäten ergeben. In beiden Teilabschnitten werden Referenznetzmodelle (beziehungsweise Fragmente) auf der Abstraktionsstufe I präsentiert. Dementsprechend sind sie direkt ausführbar und stellen softwaretechnische Prototypen dar. Dieser Detaillierungsgrad wird anschließend im Rest der Arbeit weitestgehend vermieden. Insofern stellen die folgenden Ausführungen das technische Fundament für die weiteren Modelle der Arbeit bereit.

9.1.2.1 Aktionen auf Ebene der Kollektivitätseinheiten

Das vollständige ausführbare Grundmodell einer Kollektivitätseinheit ist in Abbildung A.1 im Anhang der Arbeit zu finden. Hier werden die verschiedenen Aspekte punktuell betrachtet. Zunächst zeigt Abbildung 9.3 eine Verfeinerung der für interne Aktionen relevanten Anschriften aus Abbildung 9.1. Jeder Akteur und jede Aktivität ist mit einem Identifikator versehen.

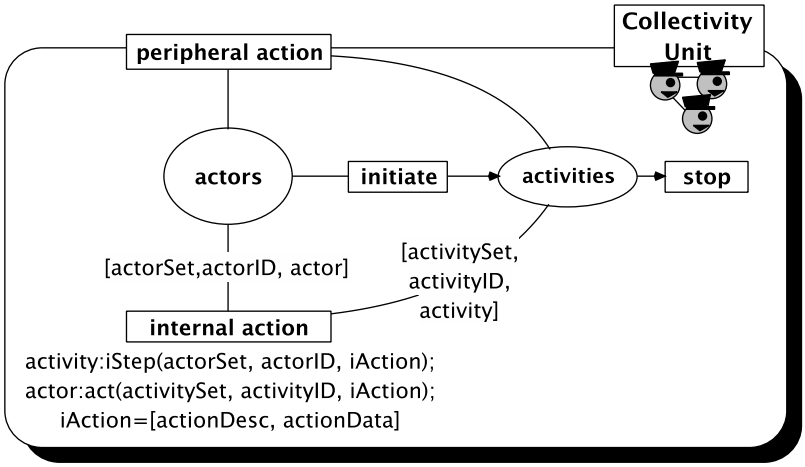

Abbildung 9.3: Operationale Details für interne Aktionen

Diese Identifikatoren sind jeweils eindeutig. Entsprechend der Ausführungen des vorigen Kapitels sind Akteure und Aktivitäten darüber hinaus in bestimmten Akteurs- beziehungsweise Aktivitätsmengen enthalten. Akteurs- und Aktivitätsmarken sind also eigentlich Tupel bestehend aus Mengenbezeichner, Identifikator und der Referenz auf das Akteurs- beziehungsweise Aktivitätsnetz. Gehört ein Akteur zu mehreren Akteursmengen, so ist dieser Akteur in Form mehrerer Marken, die aber alle auf die gleiche Instanz eines Akteursnetzes verweisen, im Kollektivitätseinheitsnetz eingebettet. Dies gilt analog für Aktivitäten.

Üblicherweise sind mehrere Akteure und Aktivitäten im Kontext einer Kollektivitätseinheit aktiv. Daher ist Sorge zu tragen, dass die richtigen Akteure und Aktivitäten für eine Synchronisation zusammen finden. Für die Ausführung einer internen Aktion werden ein Akteur über einen :act()-Kanal und eine Aktivität über einen iStep()-Kanal eingebunden. Dabei sind Mengenbezeichner und Identifikatoren für einen Akteur und eine Aktivität als Kanalparameter eingebunden und müssen mit den tatsächlichen Mengenbezeichnern und Identifikatoren des beteiligten Akteurs und der beteiligten Aktivität unifizieren. Die Mengenbezeichner spie-

len dabei in der Regel nur für die erstmalige Einbindung des Akteurs in die Aktivität eine Rolle (dies entspricht der Berechtigung zur Besetzung von bestimmten Aktivitätsrollen in Aktivitäten einer bestimmten Aktivitätsmenge durch Akteure einer bestimmten Akteursmenge, vergleiche Teilabschnitt 8.1.1 des vorigen Kapitels). Die Aktion selber ist eine Datenstruktur (ein Tupel) bestehend aus einer Aktionsbeschreibung und den Aktionsdaten. Die Aktion ist Parameter beider Kanäle und muss entsprechend unifiziert werden.

Für den Fall einer peripheren Aktion illustriert Abbildung 9.4 einen komplizierteren Sachverhalt. Die Aktion wird einerseits wieder durch einen eingebetteten Akteur im Kontext ei-

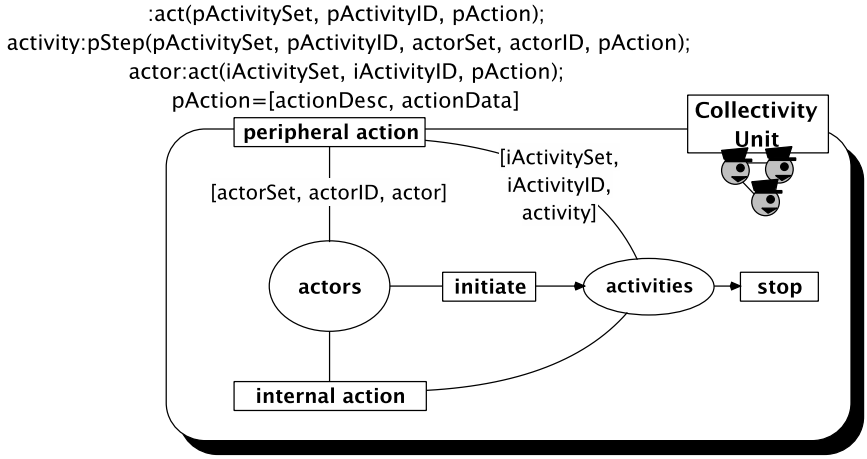

Abbildung 9.4: Operationale Details für periphere Aktionen

ner Aktivität ausgeführt. Andererseits bedeutet das Ausführen dieser Aktion gleichzeitig das Ausführen einer Aktion der Kollektivitätseinheit als Ganzem im Kontext einer Aktivität einer umschließenden Kollektivitätseinheit. Daher muss die entsprechende Aktivität auf der nächst höheren Kollektivitätsebene (mittels Aktivitätsmenge und -identifikator) in die unifizierende Bindungssuche eingebunden werden. Diese Aktivitätsverzahnung wurde bereits mehrfach konzeptionell behandelt. Unter der technischen Betrachtung in Abbildung 9.4 bedeutet dies dementsprechend, dass sowohl eine Aktivität auf Ebene der betrachteten Kollektivitätseinheit (iActivitySet, iActivityID) als auch eine Aktivität auf Ebene einer umschließenden Kollektivitätseinheit (pActivitySet, pActivityID) für die Ausführung einer peripheren Aktion relevant sind. Gegenüber dem :iStep()-Kanal hat der :pStep()-Kanal somit zwei zusätzliche Parameter mehr für die höherrangige Aktivität. Beim :act()-Kanal des eingebetteten Akteur ändert sich nichts, so dass hier auf technischer Ebene nicht unterschieden wird, ob eine interne oder eine periphere Aktion ausgeführt wird.

Schließlich fokussiert Abbildung 9.5 auf die Aktivierung von Aktivitäten. Eine solche Aktivierung erfolgt mittels einer internen Aktion eines Akteurs. Die Aktionsbeschreibung ist in diesem Fall auf "initiate" festgelegt. Im Gegensatz zu den beiden bisherigen Fällen der Akteurs-Aktivitäts-Synchronisation ist in diesem Fall zunächst noch keine Aktivität vorhanden. Stattdessen wird sie erst im Zuge des Schaltvorgangs durch den Unteraufruf this:newActivity() erzeugt. Dabei wird aus einer Aktivitätsbeschreibung, die Teil der Aktionsdaten ist, ein Aktivitätsnetz instanziiert. Details bezüglich des Unteraufrufs werden hier nicht behandelt (im einfachsten Fall verweist die Aktivitätsbeschreibung schlicht auf eine bestimmte existierende

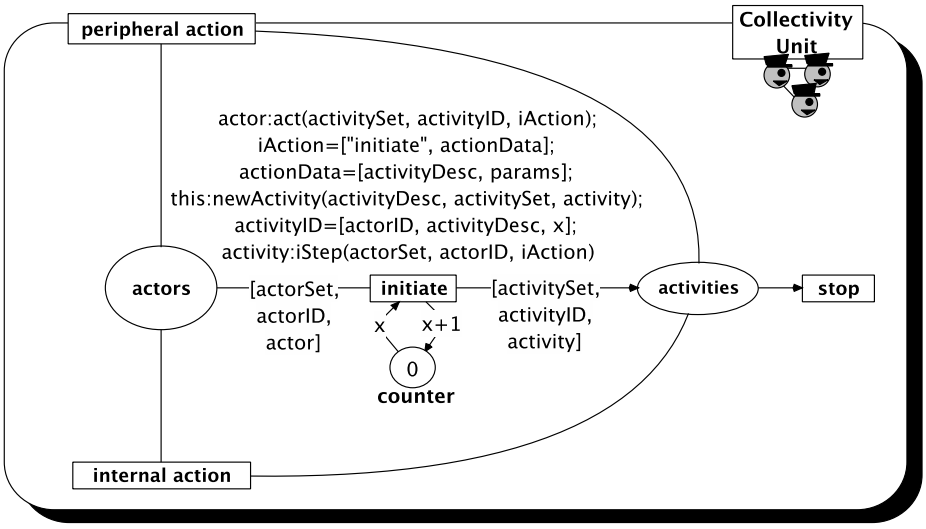

Abbildung 9.5: Operationale Details der Aktivitätsaktivierung

Netzschablone). Erst nach Instanziierung des Aktivitätsnetzes kann es über den :iStep()-Kanal angesprochen werden.[1] Der Identifikator der neuen Aktivität ergibt sich aus dem Identifikator des Akteurs in Kombination mit einem für die Kollektivitätseinheit globalen Zähler. Unter der Annahme eindeutiger Akteursidentifikatoren (vergleiche hierzu Teilabschnitt 9.1.4) ergeben sich somit auch eindeutige Aktivitätsidentifikatoren.

9.1.2.2 Aktionen auf Ebene der Aktivitäten

Das Kollektivitätseinheitsnetz stellt einen universell verwendbaren Rahmen für konkrete Kollektivitätseinheiten bereit. Die Konkretisierung erfolgt dabei in Form der eingebetteten Akteure und möglichen Aktivitäten. Im Falle der Akteure setzt sich eine solche Konkretisierung gegebenenfalls rekursiv über mehrere Kollektivitätsebenen hinweg bis hin zu den individuellen Akteuren fort. Für die Spezifikation von Aktivitäten sowie von individuellen Akteuren werden im Folgenden universelle Netzkomponenten vorgestellt.

Abbildung 9.6 behandelt den Fall der Aktivitäten und zeigt generische Netzkomponenten für die Aktivitätskomposition. Die Teile (a) und (b) der Abbildung zeigen die Komponenten für interne und periphere Aktionen, die über die Kanäle :iStep() beziehungsweise :pStep() in Synchronisationen auf Ebene des Kollektivitätseinheitsnetzes eingebunden werden. Die Komponente für periphere Aktionen ist dabei zu illustrativen Zwecken mit einem nach oben gerichteten Dreieck versehen. Die Teile (c) und (d) zeigen die Komponenten für die Aktionsinitiierung und -beendigung. Die Initiierung erfolgt wiederum mittels einer internen Aktion, bei der die Aktions-Beschreibung mit "initiate" fest vorgegeben ist. Die Beendigung erfolgt ohne weitere Akteursbeteiligung, sobald die Bedingungen dafür erfüllt sind.

[1] Tatsächlich ist die Erzeugung des Aktivitätsnetzes und der Einbezug des :iStep()-Kanals bereits für die unifizierende Bindungssuche relevant. Ist keine Unifikation für eine aktivierende Bindung möglich, so findet der Schaltvorgang nicht statt und die erzeugte Netzinstanz wird verworfen.

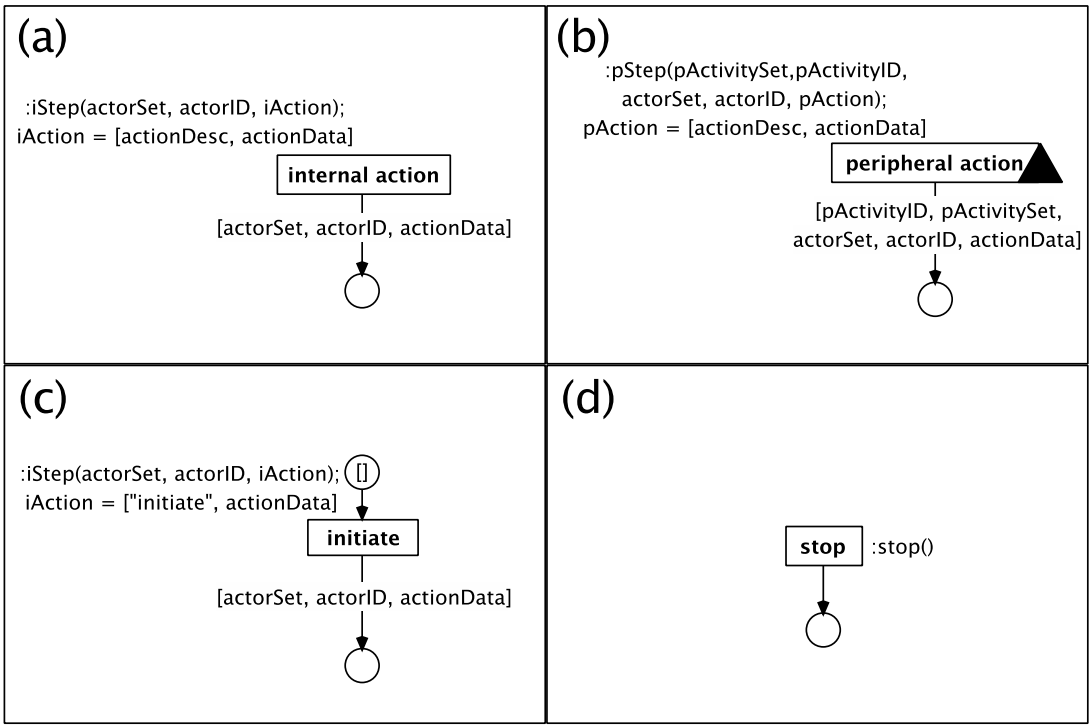

Abbildung 9.6: Netzkomponenten für die Aktivitätskomposition

9.1.2.3 Aktionen auf Ebene individueller Akteure

Die Einbettung von Kollektivitätseinheiten als kollektive Akteure in umschließende Kollektivitätseinheiten wurde bereits behandelt und die technischen Details sind gemeinschaftlich in den vorherigen Abbildungen 9.3 – 9.6 modelliert. Für individuelle Akteure gilt weiterhin, dass von ihren Interna abstrahiert wird. Allerdings zeigt Abbildung 9.7 zumindest eine Netzkomponente, die auf Seiten individueller Akteure für die Aktionsausführung im Kontext umschließender Kollektivitätseinheiten verwendet werden kann. Diese Komponente wird

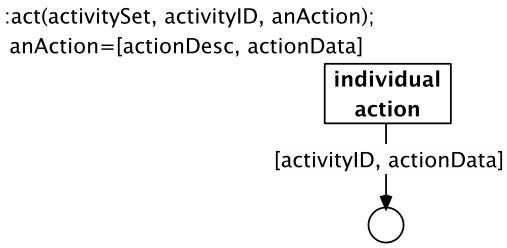

Abbildung 9.7: Netzkomponente für Aktionen individueller Akteure

unabhängig davon verwendet, ob die Aktion auf Ebene der einbettenden Kollektivitätseinheit eine interne oder periphere Aktion ist.

Die hier vorgestellten operationalen Details der Realisierung rekursiver Akteurssysteme unterstreichen die bereits im vorigen Teilabschnitt getroffene Feststellung, dass es für Akteure auf *technischer* Ebene keine Rolle spielt, ob ihre Aktionen auf der Ebene umschließender Kollektivitätseinheiten internen oder peripheren Aktionen entsprechen. Ebenso wenig ist auf Ebene einer Kollektivitätseinheit *technisch* relevant, ob eingebettete Akteure individueller oder kollektiver Natur sind. Die Akteurs-Schnittstelle ist in allen Fällen der :act()-Kanal mit der gleichen Signatur. Die Aktivitätsparameter des Kanals (Aktivitätsmenge, Aktivitätsindikator) beziehen sich immer auf die Aktivität, im Kontext derer der jeweilige Akteur die jeweilige Aktion unmittelbar ausführt. Die Unterscheidung zwischen internen und peripheren Aktionen wird im Kontext der Aktivitätsspezifikationen getroffen. Hier werden explizit die beiden Kanäle :iStep(), :pStep() mit unterschiedlichen Signaturen voneinander unterschieden.

Auf der technischen Ebene werden auf diese Weise eine modulare Modellierung und Entwicklung von Systemen von Systemen ermöglicht. Wie bereits angedeutet kann es auf *inhaltlicher* Ebene notwendig sein, dass Akteure elaboriertere Kenntnisse über die von ihnen ausgeführten Aktionen und deren Effekte besitzen. Auch für Kollektivitätseinheiten kann es *inhaltlich* relevant sein, welche ihrer eingebetteten Akteure individuell und welche kollektiv sind. In diesen Fällen müssen die entsprechenden Informationen auch auf inhaltlicher Ebene kommuniziert werden, beispielsweise durch Zugang zu Aktionsbeschreibungen und Aktivitätsspezifikationen oder insbesondere durch Daten- und damit Wissensaustausch zwischen Akteuren.

9.1.3 Synchronisationsdynamik anhand von Beispielen

Die in diesem Teilabschnitt behandelten Beispiele dienen zur Illustration der Universalität und der Mächtigkeit, die dem bisher vorgestellten operationalen Modell innewohnen. Für die Beispiele dieses Unterabschnittes erfolgt ein Rückgriff auf die in Kapitel 8 wiederholt aufgegriffene Aktivitätsspezifikation transaction. Abbildung 9.8 zeigt die Ausführung einer konkreten transaction-Aktivität im Kontext des operationalen Modells, wobei zunächst individuelle Akteure die entsprechenden Aktivitätsrollen sender und receiver einnehmen. Die Abbildung beinhaltet nahezu alle technischen Details bezüglich des Kollektivitätseinheitsnetzes, des Aktivitätsnetzes und der relevanten Netzfragmente der individuellen Akteure. Die chronologisch angeordneten Zeitpunkte der Aktivitätsausführung sind explizit gekennzeichnet. Im Folgenden wird für jeden dieser Zeitpunkte die Dynamik der Akteur-Aktivitäts-Synchronisation betrachtet, die sich aus der Unifikation zur Bindungssuche im Kontext des Referenznetzformalismus ergibt.

- **Zeitpunkt t_1:** Zu diesem Zeitpunkt initiiert der individuelle Akteur X eine transaction-Aktivität. Dies geschieht aufgrund des von X gewählten Schlüsselwortes "initiate" als Aktionsbeschreibung. Auf Ebene der Kollektivitätseinheit A wird ein neuer Aktivitätsidentifikator in Abhängigkeit vom Akteursidentifikator generiert, eine neue Instanz des Aktivitätsnetzes erzeugt und der Aktivitätsmenge "transaction" zugeteilt (siehe dazu die vorherige Abbildung 9.5, die technischen Details der Aktivitätsinitiierung sind hier ausgeblendet). Dem frisch instanziierten Aktivitätsnetz wird der Identifikator und der Akteursmengenbezeichner von X übermittelt. Hier ist festgelegt, dass die Initiierungsaktion überhaupt nur ausgeführt werden kann, wenn der initiierende Akteur der Akteursmenge "sellers" angehört (was hier der Fall ist). Im Gegenzug wird der Aktivitätsidentifikator an X für die spätere Verwendung vermittelt. Die Initiierungsaktion selber kommt in diesem Fall ohne weitere Aktionsdaten aus. Es wäre allerdings möglich, zusätzliche Startparameter einzubeziehen.

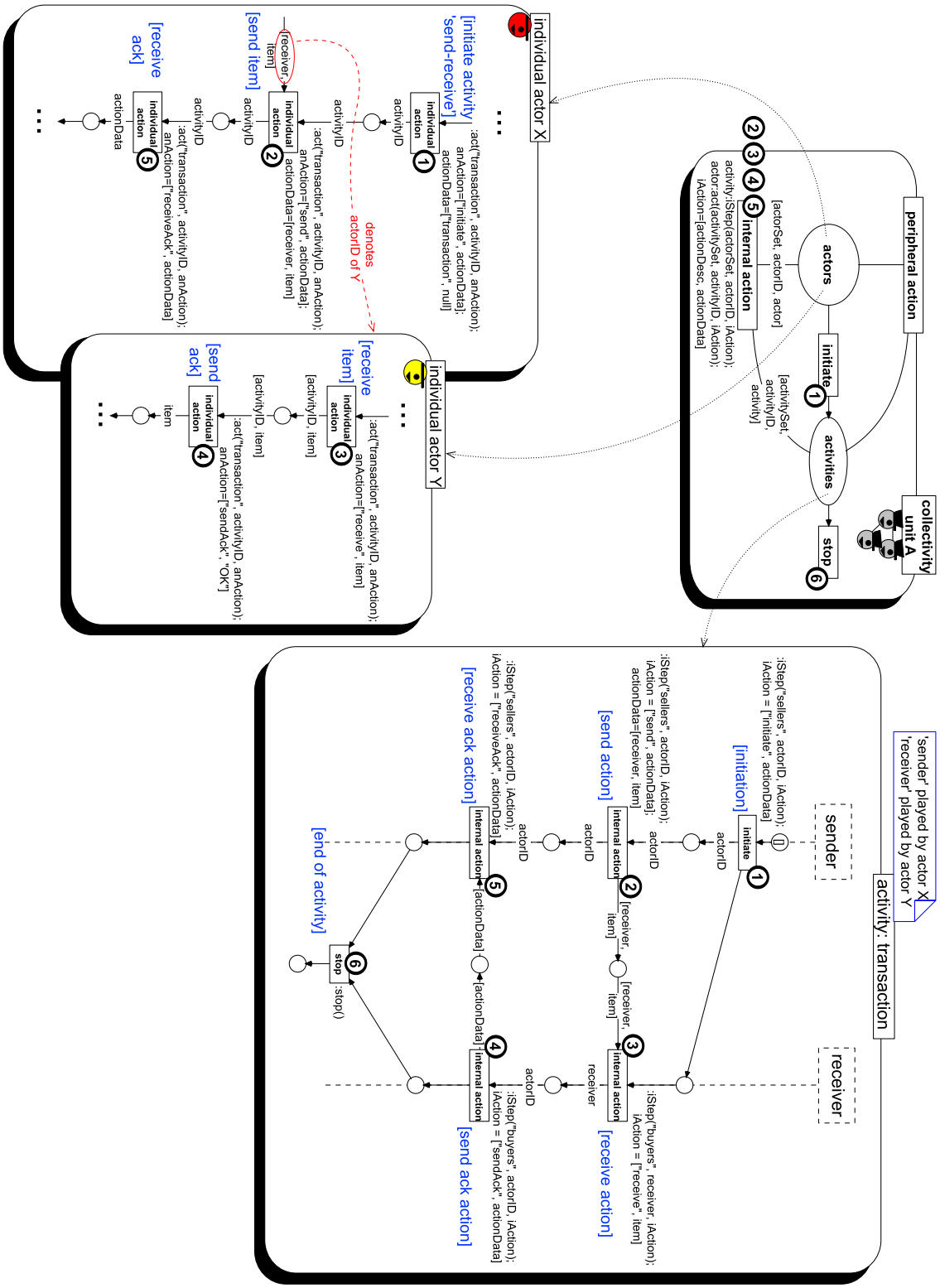

Abbildung 9.8: Beispielaktivität: send-receive

Die auf Unifikation basierende Bindungssuche erlaubt die Auflösung all der beschriebenen Abhängigkeiten und den Vollzug aller Datentransfers in einem *einzigen* Schaltschritt.

- **Zeitpunkt t_2:** Den soeben erhaltenen Aktivitätsidentifikator kann X nun verwenden, um die nächste Aktion des Versendens eines Gegenstandes gezielt im Kontext der neuen transaction-Aktivität auszuführen. Auf der anderen Seite ist für die Aktivität nun festgelegt, dass nur X berechtigt ist, den Aktivitätsschritt des Sendens auszuführen. In diesem Fall ist also von beiden Seiten aus wechselseitig eindeutig bestimmt, welcher Akteur und welche Aktivität für die Versandaktion synchronisiert werden. Die Aktions-Beschreibung als "send" ist ebenfalls von beiden Seiten aus eindeutig festgelegt. Die Aktionsdaten in Form des Empfängers (Identifikator) und des versandten Gegenstandes werden in diesem Fall hingegen vollständig von X bereitgestellt und der Aktivität übermittelt.

- **Zeitpunkt t_3:** Der Akteur, der den Gegenstand empfangen soll, ist bis zu diesem Zeitpunkt noch überhaupt nicht in die Aktivität eingebunden. Diese Einbindung erfolgt nun dynamisch während der Laufzeit der Aktivität. Der Identifikator des empfangenden Akteurs ist aufgrund der vorigen Aktion bereits bekannt. In diesem Fall handelt es sich um Akteur Y. Die Empfangsaktion kann allerdings nur stattfinden, wenn bei Y auch eine entsprechende Aktionstransition aktiviert ist. Wie in der Abbildung zu sehen, ist die Aktionbeschreibung auf beiden Seiten auf "receive" festgelegt. Auf Seiten der Aktivität ist neben dem konkreten Identifikator des empfangenden Akteurs festgelegt, dass der Akteur der Menge "buyers" angehören muss. Auf Seiten von Y ist festgelegt, dass die Empfangsaktion im Kontext einer Aktivität der Menge "transaction" ausgeführt werden soll. Ein konkreter Aktivitätsidentifikator ist auf Seiten von Y noch nicht bekannt. Stattdessen handelt es sich also um eine Aktionstransition, die auf mögliche Empfangsereignisse „wartet“ und erst beim Empfang den entsprechenden Aktivitätsidentifikator zur weiteren Verwendung erhält. Dieser Aktivitätsidentifikator wird bei Ausführung der Empfangsaktion von der Kollektivitätseinheit bereitgestellt. Die Aktionsdaten in Form des empfangenen Gegenstandes werden von der Aktivität bereitgestellt.

  Weitere Möglichkeiten der dynamischen Einbindung eines Akteurs in eine Aktivität sind denkbar. Beispielsweise könnte die Adresse des empfangenden Akteurs vor der Ausführung der Aktion noch unbekannt sein. In diesem Fall könnte jeder Akteur als Empfänger eingebunden werden, bei dem eine aktivierte Aktionstransition mit der passenden Aktionsbeschreibung (hier "receive") vorliegt und bei dem die Akteurs- beziehungsweise Aktivitätsmengen kompatibel sind. Zusätzlich könnten auch die Aktionsdaten (hier der erhaltene Gegenstand) als Grundlage einer weiteren Filterung herhalten. Generell erlaubt die unifizierende Bindungssuche des Referenznetzformalismus eine hochdynamische Handhabung der Synchronisation von Akteuren und Aktivitäten.

- **Zeitpunkte t_4 und t_5:** In beiden Fällen besteht eine wechselseitig eindeutige Bezeichnung, welcher Akteur beziehungsweise welche Aktivität in die Aktion eingebunden werden. Im Falle der Aktion zum Versand der Bestätigungsnachricht stellt der Akteur Y die Aktionsdaten zur Verfügung und diese werden im Falle des Empfanges der Bestätigung vom Aktivitätsnetz für Akteur X bereitgestellt.

- **Zeitpunkt t_6:** Nachdem alle Aktivitätsschritte ausgeführt sind, kann die Aktivität beendet werden. Für diese „Aufräumarbeit“ ist keine Akteursbeteiligung notwendig.

Das soeben behandelte Beispiel stellte den Fall dar, dass alle Akteure einer Kollektivitätseinheit als individuelle Akteure modelliert wurden und die Systemeinheit selber in keine weiteren Kollektivitätseinheiten einbettet ist. Somit liegt im Gesamtsystem lediglich eine einzige Ebene der Kollektivität vor. Die wesentliche Stärke der hier vorgestellten Modelle ist jedoch gerade die Möglichkeit einer beliebig tiefen Schachtelung von Akteuren bei gleichzeitiger Wahrung der Möglichkeit, jede Kollektivitätseinheit für sich in ihrem lokalen Kontext entwickeln zu können. Abbildung 9.9 erweitert das Beispiel dementsprechend um den Umstand, dass der Akteur, der die Aktivitätsrolle receiver einnimmt, selber eine Kollektivitätseinheit ist. Die Abbildung folgt dabei Abbildung 8.4 aus Kapitel 8. Auf Ebene der Systemeinheit B wird eine Aktivität der Menge reception verwendet, um an einer Aktivität der Menge transaction auf der nächst höheren Systemebene teilzunehmen und dabei die Aktivitätsrolle receiver einzunehmen. Die reception-Aktivität umfasst dabei ihrerseits die beiden Aktivitätsrollen distributor und inspector. Diese werden von in Kollektivitätseinheit B eingebetteten individuellen Akteuren eingenommen.

Die Abbildung fokussiert auf die im neuen Beispiel auftretenden peripheren Aktionen (aus Sicht von Kollektivitätseinheit B) und blendet die weiteren technischen Details aus.

- **Zeitpunkt t_1:** Für die Kollektivitätseinheit A ändert sich nichts im Vergleich zum vorherigen Beispiel. Aus ihrer Sicht wird die Empfangsaktion weiterhin schlicht von einem eingebetteten Akteur ausgeführt. Die *Bereitschaft* dieses Akteurs, die Aktion auszuführen, gestaltet sich nun allerdings komplizierter. Im vorigen Beispiel handelte es sich um einen individuellen Akteur, für den lediglich die entsprechende Aktionstransition aktiviert sein musste (Zeitpunkt t_3). Für den kollektiven Akteur des aktuellen Beispiels müssen dagegen zwei Voraussetzungen erfüllt sein:

  1. Um überhaupt in der Lage zu sein, die Empfangsaktion der transaction-Aktivität auf Ebene der Kollektivitätseinheit A auszuführen, muss auf Ebene von B bereits eine passende Peripherieaktivität reception aktiv sein. Für den Moment wird schlicht angenommen, auf Ebene von B gebe es Akteure, die reception-Aktivitäten initiieren, selber die distributor-Rolle einnehmen und auf Empfangsereignisse „warten“. Im Kontext der Aktivität muss dementsprechend eine Transition für die Peripherieaktion des Empfangs aktiviert sein.
  2. Gleichzeitig muss auch der Akteur der distributor-Rolle selber wiederum bereit sein, die Empfangsaktion *im Namen der Kollektivitätseinheit B* auszuführen. In diesem Beispiel handelt es sich hierbei um den individuellen Akteur Y und somit ist der Fall analog zu der Einbindung der individuellen Akteure des vorigen Beispiels. Auf *technischer* Ebene ist für diesen Akteur nicht relevant, dass seine Aktion Effekte auf weiteren Systemebenen als der von Kollektivitätseinheit B hat. Auf *inhaltlicher* Ebene liegt es allerdings nahe, dass dieser Umstand sehr wohl bekannt ist. Schließlich ist es dieser Akteur Y, der die Peripherieaktivität reception erst ins Leben gerufen hat.

  Auf Ebene der Kollektivitätseinheit A gilt wie im vorigen Beispiel, dass der Akteur für die receiver-Rolle vor der Ausführung der Empfangsaktion noch nicht in die Aktivität transaction eingebunden ist. Erst im Zuge der Aktion wird dieser Akteur eingebunden und ihm der Aktivitätsidentifikator übermittelt. Auf Ebene der Kollektivitätseinheit B wird dieser Aktivitätsidentifikator über den :act()-Kanal empfangen, über den :pStep()-Kanal an die Aktivität receive-and-ack weitergeleitet und dort für die spätere Verwendung

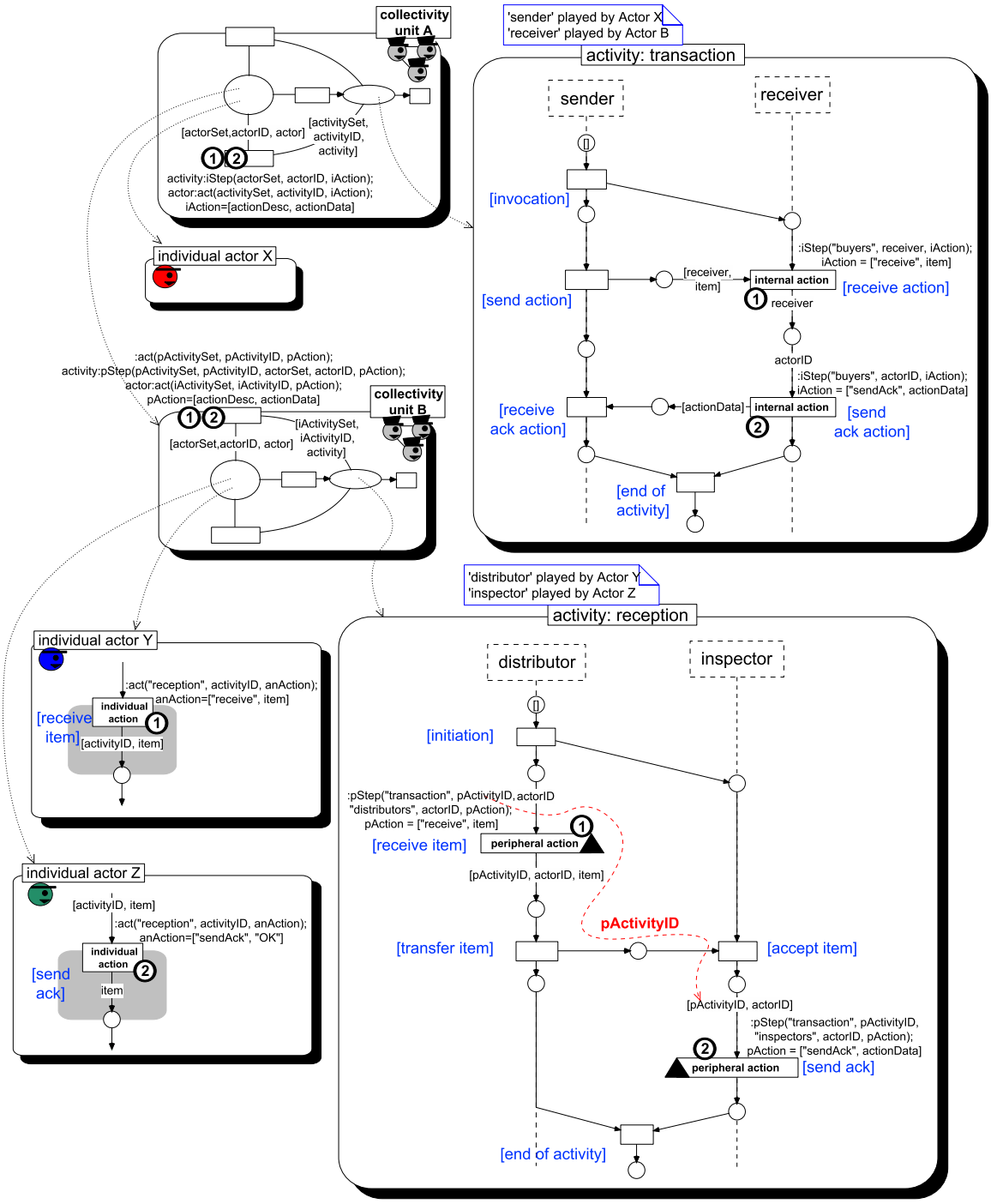

Abbildung 9.9: Periphere Aktionen in einer beispielhaften Aktivitätsverzahnung

aufbewahrt. Wie weiter unten beschrieben ist, benötigen die Akteure der Aktivitätsrollen von reception diesen Identifikator nicht notwendigerweise selber, um Peripherieaktionen im Kontext von reception ausführen zu können.

Allgemein gelten die gleichen Aussagen wie zuvor bezüglich der vielfältigen Dynamiken, die sich aus der unifizierenden Bindungssuche des Referenznetzformalismus ergeben. Die Bindungssuche wird lediglich umso komplizierter, je mehr Systemebenen einbezogen werden.

- **Zeitpunkte t_1 – t_2:** Nach Empfang des Gegenstandes zum Zeitpunkt t_1 wählt der distributor-Akteur einen Akteur für die inspector-Rolle aus und transferiert diesem den Gegenstand. Dabei können gegebenenfalls mehrere Inspekteure in Frage kommen und einer von ihnen abhängig vom empfangenen Gegenstand gewählt. Der inspector untersucht den Gegenstand (diese Aktionen sind nicht teil der Aktivitätsspezifikation reception) und versendet in Abhängigkeit von seinem Urteil eine Empfangsbestätigung zum Zeitpunkt t_2.

- **Zeitpunkt t_2:** Zu diesem Zeitpunkt führt der inspector-Akteur den Versand der Empfangsbestätigung aus. Analog zu Zeitpunkt t_1 findet hier eine Synchronisation über multiple Systemebenen hinweg statt. Obwohl es sich aus Sicht der Kollektivitätseinheit B um eine periphere Aktion handelt, benötigt der inspector-Akteur selber lediglich den Identifikator der reception-Aktivität, im Kontext derer er die Aktion *unmittelbar* ausführt. Der Identifikator der transaction-Aktivität auf Ebene der Kollektivitätseinheit A ist für ihn nicht notwendig. Dieser Identifikator wurde im Kontext der reception-Aktivität aufbewahrt und wird nun automatisch in die Aktionssynchronisation einbezogen. Er wird der Kollektivitätseinheit B über den :pStep()-Kanal übergeben und die Kollektivitätseinheit verwendet ihn für ihren :act()-Kanal, um sich ihrerseits auf die transaction-Aktivität auf Ebene von Kollektivitätseinheit A zu beziehen.

Sowohl der distributor- als auch der inspector-Akteur agieren somit *im Namen von Kollektivitätseinheit B*. Auf der nächst höheren Systemebene tritt Kollektivitätseinheit B als holistischer Akteur auf. Die Möglichkeit der lokalen Betrachtung der Belange jedes Akteurs für sich, ob individuell oder kollektiv, bleibt jederzeit gewahrt. Zusammenfassend liefert das bis hierher erarbeitete operationale Modell einer Kollektivitätseinheit modulare und mächtige Mechanismen für die exakte Modellierung der Funktionsweise von Systemen von Systemen. Gleichzeitig handelt es sich bei dem Modell um einen unmittelbar ausführbaren Prototypen. Er eignet sich, um konkrete Szenarien direkt zu beschreiben und auszuführen. Damit ist bereits die Grundlage für einen Softwareentwicklungsansatz basierend auf der Idee der rekursiven Akteurssysteme gegeben. In Abschnitt 9.2 wird das operationale Modell um die analytischen Aspekte der Operation, Integration und Governance erweitert. Damit wird der Schritt von Kollektivitäts- zu Organisationseinheiten als programmatischer Grundlage von Multiorganisationssystemen vollzogen und somit das operationale Modell dieses Abschnittes in Einklang mit dem theoretischen Denkmodells aus Abschnitt 8.2 gebracht. Zuvor behandeln die beiden folgenden Teilabschnitte allerdings noch die Dynamik von Einbettungsbeziehungen sowie Varianten des bisher vorgestellten operationalen Modells einer Kollektivitätseinheit.

9.1.4 Dynamik von Einbettungsbeziehungen

Das Hinzufügen und Entfernen von Akteuren im Kontext einer Kollektivitätseinheit wurde in den operationalen Modellen bislang ausgespart. In den einfachsten Fällen betrifft dies die Hinzufügung beziehungsweise Entfernung einer Einbettungsbeziehung. Komplexere Fälle bauen

darauf auf. Im Fall der Migration wird die Einbettung eines Akteurs in einer Kollektivitätseinheit aufgehoben und stattdessen eine Einbettung des Akteurs in eine andere Kollektivitätseinheit vorgenommen. Entsprechend der Philosophie der unterliegenden Modellierungstechnik der Referenznetze handelt es sich um *Akteursreferenzen*, die als Marken aus Kollektivitätseinheiten entfernt beziehungsweise ihnen hinzugefügt werden. Die Referenzsemantik erlaubt außerdem ausdrücklich den wiederum etwas anders gelagerten Fall der „Expansion“ eines Akteurs. Hier bleibt der Akteur seiner aktuellen Kollektivitätseinheit erhalten und weitet sein Wirken aus, indem er zusätzlich einer anderen Kollektivitätseinheit beitritt. Dieses Prinzip der nichtdisjunkten Akteurshierarchien folgt der realweltlichen Inspiration der *partiellen (nicht exklusiven) Beteiligung* sozialer Akteure an korporativen Entitäten (vergleiche Kapitel 3) und wurde im Laufe des Kapitels bereits mehrfach aufgegriffen. Aber erst an dieser Stelle kann dafür die technische Grundlage im Rahmen des operationalen Modells gelegt werden.

Man kann hier von einer Art *Mobilität* sprechen. Abbildung 7.1 aus Kapitel 7 folgend kann man zwei Arten von Mobilität unterscheiden. Einerseits werden Akteure (Agenten und Organisationseinheiten) auf Deployment-Plattformen softwaretechnisch ausgeführt und können gegebenenfalls zwischen den Plattformen migrieren. Andererseits stellt die Teilnahme an organisatorischen Kontexten eine weitere, inhaltliche Form der Ausführung eines Akteurs dar und das Wechseln/Erweitern von Kontexten stellt eine logische Form der Mobilität dar. Diese zweite Form der Mobilität, die genau mit der Betrachtung von sich ändernden Einbettungsbeziehungen korrespondiert, wird im Folgenden betrachtet.

9.1.4.1 Erweiterung des Grundmodells einer Kollektivitätseinheit

Die Mechanismen für die Dynamik von Einbettungsbeziehungen sind eine unmittelbare Erweiterung der bisher beschriebenen Mechanismen einer Kollektivitätseinheit, so dass das Verständnis nun leichter fällt. Akteursreferenzen werden nunmehr zu Objekten des Diskurses in Aktivitäten. Abbildung 9.10 zeigt die Erweiterung des bisherigen operationalen Modells einer Kollektivitätseinheit um Möglichkeiten der mobilen Akteursreferenzen. Der Übersichtlichkeit

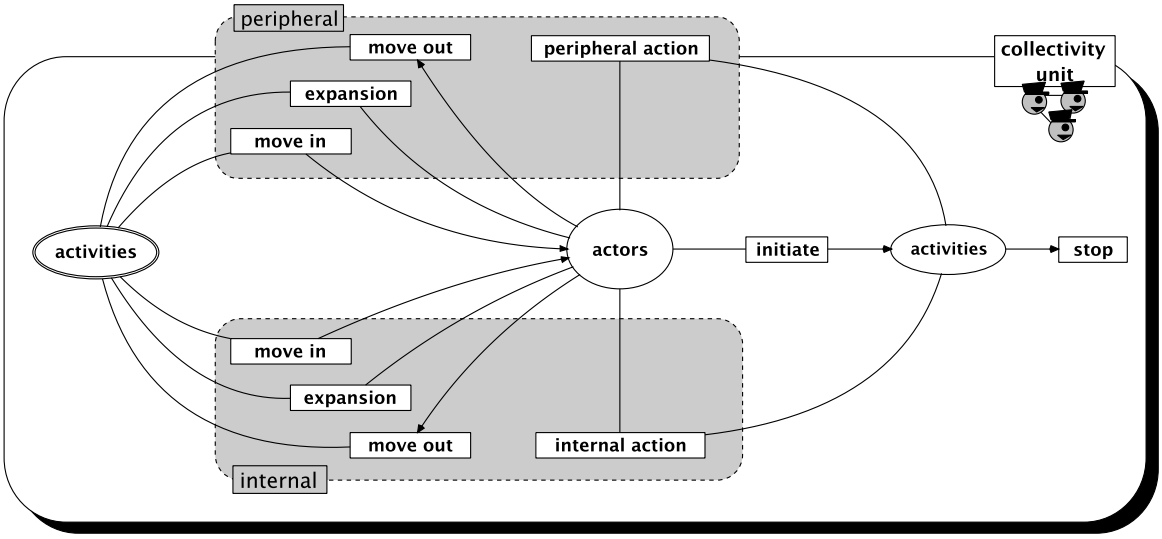

Abbildung 9.10: Erweiterung des Kollektivitätseinheitsmodells um mobile Akteursreferenzen

halber wird hier eine zweite Aktivitätsstelle als virtuelle Stelle der ursprünglichen Stelle aufgenommen. Dabei folgt die Erweiterung der Idee des bisherigen Modells. Alle Mobilitätsereignisse sind wiederum mit Aktionen von Akteuren im Kontext von Aktivitäten assoziiert. Ein Akteur kann der fokussierten Kollektivitätseinheit beitreten (move in), sie verlassen (move out) oder ihr erhalten bleiben und gleichzeitig sein Wirken auf andere Kollektivitätseinheiten ausweiten (expansion). Diese drei Fälle werden jeweils für interne und periphere Mobilitätsaktionen unterschieden. Auf technische Details in Form der notwendigen Anschriften wird hier verzichtet, sie bauen unmittelbar auf den bereits betrachteten Mechanismen auf. Der Unterschied besteht darin, dass nun Akteursreferenzen zwischen Kollektivitätseinheitsnetz und Aktivitätsnetzen ausgetauscht werden. Im Fall des Beitritts eines Akteurs (move in-Aktion) tritt der neuartige Umstand auf, dass der Akteur vor dem Beitritt noch gar nicht in die Kollektivitätseinheit eingebettet ist. In diesem Fall wird die Akteursreferenz von der Aktivität bereitgestellt.

9.1.4.2 Beispiele für die Umstrukturierung von Einbettungsbeziehungen

In der Folge werden einige Beispiele betrachtet, um die verschiedenen Szenarien, die durch Mobilitätsaktionen möglich werden, zu skizzieren. In den Beispielen werden die Interna der mobilen Akteure jeweils ausgeblendet da sich hier auf technischer Ebene keine Neuheiten gegenüber den bisherigen Ausführungen auftun. Neuartige Mechanismen auf technischer Ebene ergeben sich nur aus Perspektive der Umgebungen, die die mobilen Einheiten einbetten.

Abbildung 9.11 zeigt das erste Beispiel. In diesem Fall migriert ein Akteur „nach unten/innen“. Konkret heißt das hier, dass zunächst sowohl Kollektivitätseinheit B als auch Kollektivitätseinheit C als kollektive Akteure in Kollektivitätseinheit A eingebettet sind. Im Zuge der Migration wird die Referenz auf C von A aus eine Kollektivitätsebene tiefer nach B „verschoben“. Als abstraktes Anwendungsbeispiel kann hier eine Umstrukturierung von Hierarchieebenen einer Organisation dienen. Dabei wandert C mitsamt seinen rekursiv enthaltenen weiteren Akteursreferenzen eine Ebene tiefer.

Auf Ebene der Ursprungs-Kollektivitätseinheit A wird eine Aktivität move-downwards initiiert, im Zuge derer der migrierende Akteur etwaige Migrationsparameter mit der Ziel-Kollektivitätseinheit B verhandelt. Dabei handelt es sich aus Sicht von A ausschließlich um interne Aktionen eingebetteter Akteure. Die Aktivitätsteilnahme der Ziel-Kollektivitätseinheit B erfolgt mittels der Peripherieaktivität move-in-from-above, im Zuge derer der (individuelle) Akteur Y als admitter im Namen von B den Zugang des migrierenden Akteurs verhandelt. Aus Sicht von Kollektivitätseinheit B wird diese Verhandlung mittels peripherer Aktionen von Y geführt. Zum Zeitpunkt t_1 verlässt die Kollektivitätseinheit C schließlich Kollektivitätseinheit A über eine interne Austrittsaktion. Die Akteursreferenz wird dabei an die move-downwards-Aktivität übergeben und zum Zeitpunkte t_2 von der Ziel-Kollektivitätseinheit B empfangen. Aus Sicht von B gestaltet sich Zeitpunkt t_2 komplizierter. Hier wird die Referenz des neuen Akteurs erst mittels eines peripheren Aktivitätsschrittes empfangen. Gleichzeitig stellt dieser Schritt die Beitrittsaktion des neuen Akteurs dar. Diese Kombination wird gerade durch die Kollektivitätseinheitstransition für eine periphere Beitrittsaktion ermöglicht. Zum Zeitpunkt t_2 erhält Kollektivitätseinheit B dementsprechend die neue Akteursmarke. Die Referenz auf Kollektivitätseinheit C ist nun von Kollektivitätseinheit A aus eine Systemebene tiefer nach Kollektivitätseinheit B weitergereicht worden.

Abbildung 9.12 zeigt als weiteres Beispiel die Expansion eines Akteurs. In diesem Fall weitet Akteur X sein Wirken von Kollektivitätseinheit B nach Kollektivitätseinheit C aus. Die Kollektivitätseinheiten B und C sind beide wiederum in Kollektivitätseinheit A eingebettet und

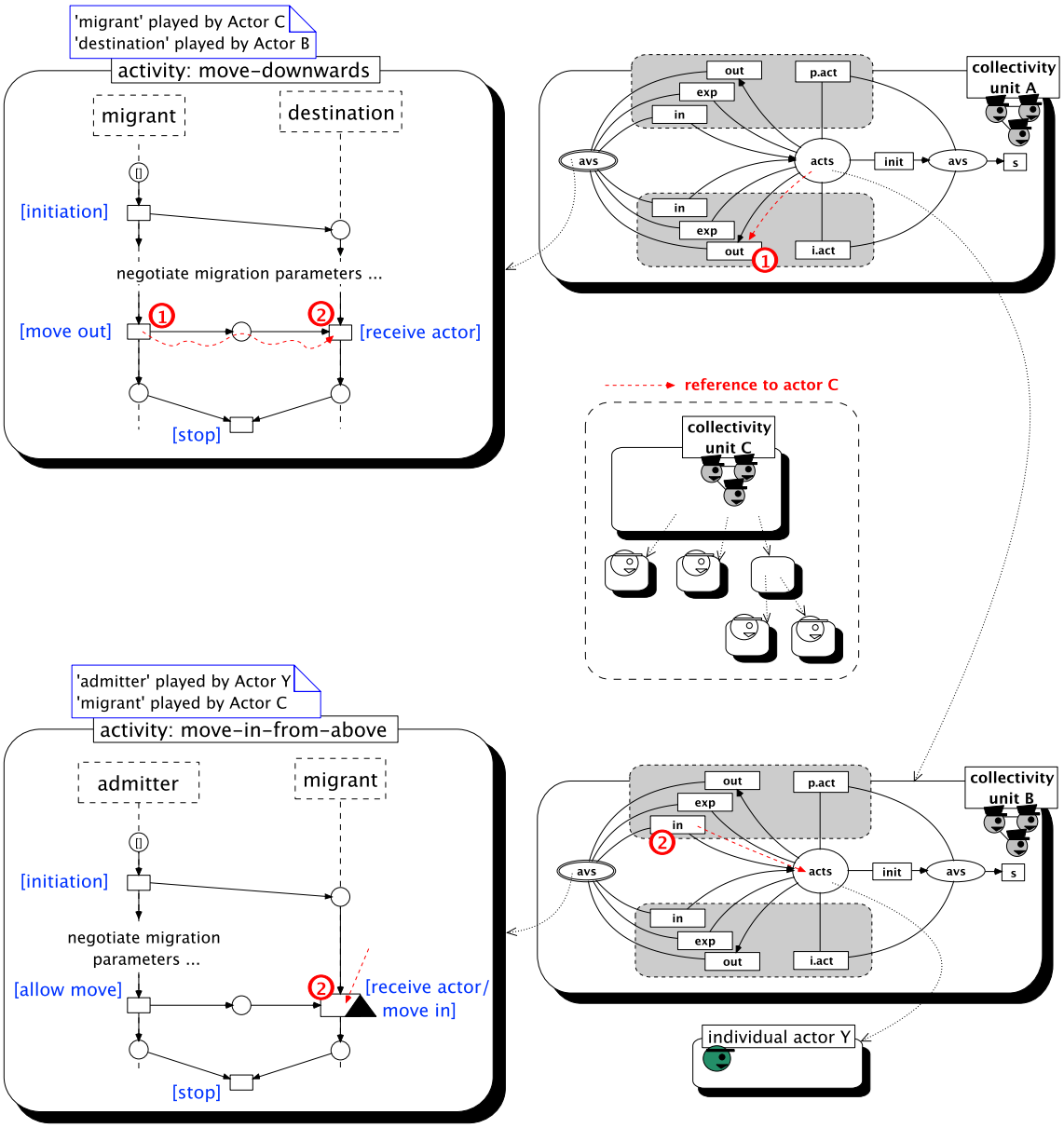

Abbildung 9.11: Migration eines Akteurs „nach unten/innen“

befinden sich somit auf derselben Systemebene. Folgerichtig findet hier eine Verzahnung dreier Aktivitäten statt. Auf Ebene von A gibt es die Aktivität move-from-to mittels derer B (als source) und C (als destination) in Kontakt treten. Die Teilnahme von B und C an dieser Aktivität wird mittels der Peripherieaktivitäten expand-upwards beziehungsweise move-in-from-above realisiert. Für diese beiden Aktivitäten ist jeweils nur die Rolle des expandierenden Akteurs aufgeführt. Zum Zeitpunkt t_1 führt Akteur X die Expansionsaktion aus. Aus Sicht von Kollektivitätseinheit B handelt es sich dabei um eine periphere Expansionsaktion, die auf der Ebene von A als gewöhnliche interne Aktion von B „als Ganzem“ wahrgenommen

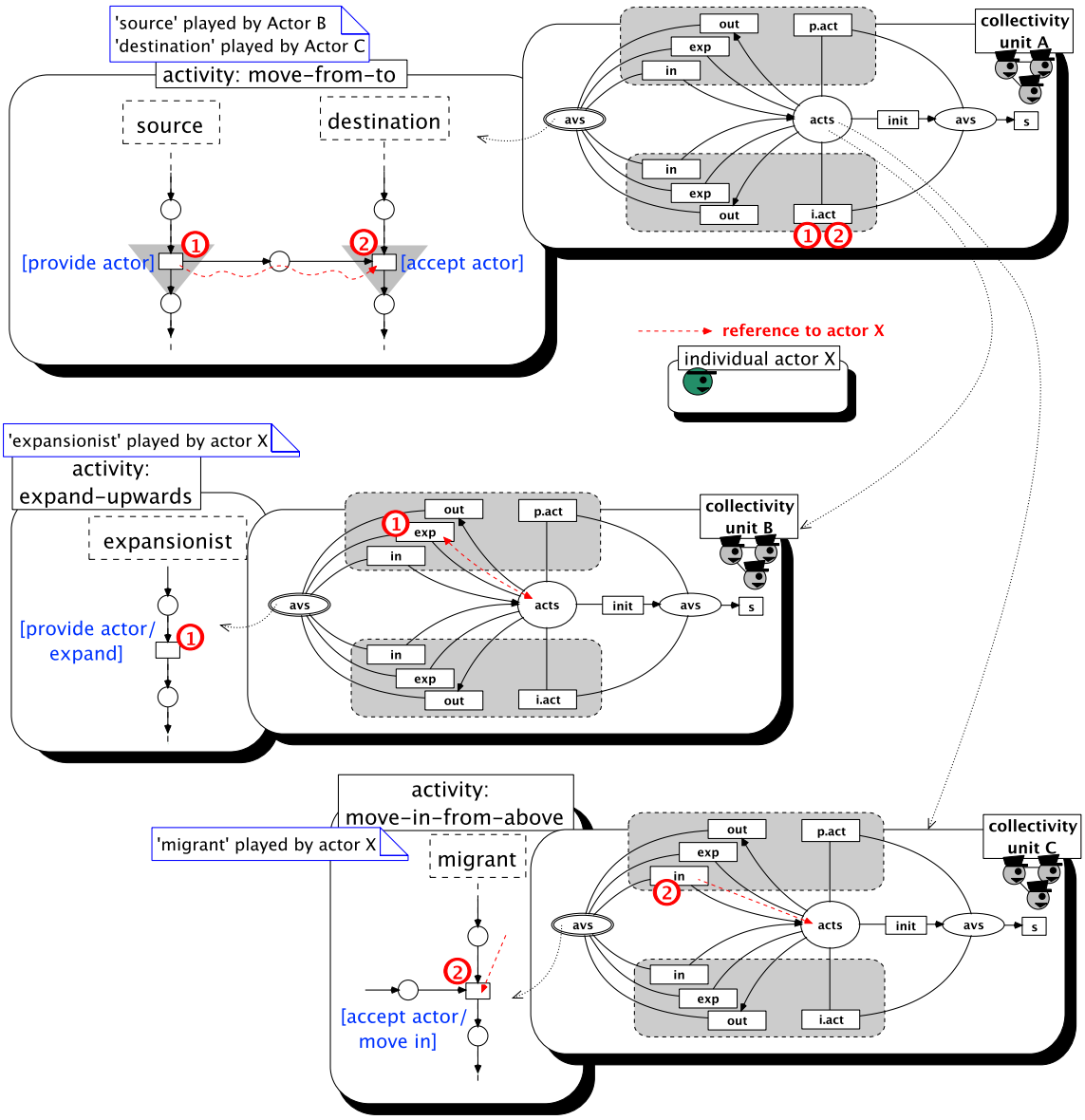

Abbildung 9.12: Expansion eines Akteurs

wird. Nach dieser Aktion bleibt B die Akteursmarke für X erhalten, eine Referenz auf X ist aber gleichzeitig an die move-from-to-Aktivität auf Ebene von A weitergereicht worden. Zum Zeitpunkt t_2 erfolgt der Beitritt von X zu C. Aus Sicht von A handelt es sich wiederum um eine gewöhnliche interne Aktion ihres eingebetteten Akteurs C. Aus Sicht von C wird die Akteursreferenz auf X im Zuge eines peripheren Aktivitätsschrittes erhalten und gleichzeitig stellt dieser Schritt eine periphere Beitrittsaktion von X zu C dar, so dass C die Referenz auf X in Form einer neuen Akteursmarke entgegen nimmt.

Bis hierher wurden lediglich Beispiele behandelt, in der neue Einbettungsbeziehungen auf Basis bereits bestehender Einbettungsbeziehungen entstehen. In diesem Zusammenhang stellt

sich die Frage, wie neu deployte Agenten und frisch formierte Kollektivitätseinheiten beziehungsweise Organisationseinheiten eines Multiorganisationssystems überhaupt zum ersten Mal in eine logische Einbettungsbeziehung treten. Hier dreht es sich also um den Übergang beziehungsweise den Zusammenhang zwischen der softwaretechnischen und logischen Ausführungsumgebung eines Multiorganisationssystems nach Abbildung 7.1 aus Kapitel 7. In Kapitel 7 wurde das Modellierungsvorhaben der vorliegenden Arbeit auf die logische Ebene eines Multiorganisationssystems eingeschränkt. Von der Ebene der Deployment-Plattformen wird abstrahiert. Der Einfachheit halber wird für den Moment angenommen, alle neuen Agenten und neuen Kollektivitätseinheiten würden bei ihrer Entstehung und damit bei ihrem Deployment automatisch logisch eingebettet. Dieser Annahme geht die Annahme voraus, jedes Multiorganisationssystem weise bereits zu Beginn eine initiale Kollektivitätsstruktur auf, von der alle weitere Evolution ausgeht.

9.1.4.3 Nichtdisjunkte Einbettung und Akteursidentitäten

Im Zusammenhang mit der Dynamik von Einbettungsbeziehungen und insbesondere der nichtdisjunkten Einbettung von Akteuren stellt sich die Frage nach der Identität eines Akteurs. Hier wird davon ausgegangen, jeder Akteur (Agent oder Organisationseinheit) sei auf Ebene der Deployment-Plattformen mit einer eindeutigen „physikalischen“ *Adresse* assoziiert.[2] Weiterhin wurde in Teilabschnitt 9.1.2 beschrieben, im Kontext einer Kollektivitätseinheit habe jeder eingebettete Akteur einen eindeutigen *Identifikator*. Dieser wird dem Akteur beim Beitritt in die Kollektivitätseinheit gegeben. Jeder von einer Kollektivitätseinheit vergebene Identifikator ist systemweit eindeutig (beispielsweise kann eine Kollektivitätseinheit Identifikatoren in Abhängigkeit von der eigenen Adresse vergeben). Ein mehrfach eingebetteter Akteur weist also neben seiner Adresse mehrere Identifikatoren gegenüber den Kollektivitätseinheiten auf, in der Akteur eingebettet ist. Seine gesammelten Identifikatoren machen gemeinsam die Identität eines Akteurs auf der (logischen) Anwendungsebene eines Multiorganisationssystems aus.

In Abbildung 9.13 ist modelliert, wie Identifikatoren der gezielten Initiierung von Aktivitäten in bestimmten Kontexten dienen. In Abbildung 9.5 aus Teilabschnitt 9.1.2 ist ersichtlich, dass der Identifikator einer neuen Aktivität im Kontext einer Kollektivitätseinheit immer in Abhängigkeit vom Identifikator des initiierenden Akteurs generiert wird. Ein Akteur kann also seine eigenen Identifikatoren nutzen, um die Initiierung von Aktivitäten gezielt auf bestimmte Umgebungen in Form von Kollektivitätseinheiten zu richten.

9.1.5 Varianten und Erweiterungen

Mit den Erläuterungen zur Dynamik von Einbettungsbeziehungen ist das operationale Grundmodell einer Kollektivitätseinheit abgeschlossen. Allerdings ist es möglich und in vielen Fällen erforderlich, Erweiterungen vorzunehmen. Dieser Unterabschnitt rundet die Vorstellung operationaler Kollektivitätseinheitsmodelle mit der Vorstellung einiger Varianten ab. Alle Varianten wahren das durch die Kanalschnittstellen :act(), :iStep() und :pStep() manifestierte operationale Grundprinzip. Insbesondere wahren sie das Prinzip der Lokalität einer Kollektivitätseinheit im Kontext eines Systems von Systemen. All die hier vorgestellten Varianten können für einzelne Kollektivitätseinheiten angewendet werden, ohne dass dies Auswirkungen auf andere Kollektivitätseinheiten hätte.

[2] Wird beispielsweise Renew als Deployment-Plattform angenommen, so ist jedes Netz mit einem eindeutigen Netzbezeichner assoziiert.

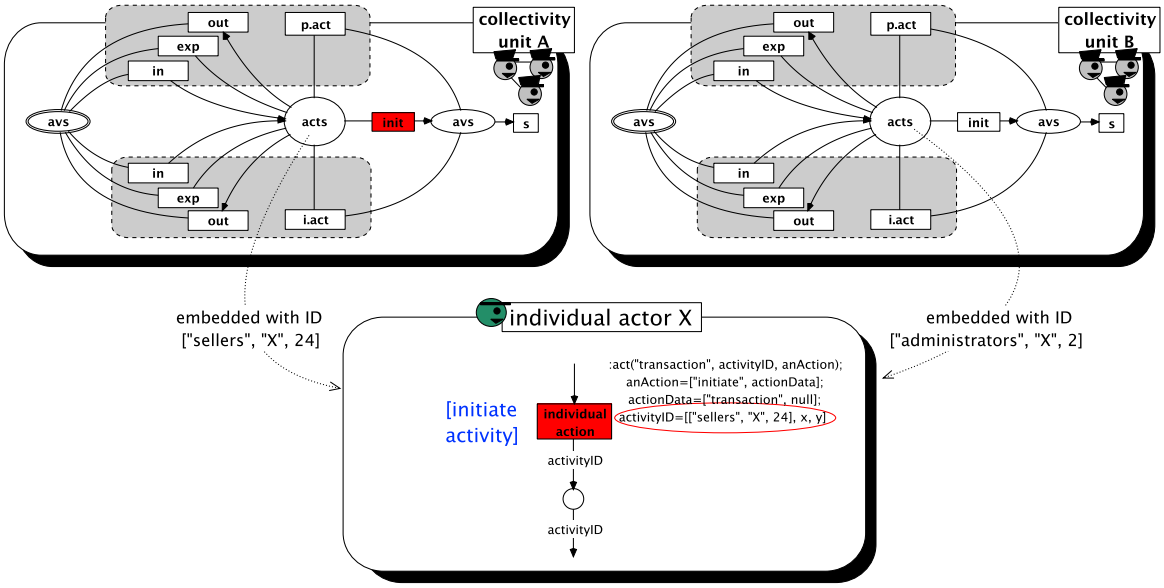

Abbildung 9.13: Gezielte Aktivitätsaktivierung über Akteursidentifikatoren

Periphere Aktionen mit doppelter Beschreibung Die technischen Details von peripheren Aktionen wurden in Unterabschnitt 9.1.2 behandelt. Dabei wird nach dem Prinzip verfahren, Aktionen eingebetteter Akteure direkt als Aktionen der Kollektivitätseinheit „als Ganzem“ auf der nächst höheren Systemebene auszugeben. In einigen Fällen kann es allerdings notwendig oder zumindest hilfreich sein, die betroffene Aktion beim Übergang zumindest insofern anzupassen, dass sie eine neue Beschreibung erhält. Abbildung 9.14 illustriert die dazu notwendige Erweiterung des bisherigen Mechanismus.

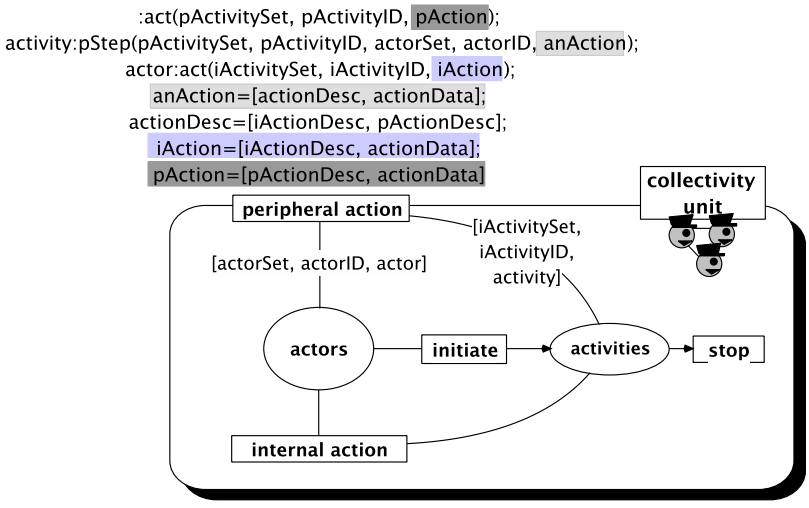

Abbildung 9.14: Periphere Aktionen mit doppelter Beschreibung

Die durch den :pStep()-Kanal übermittelte Aktion enthält zwei Beschreibungen. Die erste wird intern verwendet während die zweite auf der nächst höheren Systemebene verwendet wird. Dies ist beispielsweise hilfreich wenn Aktionen, die auf der nächst höheren Systemebene als verschiedenartig interpretiert werden, intern mit ein und derselben Form von Aktion realisiert wird. Auf diese Weise wird die Möglichkeit der lokalen Entwicklung einzelner Kollektivitätseinheiten im Kontext eines Systems von Systemen weiter manifestiert.[3]

Direkter Durchgriff Die Aktivierung von Aktivitäten findet im bisherigen Modell lediglich mittels interner Aktionen statt. Damit besteht ein gewisses Ungleichgewicht, da für alle anderen Aktions-Typen interne sowie periphere Fälle betrachtet wurden. Abbildung 9.15 zeigt eine Erweiterung, die auch den Fall einer peripheren Aktivierungsaktion beinhaltet. Der Vorteil ei-

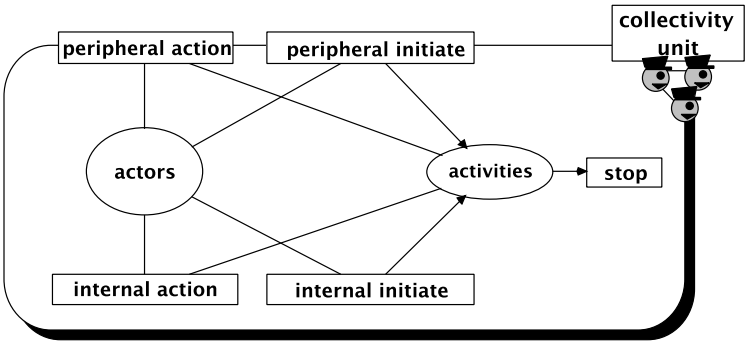

Abbildung 9.15: Periphere Aktivierungsaktion

ner solchen peripheren Aktivierungsaktion liegt darin, dass es nun möglich ist, *gleichzeitig* Aktivitäten auf verschiedenen Systemebenen zu aktivieren. Ohne diese Möglichkeit ist gegebenenfalls ein mühsames schrittweises Setup notwendig, wenn eine Aktivitätsverzahnung auf mehreren Systemebenen angestoßen werden soll.

Aktivitäten können bislang nur durch interne Aktionen einer Kollektivitätseinheit aktiviert werden. Diese Herangehensweise ist sehr hilfreich, da eine Kollektivitätseinheit auf diese Weise maximale Kontrolle über ihr Grenzmanagement besitzt. An dieser Stelle sei aber bereits ein Vorgriff auf den folgenden Abschnitt 9.2 erlaubt. In einem Organisationsszenario kann es durchaus notwendig sein, dass eine Organisationseinheit einen direkten Durchgriff auf eingebettete Organisationseinheiten hat. Dementsprechend zeigt Abbildung 9.16 eine Erweiterung des bisherigen Kollektivitätseinheitsmodells. Hier können Aktivitäten von außerhalb aktiviert werden. Es sei darauf hingewiesen, dass es sich hierbei um das erste Modell handelt, bei dem das bisherige Grundprinzip, nach dem Aktionen im Kontext einer Kollektivitätseinheit stets mit einem eingebetteten Akteur assoziiert sind, verletzt wird.

[3] In einigen Fällen ist diese Herangehensweise sogar notwendig. Beispielsweise findet sie Anwendung im Kontext der peripheren Mobilitätsaktionen, deren technische Details in Unterabschnitt 9.1.4 ausgeblendet wurden. Die Schlüsselwörter „move-out“ und „expand“ dürfen nur auf der Systemebene, auf der tatsächlich ein Austritt beziehungsweise eine Expansion stattfindet, als Aktionsbeschreibungen verwendet werden. Würden diese Aktionsbeschreibungen nach oben/außen propagiert, so würde das eine Kettenreaktion auf höheren Systemebenen auslösen.

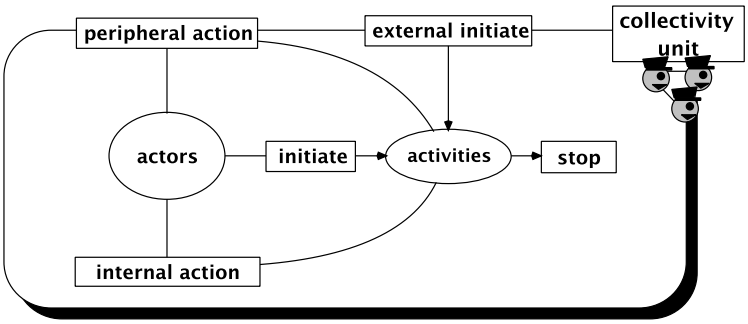

Abbildung 9.16: Externe Aktivitätsaktivierung

Gemeinsame Aktionen In Abschnitt 8.2 zum abstrakten Modell von Kollektivitäts- und Organisationseinheiten wurde die Möglichkeit *gemeinsamer* Aktionen durch mehrerer Akteure berücksichtigt. Solche Aktionen werden von den beteiligten Akteuren synchron ausgeführt. Im bisher vorgestellten operationalen Modell ist eine solche gemeinsame Ausführung nicht möglich. Dieser Umstand ist dem Referenznetzformalismus geschuldet, der nicht mehr als einen Uplink als Kanalanschrift zulässt. Daher ist es nicht ohne weiteres möglich, eine Aktivitätstransition einfach mit Anschriften für mehrere Schritte (Aktionen) zu versehen. Allerdings zeigt Abbildung 9.17 anhand des bereits behandelten Beispiels des Versandes eines Gegenstandes eine Möglichkeit, gemeinsame Aktionen mittels rekursiver Kanalaufrufe zu realisieren. Dabei zeigt die Abbildung den Fall interner gemeinsamer Aktionen. Der periphere Fall folgt analog zu den bisherigen Ausführungen.

Anstatt der üblichen Synchronisierung eines Akteurs und einer Aktivität für die Ausführung *einer Aktion* erfolgt hier eine kompliziertere Synchronisation. Für die beiden eingebetteten Akteure ändert sich auf technischer Ebene nichts gegenüber dem Fall des asynchronen Versandes und Empfangs eines Gegenstandes. Auf Ebene der Kollektivitätseinheit allerdings werden diese beiden Aktionen in einem einzigen Schaltvorgang synchronisiert. Dazu wird zunächst von Seiten der Aktivität aus eine *Liste von Aktivitätsschritten* anstatt eines einzigen Schrittes bereitgestellt. Diese Liste wird in der Folge mittels rekursiver Kanalaufrufe in einem einzigen Schaltschritt abgearbeitet, so dass die beteiligten Akteure ihre jeweiligen Aktionen tatsächlich gemeinsam ausführen.

9.2 Operationalisierung von Steuerungs- und Kontrollaspekten

Analog zum Vorgehen in Abschnitt 8.2 im Kontext der abstrakten Modellierung gilt es nun, das operationale Modell einer Kollektivitätseinheit mit Steuerungs- und Kontrollaspekten anzureichern, um zu einer Operationalisierung von Organisationseinheiten zu gelangen. Die Ausführlichkeit der bisherigen operationalen Modelle hat dabei eine umfassende Grundlage bereitet. Dies ermöglicht eine Konzentration auf die konzeptionellen Kernaspekte beim Übergang zu Organisationseinheiten. Zusätzliche technische Details müssen dabei nur vereinzelt ergänzt werden. Für das abstrakte Modell einer Organisationseinheit in Abschnitt 8.2 wurde das UML-Konzeptdiagramm aus Abbildung 8.16 herangezogen, um das Zusammenspiel der analytischen Aspekte Operation, Integration und Governance zu erläutern. Für das operationale Modell erfolgt nun stattdessen eine petrinetzbasierte Interpretation dieses Zusammenspiels. Damit

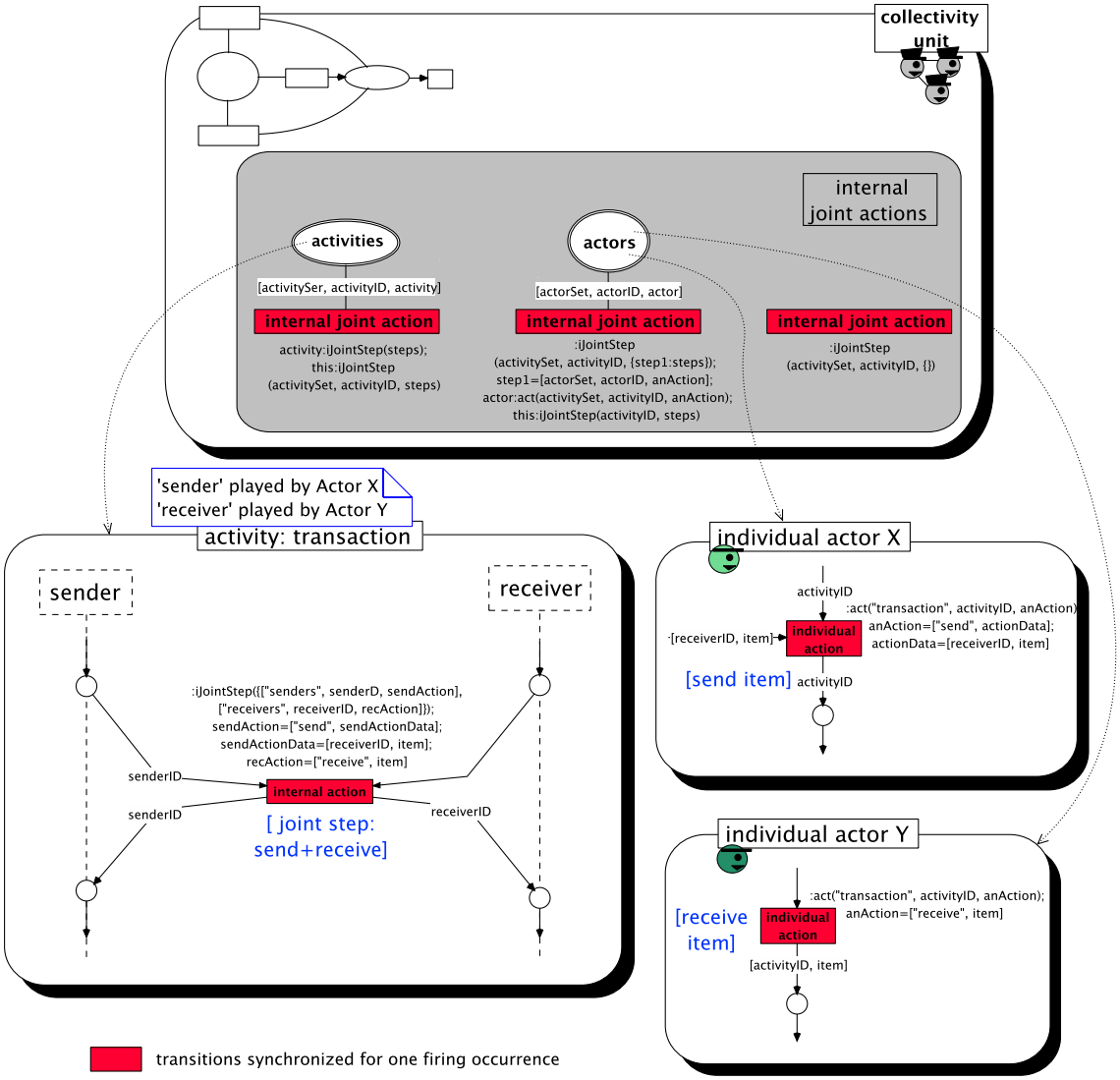

Abbildung 9.17: Technische Realisierung gemeinsamer Aktionen

einher geht dementsprechend auch ein programmatischer Umgang mit der zuvor abstrakten Beschreibung. Der Übergang vom abstrakten Denkmodell zum konkreten Programmiermodell für Organisationseinheiten findet damit seinen Abschluss.

9.2.1 Grundmodell einer Organisationseinheit

Die technische Funktionsweise einer Organisationseinheit baut unmittelbar auf den Mechanismen einer Kollektivitätseinheit auf. Um die Steuerungs- und Kontrollaspekte im Sinne des abstrakten Modells aus Abschnitt 8.2 in das operationale Modell einzuflechten, erfolgt zunächst eine konkrete Unterscheidung von Akteurs- und Aktivitätsmengen. Es werden die Mengen der Operations-, Integrations- und Governance-Akteure einerseits und die Mengen der Integrations- und Governance-Aktivitäten andererseits expliziert. Entlang dieser Mengen

werden nun auch die möglichen Aktionen differenziert. Darüber hinaus erfährt die Behandlung von Systemregeln eine prominente Stellung. Die Bedeutung des Konzeptes der Systemregeln wurde bereits im Kontext des abstrakten Modells herausgearbeitet und als zentral für das Beziehungsgeflecht der verschiedenen Akteure herausgestellt. Für das operationale Modell werden Systemregeln nun auf Ebene der Organisationseinheit expliziert.

Abbildung 9.18 zeigt das erste operationale Grundmodell einer Organisationseinheit zunächst ohne weitere Anschriften (Abbildung A.2 im Anhang der Arbeit zeigt eine Version des Modells inklusive aller für die direkte Ausführbarkeit notwendigen Anschriften). Neben den

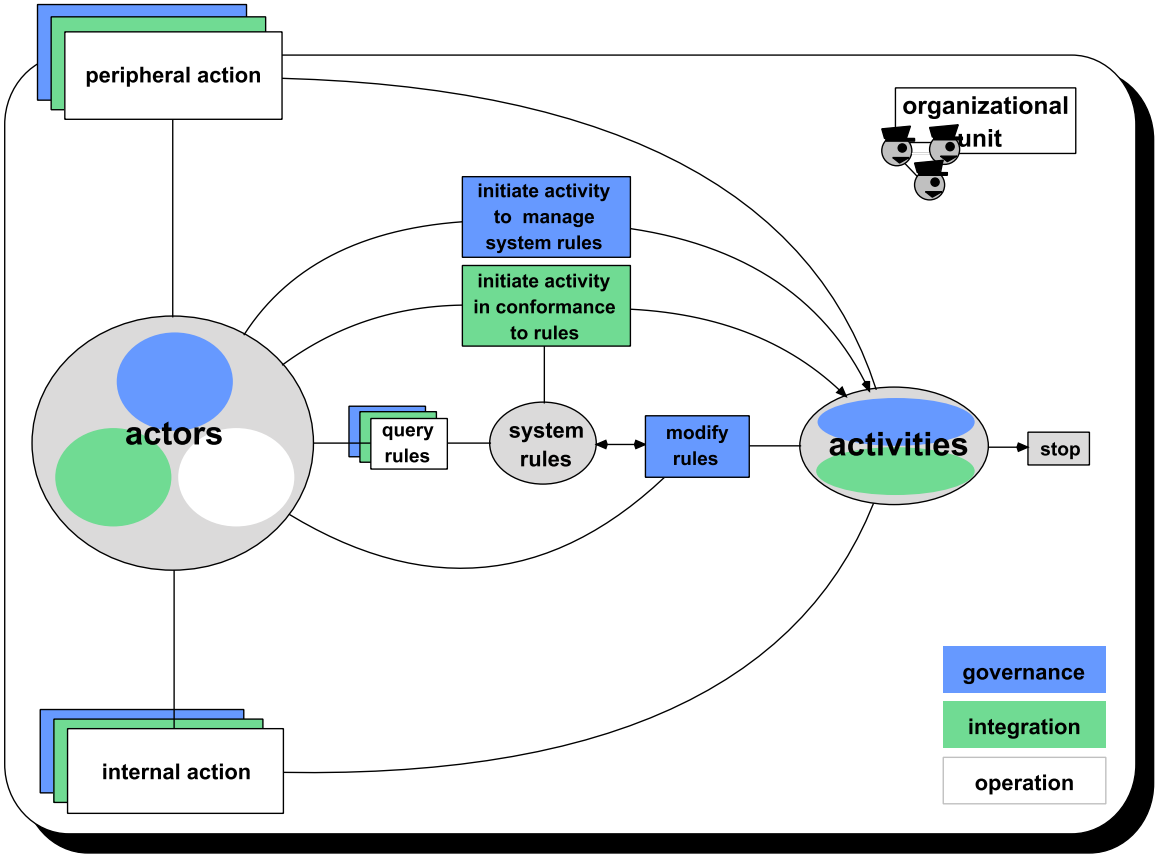

Abbildung 9.18: Erstes operationales Modell einer Organisationseinheit

bereits bekannten Stellen für Akteure und Aktivitäten gibt es nun eine Stelle für die Systemregeln. Wie im folgenden Teilabschnitt genauer erläutert wird, sind Systemregeln in Form eines eigenen Objektnetzes eingebettet. Die wesentliche Funktionsweise des operationalen Modells einer Organisationseinheit wird in diesem Teilabschnitt bereits natürlichsprachlich erläutert, bevor im folgenden Teilabschnitt einige der zu Grunde liegenden technischen Details näher erläutert werden.

Es handelt sich hier um ein erstes Grundmodell, für das in Teilabschnitt 9.2.3 Varianten und Erweiterungen betrachtet werden. Außerdem ist der Aspekt der Mobilität von Akteursreferenzen hier ausgeblendet. Die Abbildung fokussiert auf den Aspekt des Zusammenspiels der verschiedenen Mengen von Akteuren und Aktivitäten mit den Systemregeln. Die Transitionen

sind dabei derart „gefärbt“, dass sie auf die Akteursmengen verweisen, die die entsprechenden Aktionen ausführen dürfen. Dabei liefert das Modell die Grundlage für eine Operationalisierung des Konzeptdiagramms für Steuerung und Kontrolle einer Organisationseinheit aus Abbildung 8.16 in Abschnitt 8.2. Im Wesentlichen betrifft dies die Trennung von Zuständigkeiten in Hinsicht auf das *Setzen* und *Umsetzen* von Systemregeln.

Zunächst gilt es, die Initiierung von Aktivitäten näher zu betrachten. Nur Integrations- und Governance-Akteure haben die Berechtigung, neue Aktivitäten zu initiieren. Sie legen damit fest, welche Art von Aktivitäten überhaupt möglich sind und nehmen mittels der Festlegung von Aktivitätsparametern und gegebenenfalls eigener Teilnahme eine zusätzliche inhaltliche Prägung der Aktivitäten vor. Sie steuern und kontrollieren damit das gesamte Verhalten der Organisationseinheit (auf dieser Systemebene). Dabei gilt, dass die von Integrationsakteuren initiierten Integrationsaktivitäten in Übereinstimmung mit geltenden Systemregeln sein müssen. Für die von Governance-Akteuren initiierten Governance-Aktivitäten gilt diese Beschränkung nicht. Darüber hinaus können Governance-Akteure im Kontext von Governance-Aktivitäten Modifikationen an den Systemregeln vornehmen. Dieser Umstand wird im Modell aus Abbildung 9.18 bereits dadurch veranschaulicht, dass ein Zugriff auf die Stelle system rules im Zuge einer modify-Aktion über eine Reservierungskante erfolgt damit keine nebenläufigen, konfliktbehafteten Änderungsoperationen auftreten können. Ansonsten erfolgt der Zugriff über Testkanten da in diesen Fällen von einem unkritischen, rein lesenden Zugriff ausgegangen wird.[4]

Analog zu den Ausführungen aus dem vorigen Kapitel gilt weiterhin, dass eine systematische Kategorisierung von Systemregeln erst im dritten Teil der vorliegenden Arbeit vorgenommen wird. Für den Moment genügt die allgemeine Auffassung von Systemregeln als verbindliche Rahmenbedingungen für die Funktionsweise einer Organisationseinheit, beziehungsweise als ihre organisatorischen Prinzipien. Dementsprechend handelt es sich bei Integrationsaktivitäten also um „alltägliche“ Aktivitäten, in denen die Einzelakteure in Übereinstimmung mit diesen Rahmenbedingungen als Teil des Ganzen (der Organisationseinheit) auftreten. Dagegen handelt es sich bei Governance-Aktivitäten um potentiell „außerordentliche“ Aktivitäten, im Zuge derer Akteure eben diese Rahmenbedingungen in Frage stellen und neu ausrichten können.

Damit ergibt sich die Grundlage, das in Abschnitt 8.2 abstrakt modellierte und formulierte Zusammenspiel der analytischen Aspekte der Operation, Integration und Governance unmittelbar zu operationalisieren. Die Governance-Akteure geben durch ihre Hoheit über die Systemregeln einen *hochrangigen* oder auch *strategischen* „Verhaltenskorridor“ beziehungsweise Rahmen für die Organisationseinheit in ihrer Gesamtheit vor. Integrationsakteure wirken innerhalb dieses Rahmens, sorgen für die Anwendung der Systemregeln auf den operationalen Kontext der Organisationseinheit und nehmen auf diese Weise punktuelle und temporäre weitere Verengungen des Verhaltenskorridors/Rahmens vor. Die Funktion der Operationsakteure wurde in Teilabschnitt 8.2 als die Ausführung der Primär-/Kernoperationen einer Organisationseinheit charakterisiert. Sie sind in dieser Funktion allerdings abhängig davon, dass die Integrations- und Governance-Akteure ihnen einen entsprechenden Kontext bereitstellen. Konkret bedeutet dies, dass Operationsakteure nur im Kontext von Aktivitäten, die zuvor von Integrations- beziehungsweise Governance-Akteuren initiiert wurden, agieren können. Dabei agieren Integrations- und Governance-Akteure auch selbst im Kontext von Aktivitäten. Ty-

[4] Die hier für das Grundmodell formulierten Annahmen werden im Zuge der in Teilabschnitt 9.2.3 betrachteten Erweiterungen teilweise aufgehoben.

pischerweise handelt es sich dabei um Aktionen, die im Zusammenhang mit Steuerung und Monitoring des Aktivitätsverlaufs/-inhalts stehen oder die zu Prozessen der Einigung und Entscheidungsfindung beitragen.

Es fällt auf, dass für die Aktionsmöglichkeit des Abfragens von Systemregeln mit dem bisherigen Prinzip, nach dem Aktionen immer in unmittelbarem Zusammenhang mit Aktivitäten stehen müssen, gebrochen wird. Es handelt sich hier um einen ständig wiederkehrenden Fall mit ausschließlich lokalen Effekten für den abfragenden Akteur. Daher wird für diese speziellen Aktionen auf die Bedingung, sie müssten im Kontext von Aktivitäten stattfinden, verzichtet. Dabei können im Kontext der Systemregeln Beschränkungen festgelegt sein, welcher Akteur Zugang zu welchen Informationen erhält.

Die strukturelle Konfiguration aufbauend auf den Aspekten Governance, Integration und Operation ist also fest im operationalen Modell einer Organisationseinheit verankert („hart kodiert“). Die Systemregeln dienen nun dazu, konkrete Ausprägungen des universellen Modells einer Organisationseinheit zu spezifizieren („soft zu kodieren“). Im Wesentlichen wird hier eine konsistente Menge von Systemregeln gefordert, anhand derer geprüft werden kann, ob die Aktivierung einer bestimmten Aktivität zulässig ist. Diese einfache Herangehensweise wird später erweitert, erlaubt jedoch bereits in dieser Form die Operationalisierung ausdrucksstarker Systemspezifikationen, die im Wesentlichen auf drei Arten von Systemregeln fußen.

1. Bestimmungen darüber, welche Akteure die Berechtigung haben, welche Aktivitäten zu gestalten und zu initiieren.

2. Bestimmungen darüber, welche Akteure an welchen Aktivitäten teilnehmen dürfen beziehungsweise müssen.

3. Bestimmungen darüber, welche Form und Charakteristika bestimmte Aktivitäten aufweisen müssen.

Dabei handelt es sich immer um *zusätzliche* Bestimmungen, die in die zugrunde liegende Konfiguration von Governance, Integration und Operation eingebettet sind. Die ersten beiden Punkte erlauben gemeinsam eine feinere Strukturierung einer Organisationseinheit, beispielsweise in Form von Rollen-, Positions- und Gruppenspezifikationen, wie es für organisationsorientierte Multiagentensysteme üblich ist. Dort werden Rechte und Verantwortlichkeiten von Agenten häufig darüber formuliert, welche Rollen oder Positionen sie in bestimmten Gruppen einnehmen. Dementsprechend kann hier eine Verfeinerung der grundsätzlichen Governance-, Integrations- und Operations-Struktur einer Organisationseinheit vorgenommen werden. Der erste und der dritte Punkt erlauben gemeinsam eine mehr oder weniger standardisierte Behandlung von Aktivitäten. Beschränkungen für die Aktivitätsgestaltung können dabei verschiedene Formen annehmen. Beispiele sind die Forderung nach Aktivitätsspezifikationen auf Basis von Workflownetzen [vdA97] oder die Einhaltung/Berücksichtigung bestimmter *Interaktionslandmarken* (vergleiche beispielsweise [VSDD05]). Systemregeln, die Akteuren einen hohen Freiheitsgrad hinsichtlich der Gestaltung von Aktivitäten einräumen, führen schließlich zu eher organischem Systemverhalten im Gegensatz zu einem bürokratischen Verhalten resultierend aus starren Vorgaben.

In der Folge werden einige Operationalisierungsdetails beschrieben, um das technische Verständnis der Funktionsweise einer Organisationseinheit zu schärfen. Anschließend werden Erweiterungen und Varianten des bisher vorgestellten Grundmodells vorgestellt. Auf die Betrachtung von Beispielen wird in diesem Abschnitt gänzlich verzichtet. Stattdessen sei hierfür

auf das Kapitel 10 verwiesen. Dort wird das Grundmodell einer Organisationseinheit entlang einiger organisationsorientierter Ansätze aus dem Bereich der Multiagentensysteme konkretisiert.

9.2.2 Technische Details

Eine tiefgehende Betrachtung der dem operationalen Modell einer Organisationseinheit zu Grunde liegenden technischen Details erübrigt sich in vielerlei Hinsicht durch die zuvor erfolgte ausführliche Betrachtung der Mechanismen von Kollektivitätseinheiten. Die wesentlichen Erweiterungen, die sich durch den Übergang von System- zu Organisationseinheiten ergeben, werden in der Folge betrachtet.

Einbettung der Systemregeln Anhand des Modells aus Abbildung 9.18 bleibt weitestgehend offen, wie die Einbindung von Systemregeln tatsächlich realisiert wird. Abbildung 9.19 zeigt in diesem Zusammenhang die Schnittstelle zu den Systemregeln als Referenznetzmodell der Abstraktionsstufe II (die ausgeblendeten Details sind in Abbildung A.2 im Anhang der Arbeit zu finden). Dabei zeigt sich, dass Systemregeln in Form eines Objektnetzes in das Organi-

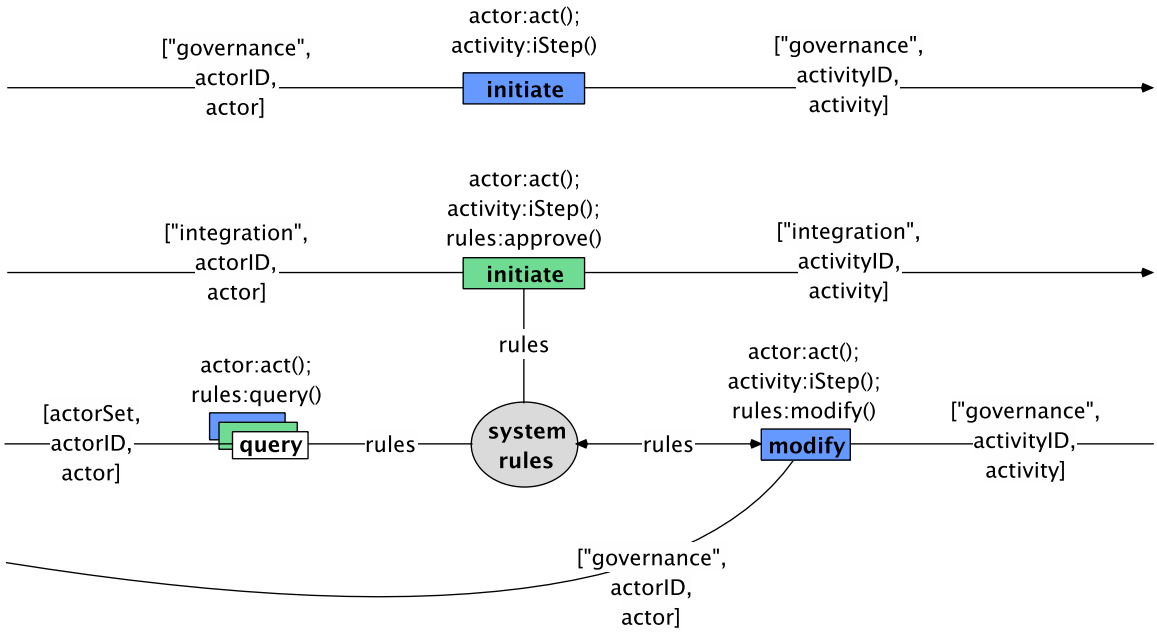

Abbildung 9.19: Schnittstelle zum Systemregelnetz

sationseinheitsnetz eingebettet werden. Das Systemregelnetz fungiert dabei als „Wrapper“, der die eigentlichen Systemregeln kapselt und die Kanäle :query(), :approve() sowie :modify() als Schnittstelle anbietet. Die Parameter für die Kanäle des Systemregelnetzes ergeben sich dabei jeweils aus den Aktionsdaten.

Dieser Ansatz des „Wrappens“ der Systemregeln erlaubt, verschiedene konkrete Organisationseinheiten nicht nur mit unterschiedlichen, sondern mit gänzlich *verschiedenartigen* Sys-

temregeln auszustatten. In einigen Fällen ist eine einfache Herangehensweise basierend auf Tupeln über primitiven Datentypen gegebenenfalls ausreichend. In anderen Fällen kann es von Vorteil sein, die Systemregeln wiederum als Petrinetz zu spezifizieren und somit eine operationale Behandlung von Systemregeln zu propagieren. In den Beispielen aus Kapitel 10 wird die Herangehensweise gewählt, Systemregeln deklarativ in Form eines logischen Formalismus zu formulieren. Ein solcher Ansatz erlaubt den systematischen Aufbau komplexer Regelsysteme bei gleichzeitigem Rückgriff auf elaborierte Schlussmechanismen. Abbildung A.4 im Anhang der Arbeit zeigt in diesem Zusammenhang das Systemregelnetz zum Wrappen einer PROLOG-Regelbasis. Weitere Erklärungen werden in Kapitel 10 geliefert. Grundsätzlich besteht auch die Möglichkeit, mehrere Systemregelnetze in eine Organisationseinheit einzubetten, um damit verschiedene Bereiche der Systemspezifikation mit verschiedenen (und gegebenenfalls verschiedenartigen) Systemregelsätzen abzudecken. Allerdings wird bei dieser Herangehensweise die Wahrung der Konsistenz der Regelsätze untereinander deutlich schwieriger.

Aktionsausführung In Teilabschnitt 9.1.2 wurden die operationalen Details der Aktionsausführung im Kontext von Kollektivitätseinheiten behandelt. Für das operationale Modell einer Organisationseinheit müssen auf technischer Ebene keine Änderungen vorgenommen werden. Inhaltlich gilt, dass die Bezeichner von Akteurs- und Aktivitätsmengen nun eingeschränkt sind auf `"operation"`, `"integration"`, `"governance"` beziehungsweise auf `"integration"`, `"governance"`. Besonders wichtig ist dabei, dass eine Operationalisierung der in Teilabschnitt 8.2.2 beschriebenen Möglichkeiten der unterschiedlichen analytischen Interpretationen von Aktivitäten unmittelbar unterstützt wird. Die Zugehörigkeit von Akteuren und Aktivitäten zu bestimmten Mengen wird immer jeweils lokal im Kontext einer Organisationseinheit vorgenommen. So kann eine Peripherieaktivität beispielsweise der Menge der Integrationsaktivitäten angehören, auf Ebene einer einbettenden Organisationseinheit jedoch mit einer Aktivität der Menge der Governance-Aktivitäten verzahnen. Analog gilt dies für Akteursmengen. Auf Ebene einer Organisationseinheit kann eine Aktivitätsrolle beispielsweise von einem Governance-Akteur eingenommen werden und dessen Aktionen sorgen dafür, dass die Organisationseinheit „als Ganzes“ im Kontext einer umschließenden Organisationseinheit eine Aktivitätsrolle als Operationsakteur einnimmt.

9.2.3 Varianten und Erweiterungen

Das Grundmodell einer Organisationseinheit aus Abbildung 9.18 kann in unterschiedliche Richtungen erweitert werden. In diesem Unterabschnitt werden zunächst einige weiterführende Aspekte auf technischer Ebene behandelt, bevor anschließend konzeptionelle Erweiterungen diskutiert werden. Dabei gilt außerdem, dass die in Teilabschnitt 9.1.5 behandelten Varianten und Erweiterungen von Kollektivitätseinheiten orthogonal zu den hier vorgestellten angewandt werden können.

Erzwungene Dynamik von Einbettungsbeziehungen Der Aspekt der Dynamik von Einbettungsbeziehungen und der mobilen Akteursreferenzen wurde ausführlich in Teilabschnitt 9.1.4 behandelt. Da in diesem Zusammenhang noch keinerlei Steuerungs- und Kontrollaspekte berücksichtigt wurden, wurde für Mobilitätsaktionen jeweils ein Konsens zwischen dem mobilen Akteur und der betroffenen Kollektivitätseinheit vorausgesetzt. Für den Fall einer Organisationseinheit sind hingegen einige zusätzliche Anwendungsfälle denkbar. So können beispielsweise Szenarien vorkommen, in denen zum Zwecke von Reorganisation eines Multiorganisa-

tionssystems Umverteilungen von Einbettungsbeziehungen vorgenommen werden. In anderen Fällen kann es notwendig sein, bestimmte Akteure aus einer Organisationseinheit auszuschließen. In solchen und ähnlichen Fällen liegt es nahe, dass Umstrukturierungen von Steuerungsakteuren (Integration, Governance) vorgenommen werden, gegebenenfalls ohne Zustimmung der betroffenen Akteure. Abbildung 9.20 modelliert die Operationalisierung solcher Umverteilungen auf Ebene einer Organisationseinheit. Dabei ist allerdings auf inhaltlicher/fachlicher Ebene dennoch häufig ein Konsens notwendig, um eine Dysfunktionalität der resultierenden Systemkonfiguration zu verhindern.

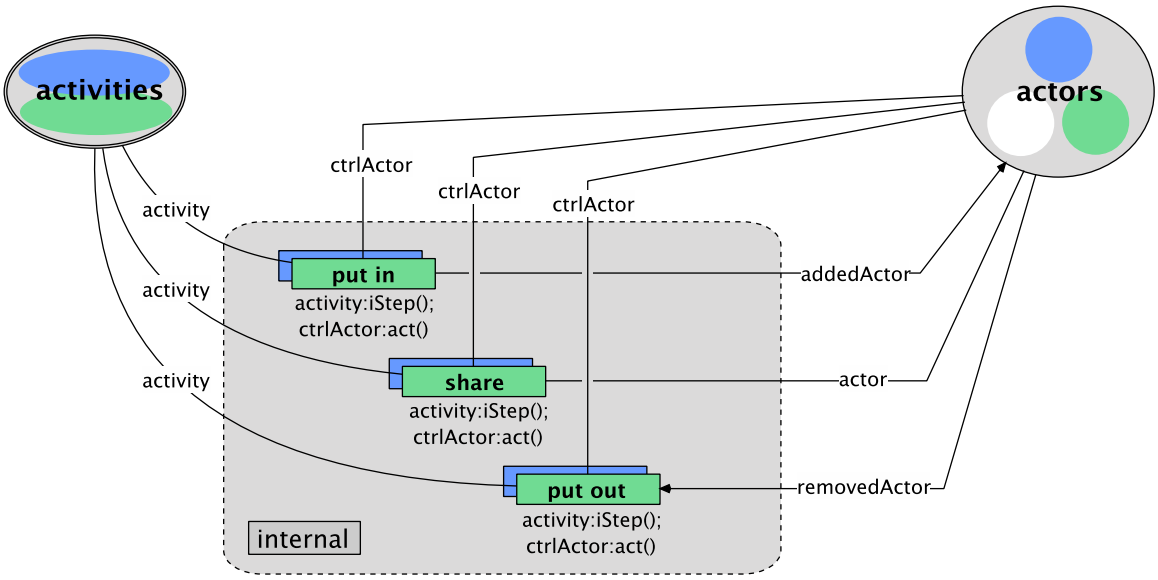

Abbildung 9.20: Erzwungene Akteursmobilität (interner Fall)

Elaborierte Behandlung von Systemregeln Die Erläuterungen aus Teilabschnitt 9.2.1 zielen auf die Verwendung von Systemregeln im ersten Grundmodell einer Organisationseinheit ab. Das grundlegende Prinzip besteht dabei darin, dass Governance-Akteure Systemregeln in einer Art und Weise gestalten, dass sie bei der Initiierung von Integrationsaktivitäten für die Überprüfung der Zulässigkeit der jeweiligen Aktivitäten herangezogen werden können.

Es gibt viele Möglichkeiten, die Handhabung von Systemregeln im Kontext einer Organisationseinheit auszubauen. Dazu gehört beispielsweise auch die Aufnahme „weicher“ Systemregeln. Damit sind solche Regeln gemeint, die keine absoluten Vorgaben, Einschränkungen und Berechtigungen betreffen. Stattdessen betreffen sie Richtlinien, Vorgehensweisen, Standards oder auch Ziele, die in der Regel Anwendung finden beziehungsweise eingehalten werden sollten, allerdings nicht erzwungen werden. Dabei handelt es sich allerdings immer noch um formale und nicht um informelle Strukturelemente. Insofern können sie gegebenenfalls mit Belohnungs- beziehungsweise Sanktionsmechanismen einhergehen. Es ist daher notwendig, konkrete Akteure damit zu beauftragen, das Akteursverhalten im Kontext einer Organisationseinheit zu beobachten, zu bewerten und gegebenenfalls einzuschreiten.

Die Handhabung solcher weichen Systemregeln im Kontext einer Organisationseinheit findet dementsprechend auf inhaltlicher Ebene statt und kann nicht direkt in das operationale Modell

einfließen. Stattdessen werden in der Folge zwei konzeptionelle Erweiterungen betrachtet, die das operationale Modell selber betreffen.

Die erste Erweiterung betrifft eine Unterscheidung, die bereits im Kontext der Organisationsinfrastrukturen für Multiagentensysteme in Abschnitt 5.2 (Abbildung 5.3) hervorgehoben wurde: Die *Organisationsspezifikation* umfasst allgemeingültige Regeln und Fakten, die für eine Agentenorganisation gelten. Zusätzlich wird der *Organisationszustand* explizit repräsentiert, so dass Organisationsspezifikationen auch kontextabhängig formuliert werden können. Der Organisationszustand bezieht sich dabei nur auf einige konkrete Fakten, die auf globaler Ebene beobachtbar sind. Der Organisationszustand in seiner Gesamtheit liegt über die Mitglieder verteilt vor.

Für das operationale Modell einer Organisationseinheit soll eine analoge Unterscheidung vorgenommen werden. Systemregeln kapseln die *Spezifikation* sowie den *Zustand* (beziehungsweise einen Teil des Zustandes) einer Organisationseinheit. Die Spezifikation betrifft dabei Regeln und Fakten, mittels derer eine Konkretisierung/Verfeinerung der grundlegenden Operations-, Integrations-, Governance-Konfiguration vorgenommen wird. Auf diese Weise ergibt sich die Spezifikation einer konkreten Organisationseinheit als Ausprägung des universellen Schemas. Der Zustand betrifft einige (zumeist temporäre) Fakten, die sich im Kontext der Ausführung einer Organisationseinheit ergeben. Die Spezifikation einer Organisationseinheit kann dabei ausschließlich durch Governance-Akteure modifiziert werden. Der Zustand jedoch kann sich im Zuge von *Effekten* jeglicher Aktivitäten einer Organisationseinheit ändern. In Kapitel 10 werden konkrete Beispiele von Organisationseinheiten behandelt. Dabei wird jeweils beschrieben, welche Art von Systemregeln es gibt und in welcher Form sich Spezifikation und Zustand dabei widerspiegeln.

Abbildung 9.21 zeigt ein entsprechend erweitertes operationales Grundmodell einer Organisationseinheit und die erweiterte Kanalschnittstelle des Systemregelnetzes. (Abbildung A.3 im Anhang der Arbeit zeigt dieses Organisationseinheitsmodell inklusive aller für die Ausführbarkeit notwendigen Anschriften). Die Kanalschnittstelle des Systemregelnetzes ist gegenüber den in Teilabschnitt 9.2.2 behandelten Kanälen :query(), :approve() und :modify() nun um den Kanal :effect() erweitert (dabei kann ein Systemregelnetz auch jeweils mehrere Transitionen für jeden Kanal anbieten). Dieser :effect()-Kanal wird auf Ebene des Organisationseinheitsnetzes an den Transitionen für interne, periphere und initiierende Aktionen aufgerufen und die Parameter ergeben sich durch ein zusätzliches Argument der Kanäle :iStep() beziehungsweise :pStep().[5] Es gilt allerdings, dass nicht *jede* dieser Aktionen notwendigerweise Effekte auf den Systemzustand hat. Stattdessen kann auch ein leerer Effekt angegeben werden. In Kapitel 10 wird die Einbindung ganz konkreter Systemregeln betrachtet. In dem Zusammenhang wird auch die Schnittstelle des Systemregelnetzes ausführlicher betrachtet.

Eine weitere Erweiterung kann darin bestehen, neben Effekten auch rein lesende Systemregelanfragen mit einer Aktion zu verknüpfen. Dies dient vor allem dem Einbezug von Aktionen, deren Ausführung an bestimmte Bedingungen der Systemregeln geknüpft sind.

Die beiden bisher betrachteten Grundmodelle einer Organisationseinheit (Abbildungen 9.18

[5] Hier wird ein Unterschied zwischen dem Auftreten von Änderungen an Spezifikation und Zustand einer Organisationseinheit deutlich. Parameter einer Spezifikationsmodifikation ergeben sich aus den Aktionsdaten (vergleiche die Erläuterungen aus Teilabschnitt 9.2.2), während sich Parameter einer Zustandsänderung als zusätzliches Argument der :iStep()/:pStep()-Kanäle ergeben. Dieser Unterschied spiegelt den Umstand wider, dass Spezifikationsmodifikationen auf „bewusste“ Aktionen von Governance-Akteuren zurückgehen, während Zustandsänderungen als „Seiteneffekte“ beliebiger Aktionen auftreten können, ohne dass dies den jeweiligen Akteuren notwendigerweise bewusst ist.

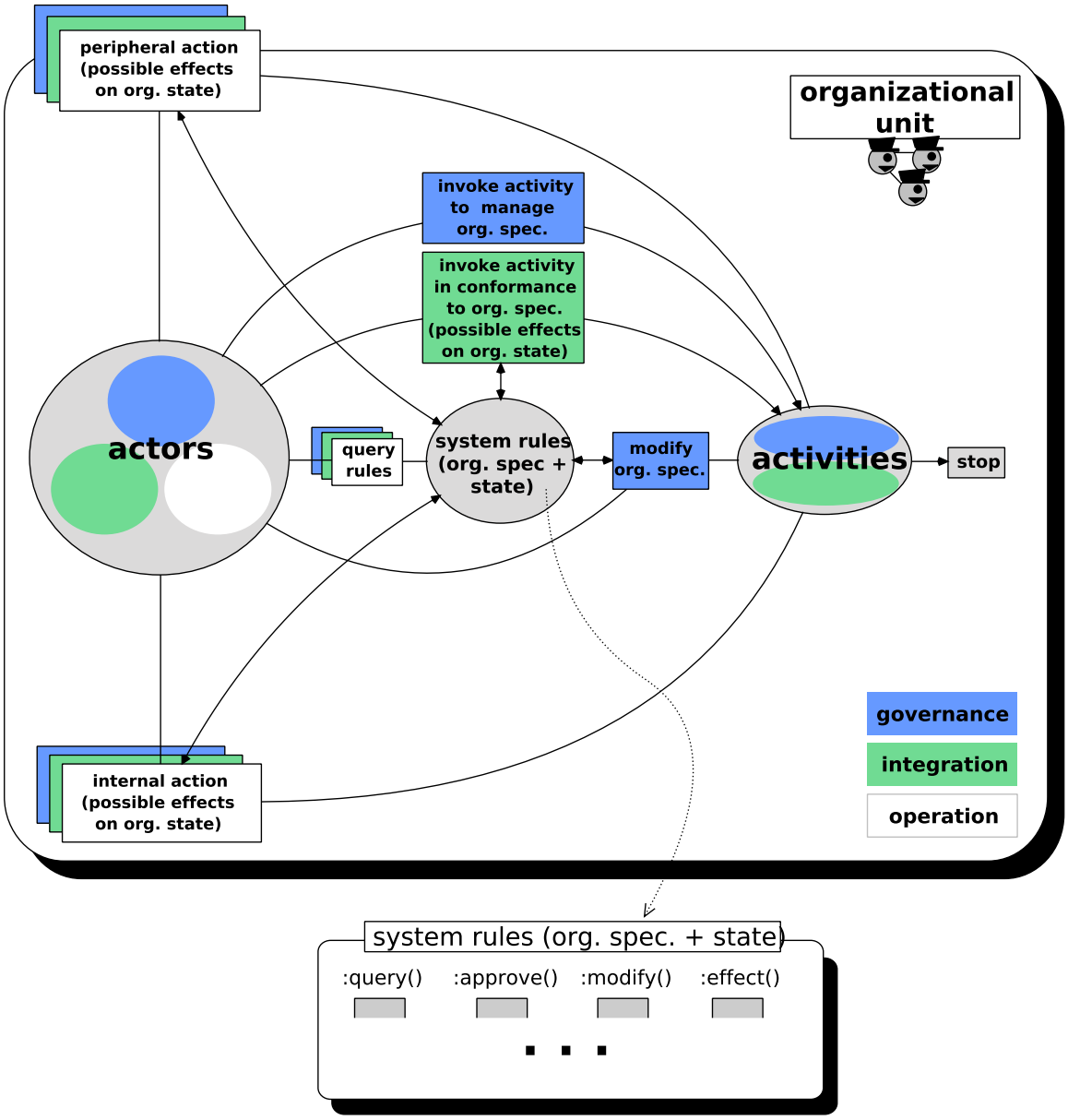

Abbildung 9.21: Trennung von Systemregeln in Spezifikation und Zustandsbeschreibung

und 9.21 beziehungsweise A.2 und A.3 im Anhang) gestatten Governance-Akteuren einen beträchtlichen Spielraum. Governance-Akteure können Aktivitäten aktivieren, ohne die Einhaltung von Systemregeln berücksichtigen zu müssen, und dürfen darüber hinaus im Kontext dieser Aktivitäten Modifikationen an den Systemregeln vornehmen. Dementsprechend lastet eine große Verantwortung auf den Schultern der Governance-Akteure einer Organisationseinheit. Für viele Anwendungsfälle ist diese Herangehensweise nicht angemessen. In einem Multiorganisationssystem gilt für viele Organisationseinheiten, dass sie äußeren Zwängen unterworfen sind. Häufig obliegt es nicht einer Organisationseinheit selber, sich diesen Zwängen anzupas-

sen oder sie zu ignorieren beziehungsweise zu umgehen. Stattdessen können externe Vorgaben *verbindlich* sein. Diese Sichtweise wurde bereits in den Kapiteln 3 und 4 für realweltliche soziale Organisationen zitiert. Die Quintessenz dieser Sichtweise ist, dass auch Governance-Aktivitäten Systemregeln unterworfen sind. Diese Regeln repräsentieren die grundsätzlichen Grenzen, innerhalb derer sich die Organisationseinheit als Ganzes bewegen darf. Man kann sie als *Meta-Regeln* aus Sicht einer fokussierten Organisationseinheit auffassen.

Abbildung 9.22 zeigt die konzeptionelle Erweiterung des operationalen Organisationseinheitsmodells um duale Systemregeln, die nun nach Integrationsregeln und Governance-Regeln unterschieden werden. Dabei ist die intuitive Perspektive modelliert, dass Governance-Regeln durch direkten Durchgriff „von oben/außen“ und damit von einer höheren Systemebene aus bestimmt werden. Die hier modellierte Dualität von Systemregeln ist orthogonal zu

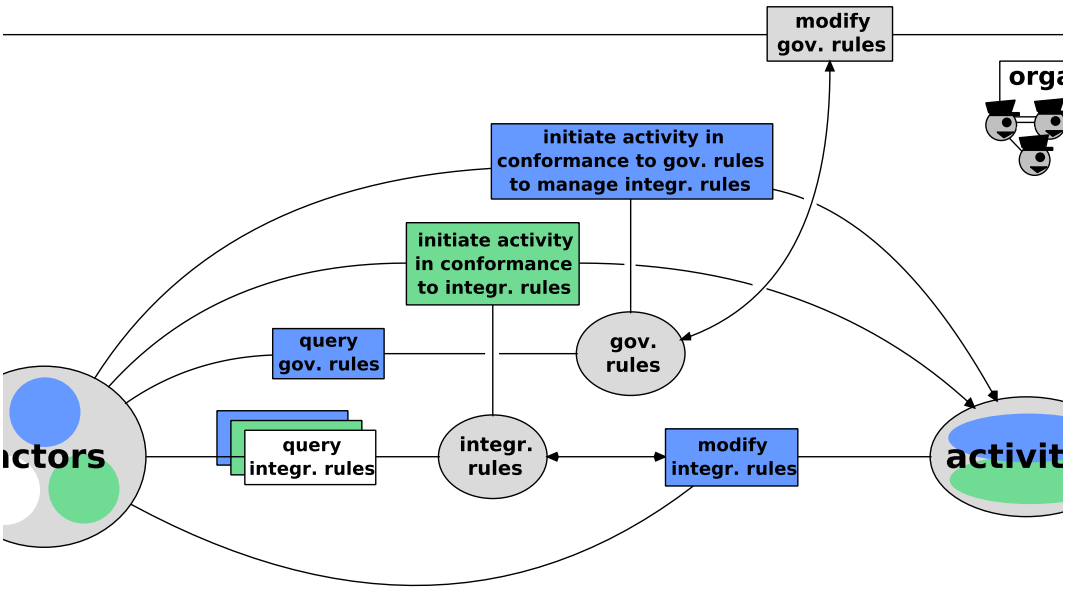

Abbildung 9.22: Duale Systemregeln mit externem Durchgriff

den vorherigen Ausführungen zu Systemregeln, die nach Spezifikation und Zustandsbeschreibungen differenziert werden können. Eine solche Differenzierung kann dementsprechend für Integrationsregeln als auch für Governance-Regeln vorgenommen werden.

Die Betrachtung von Governance-Regeln macht gegebenenfalls auch ohne den Einbezug äußerer Zwänge Sinn. Bereits in Abschnitt 8.2.1 wurde darauf verwiesen, dass das Governance-Konzept mit der Assoziation von zumindest potentiell *vielen* verantwortlichen Akteuren einhergeht. Für diese gilt, dass sie eine Einigung und Kompromisse bezüglich Steuerungsentscheidungen finden müssen. Dies kann dementsprechend mit eigenständigen Strukturen innerhalb des analytischen Governance-Anteils einer Organisationseinheit einhergehen. Die Strukturen bestimmen, welche Governance-Akteure tatsächlich die Berechtigung zur Steuerung bestimmter Systemanteile haben.

9.3 Diskussion: Kerneigenschaften und Anforderungen

Für jedes Softwareentwicklungsparadigma ist wichtig, die Kerneigenschaften der jeweils zentralen Softwareentität (des Basisbausteins) zu explizieren. Im Falle eines Organisationsparadigmas handelt es sich dabei um die Softwareentität, die von der Metapher der sozialen Organisation beziehungsweise der organisierten Kollektivität abgeleitet ist. Damit ergeben sich die Eigenschaften eigentlich aus den Ausführungen in Kapitel 4. Es gilt nun aber, eine möglichst knappe, aber präzise Charakterisierung zu finden, die eine direktere Verwendbarkeit für die Softwareentwicklung in den Vordergrund rückt. In der vorliegenden Arbeit werden Multiorganisationssysteme als konkrete Form von organisationsorientierten Softwaresystemen betrachtet. Der konkrete Basisbaustein ist hier die Organisationseinheit. Zum Abschluss der beiden Kapitel zur Modellierung von Organisationseinheiten ergibt sich ein klares Bild ihrer Kerneigenschaften. Ein Anspruch auf Verbindlichkeit für den allgemeinen Fall von organisationsorientierten Softwaresystemen geht damit nicht einher.

- **Nicht-exklusive Komposition:** Die eingebetteten Akteure als Teilnehmer an der Organisationseinheit können in weitere Organisationseinheiten eingebettet sein.
- **Freiwillige Teilnahme:** Die eingebetteten Akteure einer Organisationseinheit sind autonom und daher kann die durch sie bereitgestellte Funktionalität nicht jederzeit vorausgesetzt werden.
- **Explizite und verbindliche Formalstruktur:** Organisationseinheiten bewahren eine eigene Existenz unabhängig von teilnehmenden Akteuren, verkörpert durch eine Formalstruktur und der darauf aufbauenden Reglementierung des Verhaltens der Teilnehmer.
- **Basisfunktionalität**: Eine Organisationseinheit ist gekennzeichnet durch die Fähigkeit zur Erbringung bestimmter Basisfunktionalität. Dies manifestiert sich in der Menge der Operationsakteure.
- **Steuerung und Kontrolle:** Es *muss* eingebettete Akteure geben, die die Systemregeln interpretieren können, um regelkonforme Aktivitäten zu initiieren. Darüber hinaus *kann* es Akteure (Governance) geben, die die Systemregeln und das daraus resultierende Verhalten bewerten und gegebenenfalls modifizieren. Diese beiden Aspekte manifestieren sich in den Mengen der Integrations- beziehungsweise Governance-Akteure.
- **Kollektives Akteurswesen:** Eine Organisationseinheit ist fähig, selbst Aktionen auszuführen. Jede Aktion der Organisationseinheit ist dabei durch eine Aktion eines eingebetteten Akteurs konstituiert.

Darauf aufbauend wird nun die zum Ende des vorigen Kapitels begonnene Diskussion fortgesetzt, inwieweit die Ergebnisse des zweiten Teils der Arbeit die in Abschnitt 7.2 formulierten Anforderungen erfüllen. Bisher wurde festgehalten, dass die Betrachtung des janusgesichtigen Charakters von Organisationseinheiten als Kontext und als Akteur sowie die konkrete Berücksichtigung der drei konzeptionellen Richtungen eine besonders prominente Stellung einnehmen und die diesbezüglichen grundlegenden Anforderungen erfüllt werden. In der Folge werden die differenzierten Anforderungen/Fragestellungen entlang der Aufzählung aus Abschnitt 7.2 nacheinander betrachtet. Dabei geht es weniger darum, ob jede der Fragestellungen durch explizite Modellelemente im Grundmodell einer Organisationseinheit berücksichtigt wird. Stattdessen geht es darum, inwieweit das Grundmodell einer Organisationseinheit und der darauf

aufbauende Charakter von Multiorganisationssystemen einen angemessenen konzeptionellen Rahmen bietet, der die Adressierung der jeweiligen Fragestellungen leitet.

- *Domänendefinition:* Diese Fragestellung wird unmittelbar durch den rekursiven Charakter eines Multiorganisationssystems adressiert. Die Umwelt einer Organisationseinheit besteht aus den Organisationseinheiten, in die die fokussierte Organisationseinheit eingebettet ist. Diese umschließenden Organisationseinheiten verkörpern jeweils eigene Umgebungen/Domänen, im Kontext derer die fokussierte Organisationseinheit operiert.

- *Bestimmung von Zielen/Strategien:* Die Bestimmung von langfristigen beziehungsweise grundlegenden Zielen und Strategien einer Organisationseinheit ist grundsätzlich den Governance-Akteuren vorbehalten. Ob Ziele explizit in die Systemregeln einer Organisationseinheit aufgenommen werden und damit Teil der formalen Spezifikation werden, kann dabei von Fall zu Fall variieren. Einerseits können sie zwecks der Sichtbarkeit für andere Akteure aufgenommen werden. Andererseits können sie auch schlicht als Vereinbarung zwischen den Governance-Akteuren existieren. Im letzteren Fall würden lediglich die aus den Zielen/Strategien folgenden Struktur- und Verhaltensspezifikation für den operativen Betrieb der Organisationseinheit in die formale Spezifikation aufgenommen werden. Eine weitere Variante kann dann auftreten, wenn das um duale Systemregeln erweiterte Modell einer Organisationseinheit betrachtet wird (vergleiche Abbildung 9.22). Dann können Ziele im Kontext von Governance-Regeln formalisiert werden, während im Kontext von Integrationsregeln lediglich die abgeleiteten Spezifikationen für den operativen Betrieb festgehalten werden.

  Wenn eine starke Abhängigkeit einer Organisationseinheit von umschließenden Organisationseinheiten besteht, dann kann auch der Fall auftreten, dass Ziele und Strategien mittels direktem Durchgriff von außen vorgegeben werden. Auch dieser Fall passt insbesondere zum Organisationseinheitsmodell mit dualen Systemregeln.

- *Steuerung und Kontrolle:* In Abhängigkeit von den grundlegenden Zielen/Strategien bestimmen Governance-Akteure die formale Spezifikation für den operativen Betrieb der Organisationseinheit, um die Ziele/Strategien zu verfolgen. Es handelt sich hier um Struktur- und Verhaltensspezifikationen für die Steuerung und die Kontrolle des Organisationseinheitsverhaltens. Diese werden in Form der Systemregeln expliziert und sind in der Folge bindend. Hier kann man die Unterscheidung zwischen eher bürokratischen und organischen Spezifikationen treffen. Besonders detaillierte und umfassende Regeln sorgen für stark standardisiertes Verhalten, im Zuge dessen den Integrationsakteuren nur ein sehr enger Spielraum bei der Anwendung/Konkretisierung der Systemregeln zugestanden wird. Weitläufige und hochrangige Regeln hingegen, die eher auf dem Prinzip der Delegation von Verantwortung beruhen, lassen den Integrationsakteuren größere Freiheiten und sorgen für flexibles und dynamisches Systemverhalten (mit dem Nachteil der geringeren Vorhersagbarkeit und Verlässlichkeit).

- *Machtverteilung:* Bereits die grundsätzliche analytische Trennung in Operation, Integration und Governance stellt eine für jede Organisationseinheit verbindliche und damit formalisierte Machtverteilung unter den Akteuren dar. Governance-Akteure bestimmen die Geschicke einer Organisationseinheit auf hochrangiger/strategischer Ebene und besetzen damit die höchsten Machtpositionen. Im Kontext der daraus abgeleiteten Vorga-

ben werden Integrationsakteure eingesetzt, um die konkrete kontextabhängige Steuerung und die Kontrolle des operativen Betriebs der Organisationseinheit vorzunehmen. Operationsakteure sind zwar für die Kernaufgaben der Organisationseinheit verantwortlich, sind dabei aber vollständig abhängig von den Handlungsmöglichkeiten, die ihnen von Seiten der Akteure der anderen beiden Akteursmengen eröffnet werden. Es sei hier aber noch einmal hervorgehoben, dass die analytische Trennung von Operation, Integration und Governance keine Disjunktheit der drei Akteursmengen bedingt.

Mit der Explizierung von Struktur- und Verhaltensspezifikationen in Form von Systemregeln kann innerhalb des universellen Governance-, Integrations- und Operationsrahmens eine weitere Verfeinerung der formalisierten Machtstrukturen vorgenommen werden. Informelle Machtgefüge, die mit den formalen Strukturen wechselwirken, werden in der vorliegenden Arbeit nicht systematisch betrachtet.

- *Organisationsgrenzen:* Grundsätzlich sind die Grenzen einer Organisationseinheit im Modell klar ersichtlich, nämlich einfach in Form des Organisationseinheitsnetzes. Allerdings können zusätzlich die in Kapitel 4 diskutierten Kriterien für die Grenzziehung (Akteure, soziale Beziehungen, Aktivitäten) herangezogen werden. Werden zusätzlich zu der grundsätzlichen Aufteilung in Operation, Integration und Governance einer Organisationseinheit nur wenige weitere Spezifikationen in Form von Systemregeln vorgenommen, so ist vor allem das Kriterium der *Akteure* beziehungsweise der *Akteurszugehörigkeit* dominant. Die individuellen Charakteristika der einzelnen Akteure machen in diesem Fall die Organisationseinheit im Wesentlichen aus. Werden im Kontext der Systemregeln vor allem verfeinerte Strukturspezifikationen in Form zusätzlicher Beziehungen zwischen Positionen/Rollen vorgenommen, so dominiert das Kriterium der *sozialen Beziehungen*. Der Zusammenhalt der Organisationseinheit ergibt sich dann vor allem durch ein enges formales Strukturgeflecht zwischen Mitgliedspositionen, unabhängig von konkreten Inhabern der Positionen und davon, welche Positionen diese gegebenenfalls in anderen Organisationseinheiten besetzen. Werden im Kontext der Systemregeln hingegen vor allem Vorschriften und Standards für die Gestaltung von Aktivitäten vorgegeben, so dominiert das Kriterium der *Aktivitäten*. Eine Organisationseinheit ist dann vor allem dadurch charakterisiert, welche Aktivitäten möglich sind. Welche Akteure in diese Aktivitäten eingebunden sind und welche Abhängigkeiten zwischen diesen existieren, spielt eine untergeordnete Rolle.

- *Umwelt und Umgebungen:* Das Verhältnis einer Organisationseinheit zu ihren verschiedenen Umgebungen kann aufgrund der rekursiven Struktur eines Multiorganisationssystems systematisch und auf uniformer Basis behandelt werden. Ebenso wie eine fokussierte Organisationseinheit selber eine Umgebung in Form von Operations-, Integrations- und Governance-Aspekten aufspannt, so ist die Organisationseinheit im Kontext ihrer umschließenden Organisationseinheiten auch jeweils in mehrere solcher Kontexte eingebettet. Die Position einer Organisationseinheit in einer Umgebung ist immer die eines Operations-, Integrations- oder Governance-Akteurs (gegebenenfalls auch in mehrfacher und überlappender Art und Weise).

- *Grenzmanagement und kollektives Akteurswesen:* Auch das Grenzmanagement einer Organisationseinheit lässt sich aufgrund der rekursiven Struktur von Multiorganisationssystemen auf uniformer Basis studieren. In Kapitel 4 wurden organisatorische Umgebungen als Kombination aus Aufgabenumgebung und Institution betrachtet. Da jede Umge-

bung einer Organisationseinheit wiederum eine Organisationseinheit ist, lassen sich diese beiden Aspekte immer auf die gleiche Art und Weise identifizieren. Die Operationsakteure spannen im Wesentlichen die Aufgabenumgebung auf während die Integrations- und Governance-Akteure sowie die zusätzlichen Systemregeln im Wesentlichen die Institution aufspannen. Die Betrachtung, welcher Akteursmenge eine Organisationseinheit im Kontext einer umschließenden Organisationseinheit angehört, beantwortet weitestgehend die Frage, zu welchem Aspekt der Umgebung die Organisationseinheit selber beiträgt und welche externen Umgebungsaspekte für sie besonders relevant sind.

Welche Brückenschlags- und Pufferungsmechanismen auf Seiten einer Organisationseinheit etabliert werden, hängt von der konkreten Position im Kontext umschließender Organisationseinheiten ab. Da die Governance-Akteure im Wesentlichen für die Ausrichtung der Handlungen der Organisationseinheit an den grundlegenden Zielen/Strategien verantwortlich sind, etablieren sie auch in der Hauptsache die Brückenschlagsmechanismen, um die Position der Organisationseinheit in Relation zu externen Akteuren zu festigen und zu stärken. Die Integrationsakteure dagegen sind für den möglichst reibungslosen operativen Betrieb der Organisationseinheit verantwortlich. Pufferungsmechanismen dienen dabei als wichtige Schutzmaßnahme gegenüber Turbulenzen der Umgebung.

Welche Interaktionsmöglichkeiten eine Organisationseinheit im Kontext einer konkreten Umgebung hat, ist eindeutig durch ihre Zugehörigkeit zu bestimmten Akteursmengen (Operation, Integration, Governance) und durch zusätzliche Spezifikationen in Form von Systemregeln festgelegt. Die Informations- und Zuwendungsstruktur einer Organisationseinheit bestimmt dabei, welche der Interaktionsmöglichkeiten tatsächlich wahrgenommen werden und damit, welcher Teil der Umgebung in Betrieb genommen wird (Konzept des *enacted environment*). Die Informations- und Zuwendungsstruktur ist geprägt von den Zielen/Strategien der Organisationseinheit, den davon abgeleiteten Steuerungs- und Kontrollstrukturen sowie von den verfügbaren und durch Operationsakteure bereitgestellten Kernoperationen der Organisationseinheit. Die Informations- und Zuwendungsstruktur manifestiert sich in dauerhaften sowie wiederkehrenden Peripherieaktivitäten einer Organisationseinheit. Das Zustandekommen und das Wesen dieser Peripherieaktivitäten ergibt sich aus den Brückenschlags- und Pufferungsmechanismen der Organisationseinheit und geht somit auf Governance- und Integrations-Akteure zurück. Diese sind dabei abhängig von den Aktionsmöglichkeiten, die ihnen von Seiten der Operationsakteure zur Verfügung stehen.

- *Autonomie und Autarkie:* Autonomie lässt sich recht unmittelbar dadurch bestimmen, ob ein direkter Zugriff von außen in die Belange einer Organisationseinheit möglich ist. Betrachtet wurden in diesem Zusammenhang die Initiierung von Aktivitäten von außen, die Festlegung von Systemregeln von außen (insbesondere auf Governance-Regeln bezogen) und die erzwungene Umstrukturierung von Einbettungsbeziehungen von außen. Sind solche Eingriffe nicht möglich, so besteht zwar gegebenenfalls eine externe Beeinflussung und Einschränkung des Verhaltens einer Organisationseinheit, nicht jedoch eine unmittelbare Steuerung. Die Organisationseinheit ist dann autonom hinsichtlich der Entscheidungen über ihre Handlungsoptionen.

  Autarkie bezeichnet dabei nicht nur die Entscheidungsfreiheit hinsichtlich der Handlungs*optionen*, sondern welche und wie viele solcher Optionen gemessen an den grundsätzlichen Handlungsmöglichkeiten der Organisationseinheit überhaupt existieren. Die

Handlungsoptionen einer Organisationseinheit werden durch die Umgebung(en) bestimmt. Einschränkungen bestehen dabei durch die grundlegende Zugehörigkeit zu Operation, Integration oder Governance, durch zusätzliche Systemregeln und durch das Vorhandensein anderer Akteure, zu denen eine Dienstleistungsabhängigkeit besteht. Somit kann eine Organisationseinheit gegebenenfalls zwar autonom, aber nicht autark sein.

Insgesamt ergibt sich, dass das Grundmodell (die Grundmodelle) einer Organisationseinheit sowie das darauf aufbauende rekursive Konzept eines Multiorganisationssystems die Adressierung aller Fragestellungen aus Abschnitt 7.2 unmittelbar unterstützt und die Anforderungen mithin erfüllt werden.

9.4 Zusammenfassung

Während die Techniken zur Erstellung von abstrakten Modellen sowie die Grundmodelle des vorigen Kapitels gut dafür geeignet sind, einen schnellen und anschaulichen Zugang zu Situationen und Abläufen in Multiorganisationssystemen zu erlangen, so bedurften einige Konzepte und Mechanismen doch die Zuhilfenahme weiterer natürlichsprachlicher Erklärungen. Im vorliegenden Kapitel hingegen konnten mittels der Verwendung der Modellierungstechnik der Referenznetze detaillierte Modelle von Kollektivitätseinheiten und Organisationseinheiten erarbeitet werden, die eine direkte Operationalisierung der zuvor abstrakten Konzepte und Mechanismen beschreiben. Dies betrifft zum einen das fundamentale Konzept von Multiorganisationssystemen als rekursiv strukturierte Akteurssysteme. Zum anderen betrifft es das konkrete Zusammenspiel von Operation, Integration und Governance auf Ebene einer Organisationseinheit sowie als Kopplungsgrundlage zwischen Organisationseinheiten. Erst die operationalen Modelle dieses Kapitels erlauben die explizite Adressierung aller Aspekte einer Organisationseinheit inklusive der Behandlung elaborierter Systemregeln und der Evolution von Strukturen über die Zeit.

Das Ergebnis des Kapitels besteht in der Präsentation von mehreren operationalen Grundmodellen einer Organisationseinheit. Diese haben einen universellen Charakter und können als Basis für die Ausgestaltung unterschiedlichster konkreter Organisationseinheiten dienen. Auf Basis der Grundmodelle wurden die softwaretechnisch relevanten Kerneigenschaften einer Organisationseinheit identifiziert und demonstriert, dass die in Abschnitt 7.2 erstellten Anforderungen an das Modellierungsvorhaben des zweiten Teils der Arbeit erfüllt wurden.

Einerseits stellen die operationalen Modelle von Kollektivitätseinheiten und Organisationseinheiten technisch detaillierte *Modelle* für Strukturen und Mechanismen auf der logischen Ebene eines Multiorganisationssystems dar. Andererseits sind die Modelle reich genug, um im Kontext des Petrinetzsimulators Renew direkt ausführbar zu sein. Auf diese Weise stellen sie *softwaretechnische Prototypen* dar und können als konkretes Beispiel der technischen Realisierung von Multiorganisationssystemen auf Ebene der Laufzeitumgebung/Deployment-Plattformen verstanden werden.

Im folgenden, dritten Teil der Arbeit wird eine Anwendung der Grundmodelle vorgenommen, indem konkrete Beispiele von Organisationseinheiten betrachtet und ein konkreter Architekturvorschlag für Multiorganisationssysteme unterbreitet wird.

Teil III

Beispiele und Referenzarchitektur

10 Konkrete Organisationseinheiten

Der zweite Teil der vorliegenden Arbeit war der Vorstellung von Modellierungstechniken sowie der Erarbeitung von Grundmodellen von Organisationseinheiten gewidmet. Die Grundmodelle wurden bewusst derart gestaltet, dass sie einen universellen Charakter aufweisen. Sie sind anwendbar auf verschiedene konkrete Ausprägungen von Organisationseinheiten auf verschiedenen Ebenen eines rekursiv strukturierten Multiorganisationssystems. Die Betrachtung konkreter Ausprägungen wurde im zweiten Teil der Arbeit allerdings nicht vorgenommen. Diesem Thema ist das vorliegende Kapitel der Arbeit gewidmet. Ausprägungen von Organisationseinheiten basieren im Wesentlichen auf der Einbettung interner Akteure, der Definition von Aktivitätsmöglichkeiten sowie der Spezifikation spezieller Systemregeln.

Hier werden drei verschiedene Ausprägungsformen von Organisationseinheiten betrachtet. Dabei wird ein zweifaltiger Zweck verfolgt. (1) Zum einen wird die Tragfähigkeit der erarbeiteten (abstrakten sowie operationalen) Grundmodelle einer Organisationseinheit entlang verschiedener Paradigmen des Organisierens aufgezeigt. (2) Zum anderen dient dieses Kapitel der Vorbereitung auf das folgende Kapitel. Dort wird ein Architekturvorschlag für Multiorganisationssysteme unterbreitet. Er basiert auf der systematischen Unterscheidung verschiedener Formen von Organisationseinheiten und ihrer Kopplung.

Als Grundlage für die Betrachtung verschiedener Ausprägungen von Organisationseinheiten werden in diesem Kapitel drei der in Kapitel 5 behandelten organisationsorientierten Entwicklungsansätze für Multiagentensysteme betrachtet. Die Wahl ist dabei auf SONAR, $\mathcal{M}$OISE$^+$ und ISLANDER gefallen. Die drei Ansätze bieten jeweils eine Modellierungstechnik/-sprache für Agentenorganisationen/-institutionen an. Dabei bringen sie unterschiedliche Modellierungselemente mit sich und basieren auf unterschiedlichen Konzepten. Dementsprechend sind sie auch unterschiedlich gut zur Modellierung bestimmter organisatorischer Sachverhalte geeignet. Damit bieten die drei Ansätze eine gute Grundlage, die Tragfähigkeit des universellen Schemas einer Organisationseinheit zu belegen. In der Gegenrichtung dient das universelle Schema dazu, die drei Ansätze systematisch zu vergleichen.

Der Löwenanteil des Kapitels ist der Betrachtung des SONAR-Ansatzes in Abschnitt 10.2 gewidmet. Der Autor der vorliegenden Arbeit hat an mehreren Veröffentlichungen im Kontext des Ansatzes mitgewirkt und besitzt einen tiefen Einblick in die Thematik. Außerdem bietet SONAR als einziger der Ansätze inhärent Konzepte, die einen Übergang von der Betrachtung *einer* Organisationseinheit zur Betrachtung einer *Kopplung* von Organisationseinheiten nahe legen. $\mathcal{M}$OISE$^+$ und ISLANDER werden knapper in den Abschnitten 10.3 und 10.4 behandelt. Für jeden der Ansätze werden mögliche Ausprägungen des abstrakten sowie des operationalen Grundmodells einer Organisationseinheit betrachtet. Wie in Kapitel 9 betont, können die operationalen Modelle unmittelbar als ausführbare Prototypen verwendet werden. Diese Möglichkeit wurde im Falle von SONAR tatsächlich ausgeschöpft, so dass eine voll funktionsfähige SONAR-Organisationseinheit entwickelt wurde. Im Falle von $\mathcal{M}$OISE$^+$ und ISLANDER wurde zumindest eine eingeschränkte, punktuelle Prototypisierung vorgenommen.

In Abschnitt 10.6 erfolgt eine Zusammenfassung des Kapitels und eine Diskussion der Ergebnisse und eventueller Anschlusspunkte.

10.1 Vorbereitung

Die Ausgestaltung von Organisationseinheiten nach SONAR, $\mathcal{M}$OISE$^+$ und ISLANDER wird jeweils in mehreren Schritten vollzogen. Ausgehend von einer Konkretisierung der Akteurs- und Aktivitätsmengen im Kontext des abstrakten Modells einer Organisationseinheit werden verschiedene Aspekte in größerer Detailtiefe anhand einer Konkretisierung des operationalen Modells illustriert.

Der erste Schritt der Ausgestaltung besteht jeweils darin, die grundlegenden analytischen Aspekte der Operation, Integration und Governance zu identifizieren. Ein wesentlicher Beitrag dieses Kapitels besteht in der Untersuchung, inwiefern eine solche Identifizierung für verschiedene konkrete Organisationsmodelle möglich ist und in welcher Form sich die drei Aspekte jeweils manifestieren. Aufgrund des hochgenerischen Charakters der drei Aspekte gibt es gegebenenfalls mehrere Interpretationsmöglichkeiten.

Verknüpft mit der Identifizierung der Aspekte Operation, Integration und Governance ist die Konkretisierung von Systemregeln, die für die jeweilige Organisationseinheit gelten. Während das Konzept der Systemregeln bereits für das Verständnis des abstrakten Modells einer Organisationseinheit wichtig ist, so taucht es dort nicht als explizite Modellelemente auf. Für das operationale Modell einer Organisationseinheit nehmen Systemregeln eine noch zentralere Stellung ein, da sie hier als erstrangige Modellkonzepte auftauchen. In Teilabschnitt 9.2.3 wurde die Aufteilung von Systemregeln in Spezifikation und Zustandsbeschreibung propagiert. Das daraus resultierende Grundmodell einer Organisationseinheit ist in Abbildung 9.21 zu sehen (oder inklusive aller Anschriften in Abbildung A.3 im Anhang) und wird in diesem Kapitel zu Grunde gelegt. Hier wird eine noch feinere Aufteilung der Systemregeln betrachtet (die allerdings keine zusätzliche Erweiterung des Organisationseinheitsmodells bedingt).

- **Paradigmaregeln:** Bei diesen Regeln handelt es sich um die eigentlichen Regeln. Die drei in diesem Kapitel herangezogenen Ansätze der Organisationsmodellierung (SONAR, $\mathcal{M}$OISE$^+$, ISLANDER) können auch als verschiedene Paradigmen des Organisierens aufgefasst werden. Es geht weniger darum, unterschiedliche Möglichkeiten für die Modellierung derselben organisatorischen Sachverhalte vorzustellen. Stattdessen bringen die verschiedenen Ansätze auch eine unterschiedliche Eignung für die Modellierung bestimmter Sachverhalte mit sich. Die Paradigmaregeln kapseln genau die Kernkonzepte und -zusammenhänge des jeweiligen Ansatzes. Sie sind *unveränderlich*.

  Die Paradigmaregeln sind generisch und beziehen sich dabei auf Fakten, die eine konkrete Instanz einer Organisation entlang des jeweiligen Paradigmas beschreiben. Dabei werden zwei Typen solcher Fakten unterschieden.

- **Spezifikationsfakten:** Fakten dieses Typs beschreiben die Spezifikation einer konkreten Instanz einer Organisationseinheit entlang eines Paradigmas. Sie müssen im Einklang mit den jeweiligen Paradigmaregeln sein. Die Spezifikationsfakten können nur von Governance-Akteuren im Zuge von Governance-Aktivitäten modifiziert werden.

- **Zustandsfakten:** Diese Fakten beschreiben Zustände einer Organisationseinheit. Die Nützlichkeit solcher Zustandsbeschreibungen wurde in Teilabschnitt 9.2.3 thematisiert. Zustandsfakten ändern sich dementsprechend im Zuge von Effekten beliebiger Aktionen.

Die konkrete Formulierung und Realisierung von Systemregeln wurde in Abschnitt 9.2 weitestgehend offen gelassen. Für die in diesem Kapitel betrachteten Beispiele ist die Wahl auf eine Formulierung in PROLOG gefallen. Dafür sprechen einige Gründe.

- PROLOG bietet einen logischen Formalismus des Schließens über einer Regel- und Faktenmenge. Damit wird eine exakte Semantik für eine Menge von Regeln und deren Beziehungen gegeben. Dies erlaubt (bei geschickter Formulierung der Regeln) Anfragen an die Regelbasis, die recht komplexe Zusammenhänge betreffen.
- Eine Formulierung von Regeln auf deklarative Art und Weise stellt nach Ansicht des Autors der vorliegenden Arbeit eine intuitive und „natürliche“ Herangehensweise dar.
- Der Unifikationsmechanismus von PROLOG lässt sich sehr gut mit dem Unifikationsmechanismus für die Bindungssuche in Renew kombinieren.

Auf rein technischer Ebene lässt sich der Java-basierte PROLOG-Dialekt *tuProlog* (http://tuprolog.alice.unibo.it/) sehr bequem in Renew-Modelle einbinden. In Abschnitt 9.2.2 wurde lediglich vorgegeben, Systemregeln würden als Objektnetz, das die eigentlichen Regeln „wrappt“, in das Netz der Organisationseinheit eingebettet. Abbildung 10.1 zeigt die entsprechende Einbettung, die für die in diesem Kapitel beschriebenen Prototypen angewendet wird (Abbildung A.4 im Anhang zeigt das Systemregelnetz inklusive der Inititialisierungsdetails).

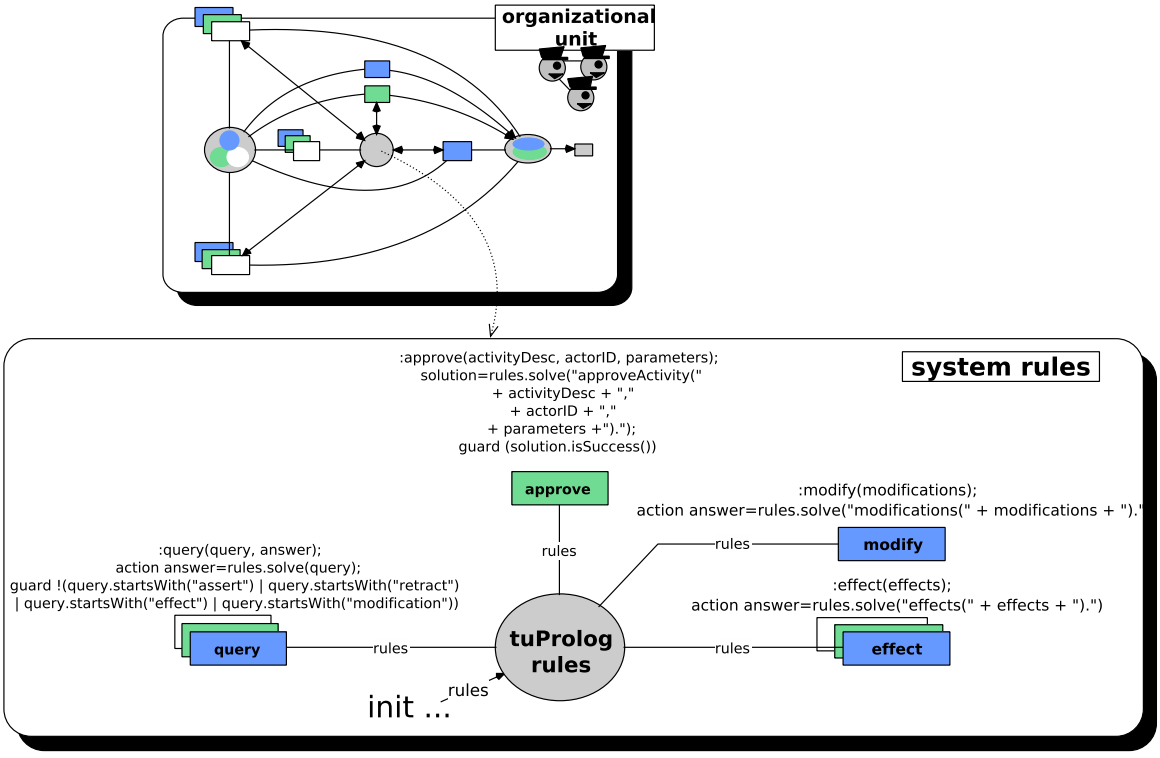

Abbildung 10.1: Einbettung einer tuProlog-Regelbasis

10.2 Organisationseinheiten nach SONAR

Köhler-Bußmeier [Köh06, KB09] stellt mit Sonar (**S**elf-**O**rganizing **N**et **Ar**chitecture) einen Ansatz für die Modellierung von Agentenorganisationen auf Basis von Petrinetzen vor. Bei ei-

nigen weiteren Arbeiten Köhler-Bußmeiers [KBWE09a, KBMWE09, KBWE09c, KBWE09b, KBWEM10] hat der Autor der vorliegenden Arbeit als Mitautor gewirkt. In den entsprechenden Artikeln geht es vornehmlich um Möglichkeiten der Operationalierung von SONAR-Modellen, zu einem geringeren Teil außerdem um Anpassungen und Erweiterungen des ursprünglichen SONAR-Formalismus.

10.2.1 Der SONAR-Ansatz: Grundlegende Konzepte und Modell-Dimensionen

Die petrinetzbasierten SONAR-Modelle haben eine exakte mathematische Semantik. Somit können Eigenschaften wie Wohlgeformtheit eines Organisationsmodells oder Bearbeitbarkeit von Aufgaben formal gefasst und verifiziert werden. Allerdings erfolgt die Vorstellung von SONAR in diesem Abschnitt ohne Rückgriff auf die formalen Definitionen. Stattdessen werden die Modellierungselemente und deren Verwendung anhand der grafischen Darstellungsweise für Petrinetze beschrieben. Insgesamt unterstützt SONAR die Modellierung struktureller, funktionaler und interaktionsbezogener Dimensionen einer Organisation. Das Laufbeispiel, welches hier herangezogen wird, ist eine Abwandlung des Beispiels, welches in den oben genannten Veröffentlichungen zu SONAR verwendet wird.

10.2.1.1 Delegationsnetze

Das Herzstück von SONAR bildet die Modellierung von Organisationen in Form von *Delegationsnetzen*. Abbildung 10.2 zeigt das ein Kooperationsszenario zwischen verschiedenen Parteien. Dabei gilt, dass es sich bei den Beschriftungen $p_1, ..., p_n$ und $t_1, ..., t_m$ einfach um Stellen-/Transitions*namen* handelt, während die weiteren Beschriftungen von Stellen und Transitionen eine inhaltliche Bedeutung für das SONAR-Modell haben.

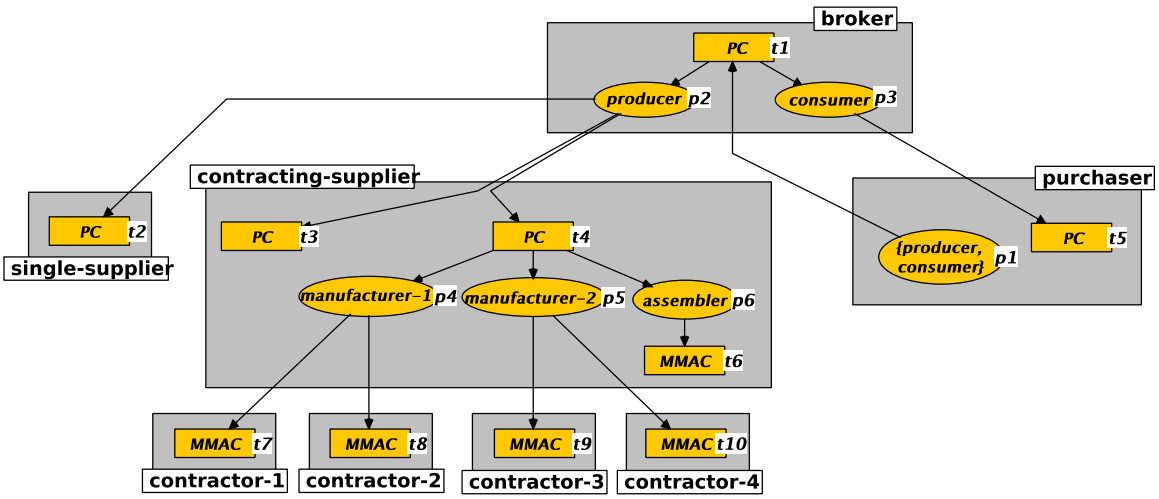

Abbildung 10.2: Eine Allianz als Delegationsnetz

In diesem Szenario kann ein Konsument (purchaser) den Erhalt eines bestimmten Produktes beantragen. Ein Vermittler (broker) stellt den Kontakt zu einem von zwei möglichen Anbietern (single-supplier, contracting-supplier) her. Beide Anbieter können das Produkt selber bereitstellen. Einer der beiden (contracting-supplier) hat die zusätzliche Option, nur einen Teil der

Produktion selber zu übernehmen und einen weiteren Teil an Sub-Unternehmer (contractor-1 – contractor-4) auszulagern. Das Szenario wird hier als eine Allianz bezeichnet da einerseits konkrete Kooperationsmuster bestehen, andererseits eine recht lose Kopplung der Teilnehmer vorliegt. Die Feinheiten von Delegationsnetzen werden in der Folge erläutert.

- Eine Stelle modelliert eine *Aufgabe* und eine Transition modelliert die *Bearbeitung* einer Aufgabe. Zu diesem Zweck hat jede Transition genau eine Stelle in ihrem Vorbereich (die zu bearbeitende Aufgabe) und jede Stelle hat mindestens eine Transition in ihrem Nachbereich (eine oder mehrere Möglichkeiten der Aufgabenbearbeitung).

- Die durch Transitionen modellierte Aufgabenbearbeitung kann zwei Formen annehmen. (1) Eine Transition mit leerem Nachbereich modelliert die *Ausführung* der Aufgabe. (2) Eine Transition mit nicht-leerem Nachbereich modelliert die *Verfeinerung/Aufteilung* einer Aufgabe: Die Aufgabe der Vorbereichsstelle wird aufgeteilt in die *Teilaufgaben* der Nachbereichsstellen.

- Stellen mit leerem Vorbereich modellieren *initiale Aufgaben*. Stellen, die im Vorbereich einer Transition mit leerem Nachbereich liegen modellieren *finale Aufgaben*.

- Das Netz wird in Form von *Positionen* (die grauen Rechtecke) partitioniert. An Positionen sind dementsprechend Verantwortlichkeiten für die Ausführung, Verfeinerung und Delegation bestimmter Aufgaben geknüpft. Dabei gilt: Falls eine Transition zu einer Position gehört, dann gehören auch die Stellen des Nachbereiches der Transition (falls vorhanden) zu der Position. Damit findet die Definition von Teilaufgaben zu einer Aufgaben immer lokal bei einer Position statt. Diese Regel führt dazu, dass Kanten von einer Position zu einer anderen Position immer Kanten von Stellen zu Transitionen sind. Damit modellieren solche Kanten die Delegation von Aufgaben zwischen Positionen.

  Positionen sind genau die Modellelemente, die im SONAR-Ansatz mit Agenten assoziiert sind (siehe weiter unten). In einem laufenden SONAR-Multiagentensystem ist jede Position mit genau einem Agenten besetzt. Allerdings kann ein Agent gegebenenfalls mehrere Positionen besetzen.

10.2.1.2 Rollen und Dienste

Bisher war nur sehr abstrakt die Rede davon, Stellen würden Aufgaben und Transitionen würden Aufgabenbearbeitungen modellieren. Ein genaueres Bild ergibt sich durch die Beschriftungen von Stellen und Transitionen. Stellen sind mit *Rollenbezeichnern* beschriftet und Transitionen sind mit *Dienstnetzen* beschriftet. Aufgaben entsprechen damit der Implementierung bestimmter Rollen. Konkretere Spezifikationen, wie Rollen zu implementieren sind, finden sich in den Dienstnetzen (die somit den zu erbringenden Dienst der Rollenimplementierung beschreiben). Ein Dienstnetz bezieht sich typischerweise auf mehrere Rollen und beschreibt deren Zusammenspiel in Form eines Interaktionsdiagramms. Rollen im SONAR-Kontext beziehen sich also auf spezifische Interaktionsanteile, die jeweils durch ein und denselben Akteur zu erbringen sind (analog zu den Aktivitätsrollen aus dem zweiten Teil der vorliegenden Arbeit).

Die Beziehung von Rollen und Dienstnetzen wird im Folgenden genauer beschrieben. Zu diesem Zweck zeigt Abbildung 10.3 die Dienstnetze PC und MMAC, mit denen die Transitionen des Delegationsnetzes aus Abbildung 10.2 beschriftet sind.

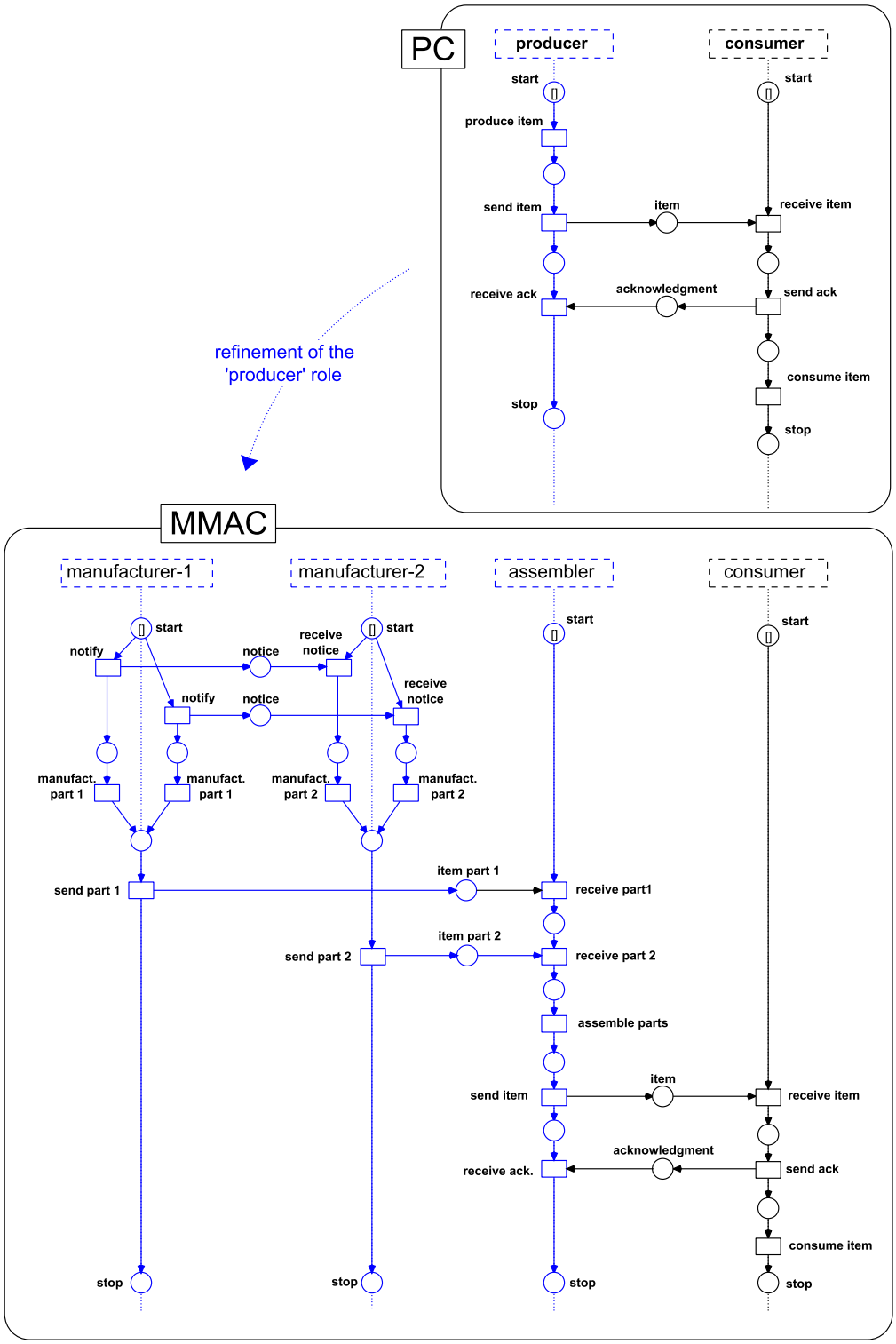

Abbildung 10.3: Die Dienstnetze PC und MMAC

1. Für jede Transition gilt, dass ihr jeweiliges Dienstnetz die Implementierung der Rollen der Vorbereichsstelle spezifiziert: Das Dienstnetz $\mathsf{D}(\mathsf{t})$ einer Transition t beschreibt ein Interaktionsprotokoll, das *mindestens* die Rollen $\mathsf{R}(\mathsf{p})$ der Vorbereichsstelle p umfasst. Dies lässt sich entsprechend für alle Transitionen und die mit ihnen assoziierten Dienstnetze des Beispiels nachprüfen. Insbesondere gilt auch für die initiale Stelle p_1, die mit einer Rollen*menge* beschriftet ist, dass die beiden in der Menge enthaltenen Rollen producer und consumer durch das Dienstnetz PC der Nachbereichstransition t_1 abgedeckt werden.

   Rollenspezifikationen sind für ein gesamtes SONAR-Modell konsistent. Dies gilt in dem Sinne, dass keine zwei Dienstnetze eine unterschiedliche Spezifikation der gleichen Rolle vornehmen.

2. Für alle Stellen im Nachbereich derselben Transition gilt: Alle Transitionen im Nachbereich dieser Stellen sind mit dem gleichen Dienstnetz beschriftet. Dies sichert zu, dass die jeweiligen Rollenspezifikationen zusammenpassen.

3. Der wichtigste Aspekt ist die *Verfeinerungsbeziehung*, die zwischen verschiedenen Dienstnetzen besteht. Verfeinerung von Dienstnetzen geht einher mit Verfeinerung/Aufteilung von Rollen. Dieser Fall tritt im Beispiel bei Transition t_4 auf. Die Vorbereichsstelle p_2 ist mit der Rolle producer beschriftet. Dementsprechend ist t_4 mit dem Dienstnetz PC aus Abbildung 10.3 assoziiert da dieses Dienstnetz die Rolle producer spezifiziert. Gleichzeitig wird an dieser Transition die Rolle producer aufgeteilt in die Teilrollen manufacturer-1, manufacturer-2 und assembler der Nachbereichs-Stellen $\mathsf{p}_4, \mathsf{p}_5$ und p_6. Die Nachbereichstransitionen dieser Stellen müssen nun entsprechend der Punkte (1.) und (2.) alle mit dem gleichen Dienstnetz beschriftet sein und dieses Dienstnetz muss eine Implementierung all der neu eingeführten Teilrollen beinhalten. Es handelt sich hier um das Dienstnetz MMAC aus Abbildung 10.3. An dieser Stelle muss nun zusätzlich gelten, dass sich die Rollenverfeinerung auf Ebene des Delegationsnetzes auf die Ebene der Dienstnetze überträgt. Konkret bedeutet das hier: Das Dienstnetz MMAC muss eine Verfeinerung des PC-Dienstnetzes sein, indem die Rolle producer durch die Rollen manufacturer-1, manufacturer-2 und assembler verfeinert wird. Die Verfeinerung muss derart sein, dass sich für die an der Verfeinerung unbeteiligte Rolle consumer hinsichtlich des Kommunikationsverhaltens nichts ändert. Die Anordnung der Dienstnetze PC und MMAC in Abbildung 10.3 illustriert, dass dies tatsächlich der Fall ist.

Um in der Folge des Abschnittes – wenn es der Klarheit dient – zwischen Aufgaben (auf Ebene der Delegationsnetze) und Rollen (auf Ebene der Dienstnetze) zu unterscheiden, wird für die Bezeichnung von Aufgaben die Mengendarstellung für Rollen (beispielsweise {producer, consumer} oder {manufacturer-1}) verwendet.

Die drei genannten Punkte sichern einen konsistenten Zusammenhang zwischen Rollenverfeinerung/-aufteilung und Dienstnetzverfeinerung. Insbesondere besteht jederzeit die Möglichkeit, ein bestehendes SONAR-Modell punktuell zu erweitern, ohne umfassende Änderungen am Gesamtmodell vornehmen zu müssen.

Beispielsweise ist es für das Modell aus Abbildung 10.2 möglich, ausgehend von der Position contractor-2 die Rolle manufacturer-1 weiter zu verfeinern. Eine solche Verfeinerung würde dementsprechend die Einführung eines neuen Dienstnetzes nach sich ziehen. Dieses Dienstnetz müsste eine Verfeinerung des Dienstnetzes MMAC hinsichtlich der Rolle manufacturer-1

sein. Für alle anderen Rollen aus MMAC (manufacturer-2, assembler, consumer) müsste das Kommunikationsverhalten unverändert bleiben.

Ein weiteres Beispiel besteht darin, ausgehend von der Position purchaser die Rolle consumer zu verfeinern. Damit müsste auch ein neues Dienstnetz eingeführt werden. Es müsste eine Verfeinerung des Dienstnetzes PC hinsichtlich der Rolle consumer sein und das Kommunikations-Verhalten der Rolle producer unverändert lassen. Damit würde der verfeinerte consumer-Anteil auf jeden Fall zu dem verfeinerten producer-Anteil gemäß MMAC „passen".

Insgesamt gilt, dass Verfeinerungen von Rollen und assoziierten Dienstnetzen *lokal* vollzogen werden können, ohne die Wohlgeformtheit des Gesamtmodells in Frage zu stellen.

10.2.1.3 Teams

Organisationsstrukturen auf Basis von Delegationsbeziehungen ließen sich auch mit traditionellen UND/ODER-Graphen modellieren. Die Modellierung mit Petrinetzen bringt allerdings den Vorteil der operationalen Semantik mit sich.

Weiter oben wurden bereits die Begriffe der initialen und finalen Aufgaben eingeführt. Initiale Aufgaben sind der Startpunkt von Aufgabenbearbeitungen. Die Bearbeitung einer initialen Aufgabe führt gegebenenfalls zu mehreren Verfeinerungen/Aufteilungen bis schließlich eine Aufteilung in Form von finalen Aufgaben erfolgt ist. Dabei gibt es typischerweise mehrere Möglichkeiten bezüglich der Verfeinerung. Jede dieser Möglichkeiten bringt ein eigenes *Team* an beteiligten Positionen mit sich.

Dies lässt sich auch auf das Petrinetz-Markenspiel zurückführen: Legt man eine anonyme Marke auf die Stelle einer initialen Aufgabe, so induzieren die möglichen Schaltvorgänge die möglichen Teams zur Bearbeitung dieser Aufgabe. Zu jedem Team gehört ein *Teamnetz* als Teilnetz des gesamten Delegationsnetzes. Abbildung 10.4 zeigt alle möglichen Teams/Teamnetze für die (hier einzige) initiale Aufgabe (Stelle p_1) des Delegationsnetzes aus Abbildung 10.2.[1]

Ein Teamnetz modelliert eindeutig, welche Positionen des Teams für die Ausführung der finalen Aufgaben verantwortlich sind. Eine Spezifikation dafür, wie diese Aufgaben auszuführen sind, ergibt sich aus dem eindeutigen *Teamdienstnetz*, mit dem jedes Teamnetz assoziiert ist. Das Teamdienstnetz ergibt sich aus der Fusion aller Dienstnetze, die zu Transitionen mit leerem Nachbereich des Teamnetzes gehören (denn diese modellieren gerade die Ausführung der finalen Aufgaben). Bei dieser Fusion werden die jeweils maximal verfeinerten Rollenanteile der Dienstnetze zusammengefügt.[2] Der konsistente Zusammenhang zwischen Rollenverfeinerung und Dienstnetzverfeinerung sichert zu, dass ein solches Zusammenfügen immer möglich ist. Dienstnetze und darauf aufbauende Teamdienstnetze können gegebenenfalls mehrere Möglichkeiten der Ausführung beinhalten. Beispielsweise lassen die beiden Rollendefinitionen für manufacturer-1 und manufacturer-2 aus dem Dienstnetz MMAC verschiedene Ausführungspfade für die Herstellung der beiden Produktanteile zu. Weitere Variationen ergeben sich natürlich schlicht durch unterschiedliche Datenwerte gefärbter Marken eines Teamdienstnetzes. Ein konkreter *Teamplan* ergibt sich, wenn ein Teamdienstnetz mit zusätzlichen Einschränkungen

[1] Um genau zu sein, handelt es sich bei den Teamnetzen um die *Petrinetzprozesse* des Delegationsnetzes. Petrinetzprozesse (selber Petrinetze der Klasse der *Kausalnetze*) sind eine Möglichkeit, das Verhalten eines Petrinetzes in Form seiner Schaltvorgänge zu beschreiben (vergleiche Goltz und Reisig [GR83]).

[2] In dem einfachen Sonar-Beispiel der Abbildungen 10.2 und 10.3 sind die Dienstnetze PC und MMAC auch gleichzeitig die beiden möglichen Teamdienstnetze. Eine Fusion mehrerer Dienstnetze ist nicht notwendig. Diese Notwendigkeit würde beispielsweise entstehen, falls die Rolle consumer weiter verfeinert werden würde.

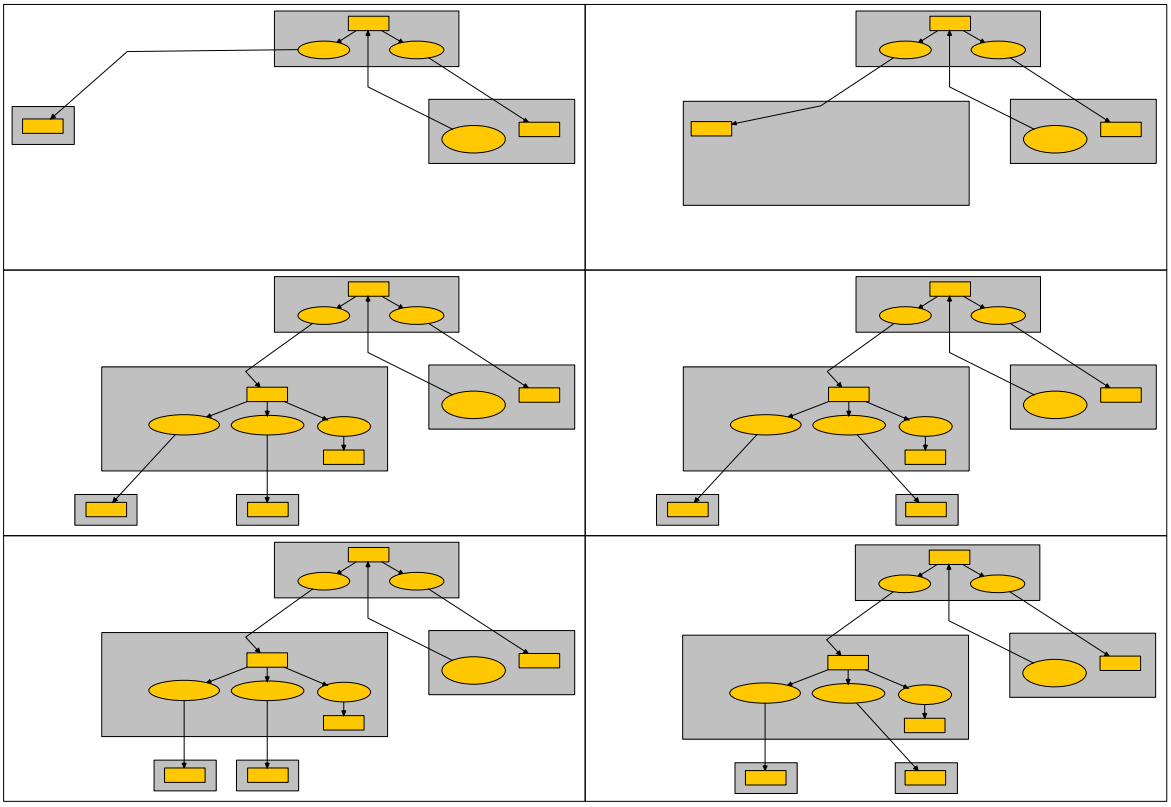

Abbildung 10.4: Mögliche Teams zum Delegationsnetz aus Abbildung 10.2

bezüglich der erlaubten (beziehungsweise geforderten) Markierungen assoziiert ist.

10.2.1.4 Stratifizierte SONAR-Modelle

In Abschnitt 5.2 wurde der Unterschied zwischen endogener und exogener Reorganisation einer Agentenorganisation hervorgehoben. Endogene Reorganisation bezieht sich darauf, dass die Reorganisation aus dem Multiagentensystem heraus geschieht, ohne weiteren Einfluss externer Entwickler. Der SONAR-Ansatz sieht die Möglichkeit der endogenen Reorganisation vor. Diesem Zweck dienen *stratifizierte* SONAR-Modelle. Ein stratifiziertes SONAR-Modell ist in mehrere Ebenen unterteilt. Es handelt sich hier allerdings nicht um Systemebenen nach dem Verständnis, das im bisherigen Verlauf der vorliegenden Arbeit angewandt wurde. Systemebenen wurden bislang im Sinne von Abstraktionsebenen eines Systems verstanden, als eine Möglichkeit der Betrachtung von Akteuren verschiedener Granularität. Im Falle stratifizierter SONAR-Modelle hingegen finden sich auf allen Ebenen die gleichen Positionen wieder (in einem laufenden SONAR-Multiagentensystem werden sie somit auch mit den jeweils gleichen Agenten besetzt). Allerdings unterscheiden sich die Delegationsnetze der verschiedenen Ebenen. Die Intention dahinter ist, dass Handlungen einer SONAR-Ebene n Transformationen der SONAR-Ebenen $< n$ bedingen. Abbildung 10.5 zeigt ein Beispiel eines stratifizierten SONAR-Modells, von dem die beiden ersten Ebenen explizit aufgeführt sind. s

Auf Ebene 1 handelt es sich um das bisher betrachtete SONAR-Modell. Das SONAR-Modell

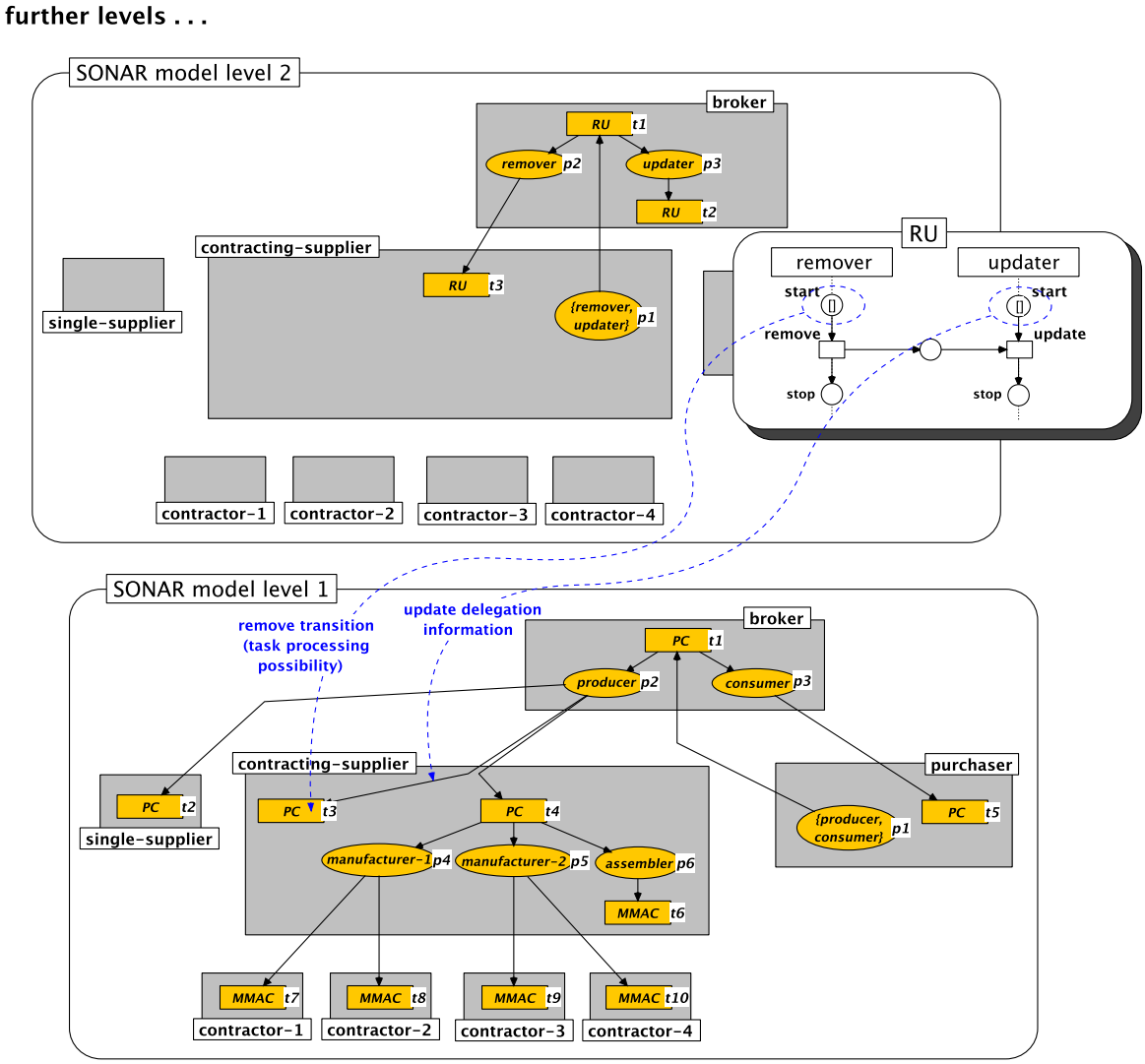

Abbildung 10.5: Organisatorische Transformationen über stratifizierte SONAR-Modelle

auf Ebene 2 hat die gleichen Positionen, enthält aber lediglich ein sehr einfaches Delegationsnetz, das nur zwei Positionen tatsächlich einbezieht. Das einzig mögliche Team umfasst dementsprechend auch nur diese beiden Positionen als Mitglieder. Das einzige Dienstnetz der Ebene 2 beschreibt eine Interaktion, bei der eine Transition des Delegationsnetzes der Ebene 1 gelöscht wird. Stratifizierte SONAR-Modelle können grundsätzlich beliebig viele Ebenen umfassen. Jede Ebene entspricht einer *Meta-Organisation* für darunter liegende Ebenen.

10.2.1.5 Eine SONAR-Organisation in Aktion

Weiter oben wurde bereits erwähnt, dass für das Multiagentensystem-Deployment eines SONAR-Modells jede Position mit genau einem Agenten besetzt wird. Dabei wird allerdings noch eine weitere Trennung vorgenommen. Deployment von SONAR-Modellen folgt dem in

Abschnitt 5.2 beschriebenen Prinzip, Organisationen in Multiagentensystemen in Form von Middleware-Schichten explizit zu vergegenständlichen. Damit ist ein Organisationsmodell nicht nur ein Entwurfs-, sondern zusätzlich ein Laufzeitartefakt. Das Middleware-Konzept MULAN4SONAR (vergleiche auch die Abschnitte 5.2 und 6.2) sieht eine Middleware-Schicht in Form von *Positionsagenten* vor. Jede Position des unterliegenden SONAR-Modells resultiert in einem eigenen Positionsagenten. Diese Agenten sind keine Domänenagenten im Sinne von Mitgliedern, sondern organisatorische Artefakte. Gemeinsam verkörpern die Positionsagenten den formalen Anteil der SONAR-Organisation im laufenden System. Jeder Positionsagent kapselt dabei die Spezifikation der Position, die er repräsentiert.

Um diese Spezifikationen auch mit Leben zu füllen, müssen Domänenagenten als Mitglieder der SONAR-Organisation ins Spiel kommen. Zu jeder Position muss es einen *Mitgliedsagenten* geben, der die jeweilige Position einnimmt. Der Positionsagent repräsentiert den organisatorischen Kontext des Mitgliedsagenten und beschränkt/leitet dessen organisationsbezogene Handlungen. Die Dynamik einer SONAR-Organisation ergibt sich in der Folge aus Interaktionen zwischen Positions- und Mitgliedsagenten sowie zwischen Positionsagenten untereinander. Basierend auf diesen Erläuterungen zeigt Abbildung 10.6 ein schematisches Deployment von SONAR-Organisationen.[3]

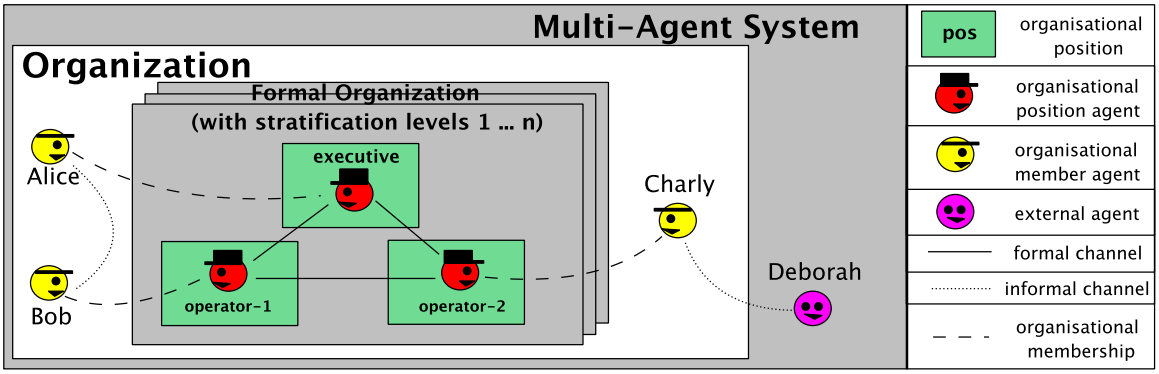

Abbildung 10.6: MULAN4SONAR Middleware-Konzept

Die Dynamik von MULAN4SONAR-Organisationen ergibt sich im Wesentlichen aus der Beschreibung der Modellierungsdimensionen von SONAR (Delegationsnetze, Dienstnetze, Teamnetze, Teamdienstnetze, Teampläne).

- *Teamformation:* Ausgehend vom Auftreten initialer Aufgaben werden Teams geformt.
- *Teamplanformation:* Jedes Team ist mit einem Teamdienstnetz assoziiert. Dieses muss gegebenenfalls zu einem Teamplan konkretisiert werden.
- *Teamplanausführung:* Die Positionen des Teams (beziehungsweise die ihnen zugewiesenen Agenten), die für die Ausführung der finalen Aufgaben zuständig sind, führen den Teamplan aus.

[3] Wie die Kopplung zwischen einem Positions- und einem Mitgliedsagenten zustande kommt (das Ausschreiben und die Besetzung der Position), ist nicht Teil der SONAR-Spezifikation.

Dabei ist zu beachten, dass diese Dynamik auf gegebenenfalls mehreren Ebenen eines stratifizierten SONAR-Modells auftritt. In der Deployment-Abbildung 10.6 sind die verschiedenen Ebenen eines stratifizierten SONAR-Modells angedeutet. Da die gleichen Positionen auf allen Ebenen wieder auftauchen, sind sie auch mit dem jeweils gleichen von Pärchen von Positions- und Mitgliedsagent assoziiert. Auf SONAR-Ebene 1 dienen die drei soeben beschriebenen Teamworkschritte der Erbringung „normaler Geschäftsfunktionen“. Für alle SONAR-Ebenen darüber gilt, dass Teamwork auf Transformationen von darunter liegenden SONAR-Ebenen abzielt: Formation eines Transformationsteams, Formation eines Transformationsteamplans, Ausführung des Transformationsteamplans.

10.2.2 Ausgestaltung von Organisationseinheiten gemäß SONAR

Für die Ausgestaltung von Organisationseinheiten nach SONAR wird den allgemeinen Vorbemerkungen aus Abschnitt 10.1 gefolgt. Ausgehend von einem abstrakten Grobmodell einer SONAR-Organisationseinheit werden verschiedene Aspekte in größerer Detailtiefe anhand eines prototypisch umgesetzten operationalen/ausführbaren Modells einer SONAR-Organisationseinheit illustriert.

10.2.2.1 Operation, Integration und Governance

Als ersten Schritt für die Ausgestaltung gilt es, die grundlegenden analytischen Aspekte der Operation, Integration und Governance zu identifizieren. Für den SONAR-Ansatz bietet sich nach Ansicht des Autors der vorliegenden Arbeit die in Abbildung 10.7 illustrierte Interpretation an. Der Aspekt des Grenzmanagements über Peripherieaktivitäten ist hier zunächst aus-

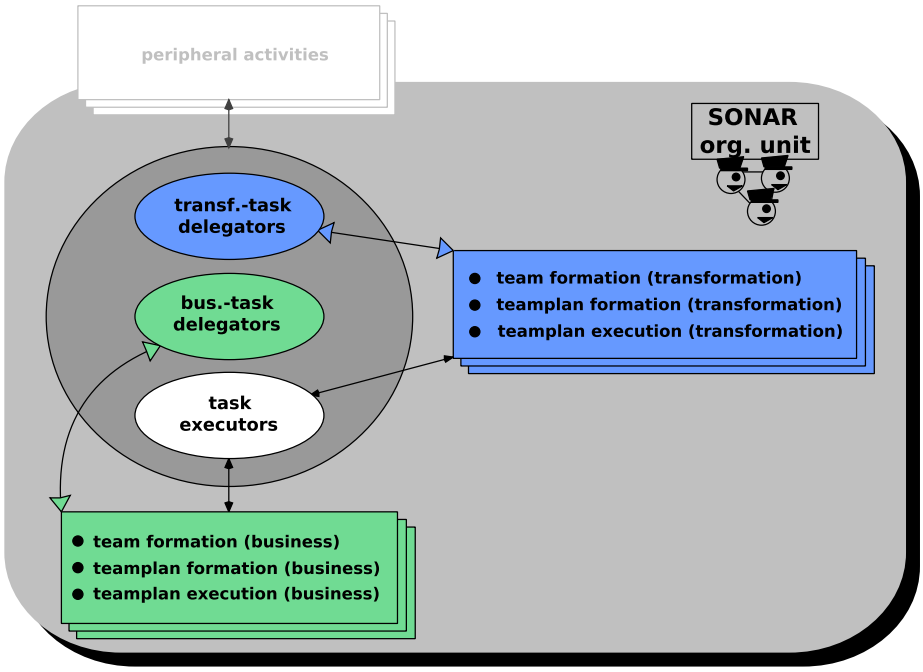

Abbildung 10.7: Erstes Grobmodell einer SONAR-Organisationseinheit

gegraut, da die ursprünglichen SONAR-Spezifikationen nichts über kollektives Akteurswesen aussagen. Allerdings wird dieser Aspekt weiter unten in diesem Teilabschnitt als Erweiterung thematisiert.

- Integrationsaktivitäten entsprechen der Teamformation, Teamplanformation und Teamplanausführung entlang der Regeln, die aus der geltenden Organisationsspezifikation folgen. Als Integrationsakteure, die diese Aktivitäten bereitstellen und kontrollieren, fungieren dabei die Akteure von Positionen, von denen ausgehend Aufgaben delegiert werden (business-task delegators): Diese Akteure sorgen für eine (regelkonforme) iterierte Aufteilung und Delegation von Aufgaben, die Bekanntmachung des so entstehenden Teams und für das Bereitstellen eines Teamplans.

- Analog entsprechen Governance-Aktivitäten den Teamaktivitäten, die zu Transformationen von Organisationsspezifikationen und den daraus folgenden Regeln für Integration und Operation führen. Bei den Governance-Akteuren handelt es sich hier um die Akteure von Positionen, von denen ausgehend Transformationsaufgaben delegiert werden (transformation-task delegators).

- Bei operationalen Akteuren handelt es sich um die Akteure von Positionen, bei denen finale Aufgaben ausgeführt werden (task executors). Sie übernehmen Rollen in Teamplänen. Ob es sich bei den finalen Aufgaben um Geschäfts- oder Transformations-Aufgaben handelt, spielt dabei keine Rolle.

Dabei gilt, dass sich alle Akteursmengen überlappen können. Beispielsweise gilt für das in Abbildung 10.2 gezeigte Modell einer SONAR-Organisation, dass Akteure der Positionen purchaser und contracting-supplier sowohl Aufgaben delegieren als auch Aufgaben ausführen und somit sowohl Integrations- als auch Operationsakteure sind. Darüber hinaus kann die Menge von Governance-Akteuren mit den beiden Akteursmengen für Operation und Integration überlappen. Dies folgt aus dem Konzept der Stratifizierung von SONAR-Modellen, bei denen die gleichen Positionen auf allen Ebenen auftauchen und auch mit den gleichen Akteuren besetzt werden. Allerdings fällt an dieser Stelle auf, dass das Modell aus Abbildung 10.7 dem Konzept der Stratifizierung nicht gerecht wird. Es umfasst eigentlich nur zwei Stratifikationsebenen. Daher erfolgt weiter unten eine elaboriertere Betrachtung des Aspektes der Governance für SONAR-Organisationseinheiten.[4]

10.2.2.2 Systemregeln einer Sonar-Organisationseinheit

In der Folge wird die Funktionsweise einer SONAR-Organisationseinheit detaillierter betrachtet. Zu diesem Zweck wird das operationale Grundmodell einer Organisationseinheit herangezogen und konkretisiert. Im Falle von SONAR hat diese Konkretisierung zu einem ausführbaren und voll funktionsfähigen Prototypen einer SONAR-Organisationseinheit geführt.

Besonders wichtig ist dabei die geeignete Formulierung von Systemregeln. In Abschnitt 10.1 ist die allgemeine Herangehensweise erläutert, der in diesem Kapitel gefolgt wird: Systemregeln werden aufgeteilt in Paradigmaregeln, Spezifikationsfakten sowie Zustandsfakten und werden als PROLOG-Regelbasis in das operationale Modell einer Organisationseinheit eingebettet.

[4] Allerdings wurden in allen Diskussionen des Autors der vorliegenden Arbeit mit Michael Köhler-Bußmeier ohnehin nur Szenarien mit zwei Stratifizierungsebenen betrachtet, so dass die Abbildung 10.7 in diesem Sinne angemessen ist.

Für den Fall einer konkreten SONAR-Organisationseinheit zeigt Abbildung 10.8 den Schnappschuss einer PROLOG-Regelbasis. Dabei ist lediglich ein Auszug zu sehen. Die verschiedenen Typen von Regeln und Fakten sind getrennt aufgeführt.

Die *Spezifikationsfakten* beziehen sich auf das bisher betrachtete SONAR-Beispiel aus den Abbildungen 10.2 und 10.3. Spezifiziert sind mögliche Aufteilungen von Aufgaben in Teilaufgaben, die Verantwortlichkeiten von Positionen bezüglich der Delegation beziehungsweise Ausführung von Aufgaben und die Assoziation von Mengen finaler Aufgaben mit Teamdienstnetzen.[5] Als *Zustandsfakten* sind zum einen die Besetzungen von Positionen mit Akteuren aufgeführt. Zum anderen wird der Verlauf von Teamformationen verfolgt und zwischenzeitliche Zustände gespeichert. Der Teil der *Paradigmaregeln* ist ziemlich umfangreich und in der Abbildung ist nur ein kleiner Auszug zu sehen. Zunächst ist festgehalten, welche Änderungen der Regelbasis im Zuge von Effekten (nur Änderungen von Zustandsfakten) und Modifikationen (nur Änderungen von Spezifikationsfakten) auftreten können. Zusätzlich zeigt die Abbildung die Regeln, die die Initiierung von Aktivitäten betreffen. Betrachtet werden hier die Initiierung von Delegations- und Teamplanaktivitäten. Dabei wird deutlich, dass diese Regeln auf verschiedenen weiteren Paradigmaregeln fußen, die hier nicht aufgeführt sind. Näheres bezüglich des Zusammenspiels von Systemregeln und Aktivitätsinitiierungen wird weiter unten anhand eines Beispiels erläutert.

10.2.2.3 Zusammenspiel von Systemregeln, Integration und Operation

Auf Basis der Erläuterungen zu Systemregeln und deren Einbettung in den Kontext einer Organisationseinheit lässt sich das Zusammenspiel von Integration und Operation nun genauer beschreiben. Abbildung 10.9 illustriert die Umsetzung der SONAR-Allianz aus Abbildung 10.2 als SONAR-Organisationseinheit. Dabei werden hier exemplarisch alle Vorgänge für die Bearbeitung einer Instanz der initialen Aufgabe {producer, consumer} gezeigt. Auf der linken Seite der Abbildung sind alle beteiligten Akteure aufgeführt, auf der rechten Seite alle erforderlichen Aktivitäten und in der unteren Mitte ein Auszug der konsultierten Systemregeln. Für die Akteure sind die Positionen, die sie im Rahmen der SONAR-Organisation einnehmen, in Klammern aufgeführt. Die für das Deployment einer SONAR-Organisation beschriebene Trennung in Positions- und Mitgliedsagenten ist hier nicht notwendig. Die Verkörperung des Formalanteils einer SONAR-Organisation wird vom Organisationseinheitsnetz und den Systemregeln übernommen. Entsprechend der weiter oben vorgenommenen Identifizierung der Aspekte der Integration und Operation nehmen einige der Akteure eine doppelte Funktion ein. Die explizite Annotation von Zeitpunkten stellt den Zusammenhang zwischen den Bereichen der Abbildung her und dienen in der Folge als Anhaltspunkte für die Erläuterungen.

1. *Zeitpunkte t_1 – t_3:* Für die Bearbeitung der Aufgabe {producer, consumer} ist die Bildung eines der Teams aus Abbildung 10.4 notwendig. Ein solches Team wird durch sukzessive Delegation von (Teil-)Aufgaben formiert. Jede Delegation einer Aufgabe geschieht im Verlauf einer eigenen Aktivität. Diese Aktivitäten werden von den Integrationsakteuren, die für die Delegation der jeweiligen Aufgabe verantwortlich sind, initiiert. Dabei werden

[5] Im aktuellen Prototypen werden Verfeinerungsbeziehungen zwischen Dienstnetzen nicht durch Systemregeln erfasst. Daher findet hier eine fixe Assoziation von Mengen finaler Aufgaben mit Teamdienstnetzen statt. Allerdings arbeitet der Autor der vorliegenden Arbeit aktuell gemeinsam mit Michael Köhler-Bußmeier an einer Überführung der Petrinetzdarstellung von Dienstnetzen in PROLOG-Notation, um zukünftig über Dienstnetzspezifikationen räsonieren zu können.

```
/* specification facts: wrapped in meta-predicate spec/1 */
/* subtasks(Position, Task, Subtasks) */
spec(subtasks(broker, producer_consumer, [producer, consumer])).
spec(subtasks(contracting_supplier, producer, [manufacturer_1, manufacturer_2, assembler])).
  ...
/* delegates(FromPosition, Task, ToPosition) */
spec(delegates(purchaser, producer_consumer, broker)).
spec(delegates(broker, producer, single_supplier)).
  ...
/* executes(Position, Task) */
spec(executes(single_supplier, producer)).
spec(executes(contracting_supplier, producer)).
  ...
/* hasTeamServiceNet(FinalTasks, TeamServiceNet) */
spec(hasTeamServiceNet([producer, consumer], activity_teamServiceNet_pc)).
spec(hasTeamServiceNet([manufacturer_1, manufacturer_2, assembler, consumer],
        activity_teamServiceNet_mmac)).
  ...

/* state facts: wrapped in meta-predicate state/1 */
/* occupiesPosition(Agent, Position) */
state(occupiesPosition(bob_alliance, purchaser)).
state(occupiesPosition(alice_alliance, broker)).
  ...
/* intermediateTeam(TeamID, TeamStructure) */
state(intermediateTeam([3, alliance], [[[], purchaser], []])).
  ...

/* paradigm rules */
/* effects */
effects([]).
effects([Effect|Effects]) :- effect(Effect), effects(Effects).

effect(delegationOccured(TeamID, Delegator, Receiver, Task)) :- ...
effect(teamPlanStarted(TeamID)) :- ...
  ...
/* modifications */
modifications([]).
modifications([Modification|Modifications]) :- modification(Modification), modifications(Modifications

modification(removeDelegationChannel(Delegator, Receiver, Task)) :- ...
modification(addDelegationChannel(Delegator, Receiver, Task)) :- ...
  ...
/* approval of activities */
approveActivity(ActivityName, Initiator, Parameters) :-
        activityType(ActivityName, ActivityType),
        approveActivity(ActivityType, ActivityName, Initiator, Parameters).

approveActivity(delegate, ActivityName, Initiator, Delegation) :-
        currentlyActivatedDelegation(Initiator, Delegation),
        delegationMode(ActivityName, DelegationMode),
        validDelegation(Initiator, Delegation, DelegationMode).

approveActivity(teamPlan, Initiator, ActivityName, [TeamID, TeamPlanConstraints]) :-
         currentlyActivatedTeam(TeamID, Team),
         headOfTeam(Initiator, Team),
         hasTeamServiceNet(Team, ActivityName),
         validTeamPlanConstraints(TeamID, TeamPlanConstraints).
  ...
```

Abbildung 10.8: Fragment der tuProlog-Regelbasis einer SONAR-Organisationseinheit

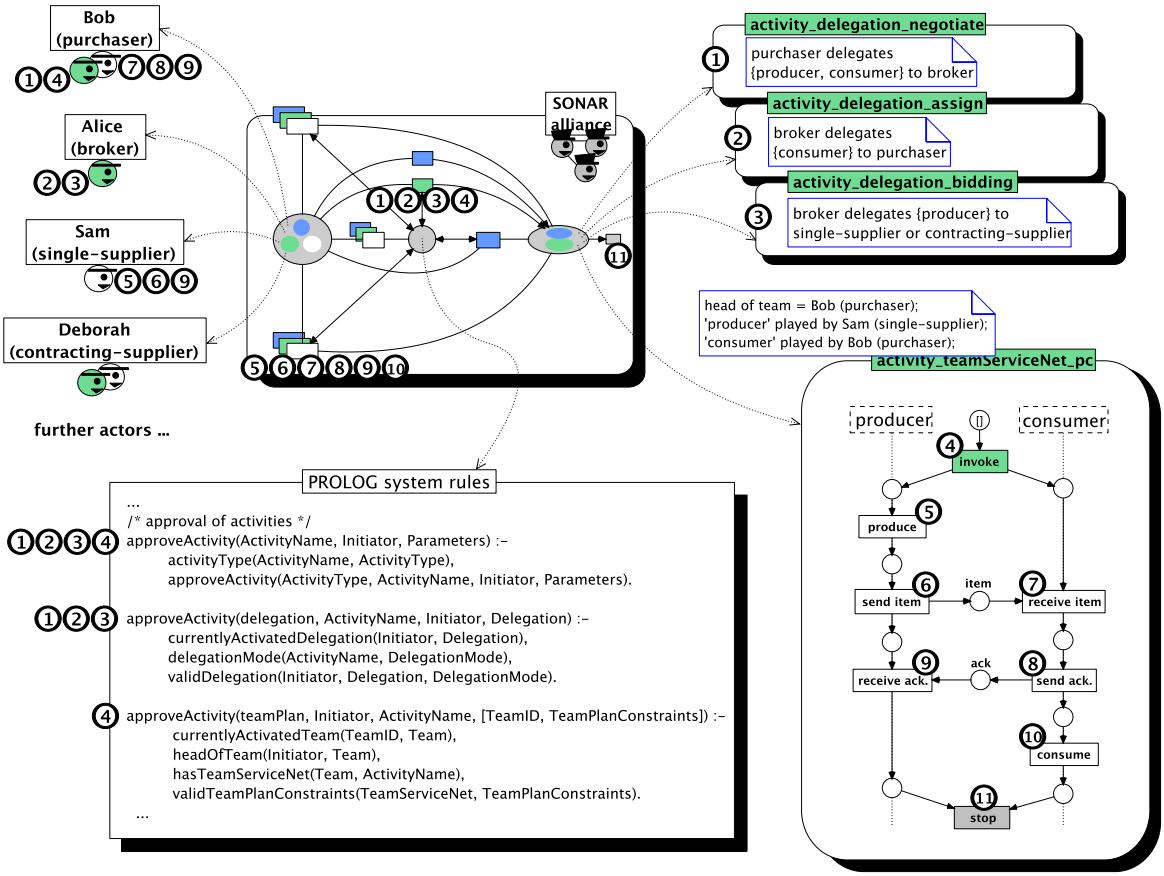

Abbildung 10.9: Bearbeitung der initialen Aufgabe {producer, consumer} aus Abbildung 10.2

die Systemregeln zur Überprüfung mehrerer Bedingungen herangezogen: (1) Der Delegierende hat die Berechtigung, die Aufgabe an den Delegationsempfänger zu delegieren; (2) Der Delegationsmodus ist zulässig[6] ; (3) Die Aufgabe (genauer, die Aufgabeninstanz) ist im Kontext einer laufenden Team-Formation entstanden und ist noch offen.

Im Zuge des exemplarischen Ablaufes aus Abbildung 10.9 aktiviert Bob (purchaser) zum Zeitpunkt t_1 eine Aktivität im Zuge derer er die Delegation der Aufgabe {producer, consumer} mit Alice (broker) verhandelt. Zum Zeitpunkt t_2 aktiviert Alice eine Aktivität, im Zuge derer sie die Teilaufgabe consumer direkt Bob zuweist. Zum Zeitpunkt t_3 aktiviert Alice eine Aktivität, im Zuge derer die beiden Akteure Sam (single-supplier) und Deborah (contracting-supplier) auf die Zuweisung der Teilaufgabe producer bieten können. In gezeigten Fall erhält Sam den Zuschlag für die Teilaufgabe und es formiert sich somit das Team, das in Abbildung 10.4 ganz links oben zu sehen ist. Die Details der

[6] Verschiedene Delegationsmodi werden von Köhler-Bußmeier [KB09, S. 458+459] für SONAR-Organisationen zwar bereits in Betracht gezogen. Allerdings handelt es sich aktuell um keinen Kernaspekt des SONAR-Ansatzes. Für den hier exemplarisch vorgestellten Prototypen einer SONAR-Organisationseinheit sind verschiedene Delegationsmodi wie Verhandlung, Versteigerung/Ausschreibung und direkte Zuweisung zwar bereits vorgesehen, allerdings haben sie noch keine wirkliche Bewandtnis, da aktuell noch kein Austausch von Gütern oder Währung implementiert ist.

Delegationsaktivitäten sind hier ausgeblendet.

2. *Zeitpunkt t_4:* Ein formiertes Team ist mit einem Teamdienstnetz assoziiert. Ein solches Teamdienstnetz wird durch die Assoziation mit zusätzlichen Einschränkungen bezüglich erlaubter/geforderter Markierungen zu einem Teamplan konkretisiert. Köhler-Bußmeier [KB09, S. 365-367] sieht für SONAR eigentlich vor, die Formation eines Teamplans solle verteilt im Team vollzogen werden. Für den hier vorgestellten Prototypen einer SONAR-Organisationseinheit wird dagegen einfach angenommen, die Einschränkung eines Teamdienstnetzes zu einen Teamplan obliege dem *Kopf* des Teams. Dabei handelt es sich um das Teammitglied, das die Delegation der ursprünglichen initialen Aufgabe angestoßen hat. Der entsprechende Integrationsakteur initiiert dementsprechend auch die Teamplanaktivität. Insgesamt werden für die Aktivierung einer Teamplanaktivität folgende Bedingungen überprüft: (1) Das angegebene Team hat sich zuvor formiert und ist wohlgeformt; (2) Der Initiator der Aktivität ist der Kopf des Teams; (3) Die Aktivität entspricht dem mit dem Team assoziierten Teamdienstnetz in Kombination mit zulässigen Teamplaneinschränkungen.

   Im Zuge des exemplarischen Ablaufes aus Abbildung 10.9 aktiviert Bob (purchaser) zum Zeitpunkt t_4 das Teamdienstnetz PC für das zuvor formierte Team. Etwaige Teamplaneinschränkungen beschränken sich hier lediglich auf den Wert der Produktmarke.

3. *Zeitpunkte t_5 – t_{10}:* Nach der Aktivierung der Teamplanaktivität werden die Rollen des Teamplans von den Akteuren eingenommen, die entsprechend des Teamnetzes (hier das Teamnetz links oben in Abbildung 10.4) die finalen Aufgaben ausführen. In diesem Fall handelt es sich um Sam (single-supplier) und Bob (purchaser), die in ihrer Funktion als operationale Akteure die Rollen producer beziehungsweise consumer einnehmen.

4. *Zeitpunkt t_{11}:* Zu diesem Zeitpunkt ist die Teamplanaktivität beendet und die Bearbeitung der initialen Aufgabe {producer, consumer} ist abgeschlossen.

10.2.2.4 Genauere Betrachtung des Aspektes der Governance

Der Governance-Aspekt einer SONAR-Organisationseinheit wurde in Unterabschnitt 10.2.2.1 als Teamwork mit dem Zweck der organisatorischen Transformation identifiziert. Dabei wurde auch bereits darauf hingewiesen, dass diese grobe Charakterisierung dem Stratifikationsprinzip von SONAR-Modellen nicht gerecht wird. Stattdessen sehen SONAR-Modelle beliebig viele Stratifikationsebenen vor, wobei Teamworkvorgänge einer Ebene n die Organisationsspezifikationen der darunter liegenden Ebenen transformieren. Es gibt nun verschiedene Möglichkeiten, dieses Prinzip im Kontext von SONAR-Organisationseinheiten umzusetzen. Eine Möglichkeit besteht darin, alle Spezifikations- und Zustandsfakten der Systemregeln mit einem Bezeichner der jeweiligen Stratifikationsebene, für die sie gelten, zu versehen. Das würde zu einem hohen Maß an Unübersichtlichkeit führen. Beispielsweise würde die Zugehörigkeit eines Akteurs zu den drei Akteursmengen (Operation, Integration, Governance) im Kontext einer Organisationseinheit stark aufgeweicht, denn diese Zugehörigkeit ist abhängig von den Spezifikationen einer bestimmten Stratifikationsebene.

Abbildung 10.10 zeigt eine weitere Möglichkeit, die eine übersichtlichere Lösung darstellt. Diese Herangehensweise steht in Zusammenhang mit der in Abbildung 9.22 (Teilabschnitt 9.2.3) modellierten Erweiterung des Grundmodells einer Organisationseinheit um die Festlegung von Governance-Regeln „von außen“. Die hier gewählte Lösung stellt eine ver-

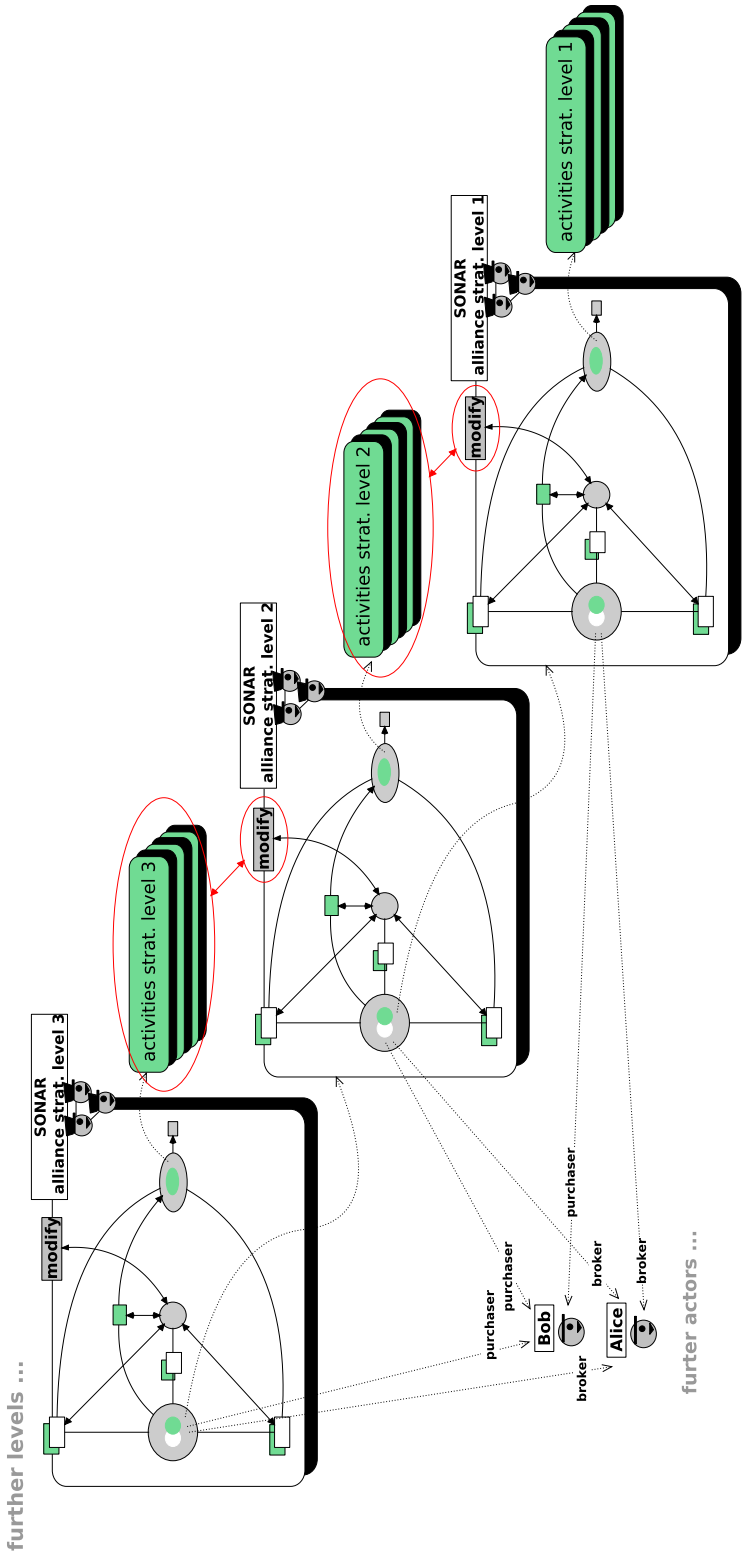

Abbildung 10.10: Umsetzung stratifizierter SONAR-Organisationen

einfachende Abwandlung dar, indem die „normalen“ Systemregeln (Integrations-Regeln) von außen festgelegt werden. Die Bezeichnung „Von außen“ bezieht sich hier auf die nächst höhere Stratifikationsebene. Jede Stratifikationsebene eines SONAR-Modells wird als eigene Organisationseinheit modelliert. Die Organisationseinheit zu einer bestimmten Stratifikationsebene bettet dabei die Organisationseinheit zur darunter liegenden Stratifikationsebene ein.[7]

Wie in der Abbildung ersichtlich ist, kommt diese Umsetzung stratifizierter SONAR-Modelle ohne explizite Governance-Akteure und -Aktivitäten aus. Jede Stratifikationsebene ist ausschließlich in Form von Integration und Operation modelliert. Implizit gilt allerdings, dass ausgehend von einer bestimmten Organisationseinheit die Aktivitäten der einbettenden Stratifikations-Organisationseinheit einen Governance-Charakter haben.

Wie bereits in Unterabschnitt 10.2.1.4 betont wurde, handelt es sich bei Stratifikationsebenen im Kontext von SONAR-Modellen ausdrücklich um eine andere Art von Systemebenen als sie ansonsten in der vorliegenden Arbeit thematisiert werden. Entsprechend sei an dieser Stelle auch für Abbildung 10.10 explizit hervorgehoben, dass die dort modellierte Einbettung von Organisationseinheiten in andere Organisationseinheiten nichts mit der Thematik des kollektiven Akteurswesen zu tun hat (der Aspekt des kollektiven Akteurswesens für SONAR-Organisationseinheiten wird im folgenden Unterabschnitt 10.2.2.5 behandelt). Es handelt sich stattdessen um operationales Modell für Selbstorganisation mit multiplen Autoritätsebenen. Dementsprechend zeigt die Abbildung auch, dass auf jeder Stratifikationsebene die gleichen Akteure für die gleichen Positionen eingesetzt werden. Die Zugehörigkeit dieser Akteure zu den Mengen der Operations- und Integrationsakteuren hingegen kann von Ebene zu Ebene variieren.

Zusammenfassend lässt sich sagen, dass sich das Konzept stratifizierter SONAR-Modelle auf das Organisationseinheits-Modell der vorliegenden Arbeit übertragen lässt, auch wenn dafür eine geringfügige Anpassung des grundlegenden Organisationseinheits-Modells notwendig ist. Allerdings sei an dieser Stelle vermerkt, dass die Theorie stratifizierter SONAR-Modelle von Köhler-Bußmeier vollständig beschrieben ist, es aktuell allerdings noch an überzeugenden Anwendungen fehlt. Daher wurde das Konzept stratifizierter Organisationseinheiten nach Abbildung 10.10 zwar prototypisch umgesetzt, aber lediglich an sehr einfachen Beispielen (vergleiche Abbildung 10.5) getestet.

Daher sei hier noch explizit auf eine mögliche Alternative hingewiesen. Da Governance in vielen Fällen ein recht diffiziler Aspekt ist, liegt es nach Autor der vorliegenden Arbeit nahe, an dieser Stelle realweltliche/soziale Akteure einzubeziehen (vergleiche hierzu auch die Ausführungen zu Autonomic/Proactive Computing aus Teilabschnitt 2.2.5). Abbildung 10.11 illustriert das generelle Schema für das Beispiel einer SONAR-Organisationseinheit: Während Integrations- und Operations-Akteure von künstlichen Softwareakteuren verkörpert werden, sind Governance-Akteure Platzhalter für soziale Akteure. Bei dem für einen Governance-Akteur eingebetteten Akteursnetz handelt es sich um einen *Netz-Proxy*, der die Anbindung an eine Benutzungsschnittstelle herstellt.

Da das Thema der sozio-technischen Integration in der vorliegenden Arbeit nicht vertiefend diskutiert wird, sei hier lediglich auf einen wichtigen Aspekt hingewiesen. Es handelt sich hier um eine Zwischenstufe zwischen endogener und exogener Governance einer Organisationseinheit. Zwar ist das Softwaresystem nicht mehr vollständig sich selbst überlassen. Doch nichtsdestotrotz handeln die sozialen/realweltlichen Governance-Akteure „aus der Organisationseinheit heraus“. Es handelt sich nicht um völlig externe Akteure, die das System

[7] Im aktuellen Prototypen werden allerdings nur zwei Stratifikationsebenen betrachtet.

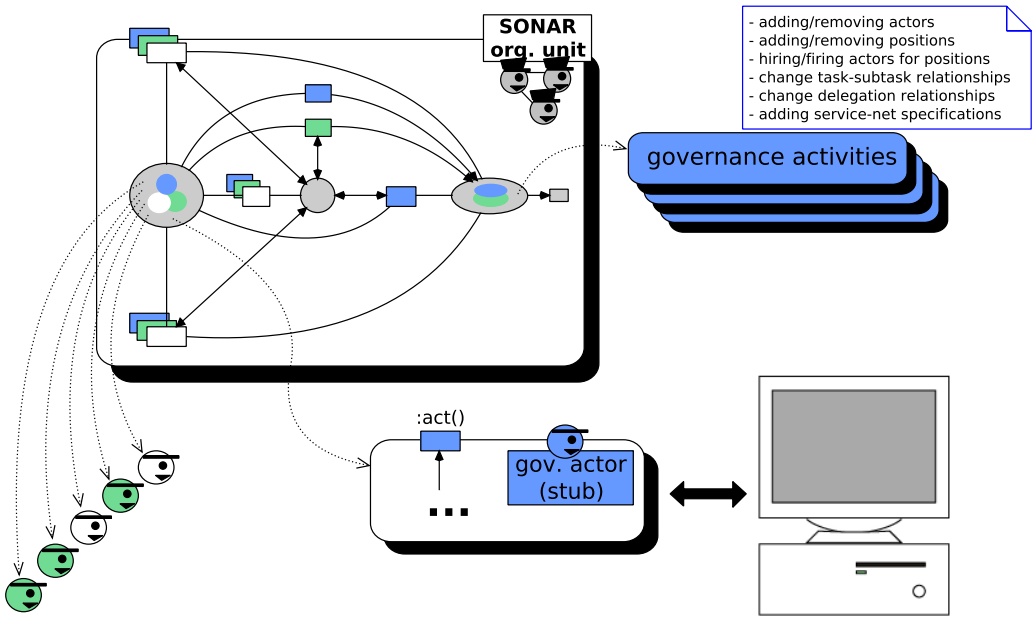

Abbildung 10.11: Einbindung sozialer Akteure als Governance-Akteure

gegebenenfalls anhalten und Offline Modifikationen vornehmen. Die von Menschen gesteuerten Governance-Akteure sind an den Kontext und die Regeln der Organisationseinheit gebunden. So können die von ihnen veranlassten Modifikationen an den Systemregeln lediglich die Spezifikations- und Zustandsfakten betreffen. Die Paradigmaregeln bleiben unberührt.

10.2.2.5 SONAR-Multiorganisationssysteme

Bisher wurde das Zusammenspiel von Integration und Operation im Kontext einer SONAR-Organisationseinheit beschrieben sowie die Behandlung von Governance-Aspekten diskutiert. Die Behandlung mehrerer Systemebenen in dem Sinne, dass SONAR-Organisationseinheiten im Kontext von umschließenden/globaleren SONAR-Organisationseinheiten agieren, wurde bislang nicht behandelt. Auch im Hinblick auf die neben SONAR in diesem Kapitel betrachteten weiteren Ansätze der Organisationsmodellierung stellt sich zunächst einmal die Frage, ob der jeweilige Ansatz einen Übergang von einer auf multiple Organisationsebenen überhaupt unmittelbar unterstützt. Im Falle von SONAR ist das tatsächlich der Fall.

Die in den Unterabschnitten 10.2.1.1 und 10.2.1.2 beschriebenen Mechanismen der Verfeinerung von Aufgaben/Rollen auf der einen und Dienstnetzen auf der anderen Seite leiten unmittelbar auf die Möglichkeit der Trennung verschiedener Systemebenen hin. Dies soll in diesem Unterabschnitt anhand eines Beispieles illustriert werden. Die Erweiterung des SONAR-Ansatzes um kollektives Akteurswesen (*holonische* SONAR-Modelle) ist Gegenstand aktueller Forschung und noch nicht vollständig in die zu Grunde liegende Theorie integriert. Somit basieren die folgenden Ausführungen zu SONAR-Organisationseinheiten als kollektiven Akteuren im Gegensatz zu den bisherigen Betrachtungen nicht unmittelbar auf der bereits bestehenden und ausgereiften SONAR-Theorie nach Köhler-Bußmeier. Stattdessen beinhalten die Erläute-

rungen einen Vorschlag des Autors der vorliegenden Arbeit, wie SONAR-Modelle mit multiplen Systemebenen prinzipiell aufzubauen sind.

Als Startpunkt zeigt Abbildung 10.12 die Verfeinerung der Position contracting-supplier in mehrere neue Positionen.[8] Zusätzlich zeigt Abbildung 10.13 das Dienstnetz MAC, das im Zuge der Positionsverfeinerung eingeführt wird. An eine Positionsverfeinerung für eine Position P

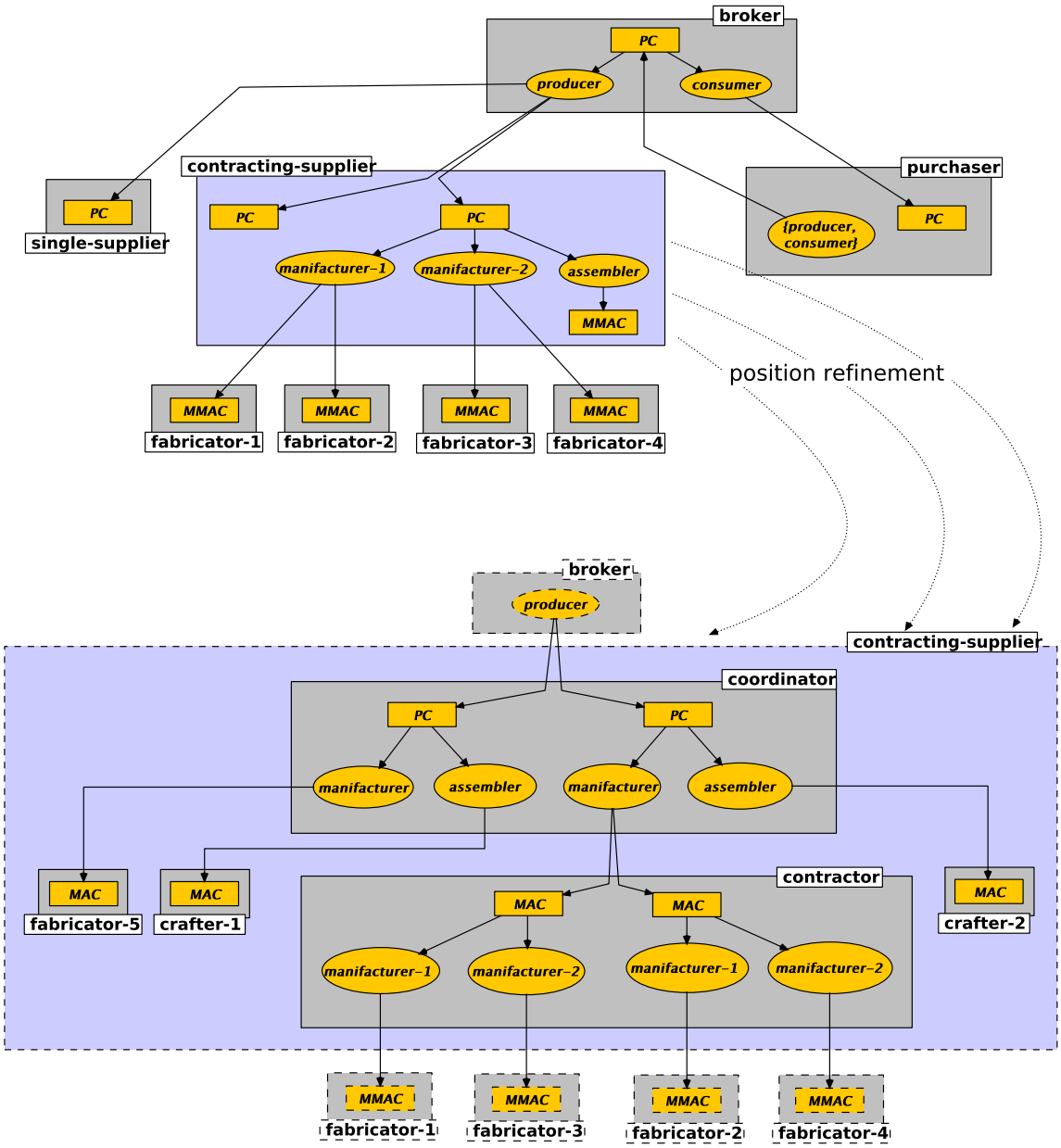

Abbildung 10.12: Verfeinerung der Position contracting-supplier

[8] Im Kontext von SONAR wird bislang lediglich die Verfeinerung von Positionen, die ausschließlich Transitionen umfassen und damit nur für die Ausführung finaler Aufgaben zuständig sind, thematisiert.

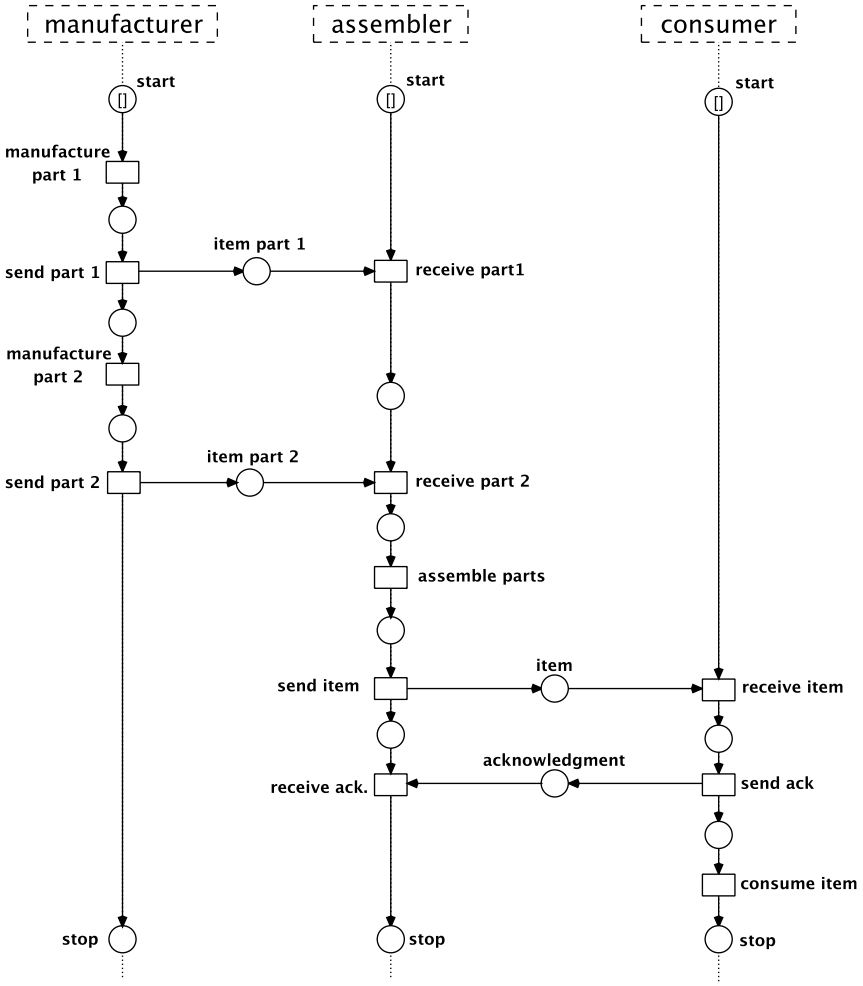

Abbildung 10.13: Das Dienstnetz MAC

in Form neu eingeführter Positionen $P_1, ..., P_n$ und (gegebenenfalls) neuer Dienstnetze werden an dieser Stelle drei grundlegende Forderungen gestellt.

1. Das durch die Verfeinerung entstehende Delegationsnetz in Kombination mit etwaig neu eingeführten Dienstnetzen ist weiterhin wohlgeformt was die (verzahnte) Verfeinerung von Aufgaben/Rollen und Dienstnetzen angeht.

2. Alle Teile des ursprünglichen Delegationsnetzes außer P bleiben von der Verfeinerung unberührt, abgesehen von etwaigen Änderungen an Delegationsbeziehungen im Zusammenhang mit den neuen Positionen $P_1, ..., P_n$ (folgender Punkt).

3. Zu jeder im neuen Delegationsnetz möglichen Delegationssequenz von Aufgaben DS_1 gibt es im ursprünglichen Delegationsnetz eine mögliche Delegationssequenz DS_2, so dass gilt: Für jede Teilsequenz von Delegationsschritten in DS_1, die von den neu eingeführten Positionen $P_1, ..., P_n$ ausgeführt werden, gibt es in DS_2 eine Teilsequenz von

Delegationsschritten, die von der ursprünglichen Position P ausgeführt werden. Abgesehen davon sind die beiden Delegationssequenzen deckungsgleich. Umgekehrt wird diese Forderung nicht gestellt, wie die folgenden Beispiele motivieren werden.

Dass die ersten beiden Bedingungen für das Beispiel erfüllt sind, ist anhand der Abbildungen 10.12 und 10.13 unmittelbar ersichtlich. Es wird die neue Aufgabe {manufacturer} eingeführt und die entsprechende neue Rolle manufacturer taucht im neuen Dienstnetz MAC auf. Gegenüber dem Dienstnetz PC (Abbildung 10.3 oben) stellt die Rolle manufacturer gemeinsam mit der Rolle assembler eine Verfeinerung der Rolle producer dar. Gegenüber dem Dienstnetz MMAC (Abbildung 10.3 unten) wird sie selber durch die beiden Rollen manufacturer-1 und manufacturer-2 verfeinert.

Bezüglich der dritten Bedingung ist festzustellen, dass Delegationsbeziehungen im Kontext der ursprünglichen Position contracting-supplier „umgebogen“ werden, Rollen- und Dienstnetzanschriften (wie gefordert) jedoch unverändert bleiben. Die Aufgabe {producer} kann weiterhin vom broker an den contracting-supplier (nun verkörpert durch neu eingeführte Positionen) delegiert werden. Entweder übernimmt dieser die Bearbeitung der Aufgabe wie zuvor komplett selbst, oder er teilt die Aufgabe derart auf, dass er selbst lediglich die Aufgabe {assembler} übernimmt und die Aufgaben {manufacturer-1} und {manufacturer-2} unter den Positionen fabricator-1 – fabricator-4 verteilt. Allerdings gilt in beiden Fällen, dass die Delegationsschritte nun über die neu eingeführten Positionen verteilt sind. Hier ist ein interessanter Umstand zu beobachten. Im ursprünglichen Delegationsnetz waren für die Bearbeitung der Aufgaben {manufacturer-1} und {manufacturer-2} vier Sub-Teams bestehend aus den Positionen fabricator-1 – fabricator-4 möglich (vergleiche Abbildung 10.4). In der verfeinerten Version sind lediglich zwei Kombinationen möglich. Die Positionsverfeinerung hat hier also auch eine Auswirkung auf mögliche Teamkompositionen für den „Rest“ der Organisation.

Eine Positionsverfeinerung kann einerseits im Kontext einer gegebenen SONAR-Organisationseinheit angewendet werden (als Transformation im Zuge von Governance-Aktivitäten). Dann entstehen neue Positionen, diese müssen mit Akteuren besetzt werden und neue Delegationspartner müssen miteinander bekannt gemacht werden. Alternativ kann die Definition einer Positionsverfeinerung aber vor allem auch als Grundlage für die Einführung neuer Systemebenen dienen. An der Spezifikation der ursprünglichen Organisationseinheit ändert sich dann nichts. Stattdessen wird die betroffene Position wiederum mit einer eigenen SONAR-Organisationseinheit besetzt. Diese ist entlang des Delegationsnetzes organisiert, das die Verfeinerung der betroffenen Position darstellt. Aus diesem Grund ist das verfeinerte Teilnetz zur Position contracting-supplier in der unteren Hälfte von Abbildung 10.12 explizit getrennt dargestellt.

Aus Perspektive der bisher betrachteten Allianz Organisationseinheit stellt sich die Situation wie zuvor dar: Die Position des contracting-supplier ist schlicht mit einem Akteur besetzt. Dass es sich dabei um einen kollektiven Akteur handelt, dessen Aktionen von mehreren seiner internen Akteure ausgeführt werden, wird nicht sichtbar. In der Abbildung ist auch angedeutet, dass es in der neuen Organisationseinheit Positionen gibt, die für das Grenzmanagement verantwortlich sind und für den Brückenschlag zwischen Delegationsvorgängen auf verschiedenen Systemebenen sorgen müssen. In diesem Fall sind die betroffenen Positionen die des coordinator und des contractor. Details bezüglich der Teamformation werden hier allerdings ausgeblendet. Stattdessen zeigen die beiden Abbildungen 10.14 und 10.15 die Verzahnung von Teamplanausführungen über Systemebenen hinweg. Zusätzlich zu der bisher betrachteten SONAR-Organisationseinheit der Allianz (SONAR alliance) ist die neue SONAR-Organisationseinheit eines Unternehmens aufgeführt (SONAR enterprise). Sie besetzt genau die Position

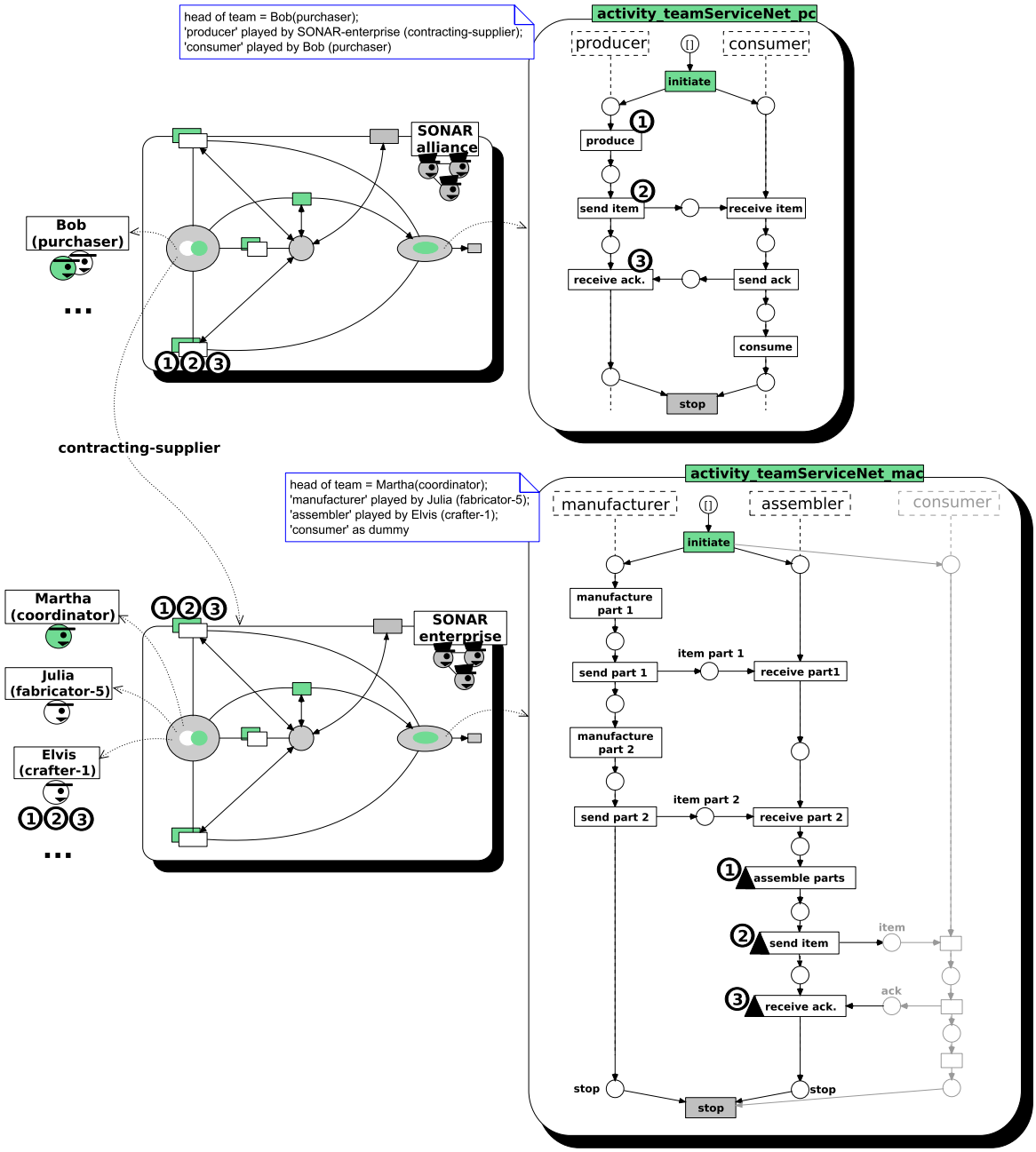

Abbildung 10.14: Verzahnte Teamplanausführung über Systemebenen hinweg (1)

des contracting-supplier der Allianz und ist intern entsprechend der unteren Hälfte von Abbildung 10.12 organisiert.

Abbildung 10.14 zeigt den Fall, bei dem auf Ebene der Allianz-Organisationseinheit das Team formiert wurde, in dem der contracting-supplier die Ausführung der finalen Aufgabe {producer} übernimmt (das Team rechts oben in Abbildung 10.4). Auf Ebene der Unternehmens-Organisationseinheit wurde das dazu passende Team formiert, bei dem die

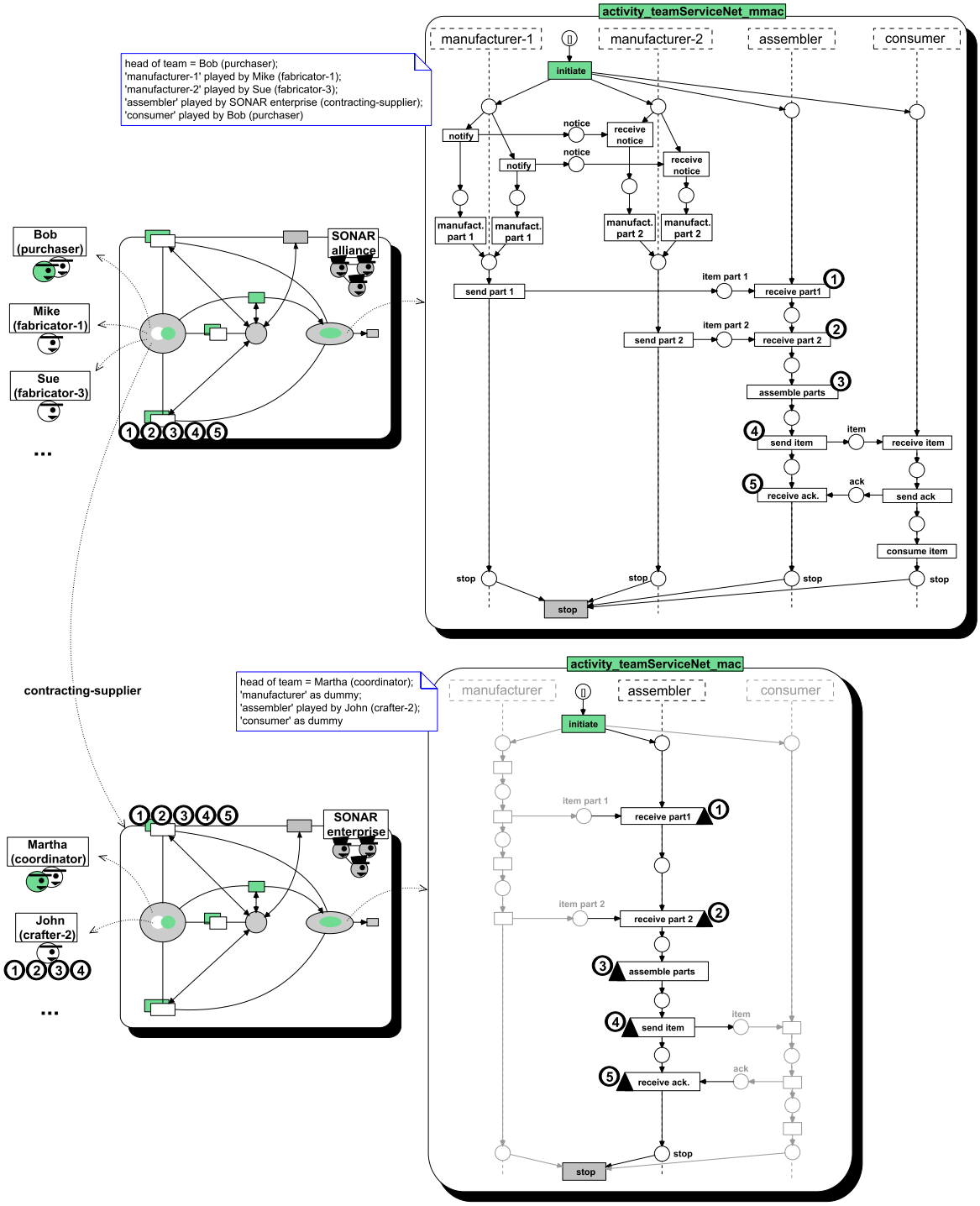

Abbildung 10.15: Verzahnte Teamplanausführung über Systemebenen hinweg (2)

Positionen fabricator-5 und crafter-1 die Ausführung der finalen Aufgaben {manufacturer} beziehungsweise {assembler} übernehmen. Auf Ebene der Allianz wird die Teamplanaktivität

bestehend aus den Rollenanteilen producer und consumer initiiert. Von der Aufteilung der producer-Rolle in die Rollen manufacturer und assembler auf Ebene des Unternehmens und der damit einhergehenden Teamplanaktivität ist die Allianz nicht betroffen. Auf Ebene des Unternehmens ist die Rolle consumer zwar bekannt (und notwendig für die Überprüfung von Wohlgeformtheit). Doch da sie nicht im Kontext des Unternehmens implementiert wird, taucht sie im Netz der Teamplanaktivität nur als Dummy auf. Das Auftreten von Peripherieaktionen auf Ebene des Unternehmens und deren Auftreten als interne Aktionen auf Ebene der Allianz ist in der Abbildung explizit hervorgehoben.

Abbildung 10.15 zeigt ein weiteres Beispiel. Hier hat sich auf Ebene der Allianz ein komplexeres Team gebildet (das Team in der Mitte der linken Seite von Abbildung 10.4): Die Aufgabe {producer} ist aufgeteilt in die Aufgaben {manufacturer-1}, {manufacturer-2} und {assembler}, wobei die Position des contracting-supplier lediglich für die Ausführung der Aufgabe {assembler} verantwortlich ist. Auf Ebene des Unternehmens übernimmt die Position crafter-2 diese Aufgabe. Im Teamdienstnetz des Unternehmens werden nun sowohl die Rolle manufacturer als auch consumer als Dummys geführt. Hier fällt auf, dass für den Teamplan des Unternehmens irrelevant ist, dass die Rolle manufacturer auf Ebene der Allianz weiter verfeinert ist.

Die beiden Beispiele zeigen ergänzend zueinander, dass in beide Richtungen einer Einbettungsbeziehung zwischen Systemebenen Details gegebenenfalls verborgen bleiben. Die Perspektive einer konkreten SONAR-Organisationseinheit kann jeweils exakt so reichhaltig sein, wie es gerade notwendig ist.

Insgesamt lässt sich feststellen, dass der SONAR-Ansatz Konzepte mit sich bringt, die den Weg für einen Übergang von einer zu multiplen SONAR-Organisationseinheiten bereiten. Die in diesem Unterabschnitt am Beispiel von zwei Systemebenen illustrierten Mechanismen ließen sich auf weitere Systemebenen ausweiten. Darüber hinaus ist es möglich, dass eine SONAR-Organisationseinheit in mehrere weitere SONAR-Organisationseinheiten eingebettet ist und dementsprechend mehrere Positionen einnimmt (sie könnte auch mehrere Positionen in einer einzigen weiteren Organisationseinheit einnehmen). In dem Fall müsste die Spezifikation der Organisationseinheit derart sein, dass die Verfeinerungen verschiedener Positionen berücksichtigt und idealerweise geeignet integriert werden.

Zuletzt ist festzuhalten, dass die in diesem Unterabschnitt veranschaulichte Trennung verschiedener Systemebenen eines SONAR-Multiorganisationssystems *orthogonal* zu der Aufteilung in Stratifikationsebenen für die Reorganisation einer einzigen SONAR-Organisation ist.

10.3 Organisationseinheiten nach $\mathcal{M}$OISE$^+$

Der ursprüngliche $\mathcal{M}$OISE$^+$-Ansatz (**M**odel of **O**rganization for mult**I**-agent **S**yst**E**ms) für die Modellierung von Agentenorganisationen, ein $\mathcal{M}$OISE$^+$-basierter Vorschlag für Reorganisation und die Middleware $\mathcal{S}$-$\mathcal{M}$OISE$^+$ für das Multiagentensystem-Deployment von $\mathcal{M}$OISE$^+$-Organisationen gehen vor allem auf Hübner, Sichman und Boissier [HSB02, HSB04, HSB06] zurück. An einigen weiterführenden Arbeiten sind zusätzliche Personen beteiligt, die Ausführungen dieses Teilabschnittes beziehen sich jedoch nur auf die drei genannten Veröffentlichungen.

10.3.1 Der $\mathcal{M}$OISE$^+$-Ansatz: Grundlegende Konzepte und Modelldimensionen

Der Zweck von $\mathcal{M}$OISE$^+$-Modellen ist dem von SONAR-Modellen ähnlich. Beide Modellierungsansätze weisen eine strukturelle und eine funktionale Dimension auf: In beiden Fällen

werden Organisationsstrukturen, die der Erfüllung spezifischer organisatorischer Funktionen dienen, beschrieben. Bei den organisatorischen Funktionen handelt es sich im Falle von SONAR um die Bearbeitung initialer Aufgaben, im Falle von $\mathcal{M}$OISE$^+$ um das Erreichen bestimmter Organisationsziele. Die Unterschiede zwischen beiden Ansätzen resultieren einerseits aus der Modellierungstechnik, andererseits aus der Beziehung zwischen den verschiedenen Modelldimensionen.

Abbildung 10.16 veranschaulicht anhand eines Beispiels die verschiedenen Dimensionen von $\mathcal{M}$OISE$^+$-Modellen. Das hier verwendete Beispiel einer Fabrik stammt vom Autor der vorliegenden Arbeit. Es wurde gewählt, um eine inhaltliche Nähe zum SONAR-Laufbeispiel zu schaffen. Die Beispiele aus den weiter oben zitierten Originalarbeiten zu $\mathcal{M}$OISE$^+$ (verschiedene Versionen eines Agenten-Fußballteams) sind hierzu ungeeignet.

10.3.1.1 Strukturelle, funktionale und deontische Spezifikation

Die linke Seite der Abbildung zeigt die *Strukturspezifikation* der Fabrik. Als Hauptkonzepte finden sich hier *Gruppen* und *Rollen*. Jede Gruppe ist aus weiteren Gruppen und/oder Rollen zusammengesetzt. So setzt sich beispielsweise die Gruppe plant für die gesamte Fabrik aus den Untergruppen production und sale-and-distribution sowie der Rolle plant-manager zusammen. Die Untergruppe production ist ihrerseits aus Rollen für die Herstellung von Produkten zusammengesetzt, während die Untergruppe sale-and-distribution aus Rollen für den Verkauf und Vertrieb der Produkte zusammengesetzt ist. Für das Deployment einer $\mathcal{M}$OISE$^+$-Organisation nehmen Agenten Rollen ein. Rollen können dabei gegebenenfalls mehrfach besetzt sein. Zu diesem Zweck ist in der Strukturspezifikation aufgeführt, wie viele Agenten eine bestimmte Rolle mindestens einnehmen müssen und höchstens einnehmen dürfen. Beispielsweise ist genau ein Manager der Fabrik (plant-manager) notwendig. Im Kontext der production-Gruppe muss es mindestens einen Vorarbeiter (foreman), mindestens einen Monteur (assembler) und mindestens drei Maschinisten (machine-operator) geben. Gleichzeitig darf es nicht mehr als zwei Vorarbeiter, nicht mehr als drei Monteure und nicht mehr als fünf Maschinisten geben.

Rollen können in verschiedenen Beziehungen zueinander stehen. Eine dieser Beziehungen ist die *Kompatibilitätsbeziehung*. Über diese Beziehung wird spezifiziert, welche Rollen gleichzeitig vom selben Agenten eingenommen werden können. Wenn Rollen nicht explizit kompatibel zueinander sind, so müssen sie von verschiedenen Agenten besetzt werden. Im Beispiel ist es möglich, dass der Manager der Fabrik gleichzeitig der Vorarbeiter und der Verkaufsmanager ist. In allen anderen Fällen müssen Rollen mit verschiedenen Agenten besetzt werden.

Eine weitere wichtige Beziehung ist die *Vererbungsbeziehung*. Im Beispiel erben alle Rollen von der abstrakten Rolle employee. Diese Rolle ist insofern abstrakt, dass nicht vorgesehen ist, sie direkt mit Agenten zu besetzen. Vererbungsbeziehungen und abstrakte Rollen dienen vor allem dem Zweck, die Definition weiterer Rollenbeziehungen zu vereinfachen beziehungsweise ausdrucksstärker zu machen. Im Beispiel wird das anhand der *Autoritätsbeziehung* verdeutlicht. Dort wird spezifiziert, die Rollen foreman und sales-manager hätten Autorität über die abstrakte Rolle employee und damit auch über alle Rollen, die von dieser Rolle erben. Allerdings sind diese Beziehungen jeweils als intra-group deklariert. Das bedeutet, die Autoritätsbeziehung gilt nur dann, wenn die jeweiligen Rolleninhaber der gleichen Gruppe angehören. Dementsprechend hat ein Vorarbeiter lediglich Autorität über Mitglieder der Produktionsgruppe. Demgegenüber ist die Rolle plant-manager über eine als inter-group deklarierte Autoritätsbeziehung mit der Rolle employee assoziiert. Dies bedeutet, dass der Fabrikmanager Autorität über *alle* anderen Rolleninhaber besitzt, unabhängig von Gruppenzugehörigkeiten.

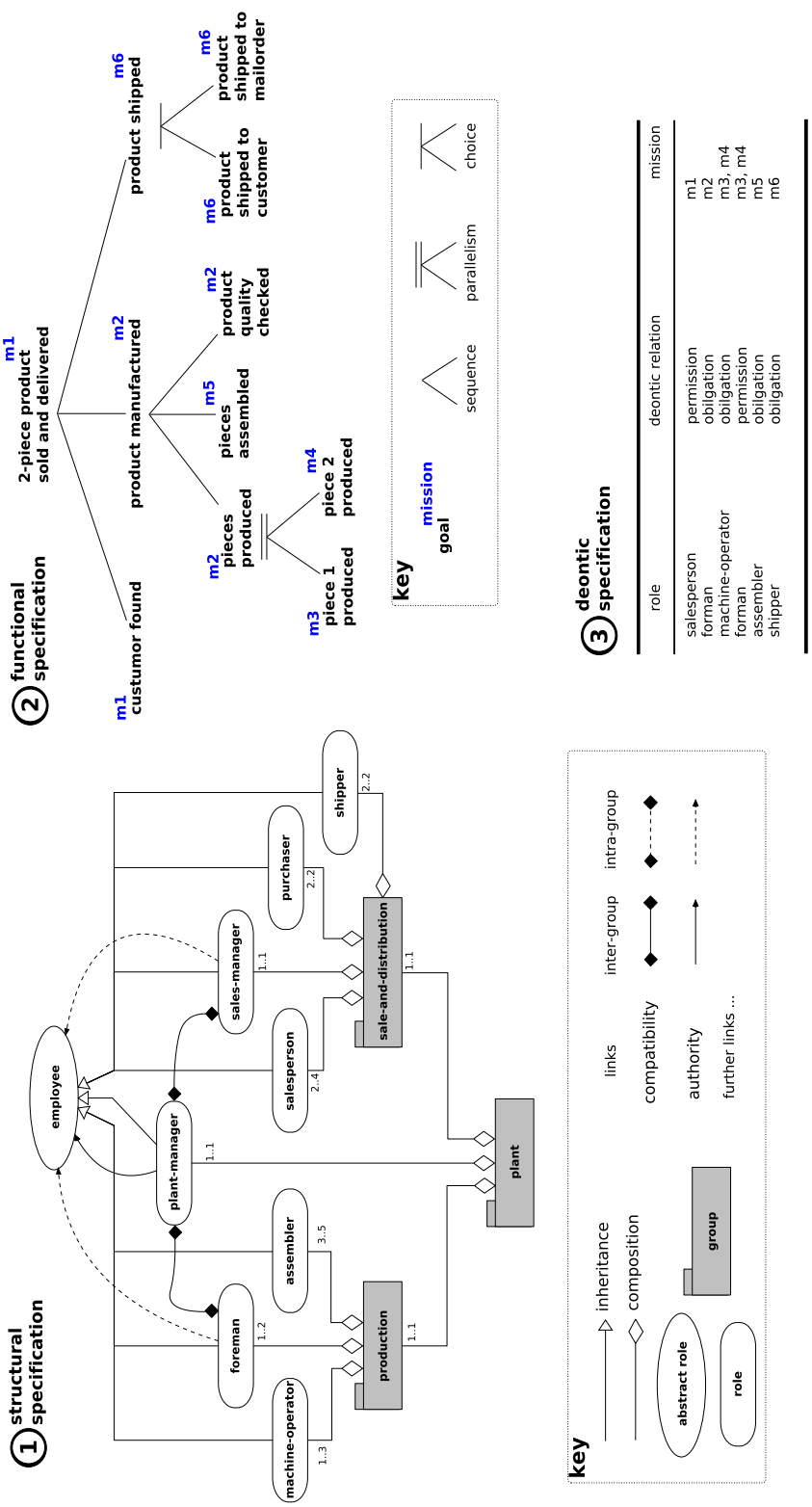

Abbildung 10.16: $\mathcal{M}$OISE$^+$ Modelldimensionen an einem Beispiel

Auf der rechten oberen Seite der Abbildung ist ein Teil der *funktionalen Spezifikation* eines $\mathcal{M}$OISE$^+$-Modells zu sehen. Die funktionale Spezifikation wird in Form von *sozialen Schemata* vorgenommen. Jedes Schema ist ein Zielbaum, der einen Plan zum Erreichen eines bestimmten organisatorischen Ziels repräsentiert. An der Wurzel des Baumes steht das eigentliche organisatorische Ziel. Die restlichen Knoten des Baumes sind Teilziele, die sich aus sukzessiver Verfeinerung des ursprünglichen Ziels ergeben. Dabei können Teilziele eines gemeinsamen Oberziels in einer von drei verschiedenen Beziehungen zueinander stehen: (1) Teilziele werden nacheinander in einer bestimmten Reihenfolge verfolgt (*Sequenz*), (2) werden parallel verfolgt (*Parallelität*) oder (3) nur eines der Teilziele wird verfolgt (*Auswahl*).

Im Beispiel ist ein beispielhaftes Schema abgebildet. Das Hauptziel dieses Schemas ist der Verkauf und die Auslieferung eines Produktes. Dieses Ziel wird in drei nacheinander zu verfolgende Teilziele aufgeteilt: (1) Einen Kunden für das Produkt gewinnen, (2) das Produkt herstellen und schließlich (3) das Produkt ausliefern. Davon ausgehend werden weitere Teilziele definiert. Parallelität der Zielverfolgung kommt dabei bei der Herstellung der Bestandteile des Produktes zum Tragen. Eine Auswahl zwischen Teilzielen wird bezüglich des Versandes getroffen: Entweder wird das Produkt direkt zum Kunden verschickt oder aber ein Versandunternehmen mit dieser Aufgabe betreut.

Die strukturelle und die funktionale Spezifikation eines $\mathcal{M}$OISE$^+$-Modells stehen zunächst ohne Bezug nebeneinander. Die Kopplung beider Dimensionen erfolgt im Zuge der dritten Spezifikation eines $\mathcal{M}$OISE$^+$-Modells, der *deontischen Spezifikation*. Zu diesem Zweck sind Ziele eines sozialen Schemas mit *Missionen* annotiert. Eine Mission besteht aus einem oder mehreren Zielen eines Schemas. Für eine konkrete *Instanz eines Schemas* obliegt die Verantwortung für eine Mission und das Erreichen der mit der Mission assoziierten Ziele genau einem Agenten. Die deontische Spezifikation deklariert nun, welche Rolleninhaber die Erlaubnis beziehungsweise die Verpflichtung haben, bestimmte Missionen zu übernehmen. Auf diese Weise wird die Verbindung zwischen struktureller und funktionaler Spezifikation hergestellt. In dem Schema aus Abbildung 10.16 gibt es beispielsweise die Mission m2, die sich aus den Zielen product manufactured, pieces produced und product quality checked zusammensetzt. Die deontische Spezifikation sagt aus, der Inhaber der Rolle foreman habe die *Verpflichtung*, diese Mission in einer Instanz des Schemas anzunehmen. Der Vorarbeiter als Inhaber dieser Rolle hat außerdem die *Erlaubnis*, die Missionen m3 beziehungsweise m4 (jeweils verknüpft mit dem Ziel, ein Teil des Endproduktes hergestellt zu haben) zu übernehmen. Eine Verpflichtung, diese beiden Missionen anzunehmen, besteht für die Inhaber der Rolle machine-operator. Eine Verpflichtung beziehungsweise eine Erlaubnis für die Annahme einer Mission einer Schemainstanz besteht natürlich nur, solange die Mission noch offen ist. Die Mission, zu der das Wurzelziel eines Schemas gehört, ist immer mit einer Erlaubnisdeklaration für eine oder mehrere Rollen verknüpft. Diese Erlaubnis ist gleichbedeutend mit der Erlaubnis, das Schema überhaupt zu instanziieren.

Die Gemeinsamkeiten und Unterschiede zwischen dem $\mathcal{M}$OISE$^+$- und dem SONAR-Ansatz für die Modellierung von Organisationen lassen sich nun gut charakterisieren.

- In beiden Fällen werden die Zwecke der Organisation expliziert. Die initialen Aufgaben im Falle von SONAR und die Wurzelziele von Schemata im Falle von $\mathcal{M}$OISE$^+$ dienen gleichermaßen dazu, die Kernfunktionen der modellierten Organisation zu repräsentieren.

- Im Falle von SONAR sind die strukturelle und die funktionale Modelldimension eng

verzahnt. Positionen und ihre Beziehungen untereinander lassen sich ausschließlich unter Bezugnahme auf die Bearbeitung von Aufgaben charakterisieren. Im Falle von $\mathcal{M}$OISE$^+$ hingegen stehen strukturelle und funktionale Spezifikationen zunächst einmal separat voneinander. Erst durch die deontischen Spezifikationen wird eine Verbindung hergestellt. Eine konkrete Strukturspezifikation kann in Kombination mit verschiedenen funktionalen Spezifikationen verwendet werden und umgekehrt. Dies erleichtert Ansätze der Reorganisation. Demgegenüber sind Transformationen im Falle von SONAR vielfach sehr kompliziert. Beispielsweise erfordert bereits die einfache Einführung einer neuen Teilaufgabe Modifikationen am Delegationsnetz sowie an gegebenenfalls mehreren Dienstnetzen. Für eine $\mathcal{M}$OISE$^+$-Spezifikation würde im analogen Fall das Einfügen eines neuen Teilzielknotens in einem Schema und eine kleine Erweiterung der deontischen Spezifikation genügen.

- Der Verzicht auf eine enge Verzahnung mit anderen Dimensionen erlaubt es im $\mathcal{M}$OISE$^+$-Ansatz, reichere Strukturspezifikationen gegenüber dem SONAR-Ansatz vorzunehmen. So können vielfältige Beziehungen zwischen Rollen definiert werden sowie Gruppierungen vorgenommen werden, ohne dass Implikationen für andere Modelldimensionen unmittelbar berücksichtigt werden müssten. In der Literatur finden sich zusätzlich zu den Vererbungs-, Kompatibilitäts- und Autoritätsbeziehungen aus Abbildung 10.16 noch weitere Rollenbeziehungen. Allerdings ist zu beachten, dass gerade aufgrund der Abgeschlossenheit der Strukturspezifikation die Semantik von Rollen-Beziehungen häufig unklar ist (das gilt auch für die Autoritätsbeziehung des hier betrachteten Beispiels, siehe dazu auch Teilabschnitt 10.3.2). Dies steht in starkem Gegensatz zu SONAR-Modellen. Die enge Verzahnung aller Modelldimensionen sowie die bewusste Einschränkung auf einige wenige Kernkonzepte und Modellelemente führen zu einer klaren Semantik.

- Im Zusammenhang mit dem letzten Punkt fällt außerdem auf, dass in $\mathcal{M}$OISE$^+$-Modellen eine Interaktionsspezifikation fehlt. Ein Schema legt mittels des Zielbaums und der Missionen zwar bestimmte Interaktionsszenarien nahe (beispielsweise Benachrichtigung bei Erreichen eines Ziels oder Einigung über die Verfolgung alternativer Ziele). Expliziert werden diese allerdings nicht. Im Falle von SONAR hingegen sind Interaktionen zwischen Rolleninhabern explizit im Form von Dienst- und Teamdienstnetzen spezifiziert. Darüber hinaus ist auch diese Modelldimension eng mit der strukturellen sowie der funktionalen Modell-Dimension verzahnt, so dass sich ein konsistenter Zusammenhang ergibt.

- Auf den ersten Blick fehlt im SONAR-Ansatz eine deontische Modelldimension. Da diese im $\mathcal{M}$OISE$^+$-Ansatz aber vor allem der Verknüpfung der strukturellen und funktionalen Dimension dient, ist sie im Falle von SONAR überflüssig. Im Grunde genommen kann man die Delegationsbeziehungen von SONAR als das Analogon zu den deontischen Spezifikationen sehen.

- Gerade die enge Verzahnung aller Modell-Dimensionen – speziell der durchgängige Zusammenhang von Verfeinerungen auf Ebene des Delegationsnetzes und auf Ebene der Dienstnetze – bot in Unterabschnitt 10.2.2.5 eine unmittelbare Unterstützung für den Übergang von einer SONAR-Organisationseinheit zu SONAR-Multiorganisationssystemen. Im Falle von $\mathcal{M}$OISE$^+$ existiert eine solche Unterstützung nicht. Es ist natürlich denkbar, eine Rolle einer $\mathcal{M}$OISE$^+$-Organisation wiederum mit

einer weiteren $\mathcal{M}$OISE$^+$-Organisation als kollektivem Akteur zu besetzen. Auf Ebene der zweiten Organisation müsste dann eine Verfeinerung von Missionen auf Ebene der ersten Organisation vorgenommen werden. Wie dies auszusehen hätte und welches ebenenübergreifende Verhalten sich daraus ergäbe, ist allerdings zunächst unklar.

- Ein weiterer – allerdings rein terminologischer – Unterschied besteht in der Verwendung des Begriffes *Rolle*. Bei SONAR tauchen Rollen lediglich in Dienstnetzen (und den darauf aufbauenden Teamdienstnetzen und Teamplänen) auf. Dabei gilt, dass eine Rolle zwar in mehreren Dienstnetzen auftauchen kann, dabei aber immer dieselbe Verhaltensspezifikation mit sich bringt. Das Konzept, welches mehrere Verhaltensspezifikationen vereint, ist in SONAR die Position. Im Falle von $\mathcal{M}$OISE$^+$ hingegen werden Rollen so verwendet wie Positionen im Falle von SONAR verwendet werden: Rollen können mit mehreren Schema-Missionen assoziiert sein und vereinigen somit multiple Verhaltensspezifikationen. Entsprechend sind es bei $\mathcal{M}$OISE$^+$ die Rollen, die in einer Multiagentensystem-Organisation mit Agenten besetzt werden, während es bei SONAR die Positionen sind.

An dieser Stelle reicht eine Aufzählung der Unterschiede zwischen dem SONAR- und $\mathcal{M}$OISE$^+$-Ansatz. In Abschnitt 10.6 wird der Vergleich (unter zusätzlicher Beachtung des ISLANDER-Ansatzes) wieder aufgegriffen und auf seine Implikationen hinsichtlich der Eignung der verschiedenen Ansätze für die Modellierung verschiedener *Archetypen* von Organisationseinheiten hin untersucht.

10.3.1.2 $\mathcal{M}$OISE$^+$-Reorganisation

Hübner et. al. [HSB04] stellen ein Konzept für endogene Reorganisation von $\mathcal{M}$OISE$^+$-Organisationen vor. Es handelt sich um einen generischen Ansatz, der auf jede beliebige $\mathcal{M}$OISE$^+$-Organisation angewendet werden kann. Zu diesem Zweck wird ein „herkömmliches“ $\mathcal{M}$OISE$^+$-Modell um die Spezifikation einer *Reorganisationsgruppe* (und zugehörigen Reorganisationsschemata) erweitert. Abbildung 10.17 zeigt die strukturelle Spezifikation der Reorganisationsgruppe und illustriert exemplarisch, wie sie in das bisherige Beispiel integriert wird.[9]

Eine Instanz der Reorganisationsgruppe besteht aus einem Manageragenten, der die Rolle orgManager einnimmt. Er ist hauptverantwortlich für Spezifikation und Zustand der gesamten $\mathcal{M}$OISE$^+$-Organisation. Zu diesem Zweck existiert die Rolle soc, von der *alle* anderen Rollen erben und zu der die Rolle orgManager eine Autoritätsbeziehung hat. Zusätzliche Mitglieder der Gruppe sind Agenten, die Buch über Spezifikations-/Zustandsänderungen der Organisation führen (Rolle historian), das Verhalten anderer Agenten beobachten (Rolle monitor), Reorganisationsvorschläge machen (abstrakte Rolle designer und deren Unterrollen) und aus mehreren Vorschlägen auswählen (Rolle selector). Die Integration der „herkömmlichen“ Strukturspezifikation der jeweiligen Organisation ist auf der rechten Seite der Abbildung illustriert. Die Rolle employee der Strukturspezifikation der Fabrik aus Abbildung 10.16 steht in Vererbungsbeziehung zu der Rolle monitored der Reorganisationsgruppe. Damit wird das Verhalten aller Agenten der Fabrik von den Inhabern der Rolle monitored verfolgt. Dadurch, dass die Rolle employee eine Kompatibilitätsbeziehung zu der Rolle org-participant der Reorganisations-Gruppe hat, können alle Agenten der Fabrik selber Vorschläge für Reorganisationsmaßnahmen einbringen.

[9] Die Strukturspezifikation der Reorganisationsgruppe ist gegenüber der Spezifikation aus der Originalliteratur etwas vereinfacht.

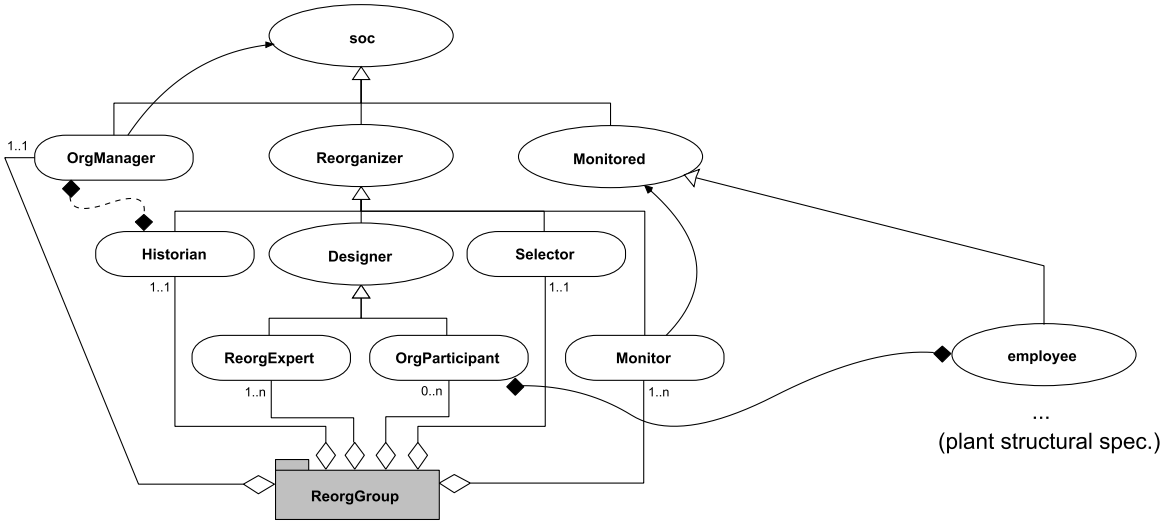

Abbildung 10.17: Reorganisations-Gruppe in $\mathcal{M}$OISE$^+$ (ursprünglich von Hübner et. al. [HSB04] und hier angepasst)

Auf weitere Feinheiten des Reorganisationskonzeptes für $\mathcal{M}$OISE$^+$ wird hier nicht eingegangen. Das beschriebene Herangehensweise stellt einen allgemeinen Rahmen für Reorganisation dar, die Inhalte von Reorganisationsmaßnahmen sind natürlich anwendungsabhängig.

10.3.1.3 Eine $\mathcal{M}$OISE$^+$-Organisation in Aktion

Hübner et. al. [HSB06] stellen mit $\mathcal{S}$-$\mathcal{M}$OISE$^+$ eine Middleware für das Deployment von $\mathcal{M}$OISE$^+$-Modellen als Agentenorganisationen vor. Die grobe $\mathcal{S}$-$\mathcal{M}$OISE$^+$-Architektur ist in Abbildung 10.18 veranschaulicht.

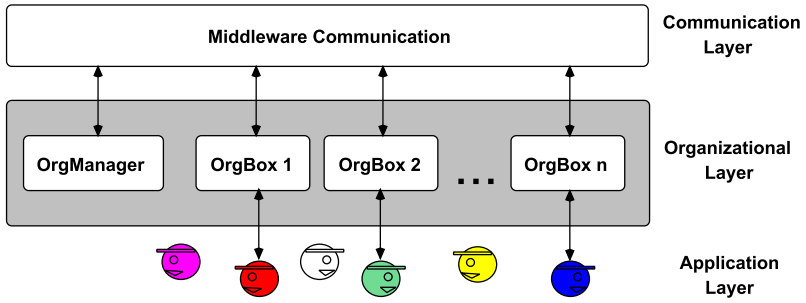

Abbildung 10.18: $\mathcal{S}$-$\mathcal{M}$OISE$^+$ Middleware-Konzept

Der wesentliche Unterschied zum Middleware-Konzept von SONAR aus Unterabschnitt 10.2.1.5 besteht in der Existenz eines zentralen Middleware-Managers (OrgManager[10]).

[10] Der hier abgebildete OrgManager ist eine Middleware-Komponente und hat nichts mit der orgManager-Rolle der Reorganisationsgruppe aus Abbildung 10.17 zu tun.

Wie im Falle von SONAR verbinden sich Domänenagenten mit einem Middleware-Proxy, um als Mitglieder der Organisation auftreten zu können. Bei SONAR sind dies die Positionsagenten. Jeder Positionsagent hat eine lokale, auf seine spezifische Position und ihren Kontext eingeschränkte Perspektive auf die Organisationsspezifikation und interagiert mit anderen Positionsagenten für positionsübergreifendes Teamwork. Zu diesem Zweck sind Positionsagenten mit einem höheren Grad an „Intelligenz“ ausgestattet als die Middleware-Proxies im Falle von $\mathcal{S}$-$\mathcal{M}$OISE$^+$. Eine OrgBox stellt nicht viel mehr als eine Schnittstelle zur Middleware zur Verfügung. Alle Bestrebungen von Agenten, als Organisationsmitglied zu handeln (Annahme von Rollen, Instanziierung von Schemata, Annahme von Missionen, Terminieren einer Zielverfolgung) werden von den OrgBox-Proxies an den zentralen Middleware-Manager weitergeleitet. Dieser hält die gesamte Organisationsspezifikation und den Organisationszustand vor und überprüft Handlungsanfragen auf Zulässigkeit. Außerdem kann der Middleware-Manager die OrgBox-Proxies nutzen, um Aufforderungen an die Domänenagenten heranzutragen (Aufforderung zur Annahme von Missionen, Aufforderung zur Verfolgung von Zielen).

Gegenüber dem Middleware-Ansatz von SONAR birgt $\mathcal{S}$-$\mathcal{M}$OISE$^+$ aufgrund des zentralen Managers die Gefahr eines Flaschenhalses. Nicht zuletzt deswegen wird in aktuelleren Arbeiten von Hübner et. al. [HBKR10] ein Ansatz beschrieben, bei dem der zentrale Middleware-Manager und die durch ihn erbrachten Dienste durch eine Reihe organisatorischer Artefakte und deren Funktionen ersetzt wird (vergleiche auch die Erläuterungen zum Agenten & Artefakte-Ansatz aus Kapitel 5).

10.3.2 Organisationseinheiten gemäß $\mathcal{M}$OISE$^+$

Im Folgenden wird die Ausgestaltung von Organisationseinheiten entlang von $\mathcal{M}$OISE$^+$-Spezifikationen erläutert. Im Gegensatz zu der detaillierten Betrachtung von SONAR-Organisationseinheiten sind die Ausführungen dieses Teilabschnittes auf eine abstraktere Charakterisierung und darauf aufbauende Bewertung ausgerichtet.

10.3.2.1 Operation, Integration und Governance

Der erste Schritt für die Beschreibung von $\mathcal{M}$OISE$^+$-Organisationseinheiten besteht wie bereits für SONAR-Organisationseinheiten darin, die analytischen Aspekte der Governance, Integration und Operation zu identifizieren. Dabei wird an dieser Stelle die Sichtweise aus Abbildung 10.19 propagiert. Wie zuvor für SONAR-Organisationseinheiten (Abbildung 10.7) ist der Aspekt der Peripherieaktivitäten zunächst ausgegraut und wird weiter unten gesondert behandelt. Desweiteren werden auch hier die Aspekte des „Einstellens/Entlassens“ von Akteuren nicht berücksichtigt. Dabei ist gerade das Besetzen von Rollen mit Akteuren in $\mathcal{M}$OISE$^+$-Organisationseinheiten komplizierteren Bedingungen unterworfen als das Besetzen von Positionen mit Akteuren in SONAR-Organisationseinheiten. Auch dieser Aspekt wird weiter unten etwas genauer beleuchtet.

Die wesentlichen Integrationsaktivitäten einer $\mathcal{M}$OISE$^+$-Organisationseinheit betreffen die Instanziierung und Ausführung der sozialen Schemata. Als Integrationsakteure fungieren dabei die Akteure, die Rollen mit Erlaubnisbeziehungen zu den Wurzelmissionen der Schemata innehaben. Bei den Operationsakteuren handelt es sich dementsprechend um die Akteure, die Rollen mit Erlaubnis-/Verpflichtungsbeziehungen zu den übrigen Missionen besetzen. Die Instanziierung eines Schemas entspricht in gewisser Weise der Formation eines Teams und des zugehörigen Teamplans in SONAR-Organisationseinheiten. Es ist allerdings einfacher. Wäh-

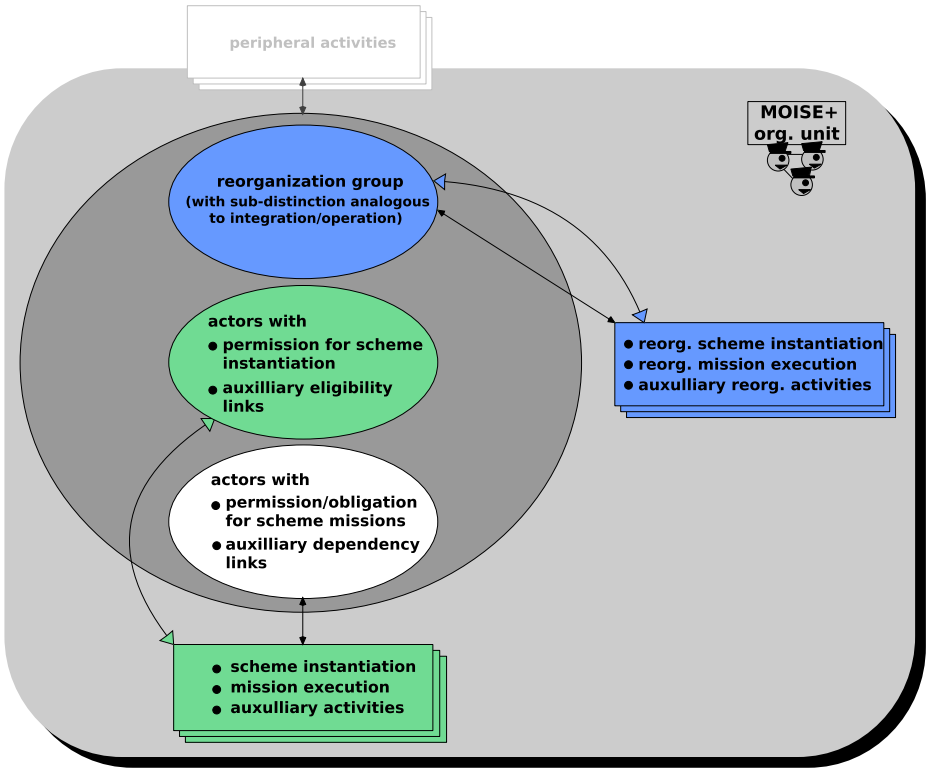

Abbildung 10.19: Grobmodell einer $\mathcal{M}$OISE$^+$-Organisationseinheit

rend bestimmte Delegationsentscheidungen zu unterschiedlich strukturierten SONAR-Teams führen können, so sind Instanzen eines $\mathcal{M}$OISE$^+$-Schemas immer gleich strukturiert. Im Falle einer $\mathcal{M}$OISE$^+$-Organisationseinheit fordert der für eine Schemainstanz verantwortliche Integrationsakteur andere Akteure auf, bestimmte Missionen zu übernehmen. Je nach Erlaubnis- oder Verpflichtungsbeziehung zu der jeweiligen Mission kann oder muss ein Akteur einer Missionsaufforderung nachkommen.

Die Ausführung eines Schemas entspricht der Ausführung des Plans, der durch den Zielbaum des Schemas aufgespannt wird. Ein solcher Zielbaum repräsentiert allerdings keine unmittelbare Interaktionsvorschrift für die an der Schemainstanz beteiligten Akteure. Allerdings zeigt Abbildung 10.20, wie die verschiedenen in Schemata auftretenden Ziel/Teilziel-Beziehungen (Sequenz, Parallelität, Auswahl) in Interaktionsmuster überführt werden können. Darauf aufbauend kann ein gesamtes Schema in eine Interaktionsspezifikation überführt werden. Die Interaktionsrollen ergeben sich hier direkt aus den Schemamissionen.

Zusätzlich zu Aktivitäten in direktem Zusammenhang mit Schemata können weitere Integrationsaktivitäten relevant sein. Dies resultiert aus den in Unterabschnitt 10.3.1.1 beschriebenen Rollenbeziehungen, die zusätzlich zu Vererbung und Kompatibilität modelliert werden können. Es wurde bereits darauf hingewiesen, dass die Semantik dieser zusätzlichen Beziehungen nicht unbedingt klar ist. In dem Beispiel aus Abbildung 10.16 wird von den Möglichkeiten, die in der $\mathcal{M}$OISE$^+$-Literatur vorgeschlagen werden, die Autoritätsbeziehung als zusätzliche Rollenbeziehung verwendet. Es gibt verschiedene Möglichkeiten, in Form welcher Aktivitäten

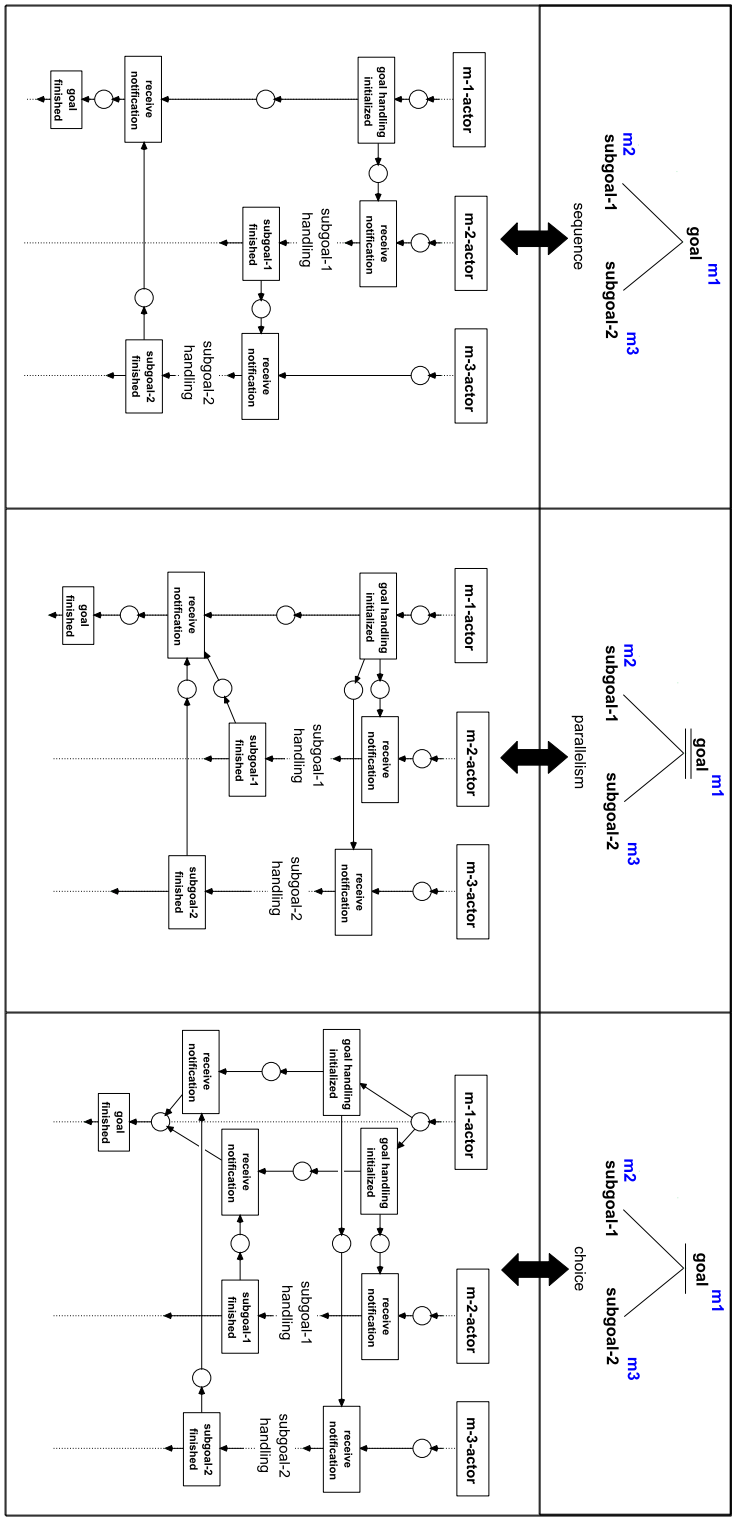

Abbildung 10.20: Überführung von $\mathcal{M}$OISE$^+$-Plänen in Interaktionsmuster

sich diese Beziehung manifestieren kann.[11] Für die Autoritätsbeziehung ist naheliegend, Aktivitäten für die (offizielle) Erteilung von Anweisungen und Aktivitäten für die Entbindung von Akteuren von ihren Rollen vorzusehen. So manifestiert sich Autorität eines Akteurs über einen anderen (resultierend aus ihren jeweiligen Rollenzugehörigkeiten) auf sehr einfache, aber effektive Art und Weise.

Der Aspekt der Governance ist in Abbildung 10.19 mit der Reorganisationsgruppe einer $\mathcal{M}$OISE$^+$-Organisation und den mit ihr assoziierten Aktivitäten vertreten. Für die Governance-Aktivitäten gilt dabei im Grunde das gleiche wie für die Integrationsaktivitäten. Auch hier gibt es Akteure, die Rollen mit Erlaubnisbeziehungen zu Wurzelmissionen von Schemata besetzen, und Akteure, die Rollen mit Erlaubnis-/Verpflichtungsbeziehungen zu den übrigen Missionen besetzen. Allerdings handelt es sich hier um Rollen der Organisationsgruppe und um Reorganisationsschemata. Eigentlich existiert also für die Menge der Governance-Akteure eine weitere Zweiteilung analog der Unterscheidung von Integrations- und Operationsakteuren. Allerdings ist dieser Aspekt in der Abbildung nur angedeutet.

10.3.2.2 Systemregeln einer $\mathcal{M}$OISE$^+$-Organisationseinheit

Wie bereits in Unterabschnitt 10.2.2.2 für SONAR-Organisationseinheiten beschrieben, werden auch im Falle des operationalen Modells einer $\mathcal{M}$OISE$^+$-Organisationseinheit Systemregeln in Form einer PROLOG-Regelbasis eingebettet. Damit ist auch der in Unterabschnitt 10.3.1.3 beschriebene Middleware-Ansatz für das Deployment von $\mathcal{M}$OISE$^+$-Organisationen in Multiagentensystem hinfällig. Die Beschränkung und Leitung von Verhalten entlang der Organisationsspezifikation werden von der Netzstruktur des operationalen Organisationseinheitsmodells in Kombination mit den Systemregeln übernommen. Auch hier gilt die Unterscheidung von Paradigmaregeln und Spezifikations- sowie Zustandsfakten.

- *Spezifikationsfakten:* Diese Fakten betreffen die konkrete strukturelle, funktionale sowie deontische Spezifikation einer $\mathcal{M}$OISE$^+$-Organisationseinheit. Hier werden Rollenbeziehungen sowie die Gruppierung von Rolleninhabern in Gruppen beschrieben. Darüber hinaus wird festgelegt, welche Schemata es gibt, aus welchen Missionen sich diese jeweils zusammensetzen und welche Rollen-Mission-Assoziationen existieren.

- *Zustandsfakten:* Zustandsfakten betreffen insbesondere die Besetzung von Rollen mit Agenten. Dabei müssen deutlich komplexere Rahmenbedingungen berücksichtigt werden als es im Falle von SONAR für die Besetzung von Positionen mit Agenten der Fall ist. Allerdings betrifft dieser Aspekt eher die weiter unten erläuterten Paradigmaregeln.

  Weitere Zustandsfakten betreffen den Umstand, ob die Instanziierung eines Schemas (die Zuteilung von Missionen) noch im Aufbau begriffen oder bereits abgeschlossen ist.

  Denkbar sind zusätzliche Zustandsfakten, die im Zusammenhang mit zusätzlichen Rollenbeziehungen stehen.

- *Paradigmaregeln:* Die Paradigmaregeln operieren auf den Spezifikations- und Zustandsfakten. Wie im Falle einer SONAR-Organisationseinheit besteht die Hauptaufgabe der Paradigmaregeln darin, die Zulässigkeit von Aktivitätsaktivierungen zu prüfen. Bei Aktivitäten, die der Instanziierung von Schemata dienen, wird geprüft, ob der jeweils initiierende Akteur eine Rolle mit Erlaubnisbeziehung zur Wurzelmission des jeweiligen

[11] Gegebenenfalls manifestieren sich Beziehungsspezifikationen ohne klare Semantik durch Interpretation und Normen auf informeller Ebene.

Schemas besetzt. Bei Aktivitäten, die der Ausführung einer Schemainstanz dienen, wird geprüft, ob das Schema korrekt instanziiert wurde (ob alle Missionen von Agenten mit entsprechenden Erlaubnis-/Verpflichtungs-Beziehungen besetzt wurden).

Ein weiterer wichtiger Punkt der Systemregeln ist die Überprüfung von Wohlgeformtheit der Besetzung von Rollen mit Agenten und darauf aufbauend der Komposition von Gruppen. Bei der Besetzung von Rollen müssen Kompatibilitätsbeziehungen berücksichtigt werden. Bei der Komposition von Gruppen gilt es die Kardinalitätsrichtlinien zu berücksichtigen.

Weitere Paradigmaregeln können eingeführt werden, um zusätzlichen Rollenbeziehungen gerecht zu werden. Dies betrifft beispielsweise die im Zusammenhang mit einer Autoritätsbeziehung stehende Berechtigung zur Aktivierung von Aktivitäten zur Entlassung von Akteuren.

10.3.2.3 $\mathcal{M}$OISE$^+$-Multiorganisationssysteme

Es stellt sich die Frage, ob $\mathcal{M}$OISE$^+$ analog zu SONAR Konzepte mitbringt, die einen unmittelbaren Übergang von der Betrachtung einer Organisationseinheit zur Betrachtung von Multiorganisationssystemen unterstützen. Im Falle von SONAR besteht diese Unterstützung vor allem in der Verzahnung von Aufgaben- und Interaktionsverfeinerungen. Diese Verzahnung kann genutzt werden, um zu einem klaren Bild von ebenenübergreifendem Systemverhalten zu gelangen wie es in den Abbildungen 10.14 und 10.15 illustriert ist.

Im Falle von $\mathcal{M}$OISE$^+$ bietet sich auf den ersten Blick das Gruppenkonzept als Grundlage für die Aufteilung eines Gesamtsystems in verschiedene Organisationseinheiten an. Allerdings ist hier explizit festzuhalten, dass $\mathcal{M}$OISE$^+$-Gruppen im Vergleich zu der soziologischen Betrachtung von Kollektivitäten in den Kapiteln 3 und 4 lediglich die Funktion als *Kontext*, nicht jedoch die Funktion als *kollektiver Akteur* mit sich bringen. Die Auffassung von Gruppen als Organisationseinheiten ist somit nicht unmittelbar mit dem $\mathcal{M}$OISE$^+$-Ansatz vereinbar beziehungsweise würde eine konzeptionelle Erweiterung erfordern.

Eine Verzahnung von Aufgaben- und Verhaltensverfeinerungen wie im Falle SONAR gibt es im $\mathcal{M}$OISE$^+$-Kontext nicht. Stattdessen fehlt eine explizite Verhaltensspezifikation gänzlich, auch wenn sie sich auf verschiedene Art und Weise aus den Schemata herleiten lässt (wie in Abbildung 10.20 exemplarisch demonstriert). Unter Umständen kann dies im Hinblick auf die Einführung verschiedener Systemebenen allerdings auch positiv ausgelegt werden. Eine Option besteht darin, dass kollektive Akteure Missionen übernehmen und die Verfolgung der Missionsziele intern jeweils durch eigene Schemata realisieren. Eine direkte Hilfestellung, welches ebenenübergreifende Verhalten daraus resultiert, ergibt sich aus dem $\mathcal{M}$OISE$^+$-Ansatz heraus nicht. Andererseits besteht dadurch auch ein größerer Spielraum für die Realisierung der Kopplung verschiedener $\mathcal{M}$OISE$^+$-Organisationseinheiten.

10.4 Organisationseinheiten nach ISLANDER

Zum Thema *Electronic Institutions* gibt es einen recht großen Unterbau an veröffentlichten Arbeiten. Die Ausführungen dieses Abschnittes berufen sich vornehmlich auf drei Arbeiten von Esteva et. al. [EPS02, EdlCS02, ERARA04], in denen eine Beschreibungssprache für elektronische Institutionen, der darauf aufbauende Editor ISLANDER und die Middleware AMELI für

das Multiagentensystem-Deployment von elektronischen Institutionen beschrieben werden.[12] Das in diesem Abschnitt zur Illustration der Konzepte herangezogene Beispiel einer elektronischen Institution aus Abbildung 10.21 ist angelehnt an die Markt- und Auktionszenarien aus der Originalliteratur, stellt jedoch eine vereinfachte Fassung des Autors der vorliegenden Arbeit dar.

10.4.1 Der ISLANDER-Ansatz: Grundlegende Konzepte und Modelldimensionen

Der Begriff der elektronischen Institution wurde bereits in Kapitel 5 im Zusammenhang mit den organisationsorientierten Ansätzen der Multiagentensystementwicklung aufgegriffen. Bei ISLANDER handelt es sich eigentlich um ein Werkzeug für die Erstellung und Analyse von Modellen elektronischer Institutionen.[13] Häufig wird der Begriff ISLANDER allerdings synonym für den Ansatz der Modellierung von elektronischen Institutionen an sich verwendet. So wird auch im vorliegenden Abschnitt verfahren. Zunächst einmal bringt der Übergang von SONAR- und $\mathcal{M}$OISE$^+$-Modellen zu ISLANDER-Modellen einen terminologischen Wandel von der Organisation zur Institution mit sich. Der Zusammenhang und insbesondere die Überlagerung der soziologischen Konzepte der Organisation und der Institution wurden bereits in Kapitel 4 behandelt. In Übereinstimmung mit den dortigen Ausführungen ist der qualitative Hauptunterschied zwischen SONAR-/$\mathcal{M}$OISE$^+$-Modellen und ISLANDER-Modellen das Fehlen einer funktionalen Dimension im Falle von ISLANDER. Es gibt keine Modellelemente, die globale Ziele/Aufgaben der Kollektivität beschreiben. Als implizites globales Ziel lässt sich höchstens identifizieren, das Verhalten der Institutionsmitglieder entlang ganz spezifischer Vorgaben (Interaktionsspezifikationen und Zusammenhänge zwischen diesen) auszurichten. Insofern liegt der Fokus auf der Regulierung des Verhaltens der beteiligten Akteure.

Die Modelldimensionen von ISLANDER umfassen eine (1) dialogische Spezifikation, eine (2) strukturelle Spezifikation (auch wenn diese gänzlich anderer Art ist als im Falle von SONAR/$\mathcal{M}$OISE$^+$), eine (3) interaktionsbezogene Spezifikation sowie eine (3) normative Spezifikation. Die dialogische Spezifikation dient der Definition einer Ontologie, einer Kommunikationssprache sowie einer Sprache zur Wissensrepräsentation. Diese Modelldimension wird im Folgenden nicht weiter betrachtet. Dort, wo sie für die übrigen Dimensionen relevant ist, wird eine abstraktere Beschreibung verwendet. Die übrigen Dimensionen werden in der Folge anhand des Beispiels aus Abbildung 10.21 beschrieben.

10.4.1.1 Strukturelle, interaktionsbezogene und normative Spezifikation

Im Falle von $\mathcal{M}$OISE$^+$ und SONAR besteht die strukturelle Spezifikation aus der Definition von „Platzhaltern“, die in einem laufenden System mit Agenten besetzt werden (Rollen, Positionen), und Beziehungen zwischen diesen Platzhaltern. Die strukturelle Spezifikation im Falle von ISLANDER ist anderer Natur. Sie setzt verschiedene Interaktionsszenarien – sogenannte *Szenen* – in Beziehung. Ein Beispiel ist in der linken Hälfte von Abbildung 10.21 zu sehen. Von Agenten einzunehmende Rollen sind dabei auch relevant: Es wird spezifiziert, wann ein Agent als Inhaber welcher Rollen auf welche Art und Weise zwischen Szenen wechseln kann.

[12] Der Autor der vorliegenden Arbeit konnte in der herangezogenen Originalliteratur keine eindeutige Auflösung darüber finden, ob es sich bei den Begriffen ISLANDER und AMELI um Akronyme handelt.

[13] Im Internet als Download unter http://e-institutions.iiia.csic.es/islander/pub/index.html verfügbar.

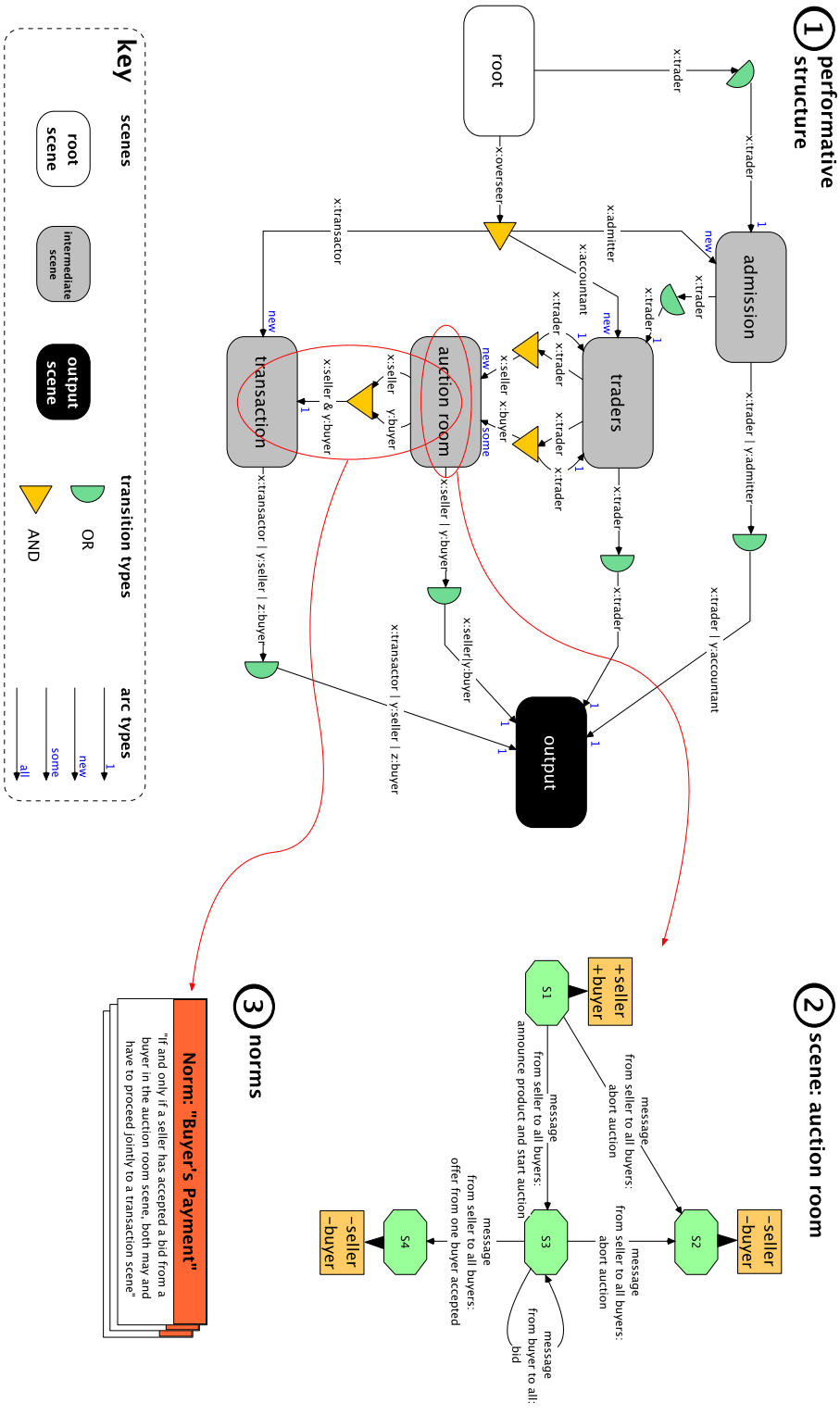

Abbildung 10.21: ISLANDER Modelldimensionen an einem Beispiel

Zu diesem Zweck werden *Transitionen* eingeführt.[14] Zwischen Szenen und Transitionen sowie zwischen Transitionen und Szenen werden Kanten definiert. Diese können als *Routingkanäle* aufgefasst werden, durch die Agenten von Szene zu Szene geleitet werden. Dabei gibt es verschiedene Möglichkeiten, dieses Routing als Modellierer zu beeinflussen.

- Kanten sind mit Tupeln jeweils bestehend aus einer Agentenvariablen und einem Rollenbezeichner beschriftet (geschrieben als x:roleA für eine Agentenvariable x und eine Rolle roleA). Um einen Agenten von Szene zu Szene zu leiten, taucht eine Agentenvariable immer mindestens auf einer eingehenden und auf einer ausgehenden Kante einer Transition auf. Die Variable kann dabei allerdings mit verschiedenen Rollen assoziiert sein. Auf diese Weise wird modelliert, dass ein Agent, der eine bestimmte Rolle in einer Szene spielt, als Inhaber einer anderen Rolle in eine andere Szene wechseln darf oder muss (die weiter unten beschriebenen Szenenspezifikationen legen zusätzliche Einschränkungen fest, wann ein solcher Wechsel möglich ist).

- Eine Kante kann mit mehreren Agentenvariable/Rolle-Tupeln beschriftet sein. Sind diese disjunktiv verknüpft (hier geschrieben mit | als Trennsymbol), so kann ein Inhaber jeder einzelnen der aufgeführten Rollen den durch die Kante modellierten Kanal *alleine* benutzen. Sind die Tupel konjunktiv verknüpft (hier geschrieben mit & als Trennsymbol), so muss für jede der aufgeführten Rollen ein Inhaber gefunden werden und alle Rolleninhaber müssen den Kanal *gemeinsam* benutzen.[15]

- Die disjunktive beziehungsweise konjunktive Verknüpfung von Agent/Rolle-Tupeln stellt ein erstes Mittel der Synchronisierung beziehungsweise Nebenläufigkeit von „Agentenbewegungen“ zwischen Szenen dar. Zusätzlich dazu werden Transitionen in *UND*- und *ODER*-Transitionen aufgeteilt. Bei einer ODER-Transition kann jeder durch eine Kante modellierte Eingangskanal unabhängig von etwaigen weiteren Eingangskanten verwendet werden. Taucht eine Agentenvariable auf mehreren Ausgangs-Kanten auf, so kann der jeweilige Agent *wählen*, welchem der möglichen Kanäle er folgt.

  Im Falle einer UND-Transition müssen alle durch Kanten modellierten Eingangskanäle gemeinsam benutzt werden. Taucht dabei eine Agentenvariable auf mehreren Ausgangskanten auf, so muss der jeweilige Agent *allen* der möglichen Kanäle folgen.

- Kanten von Transitionen zu Szenen sind zusätzlich mit genau einem Element der Menge {1, some, all, new} beschriftet. Dieser Aspekt steht in Zusammenhang mit *Szeneninstanzen*. Szenen sind Interaktionsspezifikationen/-schablonen und können in einer laufenden elektronischen Institution mehrfach instanziiert vorliegen. Die oben genannten Beschriftungen legen fest, ob beim Folgen eines Ausgangskanals einer Transition eine, mehrere oder alle Instanzen der jeweiligen Szene betreten werden oder aber ob eine neue Szeneninstanz erstellt wird.

Anhand dieser Erklärungen liest sich die Struktur-Spezifikation aus dem Beispiel in Abbildung 10.21 wie folgt. Jede elektronische Institution beinhaltet eine spezielle Wurzelszene (Szene root), die den Einstiegspunkt aller Agenten darstellt, und eine speziellen Austrittsszene

[14] Diese ISLANDER-Transitionen sind ausdrücklich nicht mit Petrinetztransitionen zu verwechseln. Sie sind mit unterschiedlicher Semantik assoziiert.

[15] Tatsächlich können Kanten mit disjunktiven Normalformen, bei denen Agent/Rolle-Tupel die Funktion der Literale einnehmen, beschriftet sein. Hier werden allerdings nur reine Disjunktionen oder Konjunktionen von Agent/Rolle-Tupeln betrachtet.

(Szene output), die alle Agenten zum Verlassen der Institution durchlaufen. Ausgehend von der Wurzelszene können Agenten in zwei verschiedenen Rollen weitere Szenen betreten. Agenten in der Rolle overseer nehmen eine Verwaltungsfunktion ein. Ein overseer-Agent startet neue Szeneninstanzen für die Zulassung neuer Händler (Szene admission), die Verwaltung vorhandener Händler (Szene traders) sowie für Transaktionen zwischen Händlern (Szene transaction). Agenten in der Rolle trader hingegen können ausgehend von der Wurzelszene zunächst lediglich eine Instanz der Szene admission betreten. Werden sie dort als Händler zugelassen, so wechseln sie in eine Instanz der Szene traders für die Verwaltung zugelassener Händler. Andernfalls wechseln sie in die Austrittsszene. Ausgehend von der traders-Szene können Agenten Auktionen initiieren oder beitreten. Auktionen finden im Kontext von Instanzen der auction room-Szene statt. Agenten, die eine Auktion initiieren, wechseln ihre Rolle von trader zu seller und initiieren eine neue Instanz der auction room-Szene. Agenten, die an Auktionen teilnehmen, wechseln ihre Rolle von trader zu buyer und treten gegebenenfalls mehreren Instanzen der auction room-Szene bei. Allerdings zeigt das Beispiel an dieser Stelle auch, dass Agenten gleichzeitig Rollen in verschiedenen Szeneninstanzen einnehmen können. Agenten, die Auktionen initiieren oder beitreten, bleiben weiterhin in der Rolle trader in einer traders-Szene. Nach der Beendigung einer Auktion wechseln der Verkäufer sowie der Käufer, der den Zuschlag erhalten hat, in eine Instanz der transaction-Szene, um die Transaktion zu vollziehen. Alle weiteren Käuferagenten (und bei Auktionsabbruch auch der Verkäufer) wechseln in die Austrittsszene. Da die jeweiligen Agenten allerdings zusätzlich noch in Instanzen der traders-Szene aktiv sind, wird hier deutlich, dass die Austrittsszene als Auffang-Becken für die Beendigung bestimmter *Lebenslinien* der Agenten dient. Erst wenn ein Agent aus allen seinen Szeneninstanzen in die Austrittsszene gewechselt ist, hat er die Institution tatsächlich verlassen.

Speziell anhand des Falles der auction room-Szene wird deutlich, dass die strukturelle Spezifikation einer elektronischen Institution alleine nicht ausreichend ist. Zwar wird hier modelliert, welche Rollen in dieser Szene möglich sind und dass Agenten die Szene auf verschiedene Art und Weise betreten und verlassen können. Allerdings bleibt hier noch unklar, wie Ein- und Austritte tatsächlich mit dem Verlauf der Auktion zusammenhängen. Zu diesem Zweck bietet ISLANDER zwei weitere Modelldimension an. Zunächst zeigt der rechte obere Teil von Abbildung 10.21 die Spezifikation der Szene auction room und damit einen Teil der Interaktionsspezifikation des ISLANDER-Modells. Hier wird deutlich, dass Interaktionen im Falle von ISLANDER anders modelliert werden als es in der vorliegenden Arbeit bisher üblich war. Bislang wurden Interaktionen/Aktivitäten angelehnt an Interaktionsdiagramme der UML modelliert: Für jede Interaktionsrolle wurde eine eigene Lebenslinie in vertikaler Richtung modelliert und die verschiedenen Lebenslinien nebeneinander angeordnet, so dass eine horizontale Verbindung als Nachrichtenaustausch zwischen Rollen interpretiert wurde. Für ISLANDER-Organisationseinheiten im folgenden Unterabschnitt wird eine andere als die bisherige Herangehensweise an die Aktivitätsmodellierung notwendig.

Eine Szene wird in Form von Interaktionszuständen und Übergängen zwischen diesen beschrieben. Übergänge finden in Form von Nachrichtenversand statt.[16] Während in den bisher betrachteten Interaktionsspezifikationen jede Rolle genau einmal auftritt, so können Rollen in Szenen in beliebiger Multiplizität auftreten. Im Falle der Szene auction room können beliebig viele Käufer auftreten (allerdings nur ein einziger Verkäufer da – wie oben beschrieben – jeder Käufer eine eigene Szeneninstanz eröffnet). Dabei gibt es ausgewiesene Zustände, in

[16] Der Versand von Nachrichten ist hier in natürlicher Sprache beschrieben. Eigentlich bietet die hier nicht behandelte dialogische Modelldimension von ISLANDER eine Kommunikationssprache an.

denen Agenten in bestimmten Rollen der Szeneninstanz beitreten oder sie verlassen können. Im Beispiel der auction room-Szene können im Startzustand s1 zunächst der Verkäufer und beliebig viele weitere Käufer beitreten. Ein Zustandsübergang findet statt wenn der Verkäufer die Auktion abbricht (Übergang nach s2) oder wenn er die Auktion eröffnet (Übergang nach s3). In beiden Fällen versendet der Verkäufer eine Nachricht an *alle* Käufer. Im Falle des Abbruchs der Auktion können Verkäufer und Käufer die Szeneninstanz verlassen. Im Falle des Auktionsstarts hingegen sind ausgehend vom neuen Zustand s3 drei Zustandsübergänge möglich: Die Auktion kann abgebrochen werden (Übergang nach s2), ein Käufer gibt ein Gebot ab (Verbleib in s3) oder der Verkäufer akzeptiert ein Gebot (Übergang nach s4). Im Falle des Akzeptierens eines Gebotes ist die Auktion beendet und Verkäufer sowie Käufer können die Szeneninstanz im Zustand s4 verlassen. Es wird deutlich, dass auf diese Weise eine sehr kompakte Interaktionsspezifikation unter Berücksichtigung beliebig vieler Interaktionsteilnehmer möglich ist.

Eine Szenenspezifikation legt Zustände fest, in denen Agenten in bestimmten Rollen einer Instanz der Szene beitreten oder sie verlassen können. Allerdings reicht dies nicht aus. So ist beispielsweise für die auction room-Szene festgelegt, in welchen Zuständen Agenten Instanzen der Szene verlassen können. Allerdings sind in der Strukturspezifikation zwei Möglichkeiten für das Verlassen der auction room-Szene vorgesehen. Welche der Möglichkeiten für Agenten beim Verlassen der Szeneninstanz zulässig sind, ergibt sich nicht aus der Szenenspezifikation. Diese Lücke wird durch die normative Modelldimension von ISLANDER geschlossen. Normen spezifizieren, welche Effekte die Handlungen von Agenten in Szenen haben. Auf der rechten unteren Seite von Abbildung 10.21 ist das Beispiel einer Norm gegeben.[17] Diese Norm "Buyer's Payment" legt eindeutig fest, dass genau der Verkäufer und der Käufer mit dem höchsten (und akzeptierten) Gebot gemeinsam in eine Instanz der Szene transaction wechseln müssen. Für alle anderen Käufer bleibt demnach nur der Weg in die Austrittsszene.

Im Gegensatz zu SONAR und $\mathcal{M}$OISE$^+$ gibt es im Falle elektronischer Institutionen nach dem Wissensstand des Autors der vorliegenden Arbeit keinen ausgereiften Ansatz der Reorganisation/Adaption. Zwar führen Campos et. al. [CLSRAE08] *autonomic electronic institutions* ein und bezeichnen damit elektronische Institutionen, bei denen Agenten als sogenannte *Adaptionsmanager* Modifikationen an der Spezifikation der elektronischen Institution vornehmen, um bestimmte institutionelle Ziele zu erreichen. Allerdings machen die Ausführungen nur einen recht kleinen Teil des Artikels aus und sowohl das Konzept von institutionellen Zielen als auch das Konzept der Transitionsfunktionen, die den Wandel einer elektronischen Institution beschreiben, bleiben sehr vage und gehen nicht über eine erste grobe Idee hinaus.

10.4.1.2 Elektronische Institutionen in Aktion

Analog zu SONAR- und $\mathcal{M}$OISE$^+$-Modellen existiert auch für ISLANDER-Modelle die Unterstützung durch eine Middleware, die für die Verkörperung der Organisation/Institution in einem laufenden Multiagentensystem sorgt. Abbildung 10.22 zeigt die grundlegende Architektur von AMELI, der Middleware für elektronische Institutionen.

Das Konzept gleicht dem von $\mathcal{S}$-$\mathcal{M}$OISE$^+$. In Falle von AMELI übernehmen die *Governors* die Rolle der *OrgBoxes* von $\mathcal{S}$-$\mathcal{M}$OISE$^+$ als Schnittstellen der Domänenagenten zur Middleware-Schicht. Während im Falle von $\mathcal{S}$-$\mathcal{M}$OISE$^+$ allerdings ein einziger *OrgManager*-Agent die gesamte Verwaltung der Organisationsspezifikation und des Organisationszustandes

[17] Auch hier wird natürliche Sprache als Ersatz für die Formulierung mittels der durch die dialogische Modelldimension eigentlich bereitgestellten Sprache verwendet.

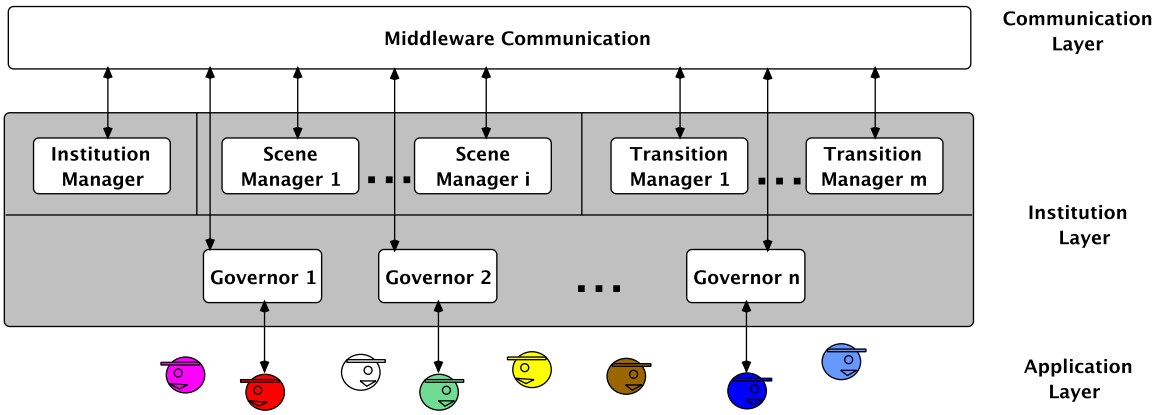

Abbildung 10.22: AMELI Middleware-Konzept

übernimmt, gibt es bei AMELI mehrere Manageragenten. Ein *Institutionsmanager* startet die gesamte elektronische Institution, autorisiert Domänenagenten zum Beitritt und verwaltet die Instanziierung von Szenen. Jede mit einer Szeneninstanz assoziierte Interaktion wird von einem eigenen *Szenenmanager* überwacht. Zusätzlich gibt es zu jeder Transition einen eigenen *Transitionsmanager*, der das Routing von Agenten zwischen Szeneninstanzen gemäß der Spezifikation übernimmt. Ein Governor verwaltet die Teilnahme des mit ihm assoziierten Domänenagenten an gegebenenfalls multiplen Szeneninstanzen und behält eine Übersicht über die seinen Agenten betreffenden Normen.

10.4.2 Organisationseinheiten gemäß ISLANDER

Im Folgenden wird die Ausgestaltung von Organisationseinheiten entlang von ISLANDER-Spezifikationen erläutert. Analog zu der Betrachtung von $\mathcal{M}$OISE$^+$-Organisationseinheiten im vorigen Abschnitt sind die Ausführungen dieses Teilabschnittes auf eine abstrakte Charakterisierung und darauf aufbauende Bewertung ausgerichtet.

10.4.2.1 Operation, Integration und Governance

Der erste Schritt für die Beschreibung von ISLANDER-Organisationseinheiten besteht wie bereits für die bisher betrachteten Typen von Organisationseinheiten darin, die analytischen Aspekte der Governance, Integration und Operation zu identifizieren. Allerdings wurde im vorausgehenden Teilabschnitt bereits darauf hingewiesen, dass spezifische Mechanismen für die Laufzeitanpassung/-modifikation von elektronischen Institutionen bisher nicht in ausreichendem Reifegrad existieren. Daher wird der Governance-Aspekt im Falle von ISLANDER-Organisationseinheiten nicht betrachtet. Natürlich ist es auch hier möglich, die in Abbildung 10.11 illustrierte Herangehensweise mit sozialen Akteuren als Governance-Akteuren zu wählen. Welche Systemregeln dabei zur Disposition stehen, wird weiter unten in diesem Teilabschnitt behandelt.

Für die Identifikation der Aspekte der Integration und Operation gibt es wiederum verschiedene Möglichkeiten. Abbildung 10.23 zeigt eine sehr vereinfachte Sichtweise. In diesem Fall wird lediglich eine Art der Integrationsaktivitäten betrachtet, nämlich die Ausführung von

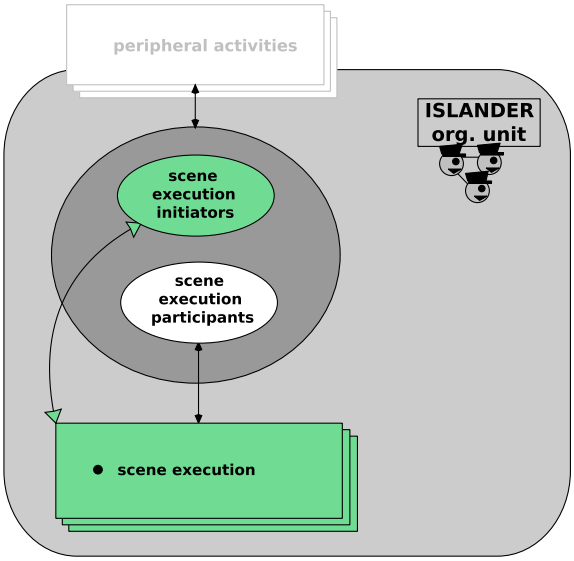

Abbildung 10.23: Grobmodell einer ISLANDER-Organisationseinheit

Szeneninstanzen. Als initiierende Integrationsakteure dienen dabei die Inhaber von Rollen, die zur Instanziierung von Szenen berechtigt sind. Akteure, die existierenden Szenen beitreten, sind die Operationsakteure.

Diese Sichtweise auf Integration und Operation von ISLANDER-Organisationseinheiten wirkt zunächst sehr schlicht und gegebenenfalls sogar *zu* vereinfachend. Vor allem fällt auf, dass der Aspekt des Wechsels zwischen Szenen über Transitionen ausgeblendet ist. Es möglich, das abstrakte Modell um diesen Aspekt zu erweitern und damit eine andere Identifizierung der Aspekte der Integration und Operation vorzunehmen. An dieser Stelle soll jedoch das einfache Modell einer ISLANDER-Organisationseinheit aus der Abbildung herhalten. Es wird bereits hier deutlich, dass dieses abstrakte Modell viele Aspekte verbirgt, die im Kontext des operationalen Modells auf die Definition von Systemregeln verlagert werden.

10.4.2.2 Systemregeln und Funktionsweise einer ISLANDER-Organisationseinheit

Wie bereits für SONAR- und $\mathcal{M}$OISE$^+$-Organisationseinheiten beschrieben, werden auch im Falle des operationalen Modells einer ISLANDER -Organisationseinheit Systemregeln in Form einer PROLOG-Regelbasis eingebettet. Damit wird der in Unterabschnitt 10.4.1.2 beschriebene Middleware-Ansatz für das Deployment von ISLANDER -Institutionen in Multiagentensystem hinfällig. Die Beschränkung und Leitung von Verhalten entlang der Institutionsvorgaben wird von der Netzstruktur des Organisationseinheitsmodells in Kombination mit den Systemregeln übernommen. Unterschieden werden Paradigmaregeln, Fakten für die Spezifikation einer konkreten Institution und Fakten für den Zustand dieser Institution.

- *Spezifikationsfakten* betreffen die konkrete Spezifikation einer ISLANDER -Organisationseinheit. Bei der Herangehensweise, die hier gewählt wurde, betrifft dies ausschließlich die Assoziation von Szenenbezeichnern mit Aktivitätsnetzen für Szenen und damit lediglich einen Teil der strukturellen Spezifikation. Spezifikationen bezüglich der mögli-

chen Übergänge zwischen Szenen und bezüglich Normen werden im Zuge der Definition von Szenen-Aktivitätsnetzen adressiert. Weiter unten wird ein Beispiel dafür geliefert.

- *Zustandsfakten* kommt eine sehr viel höhere Bedeutung zu als es im Falle von SONAR- und $\mathcal{M}$OISE$^+$-Organisationseinheiten der Fall ist. Im Falle einer ISLANDER-Organisationseinheit werden geltende Instanzen einer Norm sowie aktuell erlaubte beziehungsweise vorgeschriebene Akteursbewegungen zwischen Szenen als Zustandsfakten abgelegt. Diese Fakten werden im Zuge der Ausführung von Szene-Aktivitätsnetzen hinzugefügt und entfernt. Weitere Zustandsfakten betreffen Informationen über verschiedenen Instanzen von Szene-Aktivitätsnetzen.

- *Paradigmaregeln* betreffen die Überprüfung anhand von Spezifikations- und Zustandsfakten, ob der Beitritt oder die Initiierung von Szeneinstanzen durch Akteure gestattet ist.

Die Funktionsweise einer ISLANDER-Organisationseinheit unterscheidet sich somit wesentlich von der Funktionsweise einer SONAR- oder $\mathcal{M}$OISE$^+$-Organisationseinheit. Der Löwenanteil der Spezifikation besteht in der Definition von Szene-Aktivitätsnetzen und dabei insbesondere in der sorgfältigen Definition, welche Zustandsfakten durch den Ablauf dieser Aktivitäten betroffen sind. Es findet keine umfassende Regulierung oder Prüfung auf Wohlgeformtheit des Institutionszustandes auf Basis von Paradigmaregeln statt.

Um einen genaueren Eindruck der Funktionsweise einer ISLANDER-Organisationseinheit zu vermitteln, zeigt Abbildung 10.24 das Aktivitätsnetz (auf Abstraktionsstufe II) zur Auktionsszene aus Abbildung 10.21. Das Aktivitätsnetz spiegelt die grundlegende Struktur der Auktionsszene aus Abbildung 10.21 mit ihren vier Zuständen und den Übergängen zwischen ihnen wieder. Zusätzlich enthält das Netz diverse weitere, unterstützende Elemente. So gibt es vor allem Stellen für die Speicherung des Akteursidentifikators des Verkäufers (actor ID seller), der Akteursidentifikatoren aller Käufer (actor IDs buyers) sowie der Anzahl aller an der Ausführung der Szene beteiligten Akteure (actor quantity). Diese Stellen werden in Form virtueller Stellen (um die Übersichtlichkeit des Netzes zu wahren) an verschiedenen Stellen herangezogen. Natürlich stellt das gezeigte Netz nur eine von vielen Möglichkeiten dar, Szenenspezifikationen nach ISLANDER in Aktivitätsnetze zu überführen. In diesem Falle wurden beispielsweise die in Szenen verwendeten Broadcasts von Nachrichten lediglich implizit umgesetzt: Durch Interaktion mit dem Aktivitätsnetz haben die beteiligten Akteure Zugriff auf den Zustand und Fortgang der Szenenausführung.

Viele Aktionen im Kontext des Aktivitätsnetzes haben Effekte auf die Zustandsfakten der Organisationseinheit. Die wichtigste Funktion von Zustandseffekten ist das Hinzufügen oder Löschen von Fakten, die Aussagen über die Berechtigung von Akteuren zum Wechsel in bestimmte Szeneninstanzen und der damit einhergehenden Einnahme bestimmter Rollen treffen. Als Beispiel ist der Teil des Netzes, der das Verlassen der Szene durch den Verkäufer und den Käufer mit dem höchsten (und akzeptierten) Gebot betrifft, durch einen blauen Kasten hervorgehoben. Die beiden Akteure terminieren ihre Teilnahme an der Aktivität gemeinsam. Dies resultiert in folgendem neuen Zustandsfakt (unter der Voraussetzung, dass der Verkäufer den Akteursidentifikator alice und der Käufer den Akteursidentifikator bob haben):

state(mayEnterScene([[alice, seller], [bob, buyer]], transaction, 1))

Dieser Zustandsfakt legt fest, dass Verkäufer und Käufer gemeinsam einer Instanz einer Transaktionsszene betreten müssen. Sobald ein solcher Beitritt erfolgt, wird der Zustandsfakt ge-

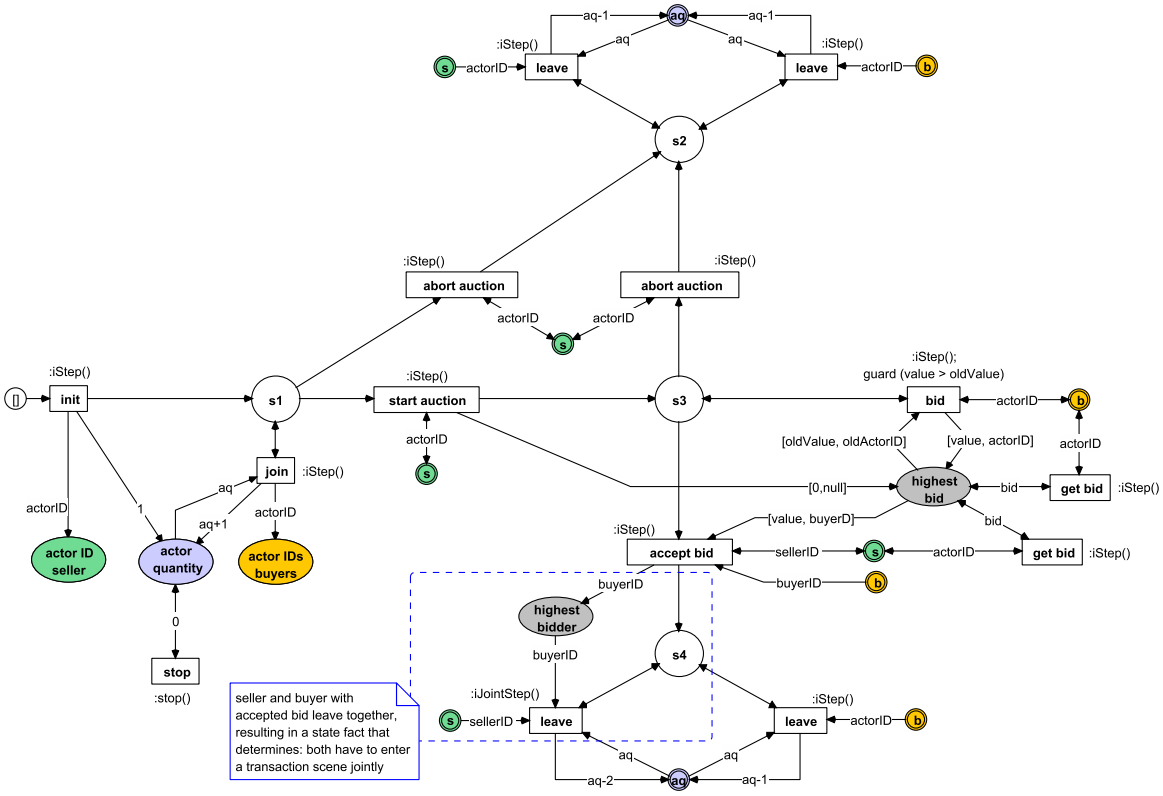

Abbildung 10.24: Aktivitätsnetz zur Auktionsszene aus Abbildung 10.21

löscht. Das operationale Grundprinzip für gemeinsame Aktionen (hier für das Betreten und Verlassen von Szenen) wurde in Teilabschnitt 9.1.5 im Zuge der Erweiterungen des Modells einer Kollektivitätseinheit behandelt.

Im Falle aller übrigen Austritte von Akteuren aus der Instanz der Auktionsszene werden Zustandsfakten hinzugefügt, die es den einzelnen Akteuren erlauben, die Austrittsszene zu betreten. Zusammenfassend bleibt festzuhalten, dass durch das beschriebene Zusammenspiel von Szenen-Aktivitätsnetzen und Zustandsfakten sowohl der Aspekt des Wechsels zwischen Szenen als auch der Aspekt der Normen abgedeckt werden (zumindest solange sich Normen lediglich in Form von zusätzlichen Richtlinien für den Wechsel zwischen Szenen manifestieren). Im umgesetzten operationalen Modell einer ISLANDER-Organisationseinheit sind bislang die Fälle abgedeckt, dass der Austritt eines Akteurs aus einer Szeneninstanz zur Berechtigung entweder des Beitritts oder der Initiierung von anderen Szeneninstanzen führt. Weitere Feinheiten der strukturellen Spezifikation von ISLANDER-Modellen wie beispielsweise die Wahl zwischen verschiedenen Routen zwischen Szeneninstanzen oder die Synchronisierung von Akteuren, die aus einer Transition heraus *verschiedene* Szeneninstanzen *gleichzeitig* betreten (anstatt *gemeinsam einer* Szeneninstanz beizutreten) werden aktuell nicht berücksichtigt.

Die hier beschriebene Ausprägung des operationalen Modells einer Organisationseinheit entlang von ISLANDER-Spezifikationen stellt lediglich eine Möglichkeit dar. Gegebenenfalls mag es etwas unbefriedigend erscheinen, speziell den Aspekt der Transitionen und der verschiedenen Routen zwischen Szeneninstanzen eher implizit durch das Hinzufügen und Entfernen

von Zustandsfakten im Zuge der Ausführung von Szeneninstanzen zu berücksichtigen. Eine Alternative wäre, Transitionen und Routen explizit in die Spezifikationsfakten aufzunehmen, spezielle Aktivitäten für das Routing von Akteuren zwischen Szeneninstanzen vorzusehen und die Zulässigkeit von Agentenbewegungen anhand von Paradigmaregeln zu regulieren. Hier wurde eine andere Herangehensweise gewählt, die sich unmittelbarer umsetzen ließ. Es gilt wieder, dass der generische Charakter des operationalen Modells einer Organisationseinheit verschiedene Möglichkeiten der Ausgestaltung bietet.

10.4.2.3 ISLANDER-Multiorganisationssysteme

Wie im Falle von SONAR und $\mathcal{M}$OISE$^+$ wird nun betrachtet, inwiefern auf Basis von ISLANDER-Konzepten das Potential des kollektiven Akteurswesens von Organisationseinheiten für den Aufbau von Multiorganisationssystemen genutzt werden kann.

Bei SONAR und $\mathcal{M}$OISE$^+$ wurde vor allem untersucht, inwiefern der jeweilige Modellierungsansatz Konzepte bietet, die das Auftreten einer Organisationseinheit als Akteur in beliebiger Position/Rolle in umschließenden Organisationseinheiten unterstützt. In beiden Fällen (im Falle von SONAR in höherem Maße) konnten solche unterstützenden Modellierungskonzepte identifiziert werden. Dies trifft im Falle von ISLANDER nicht zu. Zwar ist grundsätzlich vorstellbar, dass bei der Ausführung von Szeneninstanzen Peripherieaktionen auftreten und dazu führen, dass eine ISLANDER-Organisationseinheit als kollektiver Akteur in umschließenden Organisationseinheiten Rollen in Szeneninstanzen einnimmt. Der Modellierungsansatz ISLANDER bietet allerdings keine Anhaltspunkte dafür, wie die Verzahnung der Ausführung verschiedener Szeneninstanzen zwischen Systemebenen *systematisch* geschehen sollte.

Nach Ansicht des Autors der vorliegenden Arbeit bietet sich eine andere Möglichkeit eher an. Jedes ISLANDER-Modell beschreibt eine eigene elektronische Institution. Jede Institution ist durch ihre root- und output-Szenen klar abgegrenzt. Diese beiden Szenen eignen sich als Ankerpunkte, um mehrere Institutionen zu verknüpfen. Sie können als Portale zwischen verschiedenen Institutionen dienen. Zu diesem Zweck muss es ein übergeordnetes System geben, das die Institutionen einbettet und verknüpft. Dabei bietet es sich an, dass es sich bei dem übergeordneten System wiederum um eine Art Institution handelt. In dieser Institution werden nun nicht mehr die Bewegungen von Akteuren zwischen herkömmlichen Szeneninstanzen reguliert, sondern die Bewegungen von Akteuren zwischen Instanzen ganzer Institutionen (allerdings weist jede Institution weiterhin eine root- und eine output-Szene auf). Diese Form der Einbettung von Institutionen lässt sich dann wiederum rekursiv nach oben fortsetzen. Abbildung 10.25 zeigt eine Veranschaulichung dieser Idee.

Auf diese Weise lassen sich großangelegte Institutionen mit multiplen Systemebenen und Anwendungsdomänen bilden. Die Überführung von ISLANDER-Spezifikationen in das operationale Modell einer Organisationseinheit eröffnet genau die Möglichkeit der Modellierung dieser Art von Multiorganisationssystemen (eigentlich *Multiinstitutionssysteme*). Jede ISLANDER-Organisationseinheit tritt in einem solchen System tatsächlich als kollektiver Akteur auf. Die möglichen Aktionen einer ISLANDER-Organisationseinheit beschränken sich allerdings auf das Transferieren von individuellen Akteuren. In den herkömmlichen Szenen auf der untersten Systemebene agieren weiterhin individuelle Akteure.

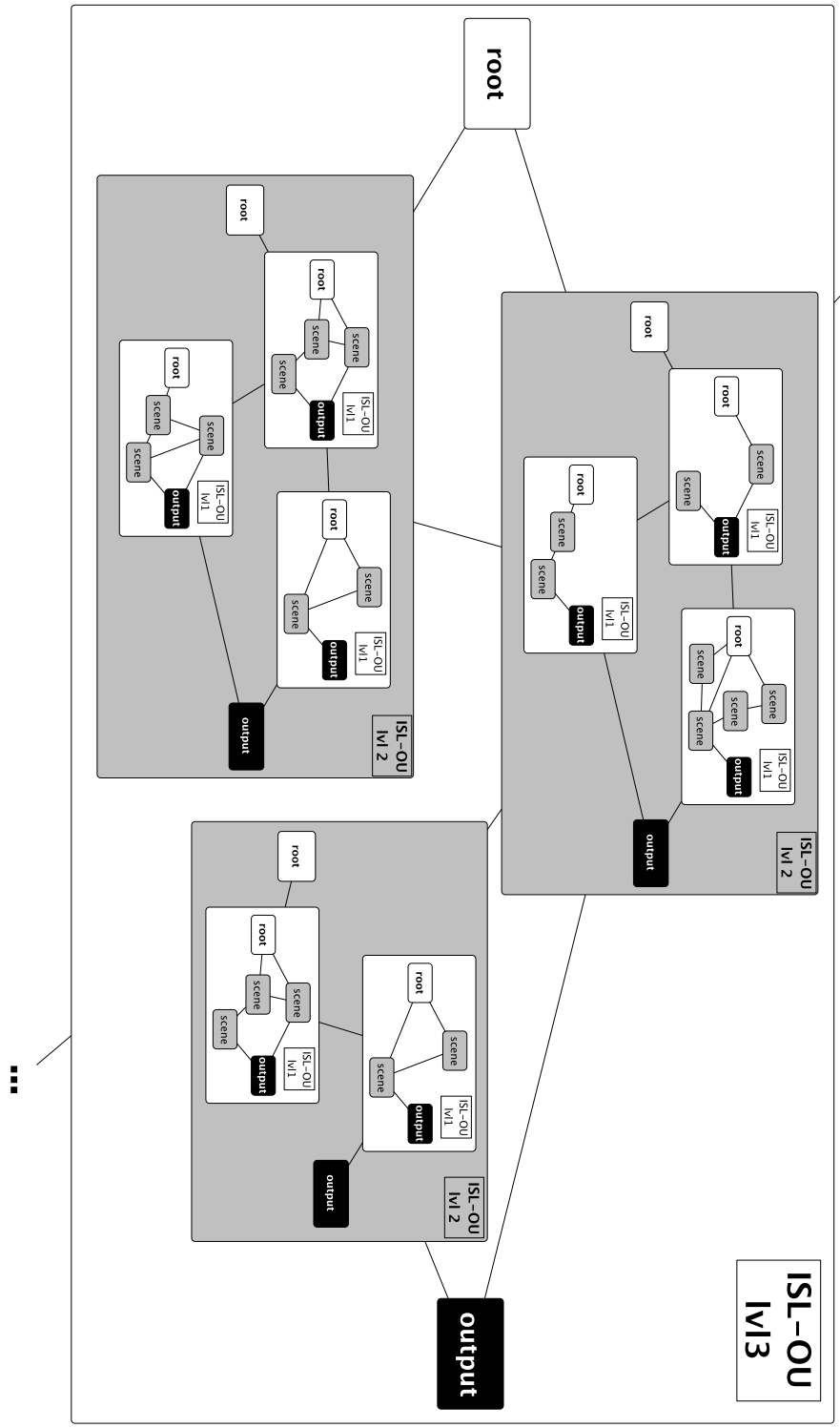

Abbildung 10.25: Skizze eines Multiorganisationssystems auf Basis von ISLANDER-Organisationseinheiten

10.5 Diskussion: Vergleich und hybride Kombination

Bislang wurde in diesem Kapitel lediglich die Möglichkeit von Multiorganisationssystemen innerhalb *eines* Paradigmas des Organisierens untersucht (SONAR-, $\mathcal{M}$OISE$^+$-, ISLANDER-Multiorganisationssysteme). Zum Abschluss und als Überleitung zum folgenden Kapitel soll hier noch die weitere Möglichkeit von hybriden Multiorganisationssystemen angedacht werden. Weiter oben wurde betont, den drei betrachteten Modellierungsansätzen würden verschiedene Paradigmen des Organisierens zu Grunde liegen. Die Unterschiede wurden im Laufe des Kapitels bereits recht detailliert aufgezeigt. Hier werden die wichtigsten Unterscheidungsmerkmale zusammengefasst.

- Als offensichtlichster Unterschied fällt zunächst auf, dass SONAR- und $\mathcal{M}$OISE$^+$-Modelle einen *konstruktiven* Charakter aufweisen, während ISLANDER-Modelle einen eher *deklarativen* Charakter aufweisen. SONAR- und $\mathcal{M}$OISE$^+$-Modelle sind konstruktiv in dem Sinne, dass Organisationen im Hinblick auf die Definition globaler Organisationsziele/-aufgaben und die Erreichung dieser globalen Zwecke konstruiert werden. ISLANDER-Modelle hingegen sind deklarativ in dem Sinne, dass jeweils eine Institution und damit die Regulierung von Akteursverhalten entlang bestimmter Regeln beschrieben/deklariert wird, ohne dass die Ausrichtung hin zu oder entlang bestimmter Zwecke expliziert wird.

- SONAR- und $\mathcal{M}$OISE$^+$-Modelle wiederum unterscheiden sich im Wesentlichen hinsichtlich ihrer Kopplung ihrer strukturellen und funktionalen Modelldimensionen. Im Falle von $\mathcal{M}$OISE$^+$ stehen die beiden Dimensionen zunächst unabhängig nebeneinander und werden erst durch die zusätzliche deontische Dimension gekoppelt. Im Falle von SONAR hingegen sind beide Dimensionen untrennbar verzahnt.

- Als Folge erlaubt $\mathcal{M}$OISE$^+$ eine sehr reiche und feingranulare Strukturspezifikation, allerdings ist die Semantik hinter einigen Modellelementen/-kombinationen unklar. Außerdem ist es möglich, strukturelle und funktionale Spezifikationen unabhängig voneinander zu modifizieren oder auszutauschen (solange weiterhin eine Kopplung mittels der deontischen Spezifikationen möglich ist). So könnten in einer $\mathcal{M}$OISE$^+$-Organisation neuartige Ziele und darauf aufbauende Pläne definiert werden, ohne dass die Strukturspezifikation davon unmittelbar betroffen wäre. Umgekehrt könnten Strukturen modifiziert werden, ohne Anpassungen an Zielen und Plänen vornehmen zu müssen.

- Im Falle von SONAR hingegen sind die verfügbaren Modellelemente relativ eingeschränkt. Dafür sind alle Modellsituationen mit einer klaren Semantik hinterlegt. Dies wird zusätzlich durch eine interaktionsbezogene Modelldimension unterstützt, die im Falle von $\mathcal{M}$OISE$^+$ fehlt. Modifikationen an einer Art Spezifikation (Struktur, Funktion, Interaktion) zieht in den meisten Fällen unweigerlich Modifikationen an allen übrigen Arten der Spezifikationen nach sich. Insgesamt dominiert im Falle von SONAR eine prozess-/verhaltensorientierte Sichtweise, aus der sich ein konsistenter Zusammenhang aller Modelldimensionen ergibt.

Anhand dieser Unterschiede liegt es nun nahe, zu untersuchen, ob die verschiedenen Ansätze gegebenenfalls auch eine unterschiedlich starke Eignung für die Modellierung bestimmter Anwendungskontexte aufweisen und ob es lohnenswert ist, Organisationen/Institutionen basierend auf verschiedenen Ansätzen in einem großangelegten Multiorganisationssystem zu kom-

binieren. Insbesondere erlaubt die Überführung in das Modell einer Organisationseinheit sowohl die horizontale als auch die vertikale Kopplung von Organisationseinheiten verschiedenen Typs. So ist beispielsweise denkbar, dass multiple $\mathcal{M}$OISE$^+$-Organisationseinheiten in eine SONAR-Organisationseinheit eingebettet sind und dort als kollektive Akteure auftreten. Dabei stellt sich natürlich sofort die Frage, wie eine Integration von Organisationseinheiten basierend auf gänzlich unterschiedlichen Modellierungsansätzen und damit auch Paradigmen des Organisierens überhaupt bewerkstelligt werden kann. Schließlich war es bereits nicht trivial, Möglichkeiten für den Übergang zu Multiorganisationssystemen *innerhalb* eines Ansatzes zu identifizieren. Grundsätzlich liegt diese Fragestellung außerhalb des Skopus der vorliegenden Arbeit und stellt ein eigenständiges Forschungsthema dar. Allerdings wurde in Kapitel 2 ausdrücklich betont, das Auftreten heterogener Systemkomponenten sei in einem System von Systemen ein inhärent auftretendes Problem. Deswegen wurde im zweiten Teil der Arbeit so viel Wert darauf gelegt, dass das Modell einer Organisationseinheit ein in sich geschlossenes und weitgehend abgeschlossenes Gebilde darstellt. Die Kopplung und erst recht die wirkliche Integration von Organisationseinheiten verschiedenen Typs wird immer wieder ein Problem darstellen, das nach Ansicht des Autors der vorliegenden Arbeit nicht uniform gelöst werden kann. Allerdings können punktuelle und proprietäre Lösungen in vielen Fällen bereits eine zufriedenstellende Herangehensweise darstellen.

Angesichts dieser Voraussetzungen wird die Kopplung von Organisationseinheiten verschiedenen Typs hier nicht im Detail betrachtet. Stattdessen wird eine abstraktere Perspektive eingenommen und eine Charakterisierung der Eignung der hier betrachteten Organisationseinheitstypen als spezifische Archetypen von Teilsystemen in einem System von Systemen (genauer: einem Multiorganisationssystem) vorgenommen. Bei den folgenden Ausführungen handelt es sich um eine hypothetische Einschätzung des Autors der vorliegenden Arbeit. Inwiefern sich diese als tragfähig erweist, muss noch gezeigt werden. Außerdem stellt die Einschätzung einen Ausblick auf das folgende Kapitel dar und wird dort wieder aufgegriffen.

- $\mathcal{M}$OISE$^+$-Organisationseinheiten eigenen sich besonders gut für niedrige Systemebenen eines Multiorganisationssystems. Idealerweise betten sie individuelle Akteure ein. Der Grund dafür ist, dass $\mathcal{M}$OISE$^+$-Spezifikationen von den hier betrachteten Spezifikationen den größten Raum für Dynamik und Flexibilität bieten. Die Trennung von strukturellen und funktionalen Spezifikationen erlaubt leichtgewichtige Modifikationen. Dieser Aspekt ist für niedrige Systemebenen besonders relevant, da hier die Operationen der jeweiligen Kernanwendungen unter potentiell dynamischen und unsicheren Bedingungen ausgeführt werden. Darüber hinaus können Beziehungen zwischen Akteuren (über ihre Rollen) auf sehr detailreiche Art und Weise beschrieben werden. Dies wird den vielfältigen Aspekten, Bedingungen und Optionen, die es in konkreten Anwendungsszenarien zu berücksichtigen gibt, gerecht. Nicht alle Strukturspezifikationen eines $\mathcal{M}$OISE$^+$-Modells sind mit einer klaren Semantik assoziiert, so dass auch hier wieder ein hohes Maß an Flexibilität vonnöten ist und Akteure die unterspezifizierten Spezifikationen durch wechselseitige Abstimmung mit Leben füllen müssen.

  Diese Ausführungen machen auch deutlich, warum es schwierig ist, kollektive Akteure in $\mathcal{M}$OISE$^+$-Organisationseinheiten, deren Spezifikationen sich (zumindest tendenziell) relativ häufig ändern, einzubetten. Zwar gestalten sich Modifikationen aus Sicht einer $\mathcal{M}$OISE$^+$-Organisationseinheit selber als leichtgewichtig. Die Annahme und Inkorporierung solcher Modifikationen durch die eingebetteten Akteure gestaltet sich allerdings gegebenenfalls als schwieriger. Adaption ist für individuelle Akteure leichter zu bewerk-

stelligen als für Kollektivitäten. Dies gilt für künstliche Softwareakteure ebenso wie für natürliche/soziale Akteure (auch wenn es natürlich insbesondere im Falle von Organisationsformen Abstufungen bezüglich ihrer Eignung zum Operieren in dynamischen, unsicheren Umgebungen gibt wie die Erläuterungen in den Kapiteln 3 und 4 gezeigt haben).

- SONAR-Organisationseinheiten eigenen sich besonders gut, weitere Organisationseinheiten als kollektive Akteure einzubetten. SONAR bietet einen eingeschränkten Satz an Möglichkeiten, das Beziehungsgeflecht zwischen Akteuren zu beschreiben. Dabei ist eine eindeutige Semantik stets gewahrt. In Kombination sorgt dies für eine leichtere Inkorporierung durch eingebettete Akteure einer SONAR-Organisationseinheit. Modifikationen von SONAR-Spezifikationen zum Zwecke der Adaption sind durch die enge Verzahnung aller Modelldimensionen recht schwergewichtig. Insofern sollten SONAR-Organisationseinheiten eher dort eingesetzt werden, wo weitreichende Modifikationen nicht zu hochfrequent notwendig werden. Insgesamt beschreiben SONAR-Modelle Akteurskonfigurationen auf eine abstraktere Weise als es im Falle der feingranularen $\mathcal{M}$OISE$^+$-Modelle geschieht. Details der Umsetzung können somit auf tiefer liegende Systemebenen ausgelagert werden. Aufgrund dieser Erläuterungen ergibt sich gegenüber $\mathcal{M}$OISE$^+$-Organisationseinheiten eine bessere Eignung zur Einbettung kollektiver Akteure.

  Da SONAR-Organisationseinheiten zweckgerichtete Kollektivitäten sind, muss es auch möglich sein, Zwecke in Form von Aufgaben und Zielen auf eingebettete Akteure zu übertragen. Daher eigenen sich ISLANDER-Organisationseinheiten weniger als $\mathcal{M}$OISE$^+$- oder weitere SONAR-Organisationseinheiten als eingebettete Kollektivitäten.

  Insgesamt ergibt sich für SONAR-Organisationseinheiten die Eignung für „mittlere“ Ebenen eines Multiorganisationssystems. Sie operieren oberhalb der Ebenen, die unmittelbar mit der Ebene der Kernoperationen und -anwendungen verknüpft sind. Sie operieren aber noch unterhalb der Ebenen, auf der Regulierung die explizite Konstruktion ablöst.

- ISLANDER-Organisationseinheiten eignen sich für höhere Ebenen eines Multiorganisationssystems. In Kapitel 2 wurde im Zusammenhang mit der Metapher der Softwarestädte auf die Forderung verwiesen, Softwaresysteme ab einer gewissen Größenordnung eher durch Regulierung anstatt durch gezielte Konstruktion zu beherrschen. Dort wurde als Metapher der Bau eines Gebäudes (Konstruktion) mit der Entwicklung einer ganzen Stadt (Regulierung) herangezogen. Das entsprechende Pendant unter einer Organisationsperspektive ist der Übergang von zielgerichtet konstruierten Organisation zu regulierenden Institutionen. ISLANDER-Organisationseinheiten bieten sich somit an, kollektive Akteure in Form von SONAR- oder $\mathcal{M}$OISE$^+$-Organisationseinheiten einzubetten und damit eine Regulierung von deren Verhalten im Kontext der verschiedenen Domänen des Gesamtsystems vorzunehmen. In Unterabschnitt 10.4.2.3 wurde eine Idee skizziert, wie großangelegte Institutionen mit multiplen Systemebenen auf Basis von ISLANDER-Organisationseinheiten realisiert werden können, so dass sich hier eine Möglichkeit bietet, den oberen Abschluss eines Multiorganisationssystems vollständig in Form von ISLANDER-Organisationseinheiten zu realisieren.

  Die durch ISLANDER-Organisationseinheiten verkörperten institutionellen Rahmenbedingungen sollten sich idealerweise nur selten ändern, damit sie einen stabilen und für die eingebetteten kollektiven Akteure verlässlichen Rahmen bieten.

10.6 Zusammenfassung

In diesem Kapitel wurden konkrete Ausprägungen des universellen Grundmodells einer Organisationseinheit aus dem vorhergehenden Teil der Arbeit vorgenommen. Die Ausprägung erfolgte dabei entlang verschiedener Ansätze zur Modellierung von Organisationen/Institutionen aus dem Bereich der Multiagentensysteme. Jeder der herangezogenen Ansätze bringt ein eigenes Paradigma des Organisierens mit sich, so dass daraus unterschiedliche Typen von Organisationseinheiten resultieren: SONAR-, $\mathcal{M}$OISE$^+$- und ISLANDER-Organisationseinheiten. Insgesamt diente das Kapitel somit dem Zweck, die Tragfähigkeit des universellen Modells einer Organisationseinheit sowie die Anschlussfähigkeit an die aktuelle Multiagentensystemforschung zu demonstrieren. Außerdem wurden in diesem Kapitel erstmals überhaupt konkrete Beispiele von Organisationseinheiten vertiefend behandelt.

Die herangezogenen Modellierungsansätze unterscheiden sich beispielsweise hinsichtlich der strukturellen Anordnung ihrer Akteure, hinsichtlich der Definitionen von Organisationsfunktionen/-zielen, hinsichtlich des Grades an Verhaltensvorgaben und auch hinsichtlich der Verzahnung verschiedener Modelldimensionen. Eine eher grobe Charakterisierung der verschiedenen Typen von Organisationseinheiten wurde anhand des abstrakten Organisationseinheitsmodells vorgenommen. Der Schwerpunkt des Kapitels lag allerdings auf der Betrachtung der Unterschiede, die sich im Hinblick auf das operationale Organisationseinheitsmodell und damit auch für die prototypische *Programmierung* der verschiedenen Organisationseinheitstypen ergeben. Die Unterschiede manifestieren sich dabei vor allem in der Definition verschiedener Systemregeln. Um den diesbezüglichen Untersuchungen eine gewisse Systematik zu verleihen, wurden dabei drei verschiedene Arten von Systemregeln explizit unterschieden. Weitere wesentliche Unterschiede konnten bezüglich den Aktivitätsspezifikationen der jeweiligen Organisationseinheitstypen ausgemacht werden. Für alle Ausprägungen des operationalen Modells waren kleinere Modifikationen/Erweiterungen des Grundmodells einer Organisationseinheit aus Abschnitt 9.2 (speziell wurde dem Kapitel das Grundmodell aus Abbildung 9.21 zu Grunde gelegt) notwendig. Die Möglichkeit solcher punktuellen Erweiterungen sind jederzeit zulässig, solange alle Änderungen die fundamentalen Mechanismen des Grundmodells respektieren.

Für jeden der hier betrachteten Organisationseinheitstypen wurde außerdem betrachtet, ob die unmittelbare Möglichkeit besteht, von der Betrachtung einer einzigen Organisationseinheit zur Betrachtung von Multiorganisationssystemen überzugehen. Genauer gesagt betrifft das die Frage, ob die zu Grunde liegenden Modellierungsansätze aus dem Multiagentensystembereich einen solchen Übergang unterstützen. Initial ist dieser Aspekt in keinem der Ansätze verwurzelt. Zusätzlich zur oben betonten Anschlussfähigkeit des Konzeptes einer Organisationseinheit an die aktuelle Forschung wird hier also auch die Innovation deutlich, die es gegenüber der klassischen Auffassung von Organisationen im Multiagentensystembereich mit sich bringt. Zumindest aber konnten für alle drei Multiagentensystemansätze unterstützende Modellierungskonzepte für den Übergang zu Multiorganisationssystemen identifiziert werden. Im Falle von SONAR ist dies besonders ausgeprägt. Abschließend zum Kapitel wurde ein Vergleich der verschiedenen Ansätze hinsichtlich ihrer Eignung für verschiedene Abstraktions-/Granularitätsebenen eines Multiorganisationssystems vorgenommen. Darauf aufbauend konnten Hypothesen bezüglich der Kombination von Organisationseinheiten verschiedenen Typs in einem Multiorganisationssystem aufgestellt werden.

11 Referenzarchitektur Organ für Multiorganisationssysteme

Das vorige Kapitel schloss mit einem Vergleich von verschiedenen Ausprägungen des universellen Modells einer Organisationseinheit hinsichtlich ihrer Eignung für den Einsatz auf verschiedenen System-/Abstraktionsebenen eines Multiorganisationssystems. In diesem Kapitel wird die Thematik der Kombination/Integration von Organisationseinheiten verschiedenen Typs in ein Gesamtsystem mit größerer Systematik behandelt. Die Thematik ist von großer Bedeutung. Gerade im Falle eines universell einsetzbaren und generischen Basisbausteins wie der Organisationseinheit der vorliegenden Arbeit bedarf es spezifischer Richtlinien und konkreter Vorschläge, welche Ausprägungen in welcher Kombination sich für den Einsatz eignen.

Natürlich kann es dabei wiederum verschiedene Möglichkeiten in Form verschiedener *Entwurfsmuster* oder *Referenzmodelle* geben. In diesem Kapitel wird die sozial motivierte *Referenzarchitektur* ORGAN[1] *für Multiorganisationssysteme* vorgestellt. Dabei wird das Wesen der vorgestellten Architektur als *Referenzarchitektur* explizit hervorgehoben. Nach Lilienthal [Lil08, S. 35] werden bei einer Referenzarchitektur die Architekturelemente in *Elementarten* eingeteilt und Regeln angegeben, welche Verantwortlichkeiten die Vertreter der einzelnen Elementarten haben und wie sie miteinander interagieren dürfen. Bei Humm et. al. [HVH06] wird eine Referenzarchitektur mit einer *idealisierten Blaupause* verglichen: Reale Systeme entlang einer Referenzarchitektur weichen gegebenenfalls punktuell von den durch die Architektur induzierten Kategorien und Relationen ab. Man kann eine Referenzarchitektur demnach als ein Referenzmodell für eine ganze Klasse von Architekturen auffassen. Sie bietet ein *idealtypisches* Modell für die Klasse der zu modellierenden Architekturen. Auf Basis des allgemeinen Modells können spezielle Modelle für bestimmte Sachverhalte geplant werden. Das allgemeine Modell kann hernach als Vergleichsobjekt herangezogen werden. Spezialisierte Architekturen können sich in einzelnen Punkten von der Referenzarchitektur unterscheiden. Die Referenzarchitektur bietet jedoch einen Leitfaden, der von den spezialisierten Architekturen im Wesentlichen befolgt wird. So ist beispielsweise die Von-Neumann-Architektur nach Burks, Goldstine und von Neumann [BGvN46] eine im Jahr 1946 veröffentlichte und bis heute überwiegend verwendete Referenzarchitektur für den Rechnerbau.

Ganz im Sinne des bisherigen Vorgehens im Kontext der vorliegenden Arbeit fußt auch die Definition der Referenzarchitektur ORGAN für Multiorganisationssysteme auf Grundlagen aus dem Bereich der realweltlichen (natürlichen) sozialen Systeme. Damit wird also das sozial fundierte Modell einer Organisationseinheit um einen sozial fundierten Vorschlag für die Integration mehrerer konkreter Organisationseinheitstypen in ein Multiorganisationssystem bereichert.

Die hier vorgestellte Referenzarchitektur hat der Autor in mehreren Publikationen [WEM08c, WEKBM09, WEM08b] in verschiedenen Stadien ihrer Konzeptionalisierung vorgestellt. Als Mitautoren wirkten Daniel Moldt und Michael Köhler-Bußmeier (letzterer in einem Fall).

[1] **Org**anisations-**A**rchitektur mit **N**etzen

11.1 Architekturgerüst

Als eine der Grundannahmen der vorliegenden Arbeit gilt, dass die Modellierung komplexer Systeme unterschiedliche Ebenen der Abstraktion erfordert. Dabei sollen die Inhalte der jeweiligen Ebenen so dargestellt werden, dass sie auf dieser (Abstraktions-)Ebene eine weitgehend vollständige und homogene Darstellung der gewählten Perspektive umfassen. In Hinblick auf Softwarearchitekturen ist daher eine gezielte Unterscheidung verschiedener Systemebenen vorzunehmen. Ein Vorschlag für eine solche Unterscheidung für die Architektur von Multiagentensystemen wurde beispielsweise in Form der MULAN-Architektur in Kapitel 5 vorgestellt. An dieser Stelle wird der Vorschlag ORGAN für die Architektur von Multiorganisationssystemen vorgestellt. In gewisser Weise kann ORGAN als eine Reinterpretation der vier Ebenen der MULAN-Architektur verstanden werden. Dabei ist nicht mehr der Agent sondern die Organisation (im Sinne eines Unternehmens) die Kerneinheit der Strukturgebung. Darauf aufbauend ergeben sich auch andere Interpretationen der übrigen drei Ebenen und der Beziehung zwischen den einzelnen Ebenen. Die Reinterpretation erfolgt dabei anhand der Betrachtung verschiedener Systemebenen und Beziehungen in real existierenden sozialen Organisationssystemen.

Bereits in den Kapiteln 3 und 4 wurde herausgearbeitet, dass eine organisationsorientierte Betrachtung sozialer Gesellschaften die Unterscheidung verschiedener Systemebenen in Form verschiedener kollektiver Akteure inhärent mit sich bringt. Als gemeinsamer Oberbegriff für die dabei betrachteten Entitäten wurde dabei in Abschnitt 4.1 der Begriff der *organisierten Kollektivität* eingeführt und die in diesem Zusammenhang identifizierten Konzepte und Mechanismen bildeten im zweiten Teil der Arbeit die Grundlage für das softwaretechnische Konzept der *Organisationseinheit*. Dabei standen allerdings generische Konzepte im Mittelpunkt und die Frage, welche konkreten Typen von organisierten Kollektivitäten beziehungsweise Organisationseinheiten in großangelegten Systemen bevorzugt kombiniert werden sollten, wurden erstmals zum Ende des vorigen Kapitels behandelt. Diese Thematik wird hier vertieft. Als Grundlage dient dabei die von Scott [Sco08, S. 192] übernommene Abbildung 11.1 zu *Strukturationsprozessen* zwischen Institutionsebenen in sozialen Organisationssystemen.

Der Begriff der Strukturation wurde bereits in Abschnitt 3.1 als von Giddens [Gid84] geprägt eingeführt. Strukturation bezieht sich auf die rekursive Interdependenz zwischen sozialer Struktur und sozialen Handlungen. Die Verbform soll darauf hinweisen, dass soziale Strukturen nur insofern existieren, als dass sie durch soziale Aktivitäten produziert und reproduziert werden. In Abschnitt 3.1 wurde auch darauf verwiesen, dass sich die wechselseitige Konstitution zwischen sozialer Struktur und sozialen Handlungen auf verschiedenen Hierarchieebenen eines sozialen Systems wiederfinden. Abbildung 11.1 expliziert diese Sichtweise anhand des Ausschnittes des sozialen Lebens, der institutionelle Kollektivitäten im Kontext von Organisationsszenarien betrifft.

Zwischen der Ebene der Organisationen/Unternehmen und der Ebene der umschließenden Gesellschaft wird die Ebene der *organisatorischen Felder* identifiziert. Sie stellt eine wichtige institutionelle Zwischenebene für den Brückenschlag zwischen der Ebene der Organisationen und der Gesellschaft dar. Institutionen auf Gesellschaftsebene oder sogar auf übergesellschaftlicher Ebene stellen eine allgemeine, weitläufige institutionelle Umgebung dar. Im Kontext dieser Umgebung existieren weitere, spezifischere Institutionen in Form von organisatorischen Feldern. Sie bieten unmittelbare Kontexte für Organisationen. Diese wiederum stellen ihrerseits institutionelle Kontexte für individuelle Akteure dar. Im Normalfall können allerdings auch innerhalb einer Organisation weitere Kollektivitäten wie Abteilungen oder Teams identifiziert werden.

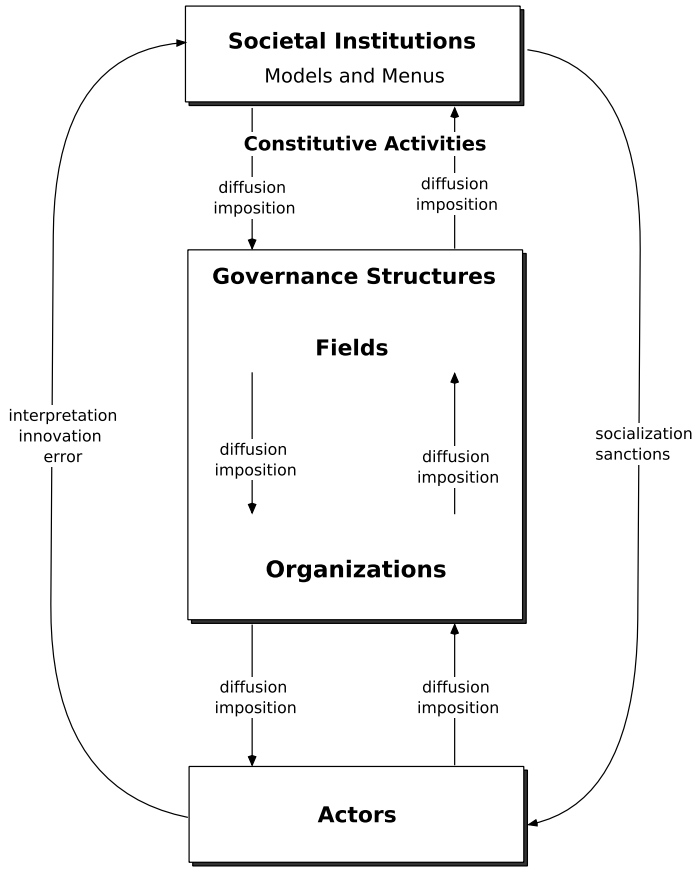

Abbildung 11.1: *Top-Down*- und *Bottom-Up*-Prozesse zwischen institutionellen Systemebenen in sozialen Gesellschaften, übernommen von Scott [Sco08, S. 192]

Verschiedene Top-Down-Prozesse ermöglichen Akteuren auf höheren/umfassenderen Systemebenen die Formgebung (sowohl im Sinne von Einschränkung als auch von Bevollmächtigung) bezüglich Strukturen und Aktivitäten von Akteuren auf tieferen Systemebenen. Zu diesen Prozessen gehören konstituierende Aktivitäten, Übersetzung/Umsetzung, Durchdringung, Sozialisierung, Auferlegung, Autorisierung, Beeinflussung/Veranlassung und Prägung. Demgegenüber stehen Bottom-Up-Prozesse, im Zuge derer Akteure auf tiefer gelegenen Systemeben ihre Kontexte formen (produzieren und reproduzieren). Zu diesen Prozessen zählen selektive Zuwendung/Beachtung, Interpretation, Wissenskonstruktion, Identitätsschaffung, Auftreten von Störungen/Fehlverhalten, Phantasie/Erfindung, Konformität, Reproduktion von Mustern, Kompromiss, Vermeidung, Widerstand und Manipulation.

Basierend auf dieser Perspektive auf soziale Institutionen erfolgt nun eine organisationsorientierte Reinterpretation der MULAN-Architektur. Für die resultierende ORGAN-Architektur wird dabei anstelle des Agenten die Organisation in den Mittelpunkt gerückt. Der Fokus auf die Organisation als zentrale Einheit der Architektur führt unmittelbar zu drei notwendigen Ebenen der Betrachtung: die Organisation selber, ihre Interna und ihre Umwelt. Je nachdem, wie vielfältig die Interaktionen mit externen Einheiten sind und wie breitgefächert das Spektrum der möglichen Interaktionspartner ist, kann die Umwelt einer Organisation weiter

differenziert werden. In Übereinstimmung mit den Ausführungen aus Abschnitt 4.2 besteht die Umwelt einer Organisation dann aus mehreren konzeptionellen Umgebungen, abhängig davon, in welchen Domänen sie operiert. Außerdem ist das Ziel hier ein Architekturvorschlag für Multiorganisationssysteme und damit für multiple Organisationen. Daraus folgt eine ganze Fülle von verschiedenen, sich teilweise überlappenden konzeptionellen Umgebungen von multiplen Organisationen. Es wird daher noch eine vierte Architekturebene als oberer Systemabschluss in Form einer Integration sämtlicher verschiedenen Umgebungen erforderlich.

Jede der Architekturebenen wird mit Einheiten besetzt, die Ausprägungen des universellen Grundmodells einer Organisationseinheit sind. Damit ergeben sich für die ORGAN-Architektur vier Typen von Organisationseinheiten, die unter Einbezug der bisherigen Erläuterungen wie folgt benannt sind: *Abteilung, Organisation, organisatorisches Feld* und *Gesellschaft*. Abbildung 11.2 zeigt einen Überblick über die Architektur und damit über die allgemeine Form der daraus resultierenden Softwaresysteme. Es handelt sich hier um eine Spezialisierung/Einschränkung der in Abbildung 7.1 aus Kapitel 7 illustrierten kanonischen Sichtweise auf Multiorganisationssysteme.

Jede Abteilung ist konzeptionell einer einzigen Organisation exklusiv zugeordnet. Jede Organisation kann aus mehreren Abteilungen zusammengesetzt und auf mehreren organisatorischen Feldern beheimatet sein. Jedes organisatorische Feld kann mehrere Organisationen umfassen und ist in die integrierende Gesellschaft eingebettet. Die Abteilung als Organisationseinheit der niedrigsten Architekturebene stellt dabei die Verbindung zur Multiagentensystemforschung dar. Softwaretechnisch gesehen ist jede Abteilung ein „klassisches“ organisationsorientiertes Multiagentensystem nach Abschnitt 5.2 (sofern man von ihrem kollektiven Akteurswesen absieht). Somit ergibt sich ein nahtloser Übergang von der Agenten- zur Organisationsorientierung. Die Abbildung deutet außerdem an, dass es oberhalb der Ebene der Gesellschaft gegebenenfalls weitere Systemebenen geben kann. Allerdings liegen diese nicht im Skopus der ORGAN-Architektur. Für ORGAN-Multiorganisationssysteme bildet die Gesellschaft einen oberen Systemabschluss.

Wie bereits in der Einleitung des Kapitels geschrieben, handelt es sich bei der ORGAN-Referenzarchitektur um ein idealisiertes Modell. Konkrete ORGAN-Systeme können gegebenenfalls punktuell abweichen. So könnte beispielsweise die Beschränkung, eine Abteilung sei *genau* einer Organisation zugewiesen, für bestimmte Anwendungsfälle aufgeweicht werden. Gegenüber dem allgemeinen Konzept eines Multiorganisationssystems aus Abbildung 7.1 fällt zudem auf, dass individuelle Akteure ausschließlich in Abteilungen und damit in Organisationseinheiten niedrigsten Kollektivitätsebene eingebettet werden. Auf allen darüber liegenden Ebenen agieren ausschließlich kollektive Akteure. Auch hier gilt, dass diese Beschränkung für bestimmte Anwendungsfälle gegebenenfalls aufgehoben werden kann. Im Allgemeinen aber liegt der ORGAN-Architektur die folgende Annahme zu Grunde. Natürlich kann es besonders dominante und einflussreiche individuelle Akteure geben, die einen unmittelbaren Einfluss auf Organisationseinheiten der Organisations-, Feld- oder sogar Gesellschaftsebene haben. Allerdings gilt für diese nichtsdestotrotz (in der Regel), dass sie diesen Einfluss gerade *wegen* ihrer Einbettung in bestimmte Organisationseinheiten haben. Als realweltliche Analogie kann hier der Geschäftsführer eines Unternehmens, der die Geschicke des Unternehmens auf Feldebene weitestgehend im Alleingang leitet, gelten. Die entsprechende Machtfülle wird ihm dennoch gerade in seiner Funktion als Inhaber der Position des Geschäftsführers zuteil. Insofern ist die konzeptionelle Betrachtung, es sei das Unternehmen, welches als kollektiver Akteur auf dem Feld agiert, weiterhin angemessen.

Um einen plastischen Eindruck der sozialen Grundlagen und ihres Bezuges zur ORGAN-

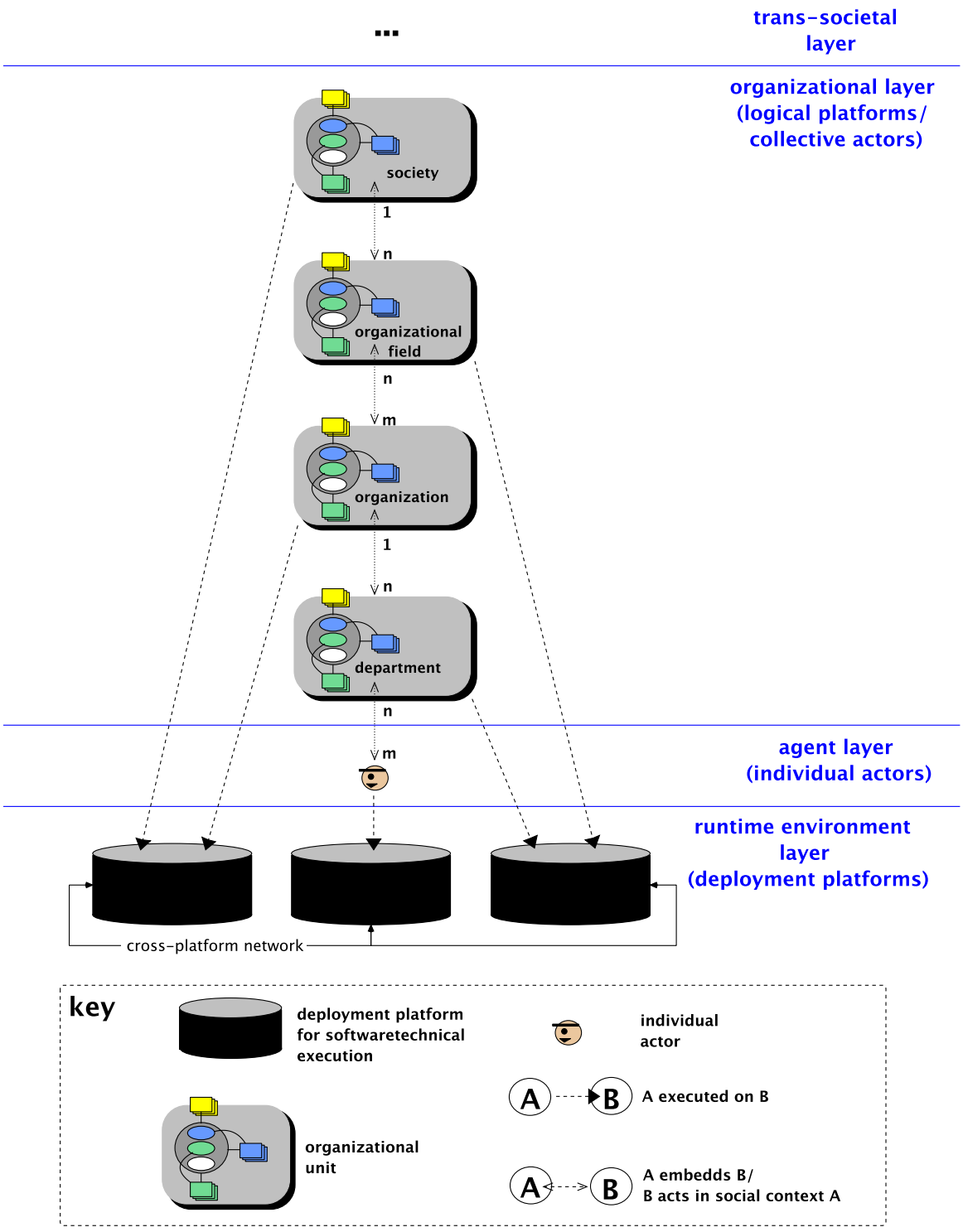

Abbildung 11.2: ORGAN-Referenzarchitektur für Multiorganisationssysteme

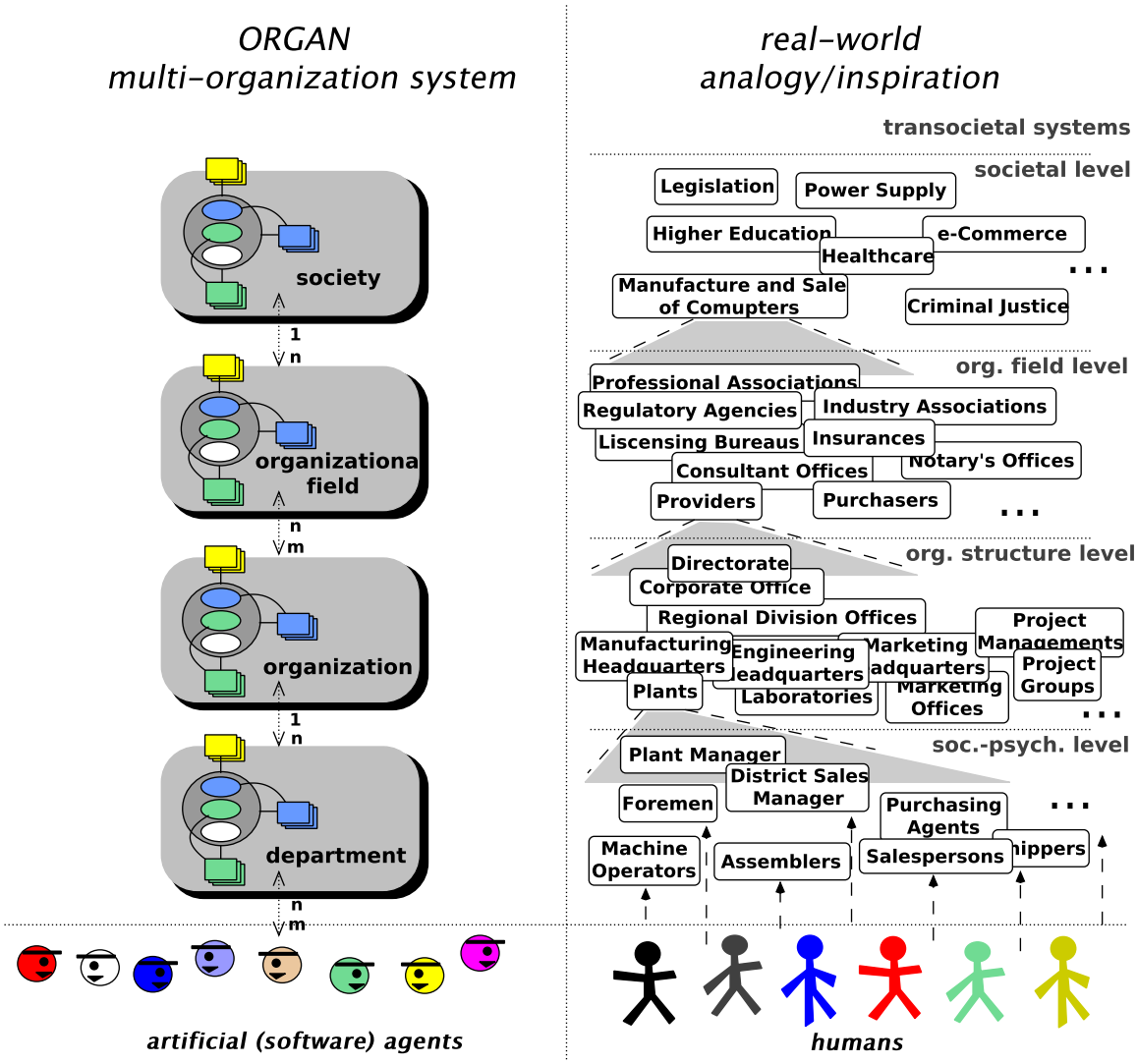

Abbildung 11.3: Exemplarischer Auszug der realweltlichen Analogie zu ORGAN-Softwaresystemen

Architektur zu vermitteln, zeigt Abbildung 11.3 abschließend zu diesem Abschnitt einen exemplarischen Auszug der realweltlichen Analogien zu den Systemebenen und Organisationseinheiten eines ORGAN-Multiorganisationssystems. Die beispielhafte Zusammenstellung der Kollektivitäten auf den einzelnen Systemebenen wurde dabei aus verschiedenen Quellen [Sco03, Joh96, Min83] zusammengetragen.

Im Folgenden wird eine detailliertere Betrachtung der einzelnen Organisationseinheitstypen eines ORGAN-Multiorganisationssystems vorgenommen. Dabei werden horizontale sowie vertikale Beziehungen und die damit einhergehenden Aktivitäten betrachtet. Dabei wird das abstrakte Grundmodell einer Organisationseinheit nach Abschnitt 8.2 herangezogen.

11.2 Organisationseinheitstyp *Abteilung*

Abteilungen sind die kleingranularsten unter den Organisationseinheiten eines ORGAN-Systems. Das Konzept der Abteilungs-Organisationseinheit baut unmittelbar auf den in Kapitel 5 geschilderten organisationsorientierten Ansätzen der Multiagentensystementwicklung auf, indem individuelle Akteure eingebettet werden.

11.2.1 Organisationstheoretischer Bezug

Für die organisationstheoretischen Grundlagen bezüglich Abteilungen wird hier insbesondere auf Mintzberg [Min83] Bezug genommen. Das Grundgerüst einer Abteilung bildet die Definition von *Positionen* und deren *Gruppierung* ([Min83, S. 45]).[2] Eine Position ist dabei mit einer organisatorischen Aufgabe beziehungsweise einer Kombination mehrerer organisatorischer Aufgaben assoziiert. Positionen werden vor allem entlang der beiden Dimensionen *Spezialisierung* und *Formalisierung* charakterisiert ([Min83, Kapitel 2]). Horizontale Spezialisierung bezieht sich darauf, wie viele verschiedene Aufgaben eine Position beinhaltet und wie umfangreich beziehungsweise eingeschränkt diese Aufgaben sind. Vertikale Spezialisierung charakterisiert, wie viel Kontrolle über ihre Aufgaben einer Position anhaftet. Formalisierung korreliert mit horizontaler sowie vertikaler Spezialisierung. Die eingeschränktesten Positionen sind gleichzeitig diejenigen, die am offensten für einen hohen Grad an Formalisierung sind. Gleichzeitig führt ein hohes Maß an Formalisierung zur Verschiebung von Kontrolle über Aufgaben von denjenigen, die Aufgaben ausführen, zu denjenigen, die Aufgaben spezifizieren.

Formalisierung bezieht sich typischerweise nicht lediglich auf einzelne Positionen, sondern stattdessen auf positionsübergreifende Arbeitsabläufe und abteilungsweite Regeln ([Min83, S. 34]). Formalisierung ist damit auch ein Mittel zur Gruppierung. Positionsgruppierung ist ein fundamentales Mittel der Koordination in Organisationen. Mintzberg [Min83, S. 46+47] nennt vor allem vier wichtige Effekte der Gruppierung.

1. Gruppierung etabliert ein System *gemeinsamer Aufsicht* über Positionsinhaber. Jeder Abteilung steht ein Management vor. Es ist gegenüber der umschließenden Organisation verantwortlich für die Handlungen der Abteilung.

2. Gruppierung bringt die *gemeinsame Nutzung von Ressourcen* mit sich. Die Mitglieder einer Abteilung teilen in der Regel ein gemeinsames Budget und nutzen häufig gemeinsame Einrichtungen und technische Systeme.

3. Gruppierung schafft *gemeinsame Performanzmaße*. Ebenso wie Mitglieder einer Abteilung gemeinsame Ressourcen teilen, können ihre Aktivitäten und Ergebnisse an den gleichen Maßstäben gemessen werden.

4. Gruppierung fördert *wechselseitige Abstimmung*. Gemeinsame Ressourcen, Performanzmaße und Aufsicht bedingen häufige Interdependenzen der Mitglieder einer Abteilung. Diese wiederum fördern häufigen informellen Kontakt untereinander und ermöglichen die flexible wechselseitige Abstimmung.

[2] Mintzberg verwendet an dieser Stelle nicht den Begriff der Abteilung, sondern beschreibt allgemeiner die Bildung von *organisational units*, bezieht sich dabei aber auf Entitäten innerhalb der Grenzen eines Unternehmens.

Anhand der bisherigen Ausführungen zu Positionen und deren Gruppierung lassen sich *Standardisierung* (Spezialisierung, Formalisierung, Performanzmaße), *direkte Aufsicht* und (informelle) *wechselseitige Abstimmung* als die drei grundsätzlichen Klassen von Koordinationsmechanismen in Abteilungen identifizieren. Dabei tritt die Trennung von Positionen und Positionsinhabern – den Mitgliedern der Abteilung – deutlich zu Tage. Die Positionsspezifikationen werden erst dann mit Leben erfüllt, wenn tatsächliche Akteure die Positionen besetzen. Hier kommt es in der Folge zum in den Kapiteln 3 und 4 der vorliegenden Arbeit wiederholt thematisierten Wechsel- beziehungsweise Widerspiel zwischen Verhalten entlang formaler Organisationsstrukturen und Verhalten entlang emergierender informeller Strukturen. Positions- und Gruppierungsspezifikation kapseln standardisierte Arbeitsabläufe und definieren Beziehungen für direkte Aufsicht. Zusätzlich sind allerdings Interaktionen zwischen Mitgliedern für die informelle wechselseitige Abstimmung möglich (und geradezu erwünscht).

Gerade auf Ebene der Abteilungen mit der direkten Einbettung individueller Akteure hat die informelle Struktur einen größeren Einfluss als auf höheren Systemebenen, auf denen Kollektivitäten agieren. Nichtsdestotrotz gilt in der Folge weiterhin die in Kapitel 7 getroffene Abgrenzung für die vorliegende Arbeit: Das Zusammenspiel von formalen und informellen Strukturen nicht systematisch thematisiert.

11.2.2 Abteilungsmodell

Ausgehend von den Erläuterungen des vorhergehenden Teilabschnittes wird das in Abbildung 11.4 illustrierte abstrakte Modell einer Abteilungs-Organisationseinheit abgeleitet. Die Akteure verteilen sich auf das Management der Abteilung (management, Governance-Akteure), Inhaber von Positionen mit koordinierender Funktion (coordination position holders, Integrati-

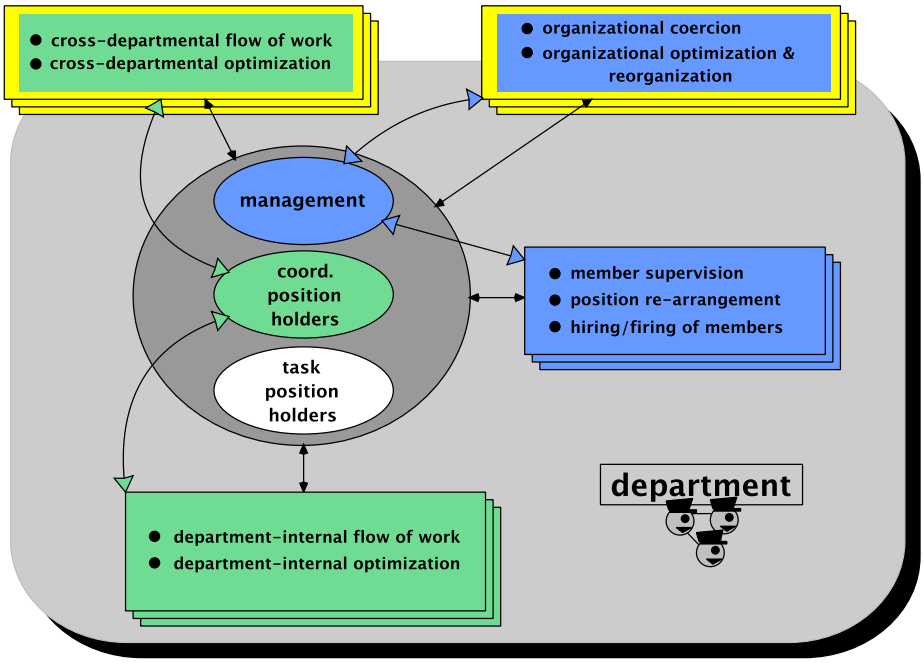

Abbildung 11.4: Abstraktes Modell einer Abteilungs-Organisationseinheit

onsakteure) im Bezug auf positionsübergreifende Arbeitsabläufe sowie Inhaber von Positionen für die reine Ausführung von Aufgaben (task position holders, Operationsakteure). Es ist gegebenenfalls nicht immer ganz einfach, diese drei Mengen von Abteilungsakteuren zu identifizieren. Im Rahmen der ORGAN-Architektur werden *alle* Kollektivitäten auf Organisationsebene als Abteilungs-Organisationseinheiten konzeptionalisiert. Dies gilt also beispielsweise sowohl für niedrigrangige Produktionsabteilungen als auch für Kollektivitäten der höheren Managementebenen wie einer Geschäftsleitung. Die Abbildung zeigt demnach ein sehr allgemeines Modell, das im Hinblick auf bestimmte konkrete Abteilungs-Organisationseinheiten geeignet interpretiert werden muss.

Dies gilt auch für die verschiedenen Arten von Abteilungsaktivitäten, die im Folgenden etwas näher beleuchtet werden. Integrationsaktivitäten setzen sich im Wesentlichen aus den abteilungsinternen alltäglichen Arbeitsabläufen zusammen. Welcher Art diese „alltäglichen Arbeitsabläufe“ sind, kann von Abteilung zu Abteilung stark variieren und hängt stark von der Stellung der jeweiligen Abteilung in der umschließenden Organisation ab. So kann es sich beispielsweise um Aktivitäten zur Herstellung bestimmter Produkte, um Administrationsaktivitäten oder auch um Aktivitäten für die Verhandlung strategischer Vorgaben und organisationsweiter Ziele handeln. Optimierungen der Arbeitsabläufe und Positionsbeziehungen können gegebenenfalls in gewissem Rahmen ebenfalls Teil der Integrationsaktivitäten sein. Weitreichende Modifikationen der Abteilungscharakteristika sind jedoch dem Management vorbehalten.

Abteilungen sind in hohem Maße abhängig von der sie umschließenden jeweiligen Organisation. Das Management wird direkt von der umschließenden Organisation eingesetzt und ist dieser gegenüber verantwortlich für das Verhalten der Abteilungsmitglieder. Zu diesem Zweck nimmt das Management die Konfiguration der Positionen und Positionsbeziehungen vor, beaufsichtigt das Mitgliederverhalten (punktuell) und nimmt die Einstellung sowie das Entlassen von Mitgliedern vor.

Bezüglich der Peripherieaktivitäten findet eine analoge Aufteilung in solche Aktivitäten mit Integrations- und Governance-Charakter statt. Bei Peripherieaktivitäten mit Integrationscharakter handelt es sich um das Spiegelbild der normalen Integrationsaktivitäten: Sie setzen sich aus abteilungsübergreifenden alltäglichen Arbeitsabläufen sowie Optimierungsbemühungen zusammen.

Peripherieaktivitäten mit Governance-Charakter dienen vor allem dem Zweck, die Abteilung entlang der globaleren Organisationsrichtlinien auszurichten. Das Management einer Abteilung kann nur innerhalb der Berechtigungen und Vorgaben, die von der Organisationsebene vorgegeben werden, agieren. Peripherieaktivitäten mit Governance-Charakter dienen dem Abteilungsmanagement darüber hinaus dazu, an Reorganisationsaktivitäten auf Ebene der Organisation teilzunehmen. Es gilt wiederum zu unterscheiden, welche Stellung eine Abteilung auf Ebene der umschließenden Organisation einnimmt. Dem Management einer niedrigrangigen Abteilungen werden typischerweise strikte Vorgaben von Organisationsebene aus auferlegt und es hat nur wenig Einfluss auf organisationsweite Reorganisationsbemühungen. Das Management einer besonders hochrangigen Abteilung hingegen hat gerade aufgrund dieser Stellung selber einen bedeutsamen Einfluss auf die Vorgaben, die von der Organisationsebene aus für Abteilungen erlassen werden, und hat damit das Schicksal seiner Abteilung zu einem großen Teil „selbst in der Hand“.

11.3 Organisationseinheitstyp *Organisation*

Eine Organisation integriert eine Menge von Abteilungen in eine gemeinsame holistische Einheit. Zu diesem Zweck werden verschiedene Anforderungen und Bedürfnisse der Organisation auf verschiedene Abteilungen abgebildet.

11.3.1 Organisationstheoretischer Bezug

Die Ausführungen dieses Abschnittes stellen zum Teil eine knappe Wiederholung der Ausführungen aus Kapitel 4 und speziell des Abschnittes 4.3 dar.

Wie im vorigen Abschnitt beschrieben, werden Abteilungen durch die Definition und Gruppierung von Positionen gebildet. Der Entwurf der gesamten Organisation beschäftigt sich nach Mintzberg [Min83, S. 45] damit, welche und wie viele Abteilungen sukzessive gruppiert werden. Zu diesem Zweck werden erstrangige Abteilungen einer gemeinsamen Dienststelle oder Zentrale zugewiesen und darauf aufbauend Dienststellen und wiederum umfassenderen, allgemeineren Dienststellen zugewiesen. Auf diese Weise entsteht die *Superstruktur* einer Organisation in Form ganzer Arbeitsbereiche, Divisionen oder Geschäftsbereiche.

Die Superstruktur vermittelt die abteilungsübergreifenden Zusammenhänge der Organisation und wird häufig in einem *Organigramm* illustriert. Mintzberg [Min83, S. 73] beschreibt die Superstruktur traditioneller Organisationen als streng vertikal und nach dem *Unity of Command*-Prinzip aufgebaut (jedes Mitglied ist genau einem weiteren Mitglied unterstellt). Bereits in Kapitel 4 wurden die Probleme, die sich im Zusammenhang mit strikten Hierarchien ergeben, und die daraus resultierenden moderneren Organisationsformen ausführlich erörtert. Modernere Organisationsprinzipien erlauben zusätzliche horizontale Verbindungen, um eine direkte Interaktion zwischen den betroffenen Parteien zu erlauben. Solche lateralen Verbindungen reichen von lokal (und häufig auch zeitlich) begrenzten Liaisonpositionen, Integrationsmanagern, Task Forces, und Projektteams über organisationsweite Performanzkontroll- und Aktionsplanungssysteme bis hin zu Netzwerkorganisationen und Matrixorganisationen mit multiplen formalen Autoritätslinien [Min83, Kapitel 4][Sco03, S. 238-244].

Es existieren verschiedene Ansätze der Charakterisierung der verschiedenen Abteilungstypen einer Organisation. So unterscheidet Scott [Sco03, Kapitel 9+10] grob zwei Klassen, den *technischen Kern* und die *Peripheriekomponenten* [Sco03, Kapitel 10]. Der technische Kern setzt sich aus den Abteilungen zusammen, die sich unmittelbar mit der Erbringung und Sicherung domänenspezifischer Leistungen auseinandersetzen, mit der Transformation von Eingaben in Ausgaben zur Produktion oder Diensterbringung. Die Peripheriekomponenten dienen einem weiten Spektrum an Aufgaben, die im weitesten Sinne zur Unterstützung des technischen Kerns dienen. Die Unterstützung des technischen Kerns durch die Peripheriekomponenten ist vielfältiger Natur. Die verschiedenen Facetten spiegeln sich in Verfeinerungen der Klasse der Peripheriekomponenten wieder. So identifiziert Parsons [Par60, S. 60–65] ebenfalls eine *technische Ebene*, die dem technischen Kern nach Scott entspricht. Als *Managementebene* charakterisiert er den Teil der Organisation, der für den Entwurf und die Kontrolle der Produktionssysteme, die Beschaffung von Eingaben, den Absatz von Ausgaben und die Allokation von Personal zu Abteilungen und Funktionen verantwortlich ist. Als *institutionelle Ebene* charakterisiert er den Teil, der die Organisation zu ihrer weiteren Umwelt in Beziehung setzt, ihre Domänen bestimmt, ihre Grenzen festlegt und ihre Legitimität nach außen sowie nach innen hin sichert. Mintzberg [Min83, Kapitel 1] nimmt eine alternative Verfeinerung vor: Managementhierarchie (bei Mintzberg als *middle line* bezeichnet), Support, Technostruktur

(verantwortlich für Standardisierung) und strategischer Apex (verantwortlich für die strategische Ausrichtung).

Die bisherigen Ausführungen geben bereits Aufschluss bezüglich der Machtverhältnisse in einer Organisation. Der strategische Apex nach Mintzberg beziehungsweise die institutionelle Ebene nach Parsons charakterisieren den Teil der Organisation, der die allumfassende Verantwortung trägt. In Kapitel 4 wurde das Thema Ziel-/Strategiesetzung, Macht und Kontrollsysteme vertiefend behandelt. Als besonders wichtiges generisches Konzept für das Verständnis von Organisationen als zielgerichtete Kollektivitäten wurde dabei die *dominante Koalition* nach [CM63, S. 27-32] zitiert. Die dominante Koalition bestimmt, in welchen Domänen sich die Organisation verankert und legt in diesem Zusammenhang die wesentlichen organisatorischen Ziele fest. In der Folge erarbeitet sie die grundsätzlichen Strategien, Strukturen und Vorgehensweisen zum Erreichen und der Wahrung dieser Ziele. Während die dominante Koalition in bestimmten Fällen klar umgrenzt, homogen, und stabil sein, gibt es generell viele verschiedene Faktoren, die ihre Größe und Zusammensetzung über Zeit beeinflussen. Die Zusammensetzung bestimmt sich dabei aus den verschiedenen Machtpositionen sowohl interner Akteure (Besitzer der Organisation, hochrangige Manager, ausgesuchte Vertreter des operativen Personals) als auch Inhabern kritischer Grenzpositionen.

11.3.2 Organisationsmodell

Ausgehend von den Erläuterungen des vorhergehenden Teilabschnittes wird das in Abbildung 11.5 illustrierte abstrakte Organisationseinheitsmodell einer Organisation abgeleitet. Als kollektive Akteure agieren dabei Abteilungen der dominanten Koalition (dominant coalition, Governance-Akteure), Abteilungen mit superstruktureller Funktion (departments with super-

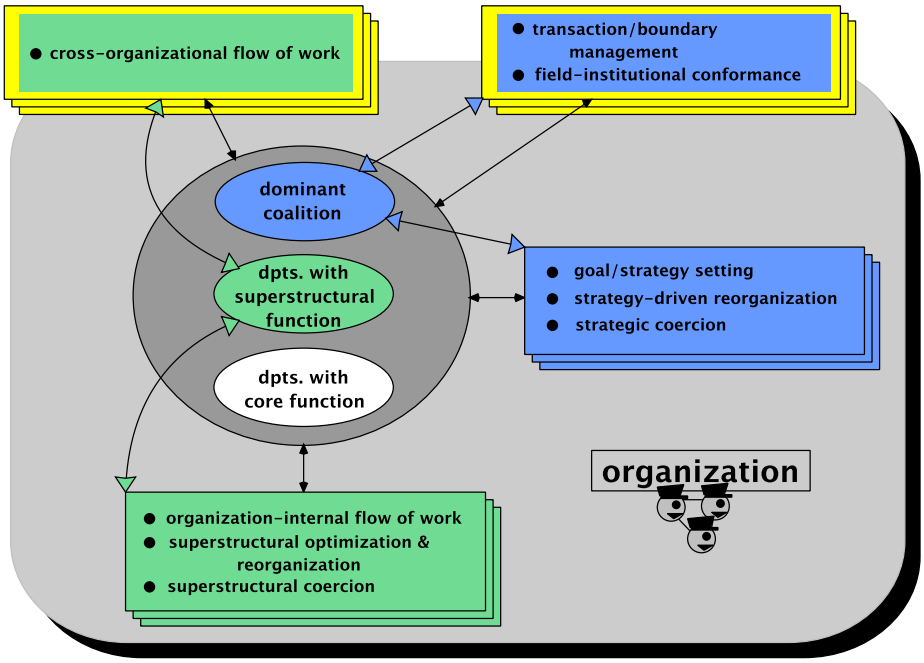

Abbildung 11.5: Abstraktes Organisationseinheits-Modell einer Organisation

structural function, Integrationsakteure) und Abteilungen mit organisatorischer Kernfunktionalität (departments with core function, Operationsakteure). Analog zur Abteilungs-Organisationseinheit gilt auch hier, dass eine Identifizierung beziehungsweise Interpretation dieser Akteursmengen für konkrete Organisationen stark von der Funktion und Stellung der jeweiligen Organisationen in ihren Umgebungen – den organisatorischen Feldern – abhängt. Dies gilt umso mehr für die im Folgenden erläuterten Aktivitäten.

Die wichtigsten Funktion der Integrationsaktivitäten besteht wie im Falle der Abteilung in der Abwicklung der alltäglichen Arbeitsabläufe der Organisation (wiederum gilt, dass die Definition dessen, was „alltägliche Arbeitsabläufe" sind, von der jeweiligen Organisation abhängt). Die superstrukturellen Verbindungen legen dabei fest, welche Abteilungen dabei Integrationsfunktionalität für andere Abteilungen übernehmen.

Darüber hinaus fällt auf, dass sowohl auf Integrations- als auch auf Governance-Seite Aktivitäten für Reorganisation und die Durchsetzung organisatorischer Bestimmungen vorgesehen sind. Dies steht im Gegensatz zur Abteilungsebene, auf der solche Aktivitäten dem Management allein vorbehalten sind. Auf Organisationsebene hingegen tritt eine verstärkte Trennung von Zuständigkeiten auf. Die dominante Koalition unterhält kontinuierlich Aktivitäten, im Zuge derer Organisationsziele und Strategien verhandelt werden. Strategische Entscheidungen werden in die Superstruktur „kodiert" , überwacht und durchgesetzt. Es handelt sich hier also um Reorganisations- und Durchsetzungsbemühungen aus *strategischen Gründen*. Demgegenüber stehen Reorganisations- und Durchsetzungsbemühungen aus *operationalen Gründen* von Seiten der Integrationsakteure. Initiatoren sind dabei typischerweise Abteilungen weit oben in der Organisationshierarchie. Superstrukturelle Reorganisation und Zwang sind dabei immer durch die strategischen Vorgaben der dominanten Koalition gerahmt.

Peripherieaktivitäten dienen der Organisation, um auf den umschließenden organisatorischen Feldern zu operieren. Einerseits betrifft das die alltäglichen organisationsübergreifenden Arbeitsabläufe im Zuge von Peripherieaktivitäten mit Integrationscharakter. Andererseits betrifft das aber auch die Rahmung dieser Arbeitsabläufe in Form von Aktivitäten für das Grenz- und Transaktionsmanagement. Diese Aktivitäten dienen dem Brückenschlag zu externen Organisationseinheiten. Es handelt sich dabei um Peripherieaktivitäten mit Governance-Charakter, deren Steuerung der dominanten Koalition vorbehalten ist. Ein weiterer Aspekt des Grenzmanagements betrifft die Konformität der fokussierten Organisation gegenüber den institutionellen Rahmenbedingungen der umschließenden Felder. Der Begriff der Konformität (conformance) ist schwächer als der für die Eingliederung von Abteilungen in Organisationen verwendete Begriff des Zwangs (coercion). Organisationen *„nehmen an organisatorischen Feldern teil"*, während Abteilungen *„Teile von Organisationen"* sind. Im Falle der Konformität einer Organisation gegenüber einem Feld hat die dominante Koalition einen größeren Spielraum, inwiefern eine institutionelle Anpassung erfolgt.

11.4 Organisationseinheitstyp *Organisatorisches Feld*

Organisatorische Felder stellen die unmittelbaren Umgebungen für Organisationen dar. Dabei kann eine Organisation in mehrere Felder eingebettet sein und Felder lassen sich recht allgemein als Verbünde kooperierender oder konkurrierender Organisationen mit gemeinsamen funktionalen Interessen charakterisieren.

11.4.1 Organisationstheoretischer Bezug

DiMaggio [DiM86] charakterisiert organisatorische Felder als soziale Einheiten, die eine wichtige Zwischenebene für den Brückenschlag zwischen Organisationen und der umfassenden Gesellschaft darstellen. Beispielsweise richten sich staatliche Regulierungen wie beispielsweise Umweltschutzrichtlinien, die eigentlich individuelle Organisationen betreffen, selten an einzelne Organisationen, sondern an ganze industrielle Sektoren und werden daher durch Strukturen und Aktivitäten auf Feldebene vermittelt.

Dementsprechend besteht ein organisatorisches Feld nicht nur aus den Organisationen, die unmittelbar an der Entwicklung, Produktion und Distribution der Kernprodukte oder -dienste entlang des funktionalen Zusammenhangs des Feldes beteiligt sind. Stattdessen beinhaltet ein Feld zusätzliche Organisationen, die als regulierende Behörden, Finanzgeber oder Anbieter ergänzender Leistungen auftreten. DiMaggio und Powell [DP83] qualifizieren ein Feld in diesem Sinne als bestehend aus einer Menge von Organisationen, die zusammen genommen einen gesellschaftlich anerkannten Bereich institutionellen Lebens darstellen.

In der Definition eines organisatorischen Feldes nach DiMaggio und Powell lassen sich bereits die beiden grundsätzlichen Klassen von Eigenschaften erkennen, die Scott [Sco03, S. 133-141] organisatorischen Feldern zu Grunde legt: *Material-Ressourcen-Eigenschaften* und *institutionelle Eigenschaften*. Diese Eigenschaften ebenso wie die konkreteren Aspekte einer institutionellen Umgebung wurden bereits in Kapitel 4 der vorliegenden Arbeit skizziert. Hier erfolgt eine etwas ausführlichere Betrachtung.

Material-Ressourcen-Eigenschaften betonen die Perspektive auf Organisationen als Produktionssysteme. Die Ziele einer Organisation stehen in engem Zusammenhang mit der Verrichtung bestimmter Arten von Tätigkeiten und diese ergeben sich aus der Transformation von Eingaben in Ausgaben. Die Umgebung wird dementsprechend in ihrer Funktion als *Aufgabenumgebung* betrachtet. Ihre Wichtigkeit ergibt sich dabei vor allem aus der offenen Systemperspektive, nach der keine Organisation vollständig selbstgenügsam ist. Stattdessen müssen Anbieter benötigter Dienste auf der einen und Marktanschlüsse auf der anderen Seite gesichert werden, um Abhängigkeitsverhältnissen sowie Unsicherheitsquellen möglichst Herr zu werden. Die Umgebung wird dementsprechend als Bestand von Ressourcen aufgefasst. Der Begriff der Ressource ist an dieser Stelle sehr weitläufig zu verstehen. Er umfasst Rohstoffe und Produkte ebenso wie Dienste und Informationen.

Demgegenüber stehen die institutionellen Eigenschaften eines organisatorischen Feldes. Scott [Sco08, S. 50–58] beschreibt eine Institution als charakterisiert durch *regulative*, *normative* und *kulturell-kognitive* Elemente, die in Kombination mit Aktivitäten und Ressourcen zu deren Durchsetzung Stabilität und Bedeutung in sozialen Systemen herbeiführen. Scott bietet dabei außerdem einen Überblick über die Wirkungsmechanismen der drei verschiedenen Typen von institutionellen Elementen. Formalisierte Regeln werden vor allem durch *externen Zwang* (Belohnung, Sanktion) durchgesetzt, die Verbindlichkeit von Normen entsteht aufgrund *sozialen Drucks* und kulturell-kognitive beziehungsweise soziale Fakten werden im Wesentlichen mittels *Nachahmung* manifestiert.

Die Klassifizierung von Feldeigenschaften in materiell-ressourcenbedingte und institutionelle soll keinesfalls nahe legen, sie seien unabhängig voneinander. Stattdessen operieren *alle* Organisationen in institutionellen Kontexten. Material-Ressourcen-Eigenschaften sind stets durch die Institution des Feldes geprägt. Austauschprozesse auf einem Markt sind beispielsweise keineswegs allein durch universelle ökonomische Prinzipien bestimmt, sondern sind vielmehr in einen Komplex aus Regeln und Praktiken eingebettet. Die beiden Eigenschaftsmengen eines

Feldes sind also nicht strikt voneinander unterscheidbar, sondern kulminieren vielmehr in der *institutionellen Logik* und den *Governance-Strukturen* des Feldes. Friedland [FA91] beschreibt die institutionelle Logik eines Feldes als Praktiken und symbolische Konstruktionen, die die Organisationsprinzipien des Feldes konstituieren und die von den teilnehmenden Organisationen zu elaborieren sind. Governance-Strukturen auf Feldebene beziehen sich nach Scott [SRMC00, S. 137] generell auf alle wie auch immer gearteten Arrangements, durch die Macht und Autorität auf Feldebene ausgeübt wird. Sie manifestieren sich in Form von Organisationen, die die institutionelle Logik *bestimmen* (Regulierungsbehörde, Berufsgenossenschaften, Standardisierungsgremien). Sie setzen die institutionelle Logik durch und administrieren deren Evolution im Laufe der Zeit. Desweiteren existieren Organisationen, die die institutionelle Logik *implementieren* (Ämter und Behörden für die Lizenzierung, Zertifizierung und Legitimation von konkreten Strukturen und Aktivitäten des Feldes, Versicherungen, Notariate, Brokeragenturen).

11.4.2 Modell eines organisatorischen Feldes

Ausgehend von den Erläuterungen des vorhergehenden Teilabschnittes wurde das abstrakte Organisationseinheitsmodell eines organisatorischen Feldes aus Abbildung 11.6 abgeleitet. Als

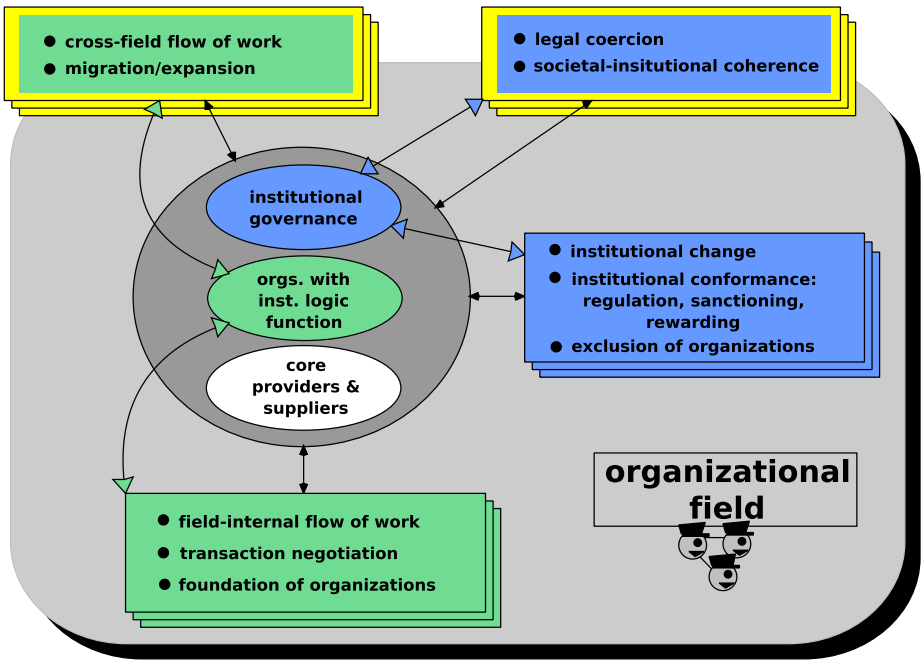

Abbildung 11.6: Abstraktes Organisationseinheits-Modell eines organisatorischen Feldes

Operationsakteure treten Anbieter und Konsumenten von Kerndiensten und -produkten (core providers & suppliers) auf. Bei Integrationsakteuren handelt es sich um solche Organisationen, die eine Funktion entlang der institutionellen Logik des Feldes einnehmen (organizations with institutional logic function). Anders ausgedrückt manifestiert sich die institutionelle Logik in der Existenz dieser Organisationen und ihrer Aufgaben. Als Governance-Akteure treten Organisationen für die Festlegung und Durchsetzung der institutionellen Logik des Feldes auf

(institutional governance). Ganz analog zu den Modellen der Abteilung und der Organisation gilt es, die einzelnen Akteursmengen für konkrete organisatorische Felder geeignet zu interpretieren. Die Interpretation ist stark abhängig von der Stellung eines konkreten Feldes auf Ebene der umschließenden Gesellschaft.

Die Integrationsaktivitäten dienen wiederum der Abwicklung der alltäglichen Arbeitsabläufe auf einem Feld. Diese Arbeitsabläufe finden im Rahmen sowohl der allgemeinen Vorgaben durch die institutionelle Logik als auch der konkreten Transaktionsabkommen zwischen Anbietern und Abnehmern von Kerndiensten und -produkten statt. Das Zustandekommen von Transaktionsabkommen ist ebenfalls Teil der Integrationsaktivitäten. Darüber hinaus sind organisatorische Felder die Quellen für die Gründung neuer Organisationen in den durch die Felder jeweils repräsentierten Systemdomänen.

Governance-Aktivitäten beziehen sich auf die Bestimmung, Festlegung sowie Durchsetzung der institutionellen Logik. Da von Seiten der Feldebene kein direkter Eingriff auf Organisationen erfolgen kann (im Gegensatz zum Verhältnis einer Organisation zu ihren Abteilungen), kann die Konformität von Organisationen zur institutionellen Logik lediglich auf indirekte Weise über Regulierung, Sanktionen und Belohnungen erfolgen. Im äußersten Fall können Organisationen vom Feld ausgeschlossen werden.

Peripherieaktivitäten mit Integrationscharakter beinhalten die Abwicklung der feldübergreifenden Arbeitsabläufe. Da ein organisatorisches Feld eine in sich weitestgehend abgeschlossene (funktional zusammenhängende) Domäne darstellt, kommt feldübergreifenden Arbeitsabläufen eine eher geringe Bedeutung zu. Von größerer Wichtigkeit sind Aktivitäten, im Zuge derer Organisationen zwischen Feldern migrieren beziehungsweise von einem Feld aus auf weitere Felder expandieren. Dies betrifft also die Ausweitung des Operationsskopus einzelner Organisationen auf neue Domänen. Gesellschaftliche Richtlinien schränken dabei ein, welche Übergänge zwischen Feldern möglich sind und auf welchen Felder simultan operiert werden darf.

Peripherieaktivitäten mit Governance-Charakter beziehen sich im Wesentlichen auf den bereits in Abschnitt 11.1 dieses Kapitels thematisierten Aspekt, dass einzelne organisatorische Felder in ein institutionelles Rahmenwerk auf Gesellschaftsebene eingebettet sind. Die Governance-Strukturen verschiedener Felder müssen dementsprechend aufeinander abgestimmt sein damit sich ein kohärentes Gesamtbild ergibt. Je nach Stellung eines konkreten organisatorischen Feldes im Kontext der umschließenden Gesellschafts-Organisationseinheit kann es Zwang auf andere Felder ausüben beziehungsweise einen solchem Zwang selber ausgesetzt sein.

11.5 Organisationseinheitstyp *Gesellschaft*

Die Gesellschaft als (zunächst) höchste Architekturebene stellt den oberen Systemabschluss eines ORGAN-Multiorganisationssystems dar und hat dementsprechend keine übergeordneten Einheiten.

11.5.1 Organisationstheoretischer Bezug

Die Ebene der Gesellschaft wird in der Organisationstheorie weit weniger eingehend behandelt als die bisher betrachteten Ebenen eines realweltlichen Organisationssystems. Dies ist nicht zuletzt dem Umstand geschuldet, dass die Systemperspektive auf Organisationen als offene Systeme ohnehin erst relativ spät Einzug in die Organisationstheorie gehalten hat und

sich die damit einhergehenden Untersuchungen zunächst der Ebene der organisatorischen Felder widmeten. Allerdings wurde bereits in Abschnitt 11.1 dieses Kapitels der Übergang von Institutionen auf Feldebene zu Institutionen auf Gesellschaftsebene beziehungsweise sogar darüber hinaus thematisiert. Oberhalb der organisatorischen Felder finden sich beispielsweise der Nationalstaat beziehungsweise die Regierung als Einrichtungen, die ein institutionelles Rahmenwerk für ganze Felder anstatt für konkrete Organisationen aufspannen.

Ganz allgemein gilt, dass es schwierig ist, von einer direkten Einflussnahme der Gesellschaft auf individuelle Akteure zu sprechen da es sich um einen sehr großen Granularitätsunterschied zwischen den Einheiten auf beiden Ebenen handelt. Daher wurde bereits in Teilabschnitt 3.1 der vorliegenden Arbeit die Notwendigkeit der Betrachtung von Meso-Strukturen für einen sukzessiven Übergang von der feingranularen „Angesicht-zu-Angesicht"-Mikro-Ebene zu den globalen Gesellschaftsstrukturen auf Makro-Ebene thematisiert. Die bislang betrachteten Kollektivitätstypen der organisatorischen Felder, Organisationen und Abteilungen stellen in diesem Sinne genau solche Meso-Strukturen aus einer organisationsorientierten Perspektive bereit.

Die Gesellschaft wird hier also als eine gemeinsame Umgebung für organisatorische Felder aufgefasst. Alle Felder unterstehen den Vorgaben einer gemeinsamen Konstitution, deren gesetzliche Inhalte von einer Regierung determiniert und durchgesetzt werden. Bestimmte Gesetze gelten dabei ausnahmslos für alle Felder. Zusätzlich können sich institutionelle Gerüste für Teilmengen von Feldern herausbilden.

11.5.2 Gesellschaftsmodell

Ausgehend von den Erläuterungen des vorhergehenden Teilabschnittes wurde das abstrakte Organisationseinheits-Modell der Gesellschaft aus Abbildung 11.7 abgeleitet. Als Operations-

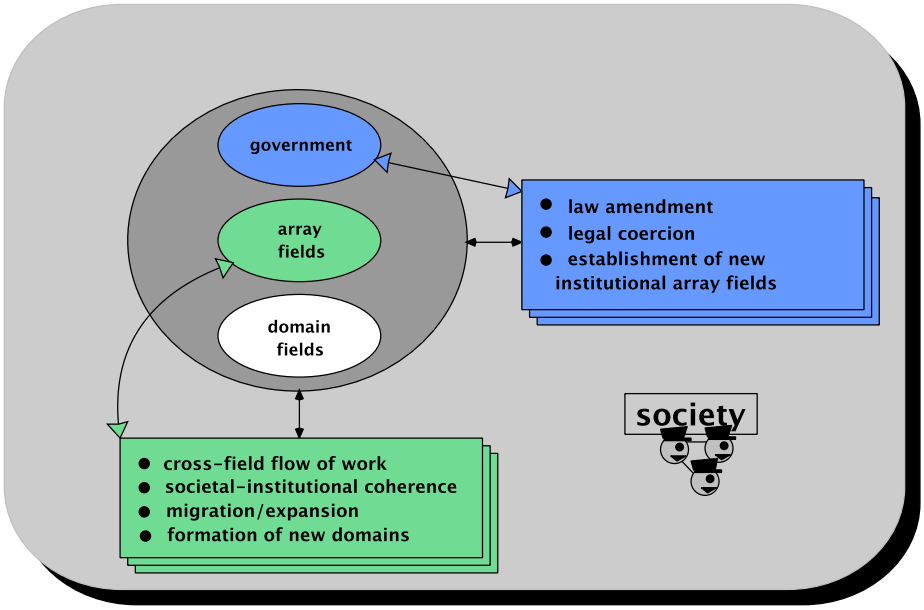

Abbildung 11.7: Abstraktes Organisationseinheits-Modell der Gesellschaft

akteure treten dabei Felder auf, die sich auf eine konkrete anwendungsspezifische Kerndomäne des Gesamtsystems beziehen (domain fields). Die Teilnehmer dieser Felder teilen gemeinsame funktionale Interessen: In der Gesellschaft sollen ganz konkrete Dienstleistungen erbracht und genutzt werden können. Als Integrationsakteure treten solche Felder auf, die globalere institutionelle Rahmenwerke für multiple Felder konstituieren (array fields). Hier teilen die Teilnehmer eher gemeinsame pragmatische Interessen: Bestimmte Typen von organisatorischen Feldern sollen entlang bestimmter einheitlicher Vorgaben ausgerichtet werden. Schließlich unterstehen alle Felder einer gemeinsamen Regierung, zusammengesetzt aus Feldern als Governance-Akteuren (government).

Die Aktivitäten auf Gesellschaftsebene beschränken sich auf Integrations- und Governance-Aktivitäten. Integrationsaktivitäten umfassen feldübergreifende Arbeitsabläufe, allerdings wird ihnen wie bereits im vorigen Abschnitt erwähnt eine eher geringe Bedeutung zugemessen. Von größerer Wichtigkeit sind Aktivitäten, die der Etablierung und Durchsetzung feldübergreifender institutioneller Rahmenwerke dienen, Migration und Expansion von Feldteilnehmern ermöglichen und zur Formation neuer Domänen (und damit Feldern) führen.

Governance-Aktivitäten beziehen sich auf gesellschaftsweit geltende *Gesetze*. Die der Regierung zugehörigen Felder können die Einhaltung dieser Gesetze auf anderen Feldern erzwingen. Gesetze können sich zwar auch ändern, allerdings geschieht das selten. Stattdessen stellen sie einen verlässlichen und stabilen Rahmen für alle weiteren Organisationseinheiten eines ORGAN-Multiorganisationssystems dar. Gegebenenfalls werden im Zuge von Governance-Aktivitäten neue integrierende Felder als Antwort auf neu entstehende Domänen etabliert.

11.6 Diskussion: Einsatz der Referenzarchitektur

Bei der Modellierung der Organisationseinheitstypen der ORGAN-Architektur wurde das abstrakte Grundmodell einer Organisationseinheit als Basis herangezogen. Dadurch ergibt sich eine sehr allgemeine Charakterisierung von ORGAN-Multiorganisationssystemen, ganz im Sinne der intendierten Referenzarchitektur als Modell für eine Vielzahl verschiedener Ausprägungen. Allerdings bleibt die konkrete Funktionsweise von ORGAN-Multiorganisationssystemen sehr vage. Dies steht im Gegensatz zu den im vorherigen Kapitel betrachteten Systemen auf Basis von Ausprägungen des operationalen Grundmodells einer Organisationseinheit.

Daher wird der Fokus in der Diskussion dieses Abschnittes auf verschiedene Aspekte des praktischen Einsatzes der Referenzarchitektur gerichtet. Es gilt hier aber zu betonen, dass tatsächliche Erfahrungen mit ernstzunehmenden praktischen Anwendungen zum gegenwärtigen Zeitpunkt ausstehen. Erste ausführbare ORGAN-Multiorganisationssysteme auf Basis der operationalen Modelle des vorigen Kapitels gehen über den Status rudimentärer Prototypen nicht hinaus. Insofern handelt es sich bei den Ausführungen dieses Abschnittes vor allem um Einschätzungen und Hypothesen des Autors der vorliegenden Arbeit und um einen Ausblick auf zukünftige Forschungsbestrebungen. Somit kann man diesen Abschnitt als Abschluss des inhaltlichen Körpers der vorliegenden Arbeit und als Überleitung zur Schlussbetrachtung der Arbeit auffassen.

11.6.1 Softwaretechnische Implikationen

In diesem Teilabschnitt werden die Architekturrollen der verschiedenen Organisationseinheitstypen unter einer softwaretechnischen Perspektive skizziert. Dazu gehören auch die softwaretechnischen Implikationen, die sich aus den erstellten Modellen ergeben. Allerdings

dürfen die hier skizzierten Implikationen nicht als obligatorisch aufgefasst werden. Dazu ist der Variationsspielraum für Ausprägungen der einzelnen Organisationseinheitstypen schlicht zu groß. Es geht hier darum, gewisse *archetypische* Eigenschaften zu postulieren.

Gesellschaft Es gibt genau eine Organisationseinheit des Typs *Gesellschaft* in einem ORGAN-Multiorganisationssystem. Die Gesellschaft repräsentiert den oberen Systemabschluss. Jedes Softwaresystem dient einem bestimmten Zweck beziehungsweise im Falle eines Systems von Systemen in der Regel mehreren Zwecken. Dies spiegelt sich in der Architekturrolle der Gesellschaft wieder. Die Gesellschaft repräsentiert die Abbildung eines bestimmten Ausschnitts der Realität und dieser Ausschnitt kann verschiedene Zwecke betreffen. Verschiedene Abgrenzungen des Gesamtsystems werden vorgenommen und Verbindungen zwischen ihnen determiniert. Für jede der Abgrenzungen wird ein eigenes organisatorisches Feld eingebettet. Systemweite Gesetze werden spezifiziert und spezielle (Regierungs-)Organisationseinheiten wachen über diese Gesetze und bedienen sich Governance-Strukturen auf Feldebene, um die Gesetze dort durchzusetzen.

In einem softwaretechnischen ORGAN-Multiorganisationssystem handelt es sich bei der Gesellschafts-Organisationseinheit um ein lose gekoppeltes Netzwerk von Feld-Organisationseinheiten. Dieses Netzwerk weist insofern eine zentrale Steuerung auf, als dass die zur Regierung gehörenden Felder allumfassende Systemgesetze schaffen und durchsetzen. Allerdings handelt es sich bei diesen Gesetzen eher um weitläufige und allgemeine Richtlinien, konkretere und in hohem Maße einschränkende Regeln werden erst auf Feldebene geschaffen. Insofern haben in die Gesellschaft eingebettete Felder einen hohen Grad an Autonomie und weisen untereinander ein hohes Maß an Diversität auf. Insgesamt entspricht dies der bereits in Kapitel 2 für großangelegte Softwaresysteme formulierten Forderung, auf hohen Systemebenen von direkter Steuerung zu einer indirekten, regulierenden Form der Steuerung überzugehen.

Organisatorisches Feld Jedes organisatorische Feld bietet ein konsistentes, weitestgehend abgeschlossenes Bild eines bestimmten Anteils des Gesamtsystems (einen eigenen „Lebensraum“). Felder stellen die unmittelbaren Umgebungen für Organisationen dar. Teilnehmende Organisationen haben funktionale Interessen gemein und begegnen einander als eigenständige, gleichberechtigte Akteure (kooperierend beziehungsweise konkurrierend aus Eigeninteresse). Ihnen wohnt ein besonders hohes Maß an Akteursautonomie inne. Daher ist es genau die Ebene der organisatorischen Felder, auf der Softwaresysteme, die verschiedenen (realweltlichen) Besitzern und Interessenvertreten zuzuordnen sind, als Organisationen integriert werden. Die durch ein organisatorisches Feld aufgespannte Umgebung manifestiert sich in Form von Mechanismen, die bestimmte Praktiken und symbolische Konstruktionen etablieren. Letztere wiederum haben eine sowohl einschränkende als auch unterstützende Funktion hinsichtlich möglicher Interaktionen zwischen Feldteilnehmern. Dies erleichtert die Konstruktion von Organisationen da diese sich auf die Elaborierung und Internalisierung der feldweiten Richtlinien stützen können.

In einem softwaretechnischen ORGAN-Multiorganisationssystem handelt es sich bei der Organisationseinheit eines organisatorischen Feldes um ein Netzwerk von Akteuren, welches die drei analytischen Aspekte der Governance, Integration und Operation weitestgehend kanonisch widerspiegelt: Organisationen für das Setzen der feldweiten Praktiken und Konstruktionen, Akteure für deren Durchsetzung und Organisationen, die unter diesem Einfluss Kerndienstleistungen erbringen und nutzen. Es gibt auf Feldebene ein relativ hohes Maß an Gleichför-

migkeit innerhalb bestimmter Teilmengen der eingebetteten Organisationen (organisatorische Populationen) da viele Organisationen ähnliche Ziele verfolgen und unter ähnlichen Bedingungen operieren. Gegebenenfalls gilt diese Gleichförmigkeit lediglich für bestimmte Anteile oder Aspekte der Organisationen da sie auf multiplen Feldern operieren können.

Neben der Tatsache, dass Operationen mehrere Felder als ihre Umgebungen auswählen können und damit die Abhängigkeit einer Organisation von einem bestimmten Feld bereits geschwächt wird, können Organisationen ihre Interna gegenüber dem Einfluss der umschließenden Felder puffern. Felder können nur indirekte Einflussnahme (Sanktionen, Belohnung, Ausschluss) auf ihre eingebetteten Organisationen ausüben.

Organisation Analog zum Agenten in der Agentenorientierung stellt die Organisationseinheit der Organisation die zentrale Modelleinheit im Kontext der ORGAN-Architektur dar. Die Organisation bildet den Kern der Abstraktion und bestimmt damit, auf welchem Abstraktionsgrad das Gesamtsystem betrachtet wird.

Organisationen sind in organisatorischen Felder eingebettet, abhängig von den Domänen, in denen sie operieren. Eine Organisation setzt sich aus Abteilungen zusammen. Die Zusammenstellung und Anordnung der Abteilungen einer Organisation spiegelt die Komplexität wider, der die Organisation in ihren gesammelten Umgebungen begegnet. Verschiedene Bedürfnisse und benötigte Funktionen der Organisation ihren Umgebungen gegenüber werden auf verschiedene Abteilungen abgebildet. Im Gegensatz zu organisatorischen Feldern, auf denen Organisationen als Akteure mit einem besonders hohen Maß an Autonomie operieren, handelt es sich bei den Abteilungen einer Organisation um Akteure, die besonders stark von der (einzigen) sie umschließenden Organisation abhängen. Abteilungen existieren ausschließlich im Auftrag beziehungsweise im Namen ihrer Organisation. Zu diesem Zweck werden Abteilungen einer Organisation unter einer gemeinsamen Superstruktur und einer gemeinsamen Strategie angeordnet.

In einem softwaretechnischen ORGAN-Multiorganisationssystem handelt es sich bei der Organisationseinheit einer Organisation um ein Netzwerk von Abteilungen, das im Wesentlichen hierarchischer Natur ist, auch wenn die Hierarchie um verschiedene Formen horizontaler Verbindung angereichert sein kann. Auf diese Weise können die Governance-Abteilungen der Organisation strategische Entscheidungen für die Organisation als Ganzes treffen und Kontrolle über deren Implementierung sukzessive weiter nach unten in der Organisationshierarchie delegieren.

Die Autonomie einer Abteilung hängt somit stark von ihrer Stellung in der Organisationshierarchie ab. Zwar gilt wie oben beschrieben, dass Abteilungen ausschließlich im Namen ihrer Organisation existieren. Allerdings haben Abteilungen weiter oben in der Hierarchie einen großen Einfluss auf die globalen Organisationscharakteristika und damit auch auf ihr eigenes Schicksal. Gleichförmigkeit tritt unter Abteilungen im Grunde genommen nur dann auf, wenn gleichartige Abteilungen aus Skalierungsgründen bewusst geschaffen werden. Ansonsten sind die Abteilungen einer Organisation sehr divers und nehmen unterschiedliche superstrukturelle Funktionen ein.

Abteilung Abteilungen bilden den unteren Abschluss eines ORGAN-Multiorganisationssystems. Auf dieser Ebene agieren individuelle Akteure, so dass hier im Grunde genommen die „eigentlichen“ Aktionen, die sich gegebenenfalls als Kollektivitätsaktionen auf darüber liegenden Ebenen manifestieren, originär auftreten.

In einem softwaretechnischen ORGAN-Multiorganisationssystem handelt es sich bei einer Abteilungs-Organisationseinheit um ein stark gekoppeltes Netzwerk individueller Akteure mit vielen verschiedenen Verbindungen. Die Existenz eines Managements sorgt zwar für eine gewisse Zentralisierung, doch in der Hauptsache bestimmen sich die Beziehungen zwischen Akteuren auf Basis der Arbeitsabläufe. Dabei wechselt die Interpretation eines Akteurs entlang der analytischen Aspekte Governance, Integration und Operation von Fall zu Fall. Allgemeine Aussagen bezüglich Gleichförmigkeit und Autonomie der Mitglieder einer Abteilung lassen sich nur schwer treffen, da sie stark von den konkreten Charakteristika der jeweiligen Abteilung abhängen. Generell lässt sich sagen, dass Akteursautonomie in gleichem Maße abnimmt, wie der Grad an Bürokratie zunimmt. Allerdings ist festzuhalten, dass dies lediglich die Autonomie eines individuellen Akteurs in Bezug auf seine Mitgliedschaft in der Abteilung betrifft. Individuelle Akteure sind nicht von Abteilungen umschlossen, sondern treten ihnen freiwillig bei, solange es ihren eigenen Zwecken dient.

11.6.2 Multiperspektivischer Einsatz der Referenzarchitektur

Die ORGAN-Referenzarchitektur umfasst vier Ebenen von Organisationseinheiten. Dabei stellt sich die Frage, ob diese Ebenen ausreichen. Unter organisationstheoretischer Betrachtung fällt beispielsweise auf, dass sich Kollektivitäten wie strategische Allianzen oder virtuelle Organisationen nicht ohne weiteres auf einer der vier Ebenen verorten lassen. Es handelt sich bei diesen Beispielen um zielgerichtete, eigenständige Kollektivitäten und sie müssten mithin eigentlich als Organisations-Organisationseinheiten umgesetzt werden. Allerdings betten sie keine (nicht-autonomen) Abteilungen sondern eigenständige Organisationen ein und müssten mithin eigentlich als Abteilungs-Organisationseinheiten umgesetzt werden.

Es bieten sich verschiedene Lösungen der Problematik an. Beispielsweise könnte die Referenzarchitektur um weitere Ebenen erweitert werden. Eine andere Möglichkeit besteht darin, das Konzept des organisatorischen Feldes breiter zu interpretieren, damit es auch Kollektivitäten wie Allianzen, Koalitionen, Merger oder virtuelle Organisationen einschließt. Nach Ansicht des Autors der vorliegenden Arbeit bietet sich eine dritte Möglichkeit allerdings eher an. Die Referenzarchitektur kann *multiperspektivisch* Verwendung finden. Es gibt vier verschiedene *Typen* von Organisationseinheiten, die im Hinblick auf eine weitere Verfeinerung und im Hinblick auf ihre Einbettung jeweils spezifische Charakteristika und Fähigkeiten aufweisen. Eine konkrete Organisationseinheit eines ORGAN-Multiorganisationssystems kann dabei unter Umständen *mehrere* Typen in sich vereinen, indem sie *mehrere* Rollen der Referenzarchitektur (gegebenenfalls sogar in *mehrfachen* Instanzen) einnehmen kann. So kann beispielsweise eine konkrete Organisationseinheit einer anderen gegenüber in der Rolle einer Organisation einem Feld gegenüber auftreten, einer wiederum anderen gegenüber aber in der Rolle einer Abteilung einer Organisation gegenüber. Die fokussierte Organisationseinheit muss in diesem Fall dementsprechend die Funktionalität beider Typen von Organisationseinheiten in sich vereinen und verschiedene Arten Einbettungsbeziehungen nach oben hin implementieren. Die gemeinsame Grundlage für alle Typen von Organisationseinheiten in Form des universellen Modells unterstützt das Auftreten konkreter Organisationseinheiten als Vertreter mehrerer Typen.

In diesem Zusammenhang ist auch denkbar, die Organisationseinheit der Gesellschaft „nach oben hin zu öffnen“ und auf diese Weise weitere Architekturebenen zuzulassen. Dieser Aspekt liegt allerdings nicht im Rahmen der vorliegenden Arbeit und es ist fraglich, ob er überhaupt noch innerhalb des Skopus eines Organisationsparadigmas liegt und oder ob nicht stattdessen ein erneuter Paradigmenwechsel erforderlich ist.

11.6.3 Konkrete Entwicklungsansätze für Organisationseinheitstypen

Die verschiedenen Typen von Organisationseinheiten der ORGAN-Referenzarchitektur wurden in diesem Kapitel unter Bezugnahme des abstrakten Modells einer Organisationseinheit spezifiziert. Damit wurde vor allem eine Überführung organisationstheoretischer Konzepte in die Nomenklatur und das Konzeptgebilde des in der vorliegenden Arbeit verwendeten Modells einer Organisationseinheit vorgenommen. Im Hinblick auf die softwaretechnische Umsetzung von ORGAN-Multiorganisationssystemen stellt sich natürlich die Frage nach spezifischen Entwicklungsansätzen. Hier bietet sich ein Rückbezug auf die Betrachtungen des vorigen Kapitels an. Dort wurden ganz konkrete Typen von Organisationseinheiten anhand des operationalen Modells einer Organisationseinheit betrachtet. Auf diese Weise ergeben sich ausführbare Prototypen. Zum Abschluss des Kapitels wurde der Frage nachgegangen, inwiefern sich die verschiedenen Typen von Organisationseinheiten für den Einsatz auf verschiedenen Systemebenen eines Multiorganisationssystems eignen. Die dort postulierte Passung wird im Folgenden auf die spezifische Betrachtung von ORGAN-Multiorganisationssystemen übertragen.

Im Falle von $\mathcal{M}$OISE$^+$-Organisationseinheiten wurde argumentiert, sie würden sich besonders gut für die Einbettung individueller Akteure auf niedrigen Systemebenen eignen. Beziehungen zwischen Akteuren können auf sehr detailreiche Art und Weise spezifiziert werden. Nicht alle möglichen $\mathcal{M}$OISE$^+$-Spezifikationen sind dabei mit einer klaren Semantik hinterlegt, so dass Spielraum für wechselseitige Abstimmung der individuellen Akteure bleibt. Adaptionen der Spezifikation einer $\mathcal{M}$OISE$^+$-Organisationseinheit können relativ leichtgewichtig und somit relativ hochfrequent erfolgen. Aufgrund dieser Eigenschaften folgt eine gute Eignung von $\mathcal{M}$OISE$^+$-Organisationseinheiten für den Einsatz als Abteilungen eines ORGAN-Multiorganisationssystems.

Im Falle von SONAR-Organisationseinheiten wurde argumentiert, sie würden sich besonders gut dafür eignen, weitere Organisationseinheiten als kollektive Akteure einzubetten. Sie bieten einen eingeschränkten Satz an Spezifikationsmöglichkeiten für das Beziehungsgeflecht zwischen Akteuren, dabei bleibt allerdings eine eindeutige Semantik stets gewahrt. Insgesamt erleichtert dies die Inkorporierung von SONAR-Vorgaben durch eingebettete Kollektivitäten. Adaptionen einer SONAR-Organisationseinheit sind möglich, gestalten sich aufgrund der engen Verzahnung aller Modelldimensionen jedoch als recht schwergewichtig und sollten daher nicht häufig vorgenommen werden. SONAR-Organisationseinheiten sind außerdem zweckgerichtete kollektive Akteure, so dass eingebettete Kollektivitäten entlang der Systemziele einer SONAR-Organisationseinheit ausgerichtet werden. Aus diesen Eigenschaften folgt eine gute Eignung von SONAR -Organisationseinheiten für den Einsatz als Organisationen eines ORGAN-Multiorganisationssystems.

Im Falle von ISLANDER-Organisationseinheiten wurde argumentiert, sie würden sich besonders gut für höhere Ebenen eines Systems von Systemen eignen. Sie stellen einen Übergang von zielgerichteten Organisationen zu regulierenden Institutionen dar. Sie bieten sich an, kollektive Akteure in Form von Organisationen einzubetten und eine Regulierung von deren Verhalten im Kontext einer spezifischen Domäne des Gesamtsystems vorzunehmen. Für ISLANDER existiert aktuell kein konkreter ausgereifter Ansatz der Adaption von Institutionsspezifikationen. Allgemein gilt aber, dass institutionelle Rahmenbedingungen idealerweise nur selten verändert werden sollten damit sie einen stabilen und für die eingebetteten kollektiven Akteure verlässlichen Rahmen bieten. Insgesamt ergibt sich eine gute Eignung von ISLANDER-Organisationseinheiten für den Einsatz als organisatorische Felder eines ORGAN-Multiorganisationssystems. Darüber hinaus wurde auch eine Idee skizziert, wie großangelegte

Institutionen mit multiplen Systemebenen auf Basis von ISLANDER-Organisationseinheiten realisiert werden können, so dass sich hier eine Möglichkeit des Einsatzes von ISLANDER-Organisationseinheiten für die Ebene der Gesellschaft eines ORGAN-Multiorganisationssystems bietet.

Die somit postulierte Passung konkreter Entwicklungsansätze für die jeweiligen ORGAN-Ebenen ist natürlich lediglich ein Beispiel. Es gibt weitere Ansätze aus dem Bereich der Multiagentensysteme, die herangezogen werden könnten. Darüber hinaus muss man die Ausführungen dieses Teilabschnittes als einen „ersten Wurf“ betrachten. Die jeweiligen Passungen wurden anhand sehr knapper Argumentationen vorgenommen. Speziell steht in Frage, ob sich die wechselseitigen Konstitutionsprozesse realweltlicher Organisationssysteme (vergleiche die Top-Down- und Bottom-Up-Prozesse in Abbildung 11.1 aus Abschnitt 11.1) tatsächlich in ORGAN-Multiorganisationssystemen in Form der oben beschriebenen Systemkomposition aus $\mathcal{M}$OISE$^+$-, SONAR- und ISLANDER-Organisationseinheiten umsetzen ließen. Zusammenfassend bleibt allerdings festzuhalten, dass die Multiagentensystemforschung bereits ein reichhaltiges Reservoir an konkreten Modellierungs- und Entwicklungsansätzen für Organisationen und Institutionen auf Softwareebene bietet. Bislang wurden diese Ansätze weitestgehend für den Einsatz jeweils *einer* Organisation/Institution für individuelle Agenten verwendet. Nach Ansicht des Autors der vorliegenden Arbeit ist es an der Zeit, die Möglichkeiten für den Einsatz in Systemen mit kollektiven Akteuren zu ergründen. Das universelle Modell einer Organisationseinheit sowie der Referenzarchitekturvorschlag ORGAN bieten hier einen allgemeinen Rahmen für eine gemeinsame Verortung.

11.7 Zusammenfassung

Dieses Kapitel stellt den Abschluss des inhaltlichen Körpers der vorliegenden Arbeit und einen Übergang zum Ausblick im folgenden Kapitel dar. Aufbauend auf den Betrachtungen konkreter Typen von Organisationseinheiten im vorigen Kapitel wurde hier eine systematischere Betrachtung der Typisierung von Organisationseinheiten und deren Kombination vorgenommen. Ausgehend von Architekturprinzipien sozialer Organisationssysteme wurde die Referenzarchitektur ORGAN für Multiorganisationssysteme erstellt und beschrieben. Allerdings erfolgte die Beschreibung auf sehr abstrakter Ebene und auf Basis des abstrakten Grundmodells einer Organisationseinheit aus Kapitel 8. Damit manifestiert sich bereits der Charakter der Ausführungen des Kapitels als Vision und als Ausgangspunkt zukünftiger Forschungsbestrebungen. Die Diskussion zum Ende des Kapitels bezüglich Möglichkeiten des praktischen Einsatzes der Referenzarchitektur hat in diesem Zusammenhang bereits erste konkretere Perspektiven offengelegt.

Interessanterweise stellte die Referenzarchitektur (in einem sehr viel früheren Stadium als hier vorgestellt) eigentlich den Ausgangspunkt der Forschung für die vorliegende Arbeit dar. In der Folgezeit war der Autor mit der Erarbeitung der Grundlagen beschäftigt, die für eine genauere Beschreibung und einen Einsatz der Architektur notwendig sind. Diese Grundlagen in Form organisationstheoretischer Kernkonzepte, verschiedener Modellierungstechniken für Multiorganisationssysteme und verschiedener Grundmodelle von Organisationseinheiten (sowohl abstrakter als auch operationaler/ausführbarer Natur) wurden in der Arbeit ausführlich vorgestellt. Erst jetzt sind damit die Grundlagen geschaffen, sich der Referenzarchitektur und ihrem Einsatz tatsächlich widmen zu können. Somit stellt die Architektur zum wiederholten Male den Ausgangspunkt zukünftiger Forschung dar.

12 Schlussbetrachtung

Zum Schluss der vorliegenden Arbeit wird eine Zusammenfassung des Inhaltes und der Ergebnisse sowie ein Ausblick auf zukünftige Forschungsbestrebungen vorgenommen.

12.1 Zusammenfassung und Ergebnisse

Zunächst werden hier der Inhalt der einzelnen Kapitel zusammengefasst und die jeweiligen Ergebnisse herausgestellt. Das Gesamtergebnis der Arbeit wird unter Bezugnahme der in der Einleitung formulierten Ziele bewertet. Schließlich werden einige wichtige Veröffentlichungen des Autors, die im Kontext der Arbeit entstanden, aufgelistet.

12.1.1 Kapitelübersicht

In der Folge werden die Inhalte und die Ergebnisse der einzelnen Kapitel zusammengefasst. Am Ende des Teilabschnittes erfolgt eine Bewertung des Gesamtergebnisses der Arbeit.

Kapitel 2: *Softwaresysteme im Großen als Systeme von Systemen* Zu Beginn des ersten Teils der Arbeit ist Kapitel 2 den Charakteristika und Problemstellungen der Art von Softwaresystemen gewidmet, die überhaupt erst den Ausgangspunkt der Forschung der vorliegenden Arbeit ausmachen. Es handelt sich hierbei um großangelegte Softwaresysteme, die als Systeme von Systemen charakterisiert sind. Diese Eigenschaft überträgt sich häufig vom realweltlichen Gesamtkontext auf die Softwaresysteme. Systeme von Systemen sind durch operationale und administrative Unabhängigkeit/Eigenständigkeit ihrer Teilsysteme gekennzeichnet, so dass das übergeordnete System zumindest zu einem gewissen Grad auf der „freiwilligen“ Teilnahme und Kooperation der Teilsysteme basiert. Dieser Umstand stellt Zweckmäßigkeit, Steuerbarkeit und Stabilität des Gesamtsystems in Frage. Das wichtigste Ergebnis des Kapitels besteht in der Identifizierung von vier Hauptproblemstellungen beziehungsweise Herausforderungen, denen sich die Softwareentwicklung in diesem Kontext stellen muss: (1) Dezentralisierung in multiplen Dimensionen, (2) Auftreten multipler Systemebenen, (3) Notwendigkeit soziotechnischer Perspektiven, (4) Notwendigkeit von Software mit Selbst-Eigenschaften.

Kapitel 3: *Motivation eines Organisationsparadigmas der Softwareentwicklung* Die in Kapitel 2 identifizierten Problemstellungen werden in Kapitel 3 herangezogen, um ein Organisationsparadigma der Softwareentwicklung zu motivieren. Die Argumentation wird entlang von vier Kernmerkmalen von Organisationen in sozialen Gesellschaften geführt: (1) Organisationen als Meso-Strukturen für einen sukzessiven Übergang der feingranularen Mikro-Ebene zur Makro-Ebene der umschließenden Gesellschaft, (2) Organisationen als Ordnungsstifter in sozialen Systemen mit eigeninteressierten Akteuren, (3) Organisationen als inhärent soziotechnische Systeme und (4) Organisationen als kollektive Akteure. Die Versprechungen dieser Merkmale werden explizit zu den vier Problemstellungen aus Kapitel 2 in Bezug gesetzt. Vereinfacht formuliert lässt sich Folgendes festhalten: Soziale Systeme sind inhärent durch

Dezentralisierung, multiple Systemebenen und sozio-technische Kontexte gekennzeichnet und es sind gerade Organisationen, die als Mittel für die Beherrschung dieser Aspekte so gefragt sind. Als zusätzliches Argument wird angeführt, Software mit Selbst-Eigenschaften würde von der Systemorganisation als explizitem Laufzeitkonzept profitieren. Das Ergebnis des Kapitels ist das vorgetragene Plädoyer an sich. Nach Kenntnisstand des Autors ist eine Argumentation für ein Organisationsparadigma in der hier vorgenommenen Form, nämlich ausgehend von *grundsätzlichen* softwaretechnischen Problemstellungen, bislang nicht geführt worden.

Kapitel 4: *Konkretisierung von organisationstheoretischen Konzepten und Mechanismen*
Die Argumentation aus Kapitel 3 beruft sich auf grundsätzliche Versprechungen eines Organisationsparadigmas. Wie organisationsorientierte Softwaresysteme konkret aussehen, bleibt eher vage. Als erster Schritt hin zu einem klareren Verständnis wird in Kapitel 4 eine eingehende Betrachtung der organisationstheoretischen Literatur vorgenommen, um relevante soziale Konzepte und Mechanismen zu identifizieren, die als konzeptionelle Grundlage von organisationsorientierten Softwaresystemen dienen sollen. Ein besonderer Schwerpunkt liegt dabei auf der organisationsorientierten Meso-Strukturierung von sozialen Systemen. Soziale Organisationssysteme sind durch multiple Ebenen des kollektiven Handelns charakterisiert. Der Begriff der Organisation ist aber zumindest traditionell eher mit der Analyseebene eines Unternehmens assoziiert. Daher wird in dem Kapitel der Begriff der *organisierten Kollektivität* als Oberbegriff für zielgerichtete und durch eine Formalstruktur gekennzeichnete kollektive Akteure in sozialen Organisationssystemen eingeführt.

Das Kapitel ist mithin generischen/universellen Konzepten und Mechanismen im Zusammenhang mit organisierten Kollektivitäten gewidmet. Dabei wird zunächst das Wesen einer organisierten Kollektivität als offenes System und als kollektiver Akteur in einer Umwelt betrachtet. Hier werden Grenzen/Grenzziehung, allgemeine Charakteristika von Umgebungen, Einflussnahme von Seiten der Umwelt, Einflussnahme auf die Umwelt von Seiten der organisierten Kollektivität sowie die generelle rekursive/hierarchische Struktur von Organisationssystemen thematisiert. Bei diesen Betrachtungen werden die Interna einer organisierten Kollektivität zwar wiederholt thematisiert, die Ausführungen bleiben jedoch unsystematisch. Daher wird anschließend eine explizite Betrachtung des Zustandekommens, der Aufrechterhaltung und der Evolution von internen Strukturen einer organisierten Kollektivität betrachtet. Hier werden Ziele/Zielsetzung, die Errichtung und Funktionsweise von Steuerungs- und Kontrollsystemen sowie die daraus folgende strukturelle Komplexität einer organisierten Kollektivität betrachtet. Darüber hinaus werden die Ausführungen zu Steuerung und Kontrolle in Bezug zu den allgemeineren Prinzipien der Kybernetik gesetzt.

Auch wenn sich die zitierten Literaturstellen zum Teil auf Organisationen im klassischen Sinne (Unternehmen) beziehen, so wurden sie nach Ansicht des Autors derart gewählt, dass sie einen generischen Charakter haben und für den allgemeinen Fall der organisierten Kollektivität gelten. Das Ergebnis des Kapitels ist somit die Schaffung einer sozial/organisationstheoretisch fundierten Grundlage für eine janusgesichtige Gestaltung von Softwaresystemen. Dabei sind die Ausführungen des Kapitels ausführlicher als es für die im zweiten Teil der Arbeit vorgenommene Identifizierung der Modellelemente einer Organisationseinheit strenggenommen notwendig wäre. Allerdings geht es in dem Kapitel auch darum, prinzipielle Charakteristika von sozialen Organisationssystemen aufzuzeigen, die sich im Angesicht eines Organisationsparadigmas der Softwareentwicklung zumindest potentiell auf Strukturen und Abläufen in Softwaresystemen übertragen.

Kapitel 5: *Organisationsstrukturen in Softwaresystemen – State of the Art* Auf Basis der nach Ansicht des Autors für organisationsorientierte Softwaresysteme fundamentalen Kernkonzepte aus Kapitel 4 wird in Kapitel 5 eine Betrachtung des *State of the Art* vorgenommen. Auf diese Weise wird im Kontext der Arbeit auch erstmals konkret vermittelt, wie organisationsorientierte Softwaresysteme (nach dem aktuellen Stand der Forschung) eigentlich aussehen. Der Löwenanteil ist dabei dem Bereich der Multiagentensysteme und dem hier vorherrschenden Trend weg vom individuellen Agenten hin zur Agentenorganisation gewidmet. Unter Berufung auf die Kapitel 3 und 4 wird herausgearbeitet, dass die Organisationsfunktionen als Ordnungsstifter und als sozio-technische Systeme bereits weitreichend ausgenutzt werden. Dagegen wird die Funktion als Meso-Struktur lediglich eingeschränkt und die Funktion als kollektiver Akteur ungenügend ausgenutzt. Neben organisationsorientierten Multiagentensystemen werden noch der Ansatz der Viable Software und betriebliche Softwaresysteme betrachtet. Auch hier können Bestrebungen erkannt werden, Konzepte aus dem Bereich sozialer Organisationen auf die Gestaltung von Softwaresystemen zu übertragen. Allerdings handelt es sich in beiden Fällen um eine weniger „reine“ Anwendung der Idee eines organisationsorientierten Softwaresystems nach Kapitel 3. Zusammenfassend lässt sich als Ergebnis dieses Kapitels zweierlei festhalten. (1) Einerseits wird der Bedarf einer Organisationsorientierung für die Entwicklung von Softwaresystemen auf breiter Basis festgestellt, so dass die Anschlussfähigkeit der vorliegenden Arbeit an die aktuelle Forschung und Praxis unterstrichen wird. (2) Andererseits werden die Defizite bei der Anwendung/Ausnutzung der Organisationsmetapher in aktuellen Ansätzen klar herausgearbeitet, so dass der Anschlusspunkt für das weitere Vorhaben der Arbeit (im zweiten und dritten Teil) deutlich identifiziert wird.

Kapitel 6: *Petrinetzmodellierung* Bevor aufbauend auf der Kritik am *State of the Art* in Kapitel 5 das Hauptvorhaben der Arbeit im zweiten Teil in Angriff genommen wird, wird der erste Teil mit Kapitel 6 zum Thema Petrinetzmodellierung abgeschlossen. Hier werden verschiedene Petrinetzformalismen und damit einhergehende Möglichkeiten der Modellierung vorgestellt. Als besonders wichtig wird dabei der Formalismus der Referenznetze hervorgehoben. Der Formalismus basiert auf dem Konzept der Netze in Netzen und bringt in diesem Zusammenhang mächtige Möglichkeiten der Modellierung von Synchronsationsdynamiken über multiple Systemebenen hinweg mit sich. Insgesamt stellen Referenznetze eine besonders attraktive Kombination der Anschaulichkeit von Petrinetzmodellen mit höheren Konzepten für die Operationalisierung komplexer Sachverhalte dar. Referenznetze haben eine Tradition und Erfolgsgeschichte hinsichtlich ihrer Verwendung für die Modellierung sozialer beziehungsweise sozial fundierter Systeme. Auf entsprechende Fälle wird im Kapitel verwiesen und somit die Verwendung von Referenznetzen für das Modellierungsvorhaben des zweiten und dritten Teils der Arbeit vorbereitend motiviert. Das Ergebnis des Kapitels besteht vor allem in der Stellungnahme zu den Vorteilen der Modellierung auf Basis des Konzeptes der Netze in Netzen und der Herausstellung der erwiesenen Eignung an verwandter Stelle.

Kapitel 7: *Eingrenzung und Anforderungen* Der zweite Teil der Arbeit beginnt mit Kapitel 7, in dem eine Vorbereitung des Modellierungsvorhabens im Hauptteil der Arbeit stattfindet. In diesem Kapitel werden die bereits in der Einleitung der Arbeit vorweggenommenen Begriffe der Organisationseinheit und des Multiorganisationssystems konkretisiert. Multiorganisationssysteme werden explizit als konzeptionelle Erweiterung von Multiagentensystemen beschrieben. Die Organisationseinheit fungiert dabei als gegenüber Multiagentensystemen neuartige Entität. Sie wird als softwaretechnisches Gegenstück zur in Kapitel 4 behandelten orga-

nisierten Kollektivität eingeführt. Somit inkorporiert sie einen janusgesichtigen Charakter und erlaubt auf dieser Basis den rekursiven Aufbau von Multiorganisationssystemen. In der Folge des Kapitels wird zunächst eine Eingrenzung des Modellierungsvorhabens für den Rest der Arbeit vorgenommen, um ein in sich geschlossenes und handhabbares Paket zu schnüren. Im Wesentlichen wird die Eingrenzung derart vorgenommen, dass sich die Modelle auf die logischen Beziehungen zwischen Organisationseinheiten in einem Multiorganisationssystem beschränken. Fragen des Deployments, die Modellierung informeller Kollektivitätsstrukturen oder die Modellierung individueller Akteure (Agenten) werden nicht vertieft betrachtet. Darauf aufbauend werden konkrete Anforderungen an die Modellierungstechniken an sich sowie an das zu modellierende Konzept einer Organisationseinheit formuliert. Die grundlegende Anforderung an die Techniken ist dabei, verschieden abstrakte Perspektiven zu unterstützen, von abstrakten und anschaulichen bis hin zu technischen und ausführbaren Modellen. An das Konzept einer Organisationseinheit werden eine Vielzahl von Anforderungen gestellt. Das Fundament ist dabei die Forderung, die drei konzeptionellen Richtungen des „nach unten/innen Schauens“, des „auf die eigene Ebene Schauens“ und des „nach oben/außen Schauens“ inhärent zu unterstützen. Weitere, feingranulare Anforderungen werden in Form von konkreten Fragestellungen formuliert, deren Beantwortung durch das Modell einer Organisationseinheit geleitet werden soll. Diese Fragestellungen werden unmittelbar von den Ausführungen in Kapitel 4 zu organisierten Kollektivitäten in sozialen Organisationssystemen abgeleitet. Die Ergebnisse des Kapitels sind genau die vorgenommene und begründete Eingrenzung des weiteren Vorhabens sowie die Formulierung von konkreten Anforderungen.

Kapitel 8: *Abstrakte Modelle von Organisationseinheiten* Kapitel 8 ist abstrakten Modellen von Organisationseinheiten (und darauf aufbauend von Multiorganisationssystemausschnitten) gewidmet. Die Modelle sind abstrakt in dem Sinne, dass von bestimmten Aspekten bei der Betrachtung von Organisationseinheiten (wie beispielsweise der expliziten Berücksichtigung von Systemregeln oder der Betrachtung der Evolution von Strukturen über die Zeit) abstrahiert wird. Darüber hinaus werden verschiedene Abstraktionsstufen unterschieden und damit einhergehend verschiedene Modellierungstechniken vorgestellt. Insgesamt erlauben die abstrakten Modelle die Darstellung von Situationen und Abläufen eines Multiorganisationssystem auf anschauliche Weise. Je höher der Abstraktionsgrad gewählt wird, desto weniger stehen konkrete Ablaufdetails im Vordergrund. Stattdessen werden vor allem analytische Systemanteile und funktionale Beziehungen zwischen ihnen modelliert.

Die Modellierungstechniken dienen eigentlich dem Zweck, sogenannte *Kollektivitätseinheiten* als Vorstufe zu Organisationseinheiten zu modellieren. Die Modelle basieren auf den Kernaspekten der Akteure, Aktivitäten und Akteursbeziehungen. Kollektivitätseinheiten werden als janusgesichtige Entitäten modelliert, die sowohl (künstlicher) sozialer Kontext für Akteure als auch selber kollektive Akteure sind. Organisationseinheiten werden als spezielle Ausprägung von Kollektivitätseinheiten modelliert. Die Ausprägung basiert auf der Benennung und Interpretation von Akteursmengen, Aktivitätsmengen und Beziehungen in Form der Governance-Integration-Operation-Konfiguration, die genau die drei in Kapitel 7 geforderten konzeptionellen Richtungen abdeckt und in der Folge der Arbeit die inhaltliche Grundlage für die Betrachtung von Organisationseinheiten bildet.

Im Kapitel werden drei Modellierungstechniken vorgestellt, die eine sukzessive Modulation der Modellabstraktion und damit den Einbezug größerer oder kleinerer Systemausschnitte in mehr oder weniger kompakte Modelle erlauben. Bei *Akteur/Aktivitätsablauf-Modellen* handelt es sich um Petrinetzmodelle (genauer: Referenznetzmodelle). Allerdings fehlen in der

Regel Operationalisierungsdetails für eine direkte Ausführbarkeit. Bei *Akteur/Aktivitätsrollen-Modellen* wird von konkreten Abläufen von Aktivitäten abstrahiert. Stattdessen werden nur Aktivitätsrollen und deren Besetzung durch Akteure betrachtet. *Akteurs-/Aktivitätsmengen-Modelle* erlauben schließlich ein beliebiges Anheben der Modellabstraktion durch die Assoziation von Akteurs- und Aktivitätsmengen. Ausnahmslos alle Modelle basieren auf operationaler Petrinetzsemantik, auch wenn diese nicht in allen Modellen sichtbar wird.

Das erste Ergebnis des Kapitels besteht in der Bereitstellung der drei Modellierungstechniken für die Erstellung abstrakter Modellen von Kollektivitätseinheiten und aus ihnen zusammengesetzten Systemen. Das zweite Ergebnis besteht in der Erarbeitung eines abstrakten universellen Grundmodells einer Organisationseinheit (als Akteurs-/Aktivitätsmengen-Modell) als konkreter Ausprägung einer Kollektivitätseinheit.

Kapitel 9: *Operationale Modelle von Organisationseinheiten* In Kapitel 9 werden operationale Modelle von Organisationseinheiten und Multiorganisationssystemen betrachtet. Auch die Modelle des vorherigen Kapitels basieren auf operationaler Petrinetzsemantik. In diesem Kapitel steht die Nutzung dieser operationalen Semantik für die Erstellung ausführbarer Modelle jedoch prominent im Vordergrund. Die Modelle werden auf Basis von Referenznetzen erstellt. Wie zuvor werden zunächst Modelle von Kollektivitätseinheiten als Vorstufe von Organisationseinheiten erstellt und dabei wird ebenfalls auf die Kernkonzepte der Akteure, Aktivitäten und Akteursbeziehungen zurückgegriffen. Die erstellten Modelle sind jedoch sehr viel technischer und detaillierter als die in Kapitel 8 erstellten abstrakten Modelle. Erst die operationalen Modelle dieses Kapitels erlauben die Adressierung aller Aspekte einer Organisationseinheit inklusive der Behandlung elaborierter Systemregeln und der Evolution von Strukturen über die Zeit. Im Gegenzug sind die Modelle im Detail schwerer zu durchdringen und bieten einen weniger anschaulichen Zugang zu Darstellungen von konkreten Situationen eines Multiorganisationssystems. Können die abstrakten Modelle des vorigen Kapitels gegebenenfalls als „Denkmodelle“ herhalten, so handelt es sich bei den operationalen Modellen dieses Kapitels um „Programmiermodelle“. Sie erlauben die Modellierung der exakten, technischen Funktionsweise von Organisationseinheiten und der Dynamik der darauf aufbauenden Multiorganisationssystemausschnitte. Die Modelle sind zudem reich genug, um anhand des Werkzeugs Renew direkt ausgeführt zu werden. Sie können als softwaretechnische Prototypen aufgefasst werden, auf Basis derer Multiorganisationssysteme programmiert werden können.

Analog zum vorigen Kapitel werden Organisationseinheiten auf Basis von Kollektivitätseinheiten modelliert. Wie zuvor besteht der Übergang in der Einführung der Governance-Integration-Operation-Konfiguration in die Modelle. War zuvor lediglich eine Ausprägung notwendig, so ist im Falle der operationalen Modelle eine Erweiterung in Form zusätzlicher Modellelemente erforderlich. Dies geschieht zu dem Zweck, Systemregeln im Modell zu explizieren und das Verhalten im Kontext einer Organisationseinheit in Abhängigkeit von den Regeln zu modellieren. Abhängig von der mehr oder weniger elaborierten Handhabung von Systemregeln werden mehrere Grundmodelle von Organisationseinheiten präsentiert.

Das Ergebnis des Kapitels besteht in der Präsentation von mehreren operationalen Grundmodellen einer Organisationseinheit. Diese haben einen universellen Charakter und können als Basis für die Ausgestaltung unterschiedlichster konkreter Organisationseinheiten dienen. Noch stärker als im vorigen Kapitel wird anhand dieser Grundmodelle ein spezifischer Beitrag zur Petrinetzmodellierung und vor allem für den Bereich der Modellierung mit Netzen in Netzen geleistet. Die vorgestellten Netzmuster und Richtlinien für ihre Kombination stellen eine eigenständige Form von *organisationsorientierten Petrinetzen* dar.

Kapitel 10: *Konkrete Organisationseinheiten* Bis zu Kapitel 10 werden Modellierungstechniken und universelle Grundmodelle vorgestellt. Die Illustration von Mechanismen wird anhand von Trivialbeispielen vorgenommen. In diesem Kapitel werden schließlich konkrete Beispiele von Organisationseinheiten und Multiorganisationssystemausschnitten betrachtet. Dies geschieht anhand von Ausprägungen des (abstrakten sowie operationalen) Grundmodells einer Organisationseinheit entlang von drei Ansätzen für Agentenorganisationen in Multiagentensystemen: SONAR, $\mathcal{M}$OISE$^+$ und ISLANDER. Auf diese Weise wird die Tragfähigkeit der Grundmodelle untermauert und eine erste inhaltliche Typisierung von Organisationseinheiten vorgenommen. Das Aufgreifen von Beispielen aus dem Multiagentensystembereich stellt einerseits die Anschlussfähigkeit der Ergebnisse der vorliegenden Arbeit an die aktuelle Multiagentensystemforschung heraus. Andererseits kommt dabei auch besonders anschaulich zum Vorschein, welche Defizite im Bereich der Agentenorganisationen noch gegenüber dem erweiterten Konzept der Organisationseinheit in Multiorganisationssystemen herrschen. Für alle drei Typen von Organisationseinheiten wird nämlich betrachtet, inwieweit der jeweils zugrunde liegende Organisationsansatz den Übergang von einer Organisationseinheit zur Kombination von mehreren Organisationseinheiten unterstützt. In keinem der Ansätze wird dieser Aspekt als erstrangig betrachtet und unmittelbar unterstützt. Allerdings lassen sich jeweils Konzepte identifizieren, die für diesen Zweck genutzt werden können (am stärksten ist dies im Falle von SONAR ausgeprägt). Darauf aufbauend können die im Kapitel betrachteten Organisationseinheiten als Bindeglied für exemplarische Betrachtungen des Übergangs von klassischen Agentenorganisationen zu Multiorganisationssystemen herhalten.

Die Ergebnisse des Kapitels bestehen dementsprechend in genau den beschriebenen Punkten: Demonstration der Tragfähigkeit des Modells einer Organisationseinheit, Demonstration der Anschlussfähigkeit an die aktuelle Multiagentensystemforschung und Demonstration der Eignung von Organisationseinheiten als Bindeglieder für die Einbettung klassischer Agentenorganisationen in einen Multiorganisationssystemkontext.

Kapitel 11: *Referenzarchitektur* Organ *für Multiorganisationssysteme* Kapitel 11 stellt das letzte Kapitel des inhaltlichen Körpers der Arbeit dar. Hier wird aufbauend auf dem abstrakten Grundmodell einer Organisationseinheit eine Referenzarchitektur für Multiorganisationssysteme vorgeschlagen. Die Fundierung dieser Architektur liegt wiederum im Bereich sozialer Organisationen und den dort betrachteten Analyseebenen. Für die Referenzarchitektur ORGAN für Multiorganisationssysteme werden vier Ebenen/Typen von Organisationseinheiten abgeleitet: Abteilungen, Organisationen, organisatorische Felder und Gesellschaft. Allerdings ist festzuhalten, dass die Architektur auf sehr hoher Abstraktionsebene beschrieben und charakterisiert wird. Der konkrete praktische Einsatz steht zum gegenwärtigen Zeitpunkt noch aus. Einerseits ist die Vorstellung der Architektur also das Ergebnis des Kapitels. Andererseits stellt die Architektur vor allem eine Vision und eine Überleitung auf den Ausblick des vorliegenden Kapitels dar. Interessanterweise stellte die Architektur (in einem rudimentären Stadium) den konzeptionellen Ausgangspunkt für die vorliegenden Arbeit dar. Erst mit den nun erarbeiteten Ergebnissen der Arbeit ist jetzt die Grundlagen geschaffen, sich der Referenzarchitektur und ihrem Einsatz tatsächlich widmen zu können.

Bewertung des Gesamtergebnisses der Arbeit Zusammenfassend gilt, dass die beiden in der Einleitung der Arbeit festgelegten übergeordneten Ziele inklusive der Teilziele erfüllt wurden.

1. *Motivation eines Organisationsparadigmas der Softwareentwicklung:* Aufbauend auf Problemstellungen einer wichtigen Kategorie von Softwaresystemen wurde ein Orga-

nisationsparadigma der Softwareentwicklung motiviert. Darauf aufbauend erfolgte eine Auswahl von relevanten Konzepten aus dem Bereich sozialer Organisationssysteme.

2. *Vermittlung eines klaren Verständnisses von organisationsorientierten Softwaresystemen und Bereitstellung von Mitteln für die Modellierung:* In der Arbeit wurde vor allem eine ausführliche Betrachtung und vertiefte Diskussion von Multiorganisationssystemen als konkreter Form von organisationsorientierten Softwaresystemen vorgenommen. Insgesamt wurden vier Modellierungstechniken für Organisationseinheiten und Multiorganisationssysteme vorgestellt. Alle basieren auf operationaler Petrinetzsemantik, decken dabei aber das gesamte Spektrum für die Erstellung sehr abstrakter und anschaulicher bis hin zu technisch detaillierten und ausführbaren Modellen ab. Auch wenn die ausführbaren Modelle in ihren Einzelheiten schwerer zu durchdringen sind, so basieren sie doch auf intuitiv zugänglichen Modellzusammenhängen. Speziell mit den ausführbaren Grundmodellen wird ein spezifischer Beitrag für die Petrinetzmodellierung und dabei vor allem für den Bereich der Modellierung mit Netzen in Netzen geliefert. Man kann hier von einer eigenständigen Form der organisationsorientierten Petrinetze sprechen. Abschließend und als Anknüpfpunkt zukünftiger Forschung wurde ein abstrakter Vorschlag einer Referenzarchitektur für Multiorganisationssysteme unterbreitet.

12.1.2 Veröffentlichungen

Über die Arbeit verteilt wurden Veröffentlichungen des Autors zitiert, die im Zuge der Forschungsbestrebungen im Kontext der Arbeit entstanden. Die wichtigeren dieser Publikationen werden hier noch einmal zusammengefasst. Es handelt sich um eine eingeschränkte Auswahl.

Die folgenden Veröffentlichungen entstanden direkt im Zusammenhang mit den Kerninhalten der vorliegenden Arbeit und der Autor der Arbeit fungiert jeweils als Hauptautor.

- Im Konferenzartikel [WEMRM07] für die *Software Engineering* Konferenz (SE'2007) wird die Vision eines Organisationsparadigmas der Softwareentwicklung erstmals präsentiert und durch eine erste grobe Version der ORGAN-Referenzarchitektur begleitet.

- Im Konferenzartikel [WEM08c] für die *International Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS'2008) wird das abstrakte Grundmodell einer Organisationseinheit vorgestellt (seinerzeit noch als Systemeinheit bezeichnet, da der Begriff der Organisationseinheit für die vier Einheitstypen der Referenzarchitektur vorbehalten war). Auf Basis dessen wird die Referenzarchitektur ORGAN genauer beschrieben.

- Im Artikel [WEKBM09] für den *International Workshop on Engineering Societies in the Agents World* (ESAW'2008) wird eine systematische Diskussion der Passung zwischen verschiedenen Ebenen der Referenzarchitektur ORGAN und verschiedenen Organisationsansätzen aus dem Multiagentensystembereich geführt.

- Im Artikel [WEM10] für die Fachtagung *Modellierung* (MOD'2010) der Gesellschaft für Informatik werden die verschiedenen Modellierungstechniken für die Erstellung abstrakter Modelle von geschachtelten Akteurssystemen vorgestellt (es handelt sich um eine frühere Version der in Abschnitt 8.1 im Zusammenhang mit Kollektivitätseinheiten vorgestellten Techniken).

- Im Artikel [WEMKB11] für die Zeitschrift *Transactions on Petri Nets and other Models of Concurrency* (ToPNoC) wird das operationale Grundmodell einer Organisationseinheit basierend auf Referenznetzen vorgestellt und die Funktionsweise anhand eines Beispielszenarios erläutert.

Neben diesen Veröffentlichungen zu den Kerninhalten der Arbeit hat der Autor noch an weiteren Veröffentlichungen mitgewirkt, die zumindest in engem Zusammenhang mit den Inhalten stehen. Dies betrifft Veröffentlichungen zu den theoretischen Grundlagen und vor allem zu Möglichkeiten der Operationalisierung von SONAR-Modellen. Besonders nennenswert sind hier der Zeitschriftenbeitrag [KBMWE09], der Konferenzbeitrag [KBWE09c] und die beiden Workshopbeiträge [KBWE09a, KBWEM10]. Als Hauptautor fungiert in allen Fällen Michael Köhler-Bußmeier. Die Mitwirkung des Autors der vorliegenden Arbeit hat die Grundlage für die ausführliche Betrachtung der Überführung von SONAR-Spezifikationen in das operationale Modell einer Organisationseinheit in Kapitel 10 bereitet.

Eine weitere nennenswerte Veröffentlichung unter Mitwirkung des Autors ist der Konferenzbeitrag [CDD+08] zum Thema der petrinetzbasierten agentenorientierten Softwareentwicklung im MULAN/CAPA-Kontext (vergleiche Abschnitt 6.2). Insbesondere für zukünftige Bestrebungen des praktischen Einsatzes der in der vorliegenden Arbeit erstellten Modelle und Techniken spielt dieser Bereich eine wichtige Rolle.

12.2 Ausblick

Im Laufe der Arbeit wurden an verschiedenen Stellen inhaltliche Eingrenzungen vorgenommen (insbesonders in der Einleitung und Kapitel 7). Diese Eingrenzungen stellen gleichzeitig Anknüpfpunkte zukünftiger Forschung dar. Die entsprechenden Möglichkeiten werden hier noch einmal knapp zusammengefasst und durch einige weitere Punkte ergänzt.

Nutzung der Ergebnisse für die Softwareentwicklung Hier sieht der Autor den wichtigsten und dringlichsten Anschlusspunkt. Daher sind die Ausführungen ausführlicher als zu allen folgenden Punkten. Wie bereits in der Einleitung erläutert, stand die Verwendung der erstellten Grundmodelle von Organisationseinheiten für den Einsatz in realen Softwareanwendungen nicht im Mittelpunkt. Die vollständig beschrifteten und ausführbaren Referenznetzmodelle von Organisationseinheiten können eingesetzt werden, um mit Hilfe des Werkzeugs RENEW Multiorganisationssysteme zu programmieren. Allerdings handelt es sich dabei um eine Prototypisierung und nicht um eine Möglichkeit der Entwicklung ernsthafter Anwendungen für den praktischen Einsatz. Zum einen fehlt aktuell eine umfassende Werkzeugunterstützung, so dass für die Programmierung jeder Organisationseinheit sehr viel Handarbeit erforderlich ist. Zum anderen führt der Rückgriff auf die mächtigen Konzepte des Referenznetzformalismus (Netze in Netzen, Synchronisation über Systemebenen hinweg, Unifikation etc.) zwar zu klaren Modellkonzepten, dies steht einem Einsatz der Modelle in realen Umgebungen aber zunächst im Weg. Als Beispiel kann hier die Aktionssynchronisation über Systemebenen hinweg in einem Schaltschritt dienen. Es ist zunächst unklar, wie dieser Mechanismus in einer Umgebung mit über mehrere Rechner *verteilten* Organisationseinheiten geschehen soll. Hier wird eine Nachbildung beziehungsweise eine Approximierung der Mechanismen notwendig.

In der Einleitung der Arbeit wurde erwähnt, dass die prototypische Modellierung/Implementierung auf Basis von Petrinetzen bereits an verwandter Stelle eingesetzt wurde (vergleiche Objektsysteme [Mol96], Multiagentensysteme [Röl04], Plugin-Systeme [Duv09],

Agentenorganisationen [KB09]). In jedem Fall mussten Folgearbeiten für eine softwaretechnische Nutzbarkeit der zuvor erstellten konzeptionellen Modelle sorgen. Besonders erfolgreich ist dies im Fall der Agentenmodelle von Rölke [Röl04] geschehen. Cabac [Cab10] bettet die Modelle (bei punktueller Erweiterung/Modifikation) in ein vollständiges Vorgehensmodell inklusive durchgängiger Werkzeugunterstützung für die Anwendungsentwicklung ein. Derartige Folgearbeiten sind auch im Falle der in dieser Arbeit vorgestellten Modelle notwendig. Als besonders lohnenswerten Anknüpfpunkt sieht der Autor genau die von Cabac bereitgestellten Möglichkeiten der petrinetzbasierten agentenorientierten Softwareentwicklung. Die dabei vorherrschenden klassischen Agentenkonzepte bedürfen einer konzeptionellen Erweiterung hin zu Multiorganisationssystemen. Einige technische Voraussetzungen dafür existieren bereits. So hat Schleinzer [Sch07] die Idee von Rölkes Plattformagenten aufgegriffen und in Form einer Erweiterung des Mulan/Capa-Frameworks umgesetzt. Auf inhaltlicher Ebene gilt es zunächst, die für Organisationseinheiten charakteristische Governance-Integration-Operation-Konfiguration einzuflechten. Darüber hinaus stellen die in der Arbeit präsentierten Modellierungstechniken den Ausgangspunkt für eine Sprache zur Beschreibung von Multiorganisationssystemen dar. Allerdings wurde der Aspekt der Anwendungsentwicklung in diesem Zusammenhang bisher nicht betrachtet. Hier bietet sich gegebenenfalls eine Erweiterung der von Cabac bereitgestellten Sprache für die Entwicklung von Multiagentenanwendungen an.

Mit den Modellen der Arbeit geht der Anspruch einher, universelle Konzepte und Mechanismen zu spezifizieren, die auch auf andere Art und Weise als mittels Petrinetzsimulation umgesetzt werden können. Als allgemeine Möglichkeit bietet sich beispielsweise die Art und Weise an, wie Organisationsinfrastrukturen in der aktuellen Multiagentensystemforschung bevorzugt umgesetzt werden (vergleiche Abschnitt 5.2). Es werden spezielle *Middleware-Manageragenten* eingesetzt, die die (formale) Organisation verkörpern. Für die Umsetzung von Organisationseinheiten könnten beispielsweise spezielle Systemregelmanager, Akteursmanager, Aktivitätsmanager und Peripheriemanager für eine verteilte, interaktive Administration der notwendigen Mechanismen sorgen. Relevant ist in diesem Zusammenhang die Arbeit von Tu [Tu00], der allgemeine Herangehensweisen für die Realisierung regelbasierter Steuerungsmechanismen in verteilten, offenen Systemen betrachtet. Allgemeiner gilt, dass die durch das Organisationseinheitnetz verkörperte „Middleware-Schicht“ schlicht auf alternative Weise implementiert werden kann, um sie für andere Deployment-Umgebungen nutzbar zu machen.

Tragweite der Modelle und Modellierungstechniken Die vorgestellten Techniken und Modelle wurden in der Arbeit mit den Anspruch präsentiert, der Modellierung (bis hin zur Prototypisierung) von Softwaresystemen zu dienen. Aufgrund der sozialen/organisationstheoretischen Fundierung stellt sich die Frage, ob sich eine Ausweitung des Modellierungsanspruchs anbietet. Eine besondere Versprechung einer solchen Untersuchung sieht der Autor darin, softwaretechnische und soziale Systeme *gemeinsam/integrativ* zu modellieren. Dies könnte insbesondere dem in Abschnitt 5.4 thematisierten Business-IT-Alignment zuträglich sein. Als noch allgemeinerer Rahmen bietet sich das Mikropolis-Modell nach Rolf [Rol08] an. Hier wird die sozio-technische Wechselwirkung von IT-Systemen, Informatiksystemen (IT-Forschung und Herstellung) und IT-anwendenden Organisationen betrachtet und darüber hinaus in den Kontext von Wechselwirkungen zwischen gesamtgesellschaftlichen Entwicklungen und der Entwicklungsdynamik der Technik gesetzt.

Petrinetztheorie Die Möglichkeit des Rückgriffs auf formale Verifikationsmethoden, die sich aus der Verwendung einer Modellierungstechnik mit einer formalen Semantik wie den Re-

ferenznetzen ergeben, wurden im Kontext der Arbeit nicht thematisiert. In Zukunft müssen zunächst einmal überhaupt erst konkrete gewünschte Eigenschaften von Organisationseinheiten und Multiorganisationssystemausschnitten definiert werden. Dies muss parallel zur Betrachtung geschehen, welche Eigenschaften nach aktuellem Forschungsstand für Referenznetze/Objektnetze effizient nachweisbar sind. Gegebenenfalls müssen die Grundmodelle geeignet/punktuell eingeschränkt werden, um sie Verifikationsverfahren zugänglich zu machen.

Archetypen von Organisationseinheiten und Organ-Referenzarchitektur Die Frage nach konkreten Archetypen von organisierten Kollektivitäten wie Abteilungen, Unternehmen oder Märkten und deren Kombination wird im Bereich der organisationsorientierten Multiagentensysteme in der Regel nicht gestellt. Beim Übergang zu Multiorganisationssystemen wird sie jedoch inhärent wichtig. In den Kapiteln 10 und 11 wurden diesbezügliche Untersuchungen angestellt. Allerdings ist fraglich, ob die Herangehensweise, Organisationseinheiten entlang gänzlich verschiedener Organisationsansätze ausgestalten und anschließend kombinieren zu wollen, wirklich lohnenswert ist. In Kapitel 10 diente diese Herangehensweise nicht zuletzt der Untermauerung der Tragfähigkeit des Organisationseinheitsmodells. Für die Zukunft sieht der Autor größeres Potential darin, Archetypen von Sonar-Organisationseinheiten zu entwickeln, die im Kontext von Organ-Multiorganisationssystemen eingesetzt werden können.

Anwendungen Es ist natürlich wichtig, die Tragfähigkeit und Rechtfertigung des Konzeptes eines Multiorganisationssystems durch die Anwendbarkeit auf beziehungsweise den Einsatz in praktischen Anwendungen zu untermauern. Hier stellt sich die Frage, ob ein rein universitäres Umfeld überhaupt einen angemessenen Rahmen bietet. Idealerweise können zukünftig Möglichkeiten erschlossen werden, Projekte in der Schnittmenge von universitärer Forschung und industriellen Anwendungen auf den Weg zu bringen.

Weitere Anknüpfpunkte Abschließend werden Anknüpfpunkte skizziert, die sich durch Arbeiten im weiteren Kontext der vorliegenden Arbeit ergeben. Ortmann [Ort10], Reese [Ree09], Wagner [Wag09] und Markwardt (in seiner laufenden Dissertation) beschäftigen sich auf petrinetzbasierter Basis mit stark prozessorientierten Betrachtungen von Agentensystemen (unter jeweiliger Hinzunahme/Betonung weiterer Konzepte wie Diensten, Workflows oder Helpern/Resources). Hier entstehen Anschlusspunkte für die Modellierung, Klassifizierung und Ablaufunterstützung von Aktivitäten in Multiorganisationssystemen.

Moldt, Hewelt und Tell [TM05, HWE09] beschäftigen sich mit einer hochgenerischen *Einheitentheorie*. Im Zentrum steht das subsumierende Konzept der *Einheit*, das eine vereinheitlichte Sicht auf beliebige konkrete Einheiten wie Objekte, Agenten, Organisationen oder Plugins bietet. In diesem Kontext stellen die Organisationseinheiten und Multiorganisationssysteme der vorliegenden Arbeit eine spezielle einheitentheoretische Perspektive dar.

Die Untersuchung von möglichen Anknüpfpunkten an die Ergebnisse der vorliegenden Arbeit und die Betrachtungen des jeweils assoziierten Potentials machen deutlich, dass die Arbeit in einen weiteren Kontext aus sowohl forschungsrelevanten als auch praxiserprobten Arbeiten eingebettet ist. Insgesamt treten hier die Anschlussfähigkeit der Arbeit einerseits und ihr Innovationspotential andererseits deutlich zutage.

A Ausführbare Referenznetze

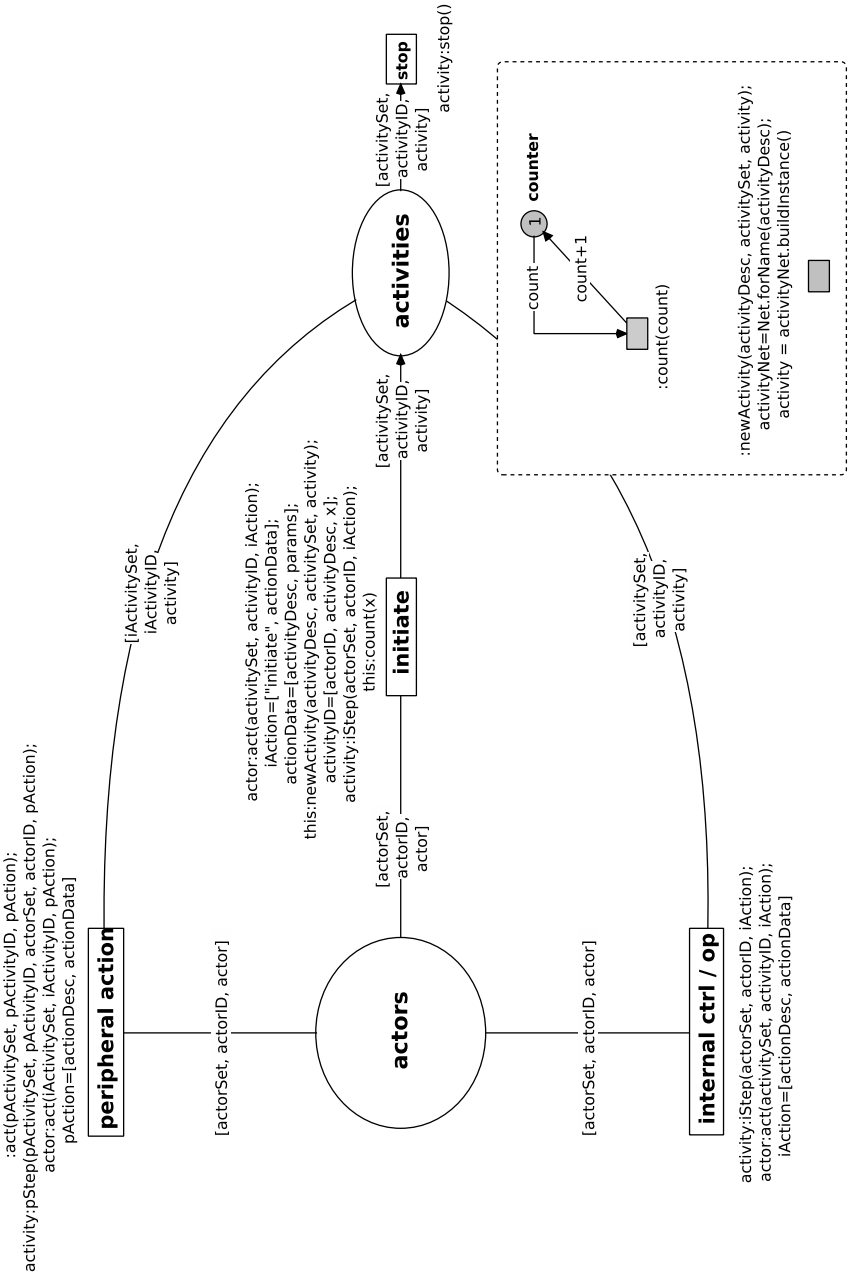

Abbildung A.1: Ausführbares Grundmodell einer Kollektivitätseinheit

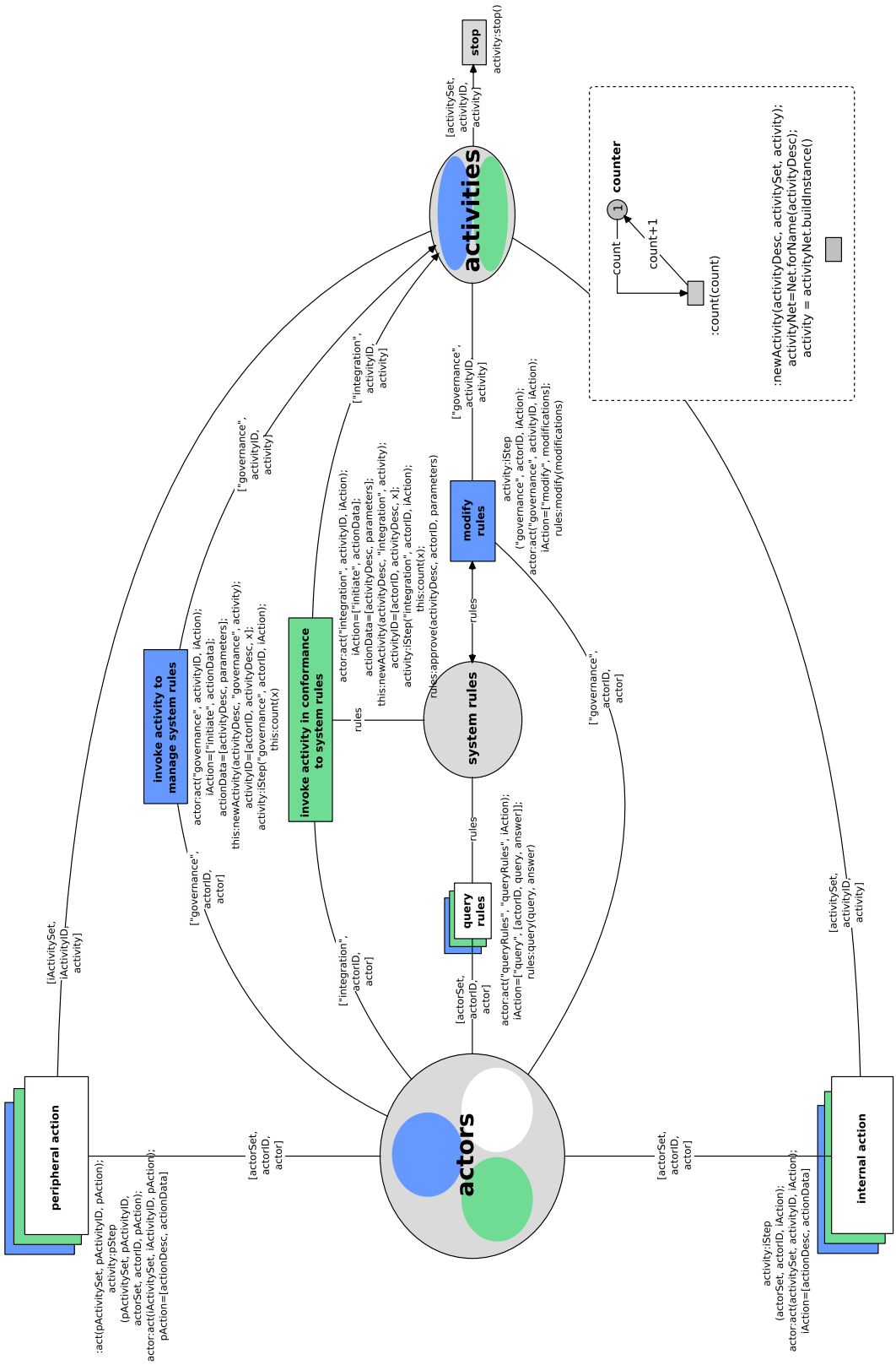

Abbildung A.2: Ausführbares Grundmodell einer Organisationseinheit (1)

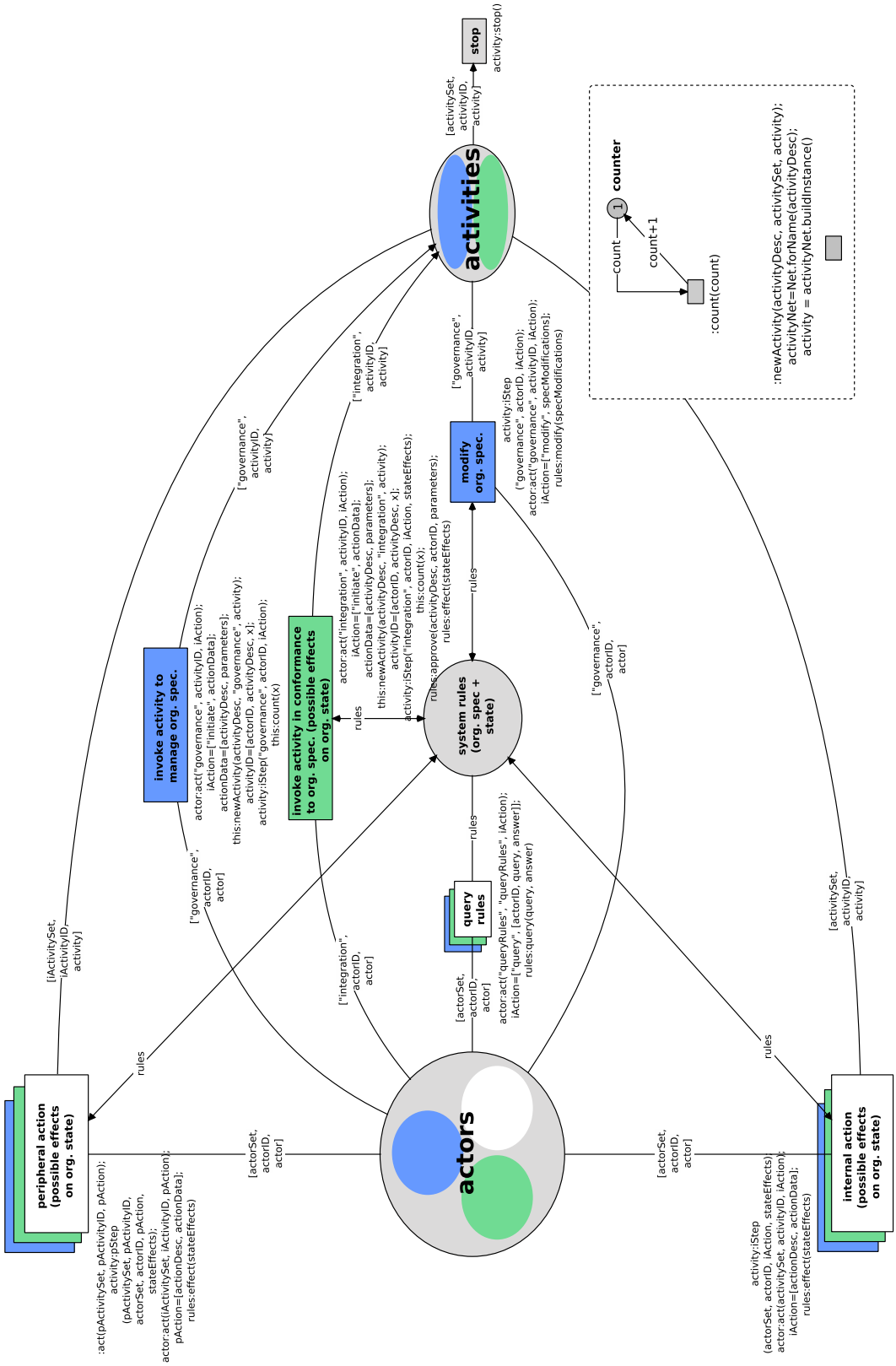

Abbildung A.3: Ausführbares Grundmodell einer Organisationseinheit (2)

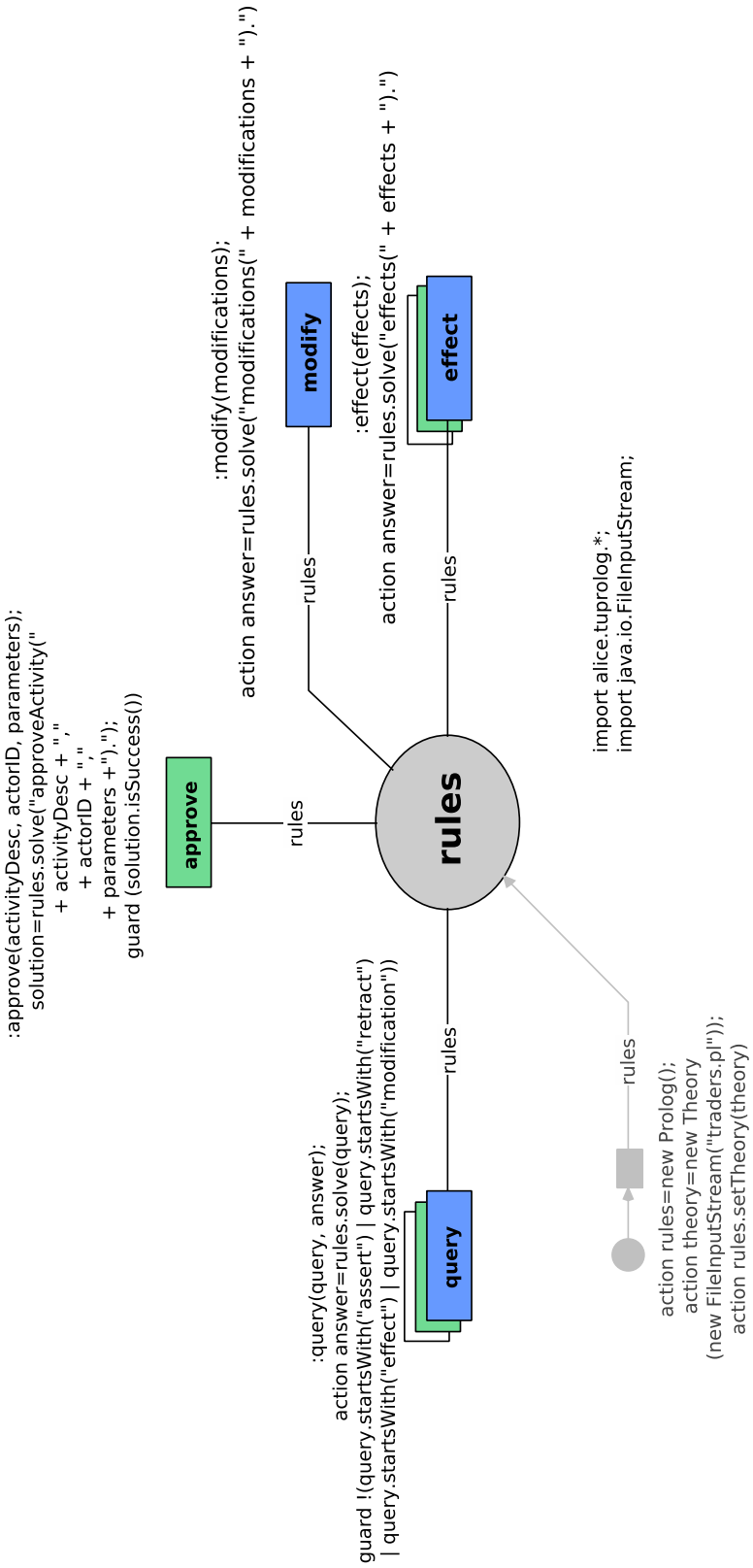

Abbildung A.4: Ausführbares Systemregelnetz mit tuProlog-Regeln

Literaturverzeichnis

[AG87] Alexander, Jeffrey und Bernhard Giesen: *From Reduction to Linkage: The Long View of the Micro-Macro Debate.* In: Alexander, Jeffrey et al. [AGMS87], Kapitel Introduction, Seiten 1–42.

[AGMS87] Alexander, Jeffrey, Bernhard Giesen, Richard Münch und Neil Smelser (Herausgeber): *The Micro-Macro Link.* University of California Press, 1987.

[AS78] Argyris, Chris und Donald Schön: *Organizational Learning: A Theory of Action Perspective.* Addison Wesley, 1978.

[Ash56] Ashby, William Ross: *An Introduction to Cybernetics.* Chapman and Hall, 1956.

[Ash60] Ashby, William Ross: *Design for a Brain.* Wiley, 1960.

[AW85] Albert, Stuart und David Whetten: *Organizational Identity.* In: Cummings, Larry und Barry Staw (Herausgeber): *Research in Organizational Behavior*, Band 14, Seiten 263–295. JAI Press, 1985.

[Bäu98] Bäumer, Dirk: *Softwarestrukturen für die rahmenwerkbasierte Konstruktion großer Anwendungssysteme.* Dissertation, Universität Hamburg, Department Informatik, 1998.

[BCG07] Bellifemine, Fabio Luigi, Giovanni Caire und Dominic Greenwood: *Developing Multi-Agent Systems with Jade.* Wiley, 2007.

[BDR93] Boyd, Brian, Gregory Dess und Abdul Rasheed: *Divergence between Archival and Perceptual Measures of the Environment: Cause and Consequences.* Academy of Management Review, 18:204–226, 1993.

[BE97] Brown, Shona und Kathleen Eisenhardt: *The Art of Continuous Change: Linking Complexity Theory and Time-Paced Evolution in Relentlessly Shifting Organizations.* Administrative Science Quarterly, 42:1–34, 1997.

[Bee79] Beer, Stafford: *The Heart of the Enterprise.* Wiley, 1979.

[Bee84] Beer, Stafford: *The Viable System Model: Its Provenance, Development, Methodology and Pathlogy.* Journal of the Operational Research Society, 35(1):7–25, 1984.

[Bee85] Beer, Stafford: *Diagnosing the System for Organizations.* Wiley, 1985.

[Ber56] Bertalanffy, Ludwig von: *General System Theory.* In: Bertalanffy, Ludwig von und Anatol Rapoport (Herausgeber): *General Systems: Yearbook of the Society for the Advancement of General Systems Theory*, Band 1, Seiten 1–10. The Society, 1956.

[BGvN46] Burks, Arthur, Herman Goldstine und John von Neumann: *Preliminary Discussion of the Logical Design of an Elecronic Computing Instrument.* U.S. Army Ordnance Department Report 1946, 1946.

[BHS07] Boissier, Olivier, Jomi Hübner und Jaime Simão Sichman: *Organization Oriented Programming: From Closed to Open Organizations.* In: O'Hare, G., A. Ricci, M. O'Grady und O. Dikenelli (Herausgeber): *Engineering Societies in the Agents World VII*, Band 4457 der Reihe *Lecture Notes in Computer Science*, Seiten 86–105. Springer, 2007.

[BHW07] Bordini, Rafael, Jomi Hübner und Michael Wooldridge: *Programming Multi-Agent Systems in AgentSpeak Using Jason.* Wiley, 2007.

[Bla70] Blau, Peter: *A Formal Theory of Differentiation in Organizations.* American Sociological Review, 35:201–218, 1970.

[Bla87] Blau, Peter: *Contrasting Theoretical Perspectives.* In: Alexander, Jeffrey et al. [AGMS87], Kapitel 2, Seiten 71–85.

[BMS06] Bernus, Peter, Kai Mertins und Günter Schmidt (Herausgeber): *Handbook on Architectures of Information Systems.* International Handbooks on Information Systems. Springer, 2006.

[BNS03] Bernus, Peter, Lazlo Nemes und Günter Schmidt (Herausgeber): *Handbook on Enterprise Architecture.* International Handbooks on Information Systems. Springer, 2003.

[Bou56] Boulding, Kenneth: *General Systems Theory: The Skeleton of Science.* Management Science, 2:197–208, 1956.

[BPG+04] Bresciani, Paolo, Anna Perini, Paolo Giorgini, Fausto Giunchiglia und John Mylopoulos: *Tropos: An Agent–Oriented Software Development Methodology.* Autonomous Agents and Multi-Agent Systems, 8(3):203–236, 2004.

[Bra07] Braubach, Lars: *Architekturen und Methoden zur Entwicklung verteilter agentenorientierter Softwaresysteme.* Dissertation, Universität Hamburg, Department Informatik, Verteilte Systeme und Informationssysteme, 2007.

[BvdT05] Boella, Guido und Leendert van der Torre: *Organizations as Socially-Constructed Agents in the Agent-Oriented Paradigm.* In: Gleizes, Marie Pierre, Andrea Omicini und Franco Zambonelli (Herausgeber): *Engineering Societies in the Agents World V*, Band 3451 der Reihe *Lecture Notes in Computer Science*, Seiten 1–13. Springer, 2005.

[Cab07] Cabac, Lawrence: *Multi-Agent System: A Guiding Metaphor for the Organization of Software Development Projects.* In: Petta, Paolo, Jörg Müller, Matthias Klusch und Michael Georgeff (Herausgeber): *Multiagent System Technologies, 5th German Conference, MATES 2007*, Band 4687 der Reihe *Lecture Notes in Computer Science*, Seiten 1–12. Springer, 2007.

[Cab10] Cabac, Lawrence: *Modeling Petri Net-Based Multi-Agent Applications.* Dissertation, Universität Hamburg, Department Informatik, Theoretische Grundlagen der Informatik, 2010.

[CDD+08] Cabac, Lawrence, Till Dörges, Michael Duvigneau, Daniel Moldt, Christine Reese und Matthias Wester-Ebbinghaus: *Agent Models for Concurrent Software Systems.* In: Bergmann, Ralph, Gabriela Lindemann, Stefan Kirn und Michal Pechoucek (Herausgeber): *Multiagent System Technologies, 6th German Conference, MATES 2008*, Band 5244 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 37–48. Springer, 2008.

[CH94] Christensen, Søren und Niels Hansen: *Coloured Petri Nets Extended with Channels for Synchronous Communication.* In: *Application and Theory of Petri Nets 1994*, Band 815 der Reihe *Lecture Notes in Computer Science*, Seiten 159–178. Springer, 1994.

[Cha62] Chandler, Alfred: *Strategy and Structure: Chapters in the History of the American Industrial Enterprise.* MIT Press, 1962.

[Cha90] Chandler, Alfred, with assistance of Takashi Hikino: *Scale and Scope: The Dynamics of Industrial Capitalism.* Harvard University Press, 1990.

[Chi72] Child, John: *Organizational Structure, Environment, and Performance: The Role of Strategic Choice.* Sociology, 6:1–22, 1972.

[CL90a] Cohen, Phillip und Hector Levesque: *Intention is Choice with Commitment.* Artificial Intelligence, 42(2–3):213–262, 1990.

[CL90b] Cohen, Phillip und Hector Levesque: *Persistence, Intention and Commitment.* In: Cohen, Phillip, J. Morgan und Martha Pollack (Herausgeber): *Intentions in Communication*, Seiten 33–69. MIT Press, Cambridge, 1990.

[CLS98] Cohen, Phillip, Hector Levesque und Ira Smith: *On team formation.* In: Holmström-Hintikka, Ghita und Raimo Tuomela (Herausgeber): *Contemporary Action Theory*, Band 2: Social Action, Seiten 87–114. Dordrecht: Kluwer, 1998.

[CLSRAE08] Campos, Jordi, Maite López-Sánchez, Juan Rodríguez-Aguilar und Marc Esteva: *Formalising Situatedness and Adaption in Electronic Institutions.* In: Hübner, Jomi Fred, Eric Matson, Olivier Boissier und Virginia Dignum (Herausgeber): *Coordination, Organizations, Institutions, and Norms in Agent Systems IV*, Band 5428 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 126–139, 2008.

[CM63] Cyert, Richard und James March: *A Behavioral Theory of the Firm.* Prentice Hall, 1963.

[CMR03] Cabac, Lawrence, Daniel Moldt und Heiko Rölke: *A Proposal for Structuring Petri net-based Agent Interaction Protocols.* In: Aalst, Wil van der und Eike Best (Herausgeber): *Application and Theory of Petri Nets 2003*, Band 2679 der Reihe *Lecture Notes in Computer Science*, Seiten 102–120. Springer, 2003.

[Col90] Coleman, James: *Foundations of Social Theory.* Harvard University Press, 1990.

[DeL05] DeLoach, Scott: *Engineering Organization-Based Multiagent Systems.* In: *Software Engineering for Multi-Agent Systems IV*, Band 3914 der Reihe *Lecture Notes in Computer Science*, Seiten 109–125. Springer, 2005.

[Dig09] Dignum, Virgina (Herausgeber): *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models.* Information Science Reference, 2009.

[DiM86] DiMaggio, Paul: *Structural Analysis of Organizational Fields: A Blockmodel Approach.* In: Staw, Barry und Larry Cummings (Herausgeber): *Research in Organizational Behaviour*, Band 8, Seiten 355–370. JAI Press, 1986.

[DK99] Dellarocas, Chrysanthos und Mark Klein: *Civil Agent Societies: Tools for Inventing Open Agent-Mediated Electronic Marketplaces.* In: Moukas, Alexandros, Carles Sierra und Fredrik Ygge (Herausgeber): *Agent Mediated Electronic Commerce II*, Band 1788 der Reihe *Lecture Notes in Computer Science*, Seiten 24–39. Springer, 1999.

[DLDD+07] De Laurentis, Dan, Charles Dickerson, Michael Dimario, Paul Gartz, Mo Jamshidi, Saeid Nahavandi, Andrew Sage, Elliot Sloane und Donald Walker: *A Case for an International Consortium on System-of-Systems Engineering.* IEEE Systems Journal, 1(1):68–73, 2007.

[DLPW95] Decker, Keith, Victor Lesser, Nagendra Prasad und Thomas Wagner: *MACRON: An Architecture for Multi-Agent Cooperative Information Gathering.* In: Finin, Tim und James Mayfield (Herausgeber): *Proceedings of the CIKM '95 Workshop on Intelligent Information Agents*, 1995.

[DMR03] Duvigneau, Michael, Daniel Moldt und Heiko Rölke: *Concurrent Architecture for a Multi-agent Platform.* In: Giunchiglia, Fausto, James Odell und Gerhard Weiss (Herausgeber): *Agent-Oriented Software Engineering III*, Band 2585 der Reihe *Lecture Notes in Computer Science*, Seiten 59–72. Springer, 2003.

[DP83] DiMaggio, Paul und Walter Powell: *The Iron Cage Revisited: Institutional Isomorphism and Collective Rationality in Organizational Fields.* American Sociological Review, 48:147–160, 1983.

[Duv09] Duvigneau, Michael: *Konzeptionelle Modellierung von Plugin-Systemen mit Petrinetzen.* Dissertation, Universität Hamburg, Department Informatik, Theoretische Grundlagen der Informatik, 2009.

[DVSD04] Dignum, Virgina, Javier Vázquez-Salceda und Frank Dignum: *OMNI: Introducing Social Structure, Norms and Ontologies into Agent Organizations.* In: Bordini, Rafael, Mehdi Dastani, Jürgen Dix und Amal El Fallah-Seghrouchni (Herausgeber): *Programming Multi-Agent Systems, Second International Workshop ProMAS 2004*, Band 3346 der Reihe *Lecture Notes in Computer Science*, Seiten 181–198. Springer, 2004.

[DWS01] DeLoach, Scott, Mark Wood und Clint Sparkman: *Multiagent Systems Engineering.* The International Journal of Software Engineering and Knowledge Engineering, 11(3):231–258, 2001.

[EdlCS02] Esteva, Marc, David de la Cruz und Carles Sierra: *ISLANDER: An Electronic Institutions Editor.* In: *The First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002, Proceedings*, Seiten 1045–1052. ACM, 2002.

[EHH+08] Engels, Gregor, Andreas Hess, Bernhard Humm, Oliver Juwig, Marc Lohmann, Jan-Peter Richter, Markus Voss und Johannes Willkomm: *Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten.* dpunkt.verlag, 2008.

[Eme62] Emerson, Richard: *Power-Dependence Relations.* American Sociological Review, 27:31–40, 1962.

[EPS02] Esteva, Marc, Julian Padget und Carles Sierra: *Formalizing a Language for Institutions and Norms.* In: Meyer, John-Jules C. und Milind Tambe (Herausgeber): *Intelligent Agents VIII*, Band 2333 der Reihe *Lecture Notes in Computer Science*, Seiten 348–366. Springer, 2002.

[ERARA04] Esteva, Marc, Juan Rodríguez-Aguilar, Bruno Rosell und Josef Arcos: *AMELI: An Agent-based Middleware for Electronic Institutions.* In: *Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, Seiten 236–243. IEEE Computer Society, 2004.

[ESP93] ESPRIT Consortium AMICE (Herausgeber): *CIMOSA: Open System Architecture for CIM.* Research Reports Esprit. Springer, 2. Auflage, 1993.

[Etz64] Etzioni, Amitai: *Modern Organizations.* Prentice Hall, 1964.

[FA91] Friedland, Roger und Robert Alford: *Bringing Back Society In: Symbols, Practices, and Institutional Contradictions.* In: Powell, Walter und Paul DiMaggio (Herausgeber): *The New Institutionalism in Organizational Analysis*, Seiten 232–263. University of Chicago Press, 1991.

[FGM03] Ferber, Jacques, Olivier Gutknecht und Fabien Michel: *From Agents to Organizations: An Organizational View of Multi-Agent Systems.* In: *Agent-Oriented Software Engineering IV*, Band 2935 der Reihe *Lecture Notes in Computer Science.* Springer, 2003.

[Fox81] Fox, Mark: *An Organizational View of Distributed Systems.* IEEE Transactions on Systems, Man and Cybernetics, 11(1):70–80, 1981.

[FSS03] Fischer, Klaus, Michael Schillo und Jörg Siekmann: *Holonic Multiagent Systems: A Foundation for the Organization of Multiagent Systems.* In: *Holonic and Multi-Agent Systems for Manufacturing, First International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2003*, Band 2744 der Reihe *Lecture Notes in Computer Science*, Seiten 71–80. Springer, 2003.

[Gal73] Galbraith, Jay: *Designing Complex Organizations.* Addison-Wesley, 1973.

[Gal74] Galbraith, Jay: *Organization Design: An Information Processing View.* Interfaces, 4(3):28–36, 1974.

[Gal02] Galbraith, John Kenneth: *The Technostructure.* In: Lazonick, William (Herausgeber): *American Corporate Economy: Critical Perspectives on Business and Management*, Band 2, Kapitel 19, Seiten 76–85. Routledge, 2002.

[Gas01] Gasser, Les: *Perspectives on Organizations in Multi-agent Systems.* In: Luck, Michael, Vladimír Marík, Olga Stepánková und Robert Trappl (Herausgeber): *Multi-Agent Systems and Applications, 9th ECCAI Advanced Course ACAI 2001 and Agent Link's 3rd European Agent Systems Summer School, EASSS 2001, Selected Tutorial Papers*, Band 2086 der Reihe *Lecture Notes in Computer Science*, Seiten 1–16. Springer, 2001.

[Gen97] Genesereth, Michael: *An Agent-Based Framework For Interoperability.* In: Bradshaw, Jeffrey (Herausgeber): *Software Agents*, Seiten 317–345. MIT Press, 1997.

[GF00] Gutknecht, Olivier und Jacques Ferber: *The MADKIT Agent Platform Architecture.* In: Wagner, Tom und Omer Rana (Herausgeber): *Infrastructures for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, Band 1887 der Reihe *Lecture Notes in Computer Science*, Seiten 48–55. Springer, 2000.

[Gid84] Giddens, Anthony: *The Constitution of Society: Outline of the Theory of Structuration.* University of California Press, 1984.

[GK96] Grosz, Barbara und Sarit Kraus: *Collaborative plans for complex group action.* Artificial Intelligence, 86(2):269–357, 1996.

[GPP+99] Georgeff, Michael, Barney Pell, Martha Pollack, Milind Tambe und Michael Wooldridge: *The Belief-Desire-Intention-Model of Agency.* In: Müller, Jörg, Singh Munindar und Anand Rao (Herausgeber): *Intelligent Agents V*, Band 1555 der Reihe *Lecture Notes in Computer Science*, Seiten 1–10. Springer, 1999.

[GR83] Goltz, Ursula und Wolfgang Reisig: *The non-sequential behaviour of Petri nets.* Information and Control, 57(2–3):125–147, 1983.

[GV03] Girault, Claude und Rüdiger Valk (Herausgeber): *Petri Nets for Systems Engineering: A Guide to Modelling, Verification and Applications.* Springer, 2003.

[HBKR10] Hübner, Jomi Fred, Olivier Boissier, Rosine Kitio und Alessandro Ricci: *Instrumenting multi-agent organisations with organisational artifacts and agents: Giving the organisational power back to the agents.* Autonomous Agents and Multi-Agent Systems, 20(3):369–400, 2010.

[HC95] Hannan, Michael und Glenn Carroll: *An Introduction to Organizational Ecology.* In: Hannan, Michael und Glenn Carroll (Herausgeber): *Organizations in Industry: Strategy, Structure and Selection*, Seiten 17–31. Oxford University Press, 1995.

[Her02] Herring, Charles: *Viable Software: The Intelligent Control Paradigm for Adaptable and Adaptive Architectures.* Dissertation, University of Queensland, Department of Information Technology and Electrical Engineering, 2002.

[HF77] Hannan, Michael und John Freeman: *The Population Ecology of Organizations.* American Journal of Sociology, 82(5):929–964, 1977.

[HHH09] Holtschke, Bernhard, Hauke Heier und Thomas Hummel: *Quo Vadis CIO?* Xpert.press. Springer, 2009.

[HHVE07] Hess, Andreas, Bernhard Humm, Markus Voss und Gregor Engels: *Structuring Software Cities - A Multidimensional Approach.* In: *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, Seiten 122–129, 2007.

[HL05] Horling, Bryan und Victor Lesser: *A Survey of Multi-Agent Organizational Paradigms.* The Knowledge Engineering Review, 19(4):281–316, 2005.

[HMMF08] Hahn, Christian, Cristian Madrigal-Mora und Klaus Fischer: *A Platform-Independent Metamodel for Multiagent Systems.* Autonomous Agents and Multi-Agent Systems, 18(2):239–266, 2008.

[HMS+03] Horling, Bryan, Roger Mailler, Jiaying Shen, Regis Vincent und Victor Lesser: *Using Autonomy, Organizational Design and Negotiation in a Distributed Sensor Network.* In: Lesser, Victor, Charles Ortiz und Milind Tambe (Herausgeber): *Distributed Sensor Networks: A Multiagent Perspective*, Band 9 der Reihe *Multiagent Systems, Artificial Societies, and Simulated Organizations*, Seiten 139–183. Kluwer Academic Publishers, 2003.

[HR85] Hulin, Charles und M Roznowski: *Organizational Technologies: Effects on Organizations' Characteristics and Individuals' Responses.* Research in Organizational Behavior, 7:39–85, 1985.

[HSB02] Hübner, Jomi Fred, Jaime Simão Sichman und Olivier Boissier: *A Model for the Structural, Functional and Deontic Specification of Organizations in Multiagent Systems.* In: Bittencourt, Guilherme und Geber Ramalho (Herausgeber): *Advances in Artificial Intelligence, 16th Brazilian Symposium*

on Artificial Intelligence, SBIA 2002, Band 2507 der Reihe *Lecture Notes in Computer Science*, Seiten 439–448. Springer, 2002.

[HSB04] Hübner, Jomi Fred, Jaime Simão Sichman und Olivier Boissier: *Using the $\mathcal{M}OISE^+$ for a Cooperative Framework of MAS Reorganisation.* In: Bazzan, Ana und Sofiane Labidi (Herausgeber): *Advances in Artificial Intelligence – SBIA 2004*, Band 3171 der Reihe *Lecture Notes in Computer Science*, Seiten 481–517. Springer, 2004.

[HSB06] Hübner, Jomi Fred, Jaime Simão Sichman und Olivier Boissier: *$\mathcal{S}$-$\mathcal{M}OISE^+$: A Middleware for Developing Organised Multi-Agent Systems.* In: Boissier, Olivier, Julian Padget, Virgina Dignum, Eric Lindemann, Gabriela Matson, Sascha Ossowski, Jaime Simão Sichman und Javier Vásquez-Salceda (Herausgeber): *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems, AAMAS 2005 Workshops on Agents, Norms and Institutions for Regulated Multi-Agent Systems, ANIREM 2005 and Organizations in Multi-Agent Systems, OOOP2005*, Band 3913 der Reihe *Lecture Notes in Computer Science*, Seiten 64–78. Springer, 2006.

[HVH06] Humm, Bernhard, Markus Voss und Andreas Hess: *Regeln für serviceorientierte Architekturen hoher Qualität.* Informatik Spektrum, 29(6):395–411, 2006.

[HWE09] Hewelt, Marcin und Matthias Wester-Ebbinghaus: *United – A Petri Net Based Framework for Modeling Complex and Adaptive Systems.* In: Moldt, Daniel (Herausgeber): *Petri Nets and Software Engineering, International Workshop, PNSE'09. Proceedings*, Technical Reports Université Paris 13, Seiten 207–226. Université Paris 13, 2009.

[IFI03] IFIP–IFAC Taskforce on Architectures for Enterprise Integration: *GERAM: The Generalised Enterprise Reference Architecture and Methodology.* In: Bernus, Peter, Lazlo Nemes und Günter Schmidt (Herausgeber): *Handbook on Enterprise Architecture*, International Handbooks on Information Systems, Seiten 21–63. Springer, 2003.

[IST05] IST Advisory Group: *Ambient Intelligence: From Vision to Reality.* In: Riva, G., F. Vatalaro und F. Davide (Herausgeber): *Ambient Intelligence - The Evolution of Technology, Communication and Cognition*, Kapitel 4, Seiten 45–68. IOS Press, 2005.

[Jah07] Jahnke, Isa: *Dynamik sozialer Rollen beim Wissensmanagement – Sozio-technische Anforderungen an Communities und Organisationen.* Deutscher Universitäts-Verlag, 2007.

[Jen95] Jennings, Nicholas: *Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems Using Joint Intentions.* Artificial Intelligence, 75(2):195–240, 1995.

[Jen96] Jensen, Kurt: *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use, Vol. 1: Basic Concepts.* EATCS Monographs on Theoretical Computer Science. Springer, 2. Auflage, 1996.

[Jen00] Jennings, Nicholas: *On agent-based software engineering.* Artificial Intelligence, 177(2):277–296, 2000.

[Joh96] Johannessen, Jon-Arild: *Systemics Applied to the Study of Organizational Fields.* Kybernetes, 25(1):33–50, 1996.

[Joh08] Johnson, Doyle: *Contemporary Sociological Theory: An Integrated Multi-Level Approach.* Springer, 2008.

[JV80] Jantzen, Matthias und Rüdiger Valk: *Formal Properties of Place/Transition Nets.* In: Brauer, Wilfried (Herausgeber): *Net Theory and Applications, Proceedings of the Advanced Course on General Net Theory of Processes and Systems, Hamburg, 1979*, Band 84 der Reihe *Lecture Notes in Computer Science*, Seiten 165–212. Springer, 1980.

[JV87] Jessen, Eike und Rüdiger Valk: *Rechensysteme - Grundlagen der Modellbildung.* Springer, 1987.

[KB09] Köhler-Bussmeier, Michael: *SONAR: Eine sozialtheoretisch fundierte Multiagentensystemarchitektur.* In: Lüde, Rolf von et al. [vLMV09], Kapitel 8–12, Seiten 281–463.

[KBMWE09] Köhler-Bussmeier, Michael, Daniel Moldt und Matthias Wester-Ebbinghaus: *A Formal Model for Organisational Structures Behind Process-Aware Information Systems.* In: Aalst, Wil van der und Kurt Jensen (Herausgeber): *Transactions on Petri Nets and Other Models of Concurrency II: Special Issue on Concurrency in Process-Aware Information Systems*, Band 5460 der Reihe *Lecture Notes in Computer Science*, Seiten 98–115. Springer, 2009.

[KBWE09a] Köhler-Bussmeier, Michael und Matthias Wester-Ebbinghaus: *Automatic Generation of Distributed Team Formation Algorithms from Organizational Models.* In: Hübner, Jomi Fred, Eric Matson, Olivier Boissier und Virginia Dignum (Herausgeber): *Coordination, Organizations, Institutions, and Norms in Agent Systems IV*, Band 5428 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 64–79, 2009.

[KBWE09b] Köhler-Bussmeier, Michael und Matthias Wester-Ebbinghaus: *A Petri Net based Prototype for MAS Organisation Middleware.* In: Duvigneau, Michael und Daniel Moldt (Herausgeber): *Proceedings of the Fifth International Workshop on Modeling of Objects, Components and Agents, MOCA'09, Hamburg*, Nummer FBI-HH-B-290/09 in *Bericht*, Seiten 29–44. Universität Hamburg, Department Informatik, 2009.

[KBWE09c] Köhler-Bussmeier, Michael und Matthias Wester-Ebbinghaus: *SONAR*: A Multi-Agent Infrastructure for Active Application Architectures and Inter-Organisational Information Systems.* In: Braubach, Lars, Wiebke van der Hoek, Paolo Petta und Alexander Pokahr (Herausgeber): *Multiagent System Technologies, 7th German Conference, MATES 2009*, Band 5774 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 248–257. Springer, 2009.

[KBWEM10] Köhler-Bussmeier, Michael, Matthias Wester-Ebbinghaus und Daniel Moldt: *Generating Executable Multi-Agent System Prototypes from SONAR Specifications.* In: Fornara, Nicoletta und George Vouros (Herausgeber): *11th Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems, COIN@Mallow 2010*, Seiten 82–97, 2010.

[KC03] Kephart, Jeffrey und David Chess: *The Vision of Autonomic Computing.* IEEE Computer, 36(1):41–50, 2003.

[Kea05] Keating, Charles: *Research Foundations for System of Systems Engineering.* In: *IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2005)*, Seiten 2720–2725, 2005.

[KMR01] Köhler, Michael, Daniel Moldt und Heiko Rölke: *Modelling the Structure and Behaviour of Petri Net Agents.* In: Colom, José-Manuel und Maciej Koutny (Herausgeber): *Application and Theory of Petri Nets 2001*, Band 2075 der Reihe *Lecture Notes in Computer Science*, Seiten 224–241. Springer, 2001.

[KNMB06] Kollingbaum, Martin, Timothy Norman, Nikolay Mehandjiev und Kenneth Brown: *Engineering Organisation-Oriented Software.* In: *Second International Workshop on Interdisciplinary Software Engineering Research (WISER'06)*, Seiten 23–28, 2006.

[KNS92] Keller, Gerhard, Markus Nüttgens und August-Wilhelm Scheer: *Semantische Prozessmodellierung auf der Grundlage „Ereignisgesteuerter Prozessketten (EPK)".* In: Scheer, August-Wilhelm (Herausgeber): *Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Universität des Saarlandes*, Heft 89, 1992.

[Koe67] Koestler, Arthur: *The Ghost in the Machine.* Henry Regnery Co., 1967.

[Köh04] Köhler, Michael: *Objektnetze: Definition und Eigenschaften.* Logos Verlag, 2004.

[Köh06] Köhler, Michael: *A Formal Model of Multi-Agent Organisations.* Fundamenta Informaticae, 79(3–4):415–430, 2006.

[KRU+03] Keating, Charles, Ralph Rogers, Resit Unal, David Dryer, Andrés Sousa-Poza, Robert Safford, William Peterson und Ghaith Rabadi: *System of Systems Engineering.* Engineering Management Journal, 15(3):36–45, 2003.

[Kuh62] Kuhn, Thomas: *The Structure of Scientific Revolutions.* University of Chicago Press, 1962.

[Kum02] Kummer, Olaf: *Referenznetze.* Logos Verlag, 2002.

[KWD09] Kummer, Olaf, Frank Wienberg und Michael Duvigneau: *Renew – The Reference Net Workshop.* Available at: http://www.renew.de/, 2009. Release 2.2.

[LCN90] Levesque, Hector, Phillip Cohen und José Nunes: *On acting together.* In: *Proceedings of the 8th National Conference on Artificial Intelligence, AAAI 1990*, Seiten 94–99. AAAI Press / The MIT Press, 1990.

[LH09] Langer, Roman und Otto Hüther: *Das Governance-Konzept.* In: Lüde, Rolf von et al. [vLMV09], Kapitel 13, Seiten 469–485.

[Lil08] Lilienthal, Carola: *Komplexität von Softwarearchitekturen.* Dissertation, Universität Hamburg, Department Informatik, Softwaretechnik, 2008.

[LMP83] Laumann, Edward, Peter Marsden und David Prensky: *The Boundary Specification Problem in Network Analysis.* In: Burt, Ronald und Michael Minor (Herausgeber): *Applied Network Analysis: A Methodological Introduction*, Seiten 18–34. Sage, 1983.

[LMW05] Lankes, Josef, Florian Matthes und Andre Wittenburg: *Softwarekartographie: Systematische Darstellung von Anwendungslandschaften.* In: Ferstl, Otto, Elmar Sinz, Sven Eckert und Tilman Isselhorst (Herausgeber): *Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety.* Physica-Verlag, 2005.

[Luh84] Luhmann, Niklas: *Soziale Systeme.* Suhrkamp, 1984.

[Lyg96] Lygeros, John: *Hierachical, Hybrid Control of Large Scale Systems.* University of California, PATH Research Report UCB-ITS-PRR-96-23, 1996.

[Mai99] Maier, Mark: *Architecturing Principles for Systems-of-Systems.* Systems Engineering, 1(4):267–284, 1999.

[Mar94] March, James: *A Primer on Decision Making: How Decisions Happen.* Free Press, 1994.

[Mat01] Mattern, Friedemann: *Pervasive/Ubiquitous Computing.* Informatik-Spektrum, 24(3):145–147, 2001.

[Mat08] Matthes, Florian: *Softwarekartographie.* Informatik-Spektrum, 31(6):527–536, 2008.

[MD93] Montgomery, Thomas und Edmund Durfee: *Search Reduction in Hierarchical Distributed Problem Solving.* Group Decision and Negotiation, 2(3):301–317, 1993.

[Mil93] Milner, Robin: *Elements of Interaction.* Communications of the ACM, 36(1):78–89, 1993.

[Min83] Mintzberg, Henry: *Structure in Fives: Designing Effective Organizations.* Prentice-Hall, 1983.

[Min87] Mintzberg, Henry: *The Strategy Concept I: Five Ps for Strategy.* In: Carroll, Glenn und David Vogel (Herausgeber): *Organizational Approaches to Strategy*, Seiten 7–20. Ballinger, 1987.

[ML04] Mailler, Roger und Victor Lesser: *Solving Distributed Constraint Optimization Problems Using Cooperative Mediation.* In: *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, Seiten 438–445. IEEE Computer Society, 2004.

[MMTZ06] Mamei, Marco, Ronaldo Menezes, Robert Tolksdorf und Franco Zambonelli: *Case Studies for Self-Organization in Computer Science.* Journal of Systems Architecture, 52(8–9):443–460, 2006.

[Mol96] Moldt, Daniel: *Höhere Petrinetze als Grundlage für Systemspezifikationen.* Dissertation, Universität Hamburg, Department Informatik, Theoretische Grundlagen der Informatik, 1996.

[Mol06] Moldt, Daniel: Paose*: A Way to Develop Distributed Software Systems Based on Petri Nets and Agents.* In: Barjis, Joseph, Ulrich Ultes-Nitsche und Juan Carlos Augusto (Herausgeber): *Proceedings of the 4th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, MSVVEIS 2006*, Seiten 1–2. INSTICC Press, 2006.

[MRS02] Mathieu, Phillipe, Jean-Christophe Routier und Yann Secq: *Dynamic Organization of Multi-Agent Systems.* In: *The First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002*, Seiten 451–452. ACM, 2002.

[MSB99] Morabito, Joseph, Ira Sack und Anilkumar Bhate: *Organization Modelling: Innovative Architectures for the 21st Century.* Prentice Hall, 1999.

[MT95] Moses, Yoram und Moshe Tennenholtz: *Artificial Social Systems.* Computers and AI, 14(6):533–562, 1995.

[MY04] Mao, XinJun und Eric Yu: *Organizational and Social Concepts in Agent Oriented Software Engineering.* In: Odell, James, Paolo Giorgini und Jörg Müller (Herausgeber): *Agent-Oriented Software Engineering V*, Band 3382 der Reihe *Lecture Notes in Computer Science*, Seiten 1–15. Springer, 2004.

[Nor06] Northrop, Linda: *Ultra-Large-Scale Systems: The Software Challenge of the Future.* Software Engineering Institute, Carnegie Mellon, 2006.

[Obe87] Oberquelle, Horst: *Human-Machine Interaction and Role/Function/Action-Nets.* In: Brauer, Wilfried, Wolfgang Reisig und Grzegorz Rozenberg (Herausgeber): *Petri Nets: Applications and Relationships to Other Models of Concurrency*, Band 255 der Reihe *Lecture Notes in Computer Science*, Seiten 171–190. Springer, 1987.

[Ort10] Ortmann, Jan: *Höhere Petrinetze als Modellierungstechnik für dienstbasierte Geschäftsprozesse.* Dissertation, Universität Hamburg, Department Informatik, Theoretische Grundlagen der Informatik, 2010.

[ORV08] Omicini, Andrea, Alessandro Ricci und Mirko Viroli: *Artifacts in the A&A meta-model for multi-agent systems.* Autonomous Agents and Multi-Agent Systems, 17(3):432–456, 2008.

[Par60] Parsons, Talcott: *Structure and Process in Modern Societies.* Free Press, 1960.

[Pet62] Petri, Carl Adam: *Kommunikation mit Automaten.* Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

[Pfe82] Pfeffer, Jeffrey: *Organizations and Organization Theory.* In: Pfeffer, Jeffrey (Herausgeber): *Organizations and Organization Theory*, Seiten 1–23. Pitman, 1982.

[PM79] Pondy, Louis und Ian Mitroff: *Beyond Open System Models of Organization.* In: Staw, Barry (Herausgeber): *Research in Organizational Behaviour*, Band 1, Seiten 3–39. JAI Press, 1979.

[Pok07] Pokahr, Alexander: *Programmiersprachen und Werkzeuge zur Entwicklung verteilter agentenorientierter Softwaresysteme.* Dissertation, Universität Hamburg, Department Informatik, Verteilte Systeme und Informationssysteme, 2007.

[Pow90] Powell, Walter: *Neither Market nor Hierarchy: Network Forms of Organizations.* In: Staw, Barry und Larry Cummings (Herausgeber): *Research in Organizational Behavior*, Band 12, Seiten 295–336. JAI Press, 1990.

[PS78] Pfeffer, Jeffrey und Gerald Salancik: *The External Control of Organizations.* Harper and Row, 1978.

[PT03] Pynadath, David und Milind Tambe: *An Automated Teamwork Infrastructure for Heterogeneous Software Agents and Humans.* Autonomous Agents and Multi-Agent Systems, 7(1–2):71–100, 2003.

[Ree09] Reese, Christine: *Prozess-Infrastruktur für Agentenanwendungen.* Dissertation, Universität Hamburg, Department Informatik, Theoretische Grundlagen der Informatik, 2009.

[Ric05] Richter, Jan-Peter: *Wann liefert eine Serviceorientierte Architektur echten Nutzen?* In: Liggesmeyer, Peter, Klaus Pohl und Michael Goedicke (Herausgeber): *Software Engineering 2005, Fachtagung des GI-Fachbereichs Softwaretechnik*, Band 64 der Reihe *Lecture Notes in Informatics*, Seiten 231–242. Gesellschaft für Informatik E.V., 2005.

[RM00] Rechtin, Eberhardt und Mark Maier: *The Art of Systems Architecting.* CLC Press, Inc., 2. Auflage, 2000.

[RM05] Ring, Jack und Azad Madni: *Key Challenges and Opportunities in 'System of Systems' Engineering.* In: *IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2005)*, Seiten 973–978, 2005.

[Röl04] Rölke, Heiko: *Modellierung von Agenten und Multiagentensystemen.* Logos Verlag, 2004.

[Rol08] Rolf, Arno: *Mikropolis 2010 – Menschen, Computer, Internet in der globalen Gesellschaft.* Metropolis, 2008.

[SBP+09] Sudeikat, Jan, Lars Braubach, Alexander Pokahr, Wolfgang Renz und Winfried Lamersdorf: *Systematically Engineering Self-Organizing Systems: The SodekoVS Approach.* Electronic Communication of the European Association of Software Science and Technology, 17, 2009.

[SCBG98] Smith, Ira, Phillip Cohen, Jeffrey Bradshaw und Mark Greaves: *Designing Conversation Policies Using Joint Intention Theory.* In: Demazeau, Yves (Herausgeber): *Third International Conference on Multi Agent Systems, ICMAS-1998*, Seiten 269–276. IEEE Computer Society, 1998.

[SCCD94] Sichman, Jaime Simão, Rosaria Conte, Cristiano Castelfranchi und Yves Demazeau: *A Social Reasoning Mechanism Based on Dependence Networks.* In: Cohn, Anthony (Herausgeber): *Eleventh European Conference on Artificial Intelligence, ECAI 1994*, Seiten 188–192. Wiley, 1994.

[Sch02] Scheer, August-Wilhelm: *ARIS - Vom Geschäftsprozess zum Anwendungssystem.* Springer, 2002.

[Sch07] Schleinzer, Benjamin: *Flexible und hierarchische Multiagentensysteme – Modellierung und prototypische Erweiterung von Mulan und Capa.* Diplomarbeit, Universität Hamburg, Department Informatik, Theoretische Grundlagen der Informatik, 2007.

[Sco03] Scott, W. Richard: *Organizations: Rational, Natural and Open Systems.* Prentice Hall, 5. Auflage, 2003.

[Sco08] Scott, Richard: *Institutions and Organizations: Ideas and Interests.* Sage, 3. Auflage, 2008.

[SFF+02] Schillo, Michael, Bettina Fley, Michael Florian, Frank Hillebrandt und Daniela Hinck: *Self-Organization in Multiagent Systems: From Agent Interaction to Agent Organization.* In: Lindemann, Gabriela, Catholijn Jonker und Ingo Timm (Herausgeber): *Modelling Artificial Social Societies and Hybrid Organizations (MASHO'02)*, Seiten 37–47, 2002.

[SG90] Sproull, Lee und Paul Goodman: *Technology and Organizations: Integration and Opportunities.* In: Sproull, Lee und Paul Goodman (Herausgeber): *Technology and Organizations*, Seiten 254–265. Jossey-Bass, 1990.

[SGL03] Sims, Mark, Claudia Goldman und Victor Lesser: *Self-Organization Through Bottom-up Coalition Formation.* In: *The Second International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2003*, Seiten 867–874. ACM, 2003.

[Sim62] Simon, Herbert: *The Architecture of Complexity.* Proceedings of the American Philosophical Society, 106(6):467–482, 1962.

[Sim64] Simon, Herbert: *On the Concept of Organizational Goal.* Administrative Science Quarterly, 9(1):1–22, 1964.

[SL97] Sandholm, Tuomas und Victor Lesser: *Coalitions Among Computationally Bounded Agents.* Artificial Intelligence, 94(1):99–137, 1997.

[SLA04] Sairamesh, Jakka, Alison Lee und Loretta Anania: *Information Cities (Introduction to Special Issue)*. Communications of the ACM (Special Issue on Information Cities), 47(2):28–31, 2004.

[Smi80] Smith, Reid: *The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver*. IEEE Transactions on Computers, 29(12):1104–1113, 1980.

[SNSS04] Stockheim, Tim, Jens Nimis, Thorsten Scholz und Marcel Stehli: *How to Build a Multi-Multi-Agent System: The Agent.Enterprise Approach*. In: *Proceedings of the 6th International Conference on Enterprise Information Systems, ICEIS 2004*, Band 4: Software Agents and Internet Computing, Seiten 364–371, 2004.

[SPKK08] Sousa-Poza, Andrés, Samuel Kovacic und Charles Keating: *System of Systems Engineering: An Emerging Multidiscipline*. International Journal of System of Systems Engineering, 1(1–2):1–17, 2008.

[SPS+04] Scerri, Paul, David Pynadath, Nathan Schurr, Alessandro Farinelli, Sudeep Gandhe und Milind Tambe: *Team Oriented Programming and Proxy Agents: The Next Generation*. In: Dastani, Mehdi, Jürgen Dix und Amal El Fallah-Seghrouchni (Herausgeber): *Programming Multi-Agent Systems, First International Workshop, ProMAS 2003*, Band 3067 der Reihe *Lecture Notes in Computer Science*, Seiten 131–148. Springer, 2004.

[SRMC00] Scott, Richard, Martin Ruef, Peter Mendell und Carol Caronna: *Institutional Change and Healthcare Organization: From Professional Dominance to Managed Care*. University of Chicago Press, 2000.

[Suc95] Suchmann, Mark: *Managing Legitimacy: Strategic and Institutional Approaches*. Academy of Management Review, 20:571–610, 1995.

[Swa76] Swanson, Guy: *The Tasks of Sociology*. Science, 192(4240):665–667, 1976.

[Swi74] Swinth, Robert: *Organizational Systems for Management: Designing, Planning and Implementation*. Grid, 1974.

[SZ08] Schirmer, Ingrid und Karsten Zimmermann: *Visualisierung von Projektportfolios zur Unterstützung des Architekturmanagements – der VIADUCT-PPM Ansatz*. In: Hegering, Heinz-Gerd, Axel Lehmann, Hans Jürgen Ohlbach und Christian Scheideler (Herausgeber): *INFORMATIK 2008, Beherrschbare Systeme – dank Informatik, Band 2*, Lecture Notes in Informatics, Seiten 566–572. Gesellschaft für Informatik E.V., 2008.

[Tam97] Tambe, Milind: *Towards Flexible Teamwork*. Journal of Artificial Intelligence Research, 7:83–124, 1997.

[Ten00] Tennenhouse, David: *Proactive Computing*. Communications of the ACM, 43(5):43–50, 2000.

[Tho03] Thompson, James: *Organizations in Action: Social Science Bases of Administrative Theory.* Classics in Organization and Management Series. Transaction Publishers, 2003.

[TM05] Tell, Volker und Daniel Moldt: *Ein Petrinetzsystem zur Modellierung selbstmodifizierender Petrinetze.* In: Schmidt, Karsten und Christian Stahl (Herausgeber): *Proceedings of the 12th Workshop on Algorithms and Tools for Petri Nets (AWPN 05)*, Seiten 36–41. Humboldt Universität zu Berlin, Fachbereich Informatik, 2005.

[Tri81] Trist, Eric: *The Evolution of Sociotechnical Systems as a Conceptual Framework and as an Action Research Program.* In: Ven, Andrew Van de und William Joyce (Herausgeber): *Perspectives on Organisation Design and Behaviour*, Seiten 19–75. Wiley, 1981.

[Tu00] Tu, Tuan: *Steuerung offener verteilter Anwendungssysteme.* Dissertation, Universität Hamburg, Department Informatik, 2000.

[Val87] Valk, Rüdiger: *Nets in Computer Organization.* In: Brauer, Wilfried, Wolfgang Reisig und Grzegorz Rozenberg (Herausgeber): *Petri Nets: Applications and Relationships to Other Models of Concurrency*, Band 255 der Reihe *Lecture Notes in Computer Science*, Seiten 218–233. Springer, 1987.

[Val98] Valk, Rüdiger: *Petri Nets as Token Objects: An Introduction to Elementary Object Nets.* In: Desel, Jörg und Manuel Silva (Herausgeber): *Application and Theory of Petri Nets 1998*, Band 1420 der Reihe *Lecture Notes in Computer Science*, Seiten 1–24. Springer, 1998.

[Val03a] Valk, Rüdiger: *Essential Features of Petri Nets.* In: Girault, Claude und Rüdiger Valk [GV03], Kapitel 2, Seiten 9–28.

[Val03b] Valk, Rüdiger: *Intuitive Models.* In: Girault, Claude und Rüdiger Valk [GV03], Kapitel 3, Seiten 29–40.

[Val04] Valk, Rüdiger: *Object Petri Nets: Using the Nets-within-Nets Paradigm.* In: Desel, Jörg, Wolfgang Reisig und Grzegorz Rozenberg (Herausgeber): *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, Band 3098 der Reihe *Lecture Notes in Computer Science*, Seiten 819–848. Springer, 2004.

[vdA97] Aalst, Wil van der: *Verification of Workflow Nets.* In: *Application and Theory of Petri Nets 1997*, Band 1248 der Reihe *Lecture Notes in Computer Science*, Seiten 407–426. Springer, 1997.

[vdA99] Aalst, Wil van der: *Interorganizational Workflows.* Systems Analysis - Modelling - Simulation, 34(3):335–367, 1999.

[VHR+07] Viroli, Mirko, Tom Holvoet, Alessandro Ricci, Kurt Schelfthout und Franco Zambonelli: *Infrastructures for the environment of multiagent systems.* Autonomous Agents and Multi-Agent Systems, 14(1):49–60, 2007.

[vLMV03] Lüde, Rolf von, Daniel Moldt und Rüdiger Valk (Herausgeber): *Sozionik: Modellierung soziologischer Theorie*, Band 2 der Reihe *Wirtschaft, Arbeit, Technik*. Lit Verlag, 2003.

[vLMV09] Lüde, Rolf von, Daniel Moldt und Rüdiger Valk (Herausgeber): *Selbstorganisation und Governance in künstlichen und sozialen Systemen*, Band 5 der Reihe *Wirtschaft, Arbeit, Technik*. Lit Verlag, 2009.

[VSDD05] Vasquez-Salceda, Javier, Virgina Dignum und Frank Dignum: *Organizing Multiagent Systems*. Autonomous Agents and Multi-Agent Systems, 11(3):307–360, 2005.

[Wag09] Wagner, Thomas: *Prototypische Realisierung einer Integration von Agenten und Workflows*. Diplomarbeit, Universität Hamburg, Department Informatik, Theoretische Grundlagen der Informatik, 2009.

[Web72] Weber, Max: *Wirtschaft und Gesellschaft: Grundriss der verstehenden Soziologie*. Mohr Siebeck, 5. Auflage, 1972.

[Wei79] Weick, Karl: *The Social Psychology of Organizing*. McGraw-Hill, 2. Auflage, 1979.

[Wei99] Weiss, Gerhard: *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[WEKBM09] Wester-Ebbinghaus, Matthias, Michael Köhler-Bussmeier und Daniel Moldt: *From Multi-Agent to Multi-Organization Systems: Utilizing Middleware Approaches*. In: Artikis, Alexander, Gauthier Picard und Laurent Vercouter (Herausgeber): *Engineering Societies in the Agents World IX*, Band 5485 der Reihe *Lecture Notes in Computer Science*, Seiten 46–65, 2009.

[WEM08a] Wester-Ebbinghaus, Matthias und Daniel Moldt: *A Janus-Faced Net Component for the Prototyping of Open Systems*. In: *15th German Workshop on Algorithms and Tools for Petri Nets, Algorithmen und Werkzeuge für Petrinetze, AWPN 2008*, Band 380 der Reihe *CEUR Workshop Proceedings*, Seiten 25–30. CEUR-WS.org, 2008.

[WEM08b] Wester-Ebbinghaus, Matthias und Daniel Moldt: *Modelling Multi-Agent Systems with Organizations in Mind*. In: Ultes-Nitsche, Ulrich, Daniel Moldt und Juan Carlos Augusto (Herausgeber): *Proceedings of the 6th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, MSVVEIS-2008*, Seiten 81–90. INSTICC PRESS, 2008.

[WEM08c] Wester-Ebbinghaus, Matthias und Daniel Moldt: *Structure in Threes: Modelling Organization-Oriented Software Architectures Built Upon Multi-Agent Systems*. In: Padgham, Lin, David Parkes, Jörg Müller und Simon Parsons (Herausgeber): *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Seiten 1307–1311. IFAAMAS, 2008.

[WEM09] Wester-Ebbinghaus, Matthias und Daniel Moldt: *Modelling an Open and Controlled System Unit as a Modular Component of Systems of Systems.* In: Köhler-Bussmeier, Michael, Daniel Moldt und Olivier Boissier (Herausgeber): *Organizational Modelling, International Workshop, OrgMod'09. Proceedings*, Technical Reports Université Paris 13, Seiten 81–100. Université Paris 13, 2009.

[WEM10] Wester-Ebbinghaus, Matthias und Daniel Moldt: *Modeling Systems of Systems as Nested Actor Systems Based on Petri Nets.* In: Engels, Gregor, Dimitris Karagiannis und Heinrich Mayr (Herausgeber): *Modellierung 2010*, Band 161 der Reihe *Lecture Notes in Informatics*, Seiten 67–82. Gesellschaft für Informatik E.V., 2010.

[WEMKB11] Wester-Ebbinghaus, Matthias, Daniel Moldt und Michael Köhler-Bussmeier: *Modeling Organizational Units as Modular Components of Systems of Systems.* In: Jensen, Kurt, Susanna Donatelli und Maciej Koutny (Herausgeber): *Transactions on Petri Nets and Other Models of Concurrency IV*, Band 6550 der Reihe *Lecture Notes in Computer Science*, Seiten 174–198. Springer, 2011.

[WEMRM07] Wester-Ebbinghaus, Matthias, Daniel Moldt, Christine Reese und Kolja Markwardt: *Towards Organization–Oriented Software Engineering.* In: Züllighoven, Heinz (Herausgeber): *Software Engineering Konferenz 2007 in Hamburg: SE'07 Proceedings*, Band 105 der Reihe *Lecture Notes in Informatics*, Seiten 205–217. GI, 2007.

[Wie65] Wiener, Norbert: *Cybernetics: or Control and Communication in the Animal and the Machine.* MIT Press, 2. Auflage, 1965.

[Win90] Winter, Sidney: *Organizing for Continous Improvement: Evolutionary Theory Meets the Quality Revolution.* In: Baum, Joel und Jitendra Singh (Herausgeber): *Evolutionary Dynamics of Organizations*, Seiten 90–108. Oxford University Press, 1990.

[WJ05] Weiss, Gerhard und Ralf Jakob: *Agentenorientierte Softwareentwicklung.* Xpert.press. Springer, 2005.

[WJK00] Wooldridge, Michael, Nicholas Jennings und David Kinny: *The Gaia Methodology for Agent-Oriented Analysis and Design.* Autonomous Agents and Multi-Agent Systems, 3(3):285–312, 2000.

[Zir07] Zirpins, Christian: *Interaktionsorientierte Komposition virtueller Dienstleistungsprozesse.* Dissertation, Universität Hamburg, Department Informatik, Verteilte Systeme und Informationssysteme, 2007.

[ZJW03] Zambonelli, Franco, Nicholas Jennings und Michael Wooldridge: *Developing Multiagent Systems: The Gaia Methodology.* ACM Transactions on Software Engineering and Methodology, 12(3):317–370, 2003.

Erklärung

Ich erkläre hiermit eidesstattlich, dass ich die Arbeit *„Von Multiagentensystemen zu Multiorganisationssystemen – Modellierung auf Basis von Petrinetzen“* selbständig durchgeführt und nur die angegebenen Hilfsmittel und Quellen verwendet habe.

Matthias Wester-Ebbinghaus

Hamburg, 18. Januar 2011