DIGITALES ARCHIV

ZBW – Leibniz-Informationszentrum Wirtschaft
*ZBW – Leibniz Information Centre for Economics*

Härer, Felix

**Thesis**

# Integrierte Entwicklung und Ausführung von Prozessen in dezentralen Organisationen : ein Vorschlag auf Basis der Blockchain

*Reference:* Härer, Felix (2019). Integrierte Entwicklung und Ausführung von Prozessen in dezentralen Organisationen : ein Vorschlag auf Basis der Blockchain. Bamberg : University of Bamberg Press.
urn:nbn:de:bvb:473-opus4-557214.

This Version is available at:
http://hdl.handle.net/11159/3978

**Kontakt/Contact**
ZBW – Leibniz-Informationszentrum Wirtschaft/Leibniz Information Centre for Economics
Düsternbrooker Weg 120
24105 Kiel (Germany)
E-Mail: *rights[at]zbw.eu*
https://www.zbw.eu/econis-archiv/

**Standard-Nutzungsbedingungen:**
Dieses Dokument darf zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden. Sie dürfen dieses Dokument nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, aufführen, vertreiben oder anderweitig nutzen. Sofern für das Dokument eine Open-Content-Lizenz verwendet wurde, so gelten abweichend von diesen Nutzungsbedingungen die in der Lizenz gewährten Nutzungsrechte.

https://zbw.eu/econis-archiv/termsofuse

**Terms of use:**
*This document may be saved and copied for your personal and scholarly purposes. You are not to copy it for public or commercial purposes, to exhibit the document in public, to perform, distribute or otherwise use the document in public. If the document is made available under a Creative Commons Licence you may exercise further usage rights as specified in the licence.*

ZBW Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Mitglied der Leibniz-Gemeinschaft

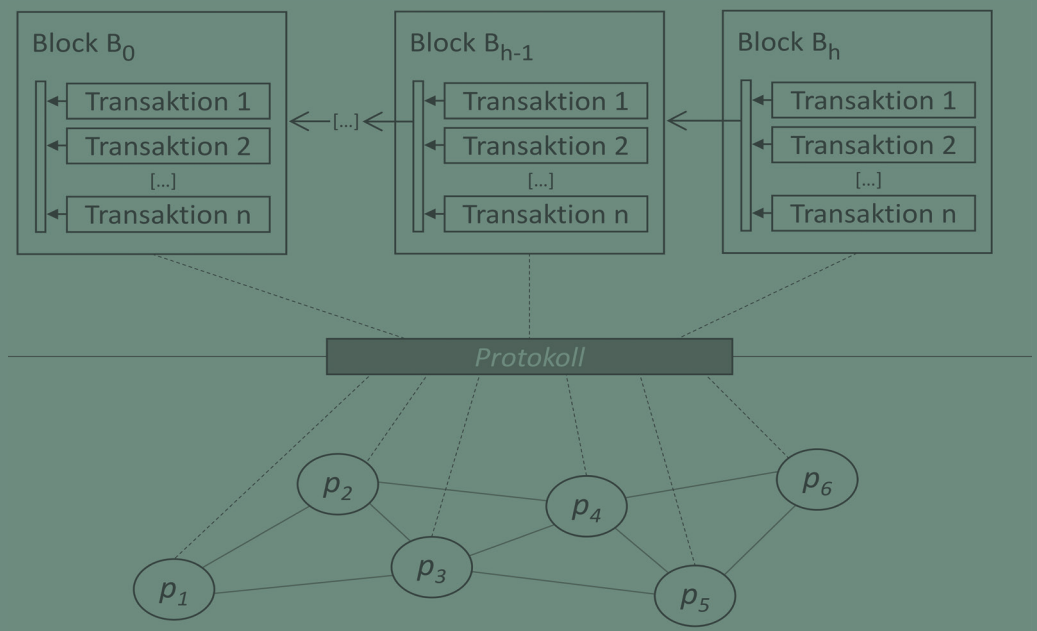
39

Schriften aus der Fakultät Wirtschaftsinformatik und
Angewandte Informatik der Otto-Friedrich-Universität Bamberg

# Integrierte Entwicklung und Ausführung von Prozessen in dezentralen Organisationen

## Ein Vorschlag auf Basis der Blockchain

Felix Härer




University of Bamberg Press

**39** Schriften aus der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg

Contributions of the Faculty Information Systems and Applied Computer Sciences of the Otto-Friedrich-University Bamberg

Schriften aus der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg

Contributions of the Faculty Information Systems and Applied Computer Sciences of the Otto-Friedrich-University Bamberg

Band 39

University of Bamberg Press
2019

# Integrierte Entwicklung und Ausführung von Prozessen in dezentralen Organisationen

Ein Vorschlag auf Basis der Blockchain

Felix Härer

University of Bamberg Press
2019

Bibliographische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Informationen sind im Internet über http://dnb.d-nb.de/ abrufbar.

Diese Arbeit hat der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg als Dissertation vorgelegen.
1. Gutachter: Prof. Dr. Elmar J. Sinz
2. Gutachter: Prof. Dr. Sven Overhage
Tag der mündlichen Prüfung: 21.08.2019

Dieses Werk ist als freie Onlineversion über das Foschungsinformationssystem (FIS; fis.uni-bamberg.de/) der Universität Bamberg erreichbar. Das Werk – ausgenommen Cover, Zitate und Abbildungen – steht unter der CC-Lizenz CC-BY.

Lizenzvertrag: Creative Commons Namensnennung 4.0
http://creativecommons.org/licenses/by/4.0

Herstellung und Druck: docupoint, Magdeburg
Umschlaggestaltung: University of Bamberg Press
Umschlagbild: © Felix Härer

© University of Bamberg Press, Bamberg 2019
http://www.uni-bamberg.de/ubp/

ISSN: 1867-7401
ISBN: 978-3-86309-682-3 (Druckausgabe)
eISBN: 978-3-86309-683-0 (Online-Ausgabe)
URN: urn:nbn:de:bvb:473-opus4-557214
DOI: http://dx.doi.org/10.20378/irbo-55721

# Inhaltsverzeichnis

**Abbildungsverzeichnis** **ix**

**Tabellenverzeichnis** **xiii**

**Abkürzungsverzeichnis** **xv**

**1 Einleitung** **1**

1.1 Motivation und Einführung . . . . . 1
1.2 Untersuchungsproblem . . . . . 3
1.3 Hypothesen . . . . . 5
1.4 Strategie . . . . . 5
1.5 Aufbau der Arbeit . . . . . 7

**2 Geschäftsprozessorientierte Entwicklung betrieblicher Systeme** **11**

2.1 Organisationstheoretische Grundlagen . . . . . 11
2.1.1 Organisation betrieblicher Aufgaben . . . . . 11
2.1.1.1 Analyse und Synthese betrieblicher Aufgaben . . . . 11
2.1.1.2 Durchführung und Kontrolle von Aufgaben . . . . . 12
2.1.1.3 Organisationsprinzipien . . . . . 13
2.1.2 Objektbasierte Aufgabenabgrenzung . . . . . 14
2.1.2.1 Objektprinzip und Verrichtungsprinzip . . . . . 14
2.1.2.2 Dezentralisation in der Organisationstheorie . . . . 15
2.1.2.3 Begriffsbestimmung dezentral organisierter Systeme 17
2.1.3 Objektorientierte Organisation betrieblicher Aufgaben . . . . 18
2.1.3.1 Objektorientierung . . . . . 18
2.1.3.2 Objektorientiertes Modell der Unternehmung . . . . 20
2.1.3.3 Objektorientiertes Aufgabenmodell . . . . . 22
2.1.4 Geschäftsprozesse und Workflows . . . . . 25

2.1.4.1 Geschäftsprozess . . . . . 25
2.1.4.2 Workflow . . . . . 28
2.2 Modellierung von Informationssystemen . . . . . 32
2.2.1 Grundlagen der Modellierung . . . . . 32
2.2.1.1 Modelltheorie und Modellbegriff . . . . . 32
2.2.1.2 Konstruktivistischer Modellbegriff . . . . . 34
2.2.1.3 Metamodellierung . . . . . 34
2.2.1.4 Modellierungsansätze und Modellierungsmethodiken 36
2.2.1.5 Modellierungssprachen . . . . . 38
2.2.2 Modellierung von betrieblichen Systemen und Geschäftsprozessen . . . . . 41
2.2.2.1 Konzeptuelle Modellierung . . . . . 41
2.2.2.2 Ansätze und Methodiken . . . . . 42
2.2.2.3 Modellebenen zur Modellierung von Geschäftsprozessen und Workflows . . . . . 46
2.2.3 Modelltransformation . . . . . 47
2.2.3.1 Einführung . . . . . 47
2.2.3.2 Generischer Architekturrahmen . . . . . 49
2.2.3.3 Model Driven Architecture . . . . . 50
2.2.3.4 Implementierung von Modelltransformationen . . . 53
2.2.4 Modellierung von Geschäftsprozessen am Beispiel des Semantischen Objektmodells (SOM) . . . . . 55
2.2.4.1 Unternehmensplan . . . . . 55
2.2.4.2 Geschäftsprozessmodell . . . . . 56
2.2.4.3 Spezifikation von Anwendungssystemen . . . . . 62
2.2.5 Modellierung von Workflows am Beispiel von Business Process Model and Notation (BPMN) . . . . . 65
2.2.5.1 BPMN Core . . . . . 66
2.2.5.2 Syntax und Notation . . . . . 68
2.2.5.3 Transformation von Geschäftsprozessmodellen des SOM in BPMN-Workflow-Schemata . . . . . 70
2.3 Kooperative Geschäftsprozesse . . . . . 72
2.3.1 Grundlagen zur Kooperation in Geschäftsprozessen . . . . . 72
2.3.1.1 Einführung . . . . . 72

2.3.1.2 Kooperationsbegriff . . . . . 73
2.3.2 Kooperation im Kontext der Wertschöpfung . . . . . 74
2.3.2.1 Grundlagen . . . . . 74
2.3.2.2 Value Co-Creation . . . . . 74
2.3.2.3 Wertschöpfung im Kontext zunehmender Vernetzung 75
2.3.2.4 Dezentrale Wertschöpfung . . . . . 76
2.3.3 Modellierung von kooperativen Geschäftsprozessen . . . . . 77
2.3.3.1 Einordnung . . . . . 77
2.3.3.2 Charakteristika der Modellierung kooperativer Geschäftsprozesse . . . . . 77
2.3.3.3 Modelle und Schemata . . . . . 79
2.3.3.4 Elemente der Modell-Syntax . . . . . 81
2.3.3.5 Abstraktionsebenen kooperativer Prozesse . . . . . 82
2.3.4 Merkmale von kooperativen Geschäftsprozessen in Netzwerken 83
2.3.4.1 Strukturierungsgrade von Geschäftsprozessen und Workflows . . . . . 83
2.3.4.2 Flexibilität in Geschäftsprozessen . . . . . 84
2.3.4.3 Evolution von Geschäftsprozess-Schemata . . . . . 86
2.3.4.4 Konzepte der Service-Orientierung . . . . . 87
2.3.4.5 Merkmale von kooperativen Geschäftsprozessen in Netzwerken . . . . . 89
2.3.5 Methoden zur Verteilung der Modellierung von Geschäftsprozessen . . . . . 90
2.3.5.1 Einordnung . . . . . 90
2.3.5.2 Modell-Management . . . . . 90
2.3.5.3 Modell-Repositories . . . . . 91
2.3.5.4 Asynchrone Versionierung von Modellen . . . . . 91
2.4 Verwandte Ansätze und Methoden . . . . . 94
2.4.1 Einordnung . . . . . 94
2.4.2 Ansätze des Adaptive Case Management . . . . . 94
2.4.3 Ansätze zur Workflow-Modellierung . . . . . 95
2.4.4 Ansätze zur Geschäftsprozessmodellierung . . . . . 98
2.5 Ergebnisdiskussion . . . . . 106

**3 Blockchain-Technologien** **109**
3.1 Blockchain-Systeme . . . 109
3.1.1 Komponenten und Merkmale . . . 109
3.1.1.1 Begriffsbestimmung . . . 109
3.1.1.2 Begriff und Komponenten von Blockchain-Systemen 110
3.1.1.3 Merkmale von Blockchain-Systemen . . . 112
3.1.2 Dezentrale Systeme . . . 114
3.1.2.1 Einordnung . . . 114
3.1.2.2 Merkmale dezentraler Systeme . . . 115
3.1.2.3 Dezentrale Koordination . . . 116
3.1.2.4 Dezentrale Blockchain-Systeme . . . 116
3.1.2.5 Smart Contracts zur Repräsentation von Vertragsbeziehungen in dezentralen Systemen . . . 118
3.2 Grundlagen zu verteilten Systemen und zur Anwendung von Kryptografie . . . 122
3.2.1 Verteilte Systeme . . . 122
3.2.1.1 Einführung . . . 122
3.2.1.2 Client-Server- und Peer-to-Peer-Architekturen . . . 123
3.2.1.3 CAP-Theorem . . . 126
3.2.1.4 Ordnung von Ereignissen . . . 128
3.2.1.5 Fehlertoleranz . . . 130
3.2.1.6 Byzantine Generals Problem . . . 130
3.2.1.7 Übereinkunft (Consensus) . . . 131
3.2.2 Kryptografie . . . 135
3.2.2.1 Einführung . . . 135
3.2.2.2 Kryptografische Hash-Verfahren . . . 135
3.2.2.3 Asymmetrische Signatur- und Verschlüsselungsverfahren . . . 137
3.3 Architektur von Blockchain-Systemen . . . 143
3.3.1 Datenstruktur . . . 145
3.3.1.1 Konzeptuelles Metamodell . . . 145
3.3.1.2 Blöcke zur Definition des Systemzustandes . . . 148
3.3.1.3 Komponenten eines Blocks in Blockchain-Systemen 150

3.3.1.4 Komponenten eines Blocks in Smart-Contract-Systemen . . . 151
3.3.1.5 Merkle-Bäume . . . 152
3.3.2 Netzwerk . . . 155
3.3.2.1 Architektur . . . 155
3.3.2.2 Typisierung beteiligter Knoten . . . 156
3.3.3 Protokoll . . . 158
3.3.3.1 Transaktionspropagation (F1) . . . 159
3.3.3.2 Blockerstellung (F2) . . . 160
3.3.3.3 Blockpropagation (F3) . . . 162
3.3.3.4 Konfliktauflösung (F4) . . . 165
3.3.4 Ausprägungen von Consensus-Verfahren . . . 169
3.3.4.1 Consensus-Verfahren in privaten Blockchain-Systemen 169
3.3.4.2 Consensus-Verfahren in öffentlichen Blockchain-Systemen . . . 172
3.3.4.3 Proof-of-Work oder Nakamoto Consensus . . . 176
3.3.5 Konsistenz und Unveränderlichkeit . . . 179
3.3.5.1 Partitionsbildung und Eventual Consistency . . . 179
3.3.5.2 Soft State und Eventual Consistency . . . 179
3.3.5.3 Unveränderlichkeit . . . 181
3.3.5.4 ACID- und BASE-Eigenschaften . . . 182
3.3.6 Limitationen zur Skalierbarkeit . . . 183
3.3.6.1 Ansätze zur Skalierung von Blockchain-Systemen . 185
3.4 Anwendungsklassen von Blockchain-Systemen . . . 188
3.4.1 Anwendungen von Blockchain- und Smart-Contract-Systemen 188
3.4.1.1 Blockchain-Systeme . . . 189
3.4.1.2 Smart-Contract-Systeme . . . 190
3.4.2 Integration anhand von Blockchain-Systemen . . . 192
3.4.2.1 Integrationskonzepte für Blockchain-Systeme . . . 193
3.4.2.2 Integration durch Transaktionen in Blockchain-Systemen . . . 194
3.4.2.3 Integration durch Smart Contracts in Smart-Contract-Systemen . . . 195

3.4.3 Kriterien zur Beurteilung des Einsatzes von öffentlichen und privaten Blockchain-Systemen . . . 196
3.4.4 Blockchain-Systeme in Verbindung mit Software-Systemen . . 197
3.4.5 Blockchain-Systeme zur Speicherung und Verarbeitung von Modellen . . . 199
3.4.5.1 Modelle in Permissioned Blockchains am Beispiel Knowledge Blockchain . . . 200
3.4.5.2 Modelle in dezentralen Blockchain-Systemen . . . 203
3.5 Ergebnisdiskussion . . . 204

**4 Ein Ansatz zur integrierten Entwicklung und Ausführung von Prozessen in dezentral organisierten Systemen 205**
4.1 Dezentrale Systementwicklung . . . 205
4.1.1 Zielarchitektur . . . 206
4.2 Architektur des Ansatzes . . . 209
4.2.1 Teilsysteme und Schichten des Ansatzes . . . 209
4.2.2 Konstruktion der Architektur . . . 210
4.2.3 Zeitbezogene Abgrenzung . . . 211
4.3 Modellsystem . . . 213
4.3.1 Generische Beschreibung des Modellsystems . . . 213
4.3.1.1 Rahmen zur Beschreibung von Modellen und Sichten der Zielarchitektur . . . 213
4.3.1.2 System-Gestaltung . . . 213
4.3.1.3 Prozess-Gestaltung . . . 215
4.3.1.4 Prozess-Ausführung . . . 216
4.3.2 Abstraktes Beispiel zur Instanziierung des Modellsystems . . 217
4.3.3 Instanziierung des Modellsystems . . . 220
4.3.3.1 Metamodelle der System-Gestaltung . . . 220
4.3.3.2 Metamodelle der Prozess-Gestaltung . . . 222
4.3.3.3 Metamodelle der Prozess-Ausführung . . . 227
4.4 Kooperationssystem . . . 230
4.4.1 Modell-Management . . . 231
4.4.1.1 Transaktionale Modellierung . . . 232

4.4.1.2 Versionsgraphen zur verteilten Verwaltung von Modellen . . . 234
4.4.1.3 Konsistenz und Integrität verteilter öffentlicher und privater Modelle . . . 237
4.4.1.4 Absicherung der Integrität und Verbindlichkeit in Smart Contracts . . . 240
4.4.2 Vereinbarung des Modellsystems . . . 243
4.4.2.1 Koordination des verteilten Commits . . . 243
4.4.2.2 Vereinbarung anhand des Commit-Verfahrens . . . 244
4.5 Ausführungssystem . . . 246
4.5.1 Instanz-Monitoring . . . 247
4.5.1.1 Modellierung von Ausführungszuständen . . . 247
4.5.1.2 Globale Identifikation von Modellelementen . . . 249
4.5.1.3 Objektspezifischer Smart Contract zur Nachverfolgung von Instanzen . . . 250
4.5.1.4 Generierung von Instanz-Protokollen . . . 252
4.6 Ergebnisdiskussion . . . 256

**5 Fallstudie zur Entwicklung und Ausführung von Geschäftsprozessen 259**
5.1 Betriebliche Prozesse in dezentralen Systemen . . . 259
5.1.1 Interorganisationale Prozesse . . . 259
5.1.2 Supply-Chain-Szenario . . . 260
5.1.3 Kriterien zur Auswahl des Szenarios . . . 261
5.2 Anwendung des Ansatzes zur System-Gestaltung . . . 262
5.2.1 Peer-Netzwerk: Netzwerkschema . . . 262
5.2.2 Wertschöpfungsnetz: Interaktionsschema . . . 263
5.2.2.1 1. Modellierung vernetzter Objekte . . . 264
5.2.2.2 2. Modellierung der gemeinsamen Ziele von kooperierenden Objekten . . . 264
5.3 Anwendung des Ansatzes zur Prozess-Gestaltung . . . 265
5.3.1 Kooperativer Prozess: Interaktions- und Vorgangs-Ereignis-Schema . . . 265
5.3.1.1 1. Modellierung des initialen Objektsystems . . . 265

5.3.1.2 2. Modellierung kooperierender Objekte in IAS und VES . . . 265
5.3.2 Peer-Workflow: Interaktionsschema und Vorgangstypschema 269
5.3.2.1 1. Detaillierung der Struktur anhand des IAS-P . . . 269
5.3.2.2 2. Detaillierung des Verhaltens anhand des VTS-P . 269
5.3.2.3 3. Implementierung der Vorgangsauslösung in objektspezifischen Smart Contracts . . . 271
5.4 Anwendung des Ansatzes zur Prozess-Ausführung . . . 273
5.4.1 Prozess-Instanz . . . 273
5.4.2 Workflow-Instanz . . . 275
5.5 Veränderung des Prozess-Schemas . . . 279
5.6 Ergebnisdiskussion . . . 282

**6 Zusammenfassung und Ergebnisdiskussion 285**
6.1 Zusammenfassung . . . 285
6.1.1 Geschäftsprozessorientierte Entwicklung betrieblicher Systeme 285
6.1.2 Blockchain-Systeme . . . 286
6.1.3 Integrierte Entwicklung und Ausführung von Prozessen . . . 287
6.2 Ergebnisdiskussion . . . 288

**A Smart Contracts 291**
A.1 Globaler Smart Contract . . . 292
A.2 Objektspezifischer Smart Contract . . . 302

**B Fallstudie 307**
B.1 Modelle . . . 308
B.2 Objektbaum-Datenstruktur . . . 316
B.3 Architektur des Software-Tools . . . 318

**Literatur 325**

# Abbildungsverzeichnis

2.1 Phasen einer Aufgabe (nach Ferstl und Sinz (2013, S. 110)) . . . . . 12
2.2 Strukturierungsformen . . . . . 17
2.3 Objektorientiertes Aufgabenmodell (nach Ferstl und Sinz (2013, S. 202) und Härer, Steffan et al. (2016, S. 94)) . . . . . 23
2.4 Betriebliche Aufgabe (nach Ferstl und Sinz (2013, S. 103)) . . . . . 24
2.5 Geschäftsprozess- und Workflow-Ebene . . . . . 29
2.6 Modell der Meta-Metaebene (nach Sinz (1996, S. 129)) . . . . . 35
2.7 Metamodellbasierte Transformation . . . . . 48
2.8 Generischer Architekturrahmen (Sinz 1995, 1997) . . . . . 50
2.9 Einordnung der Model Driven Architecture (vgl. OMG (2003), Ferstl und Sinz (2013, S. 499)) . . . . . 51
2.10 Unternehmensarchitektur und Vorgehensmodell des Semantischen Objektmodells (Ferstl und Sinz (2013, S. 195, 198)) . . . . . 56
2.11 Metamodell für Geschäftsprozesse in SOM (vgl. Ferstl und Sinz 2013, S. 219) . . . . . 57
2.12 Beispiel zu IAS und VES anhand eines Online-Shops (vereinfacht) . . 60
2.13 Metamodell für die Spezifikation von Anwendungssystemen in SOM nach Ferstl und Sinz (2013, S. 233) . . . . . 62
2.14 Beispiel zu KOS und VOS anhand eines Online-Shops (Ausschnitt) . 63
2.15 Komponenten von BPMN Core (eigene Darstellung nach OMG (2014, S. 47)) . . . . . 65
2.16 BPMN-Metamodell (erweiterte Darstellung von Pütz und Sinz (2010, S. 49)) . . . . . 66

3.1 Blockchain-System . . . . . 111
3.2 Architektur von Blockchain-Systemen . . . . . 143
3.3 Zustandsrepräsentation in Blockchain-Systemen . . . . . 144
3.4 Blockchain-Metamodell . . . . . 146

3.5 Zusammenhang zwischen System-Zuständen, Transaktionen und Blöcken . . . 149
3.6 Repräsentation von Smart Contracts in Blöcken . . . 153
3.7 Merkle-Baum (n = 8) mit Beispiel zur Integritätsprüfung von $e_5$ . . . 154
3.8 Zustandskonsistenz . . . 166
3.9 Transaktionen pro Tag für die Bitcoin-Blockchain (Blockchain.com 2018) 184
3.10 Transaktionen pro Tag für die Ethereum-Blockchain (Etherchain.org 2018) . . . 185
3.11 Integration anhand von Transaktionen . . . 194
3.12 Integration anhand von Smart Contracts . . . 195

4.1 Zielarchitektur des dezentralen Systems . . . 207
4.2 Architektur des Ansatzes . . . 209
4.3 Beschreibungsrahmen der Architektur des Modellsystems . . . 213
4.4 Transformation zwischen Sichten der System-Gestaltung . . . 214
4.5 Transformation zwischen Sichten der Prozess-Gestaltung . . . 216
4.6 Bildung von Instanz-Sichten zur Prozess-Ausführung . . . 217
4.7 Abstraktes Beispiel zur Modellsystem-Instanziierung . . . 219
4.8 Instanziierung des Modellsystems anhand von SOM . . . 220
4.9 Metamodelle und Transformation der System-Gestaltung . . . 221
4.10 Metamodelle und Transformation der Prozess-Gestaltung . . . 223
4.11 Metamodell der Prozess-Instanz . . . 227
4.12 Metamodell der Workflow-Instanz . . . 229
4.13 Architektur des Kooperationssystems . . . 230
4.14 Transaktionale Modellierung . . . 233
4.15 Commit-Operationen und Branches des Versionsgraphen . . . 236
4.16 Objektbaum-Datenstruktur . . . 239
4.17 Architektur des Ausführungssystems . . . 246
4.18 Star-Schema zur Protokollierung von Instanz-Zuständen . . . 253
4.19 Extraktionstabelle . . . 253
4.20 Beispiel eines Instanz-Protokolls . . . 254

5.1 Netzwerkschema von Produzent . . . 262
5.2 Netzwerkschemata von Logistikdienstleister, Zulieferer A und Makler 263

5.3 Interaktionsschema des globalen Wertschöpfungsnetzes aus der Sicht von „Produzent“ . . . . . 264
5.4 Initiales Interaktionsschema des kooperativen Prozesses . . . . . 266
5.5 Initiieren eines globalen Commits und Absenden einer Vote-Commit-Nachricht . . . . . 267
5.6 Interaktionsschema des kooperativen Prozesses nach Transaktions- und Objektzerlegung . . . . . 268
5.7 IAS-P von Produzent . . . . . 270
5.8 IAS-P von Zulieferer A . . . . . 271
5.9 VTS-P Zulieferer A . . . . . 272
5.10 IZS 1: Beginn des kooperativen Prozesses . . . . . 273
5.11 IZS 2: Vermittlung des Fertigungsauftrages von ZP-A . . . . . 274
5.12 IZS 3: Beginn des Produktionsvorgangs . . . . . 274
5.13 IZS-P von „Produzent“ und „Makler“ . . . . . 275
5.14 IZS-P von „Makler“ und „Logistikdienstleister“ während der Vermittlung des Fertigungsauftrages für ZP-A . . . . . 276
5.15 IZS-P von „Produzent“ vor Beginn des Produktionsvorgangs . . . . 277
5.16 Instanz-Protokoll . . . . . 278
5.17 IAS und VES des kooperativen Prozesses nach $\delta 4$ . . . . . 281

B.1 Objekt- und Transaktionszerlegung des kooperativen Prozesses . . . 309
B.2 VES des kooperativen Prozesses . . . . . 310
B.3 VES des kooperativen Prozesses (Fortsetzung) . . . . . 311
B.4 VTS-P von „Produzent“ . . . . . 312
B.5 VTS-P von „Produzent“ (Fortsetzung) . . . . . 313
B.6 VTS-P von „Makler“ . . . . . 314
B.7 VTS-P von „Logistikdienstleister“ . . . . . 315
B.8 XML-Serialisierung der Objektbaum-Datenstruktur . . . . . 316
B.9 XML-Serialisierung der Objektbaum-Datenstruktur (Fortsetzung) . . 317
B.10 Paketdiagramm des Software-Tools . . . . . 318
B.11 Paketdiagramm zur Implementierung des Modellsystems . . . . . 321
B.12 Klassendiagramm der zentralen Klassen zur Implementierung des Modellsystems . . . . . 323

B.13 Klassendiagramm zur Implementierung der Objektbaum-Datenstruktur . . . . . 324

# Tabellenverzeichnis

2.1 Beispiel zur Objekt- und Transaktionszerlegung . . . . . 60
2.2 Merkmale von kooperativen Geschäftsprozessen . . . . . 77
2.3 Modelle und Schemata kooperativer Prozesse . . . . . 79
2.4 Syntax-Elemente zur Abbildung von kooperativen Prozessen . . . . 81
2.5 Abstraktionsebenen und Abgrenzung kooperativer Prozesse . . . . . 83
2.6 Merkmale hochflexibler Geschäftsprozesse . . . . . 85
2.7 Verwandte Ansätze bestehender Modellierungssprachen . . . . . 99
2.8 Verwandte Ansätze zur kooperativen Modellierung . . . . . 102

3.1 Abgrenzung von dezentralen Systemen . . . . . 116
3.2 Verträge und Smart Contracts (Beschreibungsform nach Szabo (2018)) 119
3.3 Klassifikation von Anwendungen . . . . . 189

4.1 Zeitbezogene Abgrenzung . . . . . 211

# Abkürzungsverzeichnis

| | |
|---|---|
| **ACID** | Atomicity, Consistency, Isolation, Durability |
| **ACM** | Adaptive Case Management |
| **API** | Application Programming Interface |
| **ARIS** | Architektur integrierter Informationssysteme |
| **AS** | Aufgabensachziel |
| **ASIC** | Application-Specific Integrated Circuit |
| **AWS** | Anwendungssystem |
| **AX** | Aufgaben-externes Formalziel |
| **BASE** | Basically Available, Soft State, Eventually Consistent |
| **BC** | Blockchain |
| **BFT** | Byzantine Fault Tolerance |
| **BGP** | Byzantine Generals Problem |
| **BNF** | Backus-Naur-Form |
| **BPDM** | Business Process Definition Metamodel |
| **BPEL** | Business Process Execution Language |
| **BPM** | Business Process Management |
| **BPMN** | Business Process Model and Notation |
| **CBP** | Collaborative Business Process |
| **CIM** | Computation Independent Model |
| **CMMN** | Case Management Model and Notation |
| **DAG** | Directed Acyclic Graph |
| **DAO** | Decentralized Autonomous Organization |
| **DBVS** | Datenbankverwaltungssystem |
| **DHT** | Distributed Hash Table |
| **DLT** | Distributed Ledger Technology |
| **DMN** | Decision Model and Notation |
| **DVCS** | Distributed Version Control System |
| **DWH** | Data Warehouse System |
| **ECC** | Elliptic Curve Cryptography |
| **ECDSA** | Elliptic Curve Digital Signature Algorithm |

| | |
|---|---|
| **EMF** | Eclipse Modeling Framework |
| **EPK** | Ereignisgesteuerte Prozesskette |
| **ERC** | Ethereum Request for Comment |
| **ERM** | Entity-Relationship-Modell |
| **ERP** | Enterprise Resource Planning |
| **EVM** | Ethereum Virtual Machine |
| **GPU** | Graphics Processing Unit |
| **hGP** | hochflexibler Geschäftsprozess |
| **HTTP** | Hypertext Transfer Protocol |
| **IAS** | Interaktionsschema |
| **IAS-P** | Interaktionsschema Peer |
| **IOWF** | Interorganizational Workflow |
| **IPFS** | InterPlanetary File System |
| **IP** | Internet Protocol |
| **IS** | Informationssystem |
| **IZS** | Instanz-Zustands-Schema |
| **IZS-P** | Instanz-Zustands-Schema Peer |
| **JVM** | Java Virtual Machine |
| **KOS** | Konzeptuelles Objektschema |
| **KOT** | Konzeptueller Objekttyp |
| **LOT** | Leistungsspezifischer Objekttyp |
| **MDA** | Model Driven Architecture |
| **MDSE** | Model Driven Software Engineering |
| **MEMO** | Multi Purpose Enterprise Modelling |
| **MEP** | Message Exchange Pattern |
| **MOF** | Meta Object Factility |
| **NIST** | National Institute of Standards and Technology |
| **NWS-P** | Netzwerkschema Peer |
| **O-Ereignis** | objektinternes Ereignis |
| **OLAP** | Online Analytical Processing |
| **OMG** | Object Management Group |
| **OML** | Operational Mapping Language |
| **OOT** | Objektspezifischer Objekttyp |
| **OSI** | Open Systems Interconnection Model |
| **PBFT** | Practical Byzantine Fault Tolerance |

| | |
|---|---|
| **PIM** | Platform Independent Model |
| **PSM** | Platform Specific Model |
| **RNG** | Random Number Generator |
| **ROLAP** | Relational OLAP |
| **SCM** | Supply Chain Management |
| **SERM** | Strukturiertes Entity-Relationship-Modell |
| **SHA** | Secure Hash Algorithm |
| **SMR** | State Machine Replication |
| **SOA** | Serviceorientierte Architektur |
| **SOM** | Semantisches Objektmodell |
| **TCP** | Transmission Control Protocol |
| **T-Ereignis** | Transaktionsereignis |
| **TLS** | Transport Layer Security |
| **TOT** | Transaktionsspezifischer Objekttyp |
| **TPS** | Transaktionen pro Sekunde |
| **U-Ereignis** | Umweltereignis |
| **UDP** | User Datagram Protocol |
| **UML** | Unified Modeling Language |
| **UUID** | Universally Unique Identifier |
| **VES** | Vorgangs-Ereignis-Schema |
| **VOS** | Vorgangsobjektschema |
| **VOT** | Vorgangsobjekttyp |
| **VTS** | Vorgangstypschema |
| **VTS-P** | Vorgangstypschema Peer |
| **W3C** | World Wide Web Consortium |
| **WfMS** | Workflow Management System |
| **XMI** | XML Metadata Interchange |
| **XML** | Extensible Markup Language |
| **ZP** | Zwischenprodukt |

# Kapitel 1

# Einleitung

## 1.1 Motivation und Einführung

Diese Arbeit schlägt einen Ansatz zur Dezentralisierung der Entwicklung von betrieblichen Systemen vor. Damit wird zum einen die Gestaltung dezentraler Organisationen adressiert und zum anderen die innerhalb von solchen Systemen verteilt und nicht-zentral gesteuerte Koordination der Entwicklung. Die Dezentralisierung ist dabei sowohl auf der Aufgabenebene als auch auf der Aufgabenträgerebene (Ferstl und Sinz 2013, 4 ff.) zur Planung, Durchführung und Kontrolle der Aufgaben in Prozessen sowie zur Implementierung und technischen Realisierung von Anwendungssystemen relevant. Organisationstheoretisch beschreibt sie als *Dezentralisation* die Verlagerung von organisationalen Merkmalen „von einem Zentrum weg“ (Frese et al. 2012, S. 214 f.), die eine Verteilung der Aufbau- und Ablauforganisation bewirkt. Dieses grundlegende Prinzip ist heute zur Umsetzung der durch die Unternehmensplanung definierten Ziele in beinahe allen Organisationen mindestens operativ von wesentlicher Bedeutung, gewinnt durch die zunehmende Vernetzung der Informationssysteme von Unternehmen fortlaufend an Relevanz und wird schließlich durch Enabling-Technologien auf weitere organisationale Merkmale anwendbar. Dies zieht Weiterungen hinsichtlich der Koordination der Leistungserstellung nach sich.

Beispiele dezentral organisierter Systeme sind interorganisationale Kooperationen mehrerer Unternehmen, Unternehmensnetzwerke oder Supply Chains. Hinzu kommen beliebige weitere Systeme verteilter und nicht-zentral koordinierter Komponenten wie Organisationen, Individuen oder Software, die beispielsweise digitale Leistungen erstellen. Diesen Ausprägungen ist gemein, dass die Informationen der innerhalb eines Systems getätigten Geschäftstransaktionen Teil eines überbetrieblichen oder globalen Informationssystems sind. Eine Geschäftstransaktion wird dabei

von mehreren autonomen Systemteilnehmern gemeinsam durchgeführt und unterliegt nicht der Kontrolle Einzelner, d.h. die Koordination erfordert ein Zusammenwirken einzelner Systemkomponenten.

Auf der Ebene von Anwendungssystemen zieht dies einen Bedarf an Integration nach sich, dem beispielsweise mit dem Prinzip der Objektintegration (Ferstl und Sinz 2013, S. 249) sowie entsprechenden Service-basierten Implementierungen begegnet wird. Damit werden lokale Schnittstellen definiert, die beispielsweise Bestellungen annehmen und verarbeiten. Dabei können die hinter dezentral organisierten Systemen stehenden betrieblichen Informationssysteme heute als Komponenten eines globalen Informationssystems verstanden werden, das die Informationen verteilter Netzwerke verknüpft. Eine hierzu komplementäre und globale Realisierungsform der Integration kommt durch Blockchain-Systeme hinzu, die Transaktionen global verteilt durchführen, speichern und verarbeiten. Blockchain-Systeme etablieren eine global konsistente Sicht auf die anhand von Transaktionen persistierten Informationen.

Die globale Konsistenz einer Blockchain (Nofer et al. 2017) und deren dezentral koordinierte Validierung durch ein Protokoll erlauben nunmehr die Abbildung der Aufgabenebene dezentraler Systeme auf eine dezentral realisierte Aufgabenträgerebene. Betriebliche Transaktionen eines dezentralen Systems unterliegen damit nicht mehr der Kontrolle einzelner Teilnehmer. Die hinter einzelnen Transaktionen stehenden Informationen sind unabhängig von den mit ihnen verbundenen Unternehmen validierbar und realisieren eine verteilte Koordination. Die fachliche Gestaltung von Systemen und Prozessen ist damit nicht mehr nur auf Organisationen beschränkt, die gemeinsame Ziele verfolgen. Sie kann die Koordination des kooperativen Zusammenwirkens mehrerer Unternehmen beispielsweise in Coopetition-Beziehungen durch eine nicht-zentral kontrollierte und dennoch konsistente und globale Sicht unterstützen. Die Systementwicklung wird diesbezüglich durch die bisherigen Realisationen von verteilten Systemen in ihrer Skalierbarkeit beschränkt, da Informationssysteme und darauf basierende überbetriebliche Kooperationen weitgehend von einzelnen Organisationen entworfen und bestimmt werden. D.h., Kooperationen entstehen unter maßgeblicher Beteiligung und unter der Kontrolle einzelner Knoten, ohne die Ausnutzung von Netzwerkeffekten.

Dabei zeigen die anfangs genannten Beispiele dezentraler Systeme ein aufgrund der ansteigenden Komplexität zunehmendes Problem der Koordination auf. Der immer höhere Verteilungsgrad stellt organisationale Netzwerke vor die Herausforderung,

die zunehmende Komplexität ausgehend von Systemkomponenten zu kontrollieren und zu beherrschen. Dieses Problem der dezentralen Koordination und die damit einhergehenden Limitationen der Skalierbarkeit von soziotechnischen Systemen werden mit dieser Arbeit adressiert. Dies betrifft insbesondere die Abstimmung von Prozessen zwischen Systemkomponenten und die Übereinkunft hinsichtlich gemeinsam durchzuführender Prozesse.

In diesem Kontext entwickelt die Arbeit einen modellbasierten Ansatz zur Entwicklung und Ausführung von Prozessen in dezentralen Organisationen, der die Gestaltung von skalierenden Netzwerken ermöglicht, in denen autonome und selbstorganisierte Teilnehmer in Kooperation Leistungen erstellen. Die Koordination soll somit kooperativ und verteilt erfolgen. Hierfür systematisiert die vorliegende Arbeit theoretische Grundlagen zu den Bereichen Organisation, Geschäftsprozesse, Modellierung und Blockchain-Technologien, um auf dieser Basis den Rahmen einer Architektur und deren Implementierung in Form eines Ansatzes zur dezentralen Systementwicklung vorzuschlagen. Die technische Realisierbarkeit zeigt ein Software-Tool, welches die Bildung von Kooperationen, die Gestaltung von Prozessen sowie die Überwachung der Ausführung unterstützt. Damit wird das Ziel verfolgt, eine Abstimmung und ein Zusammenwirken verteilter Organisationen zu ermöglichen, die als Teil eines dezentralen Systems Prozesse in Kooperation gestalten und ausführen.

## 1.2 Untersuchungsproblem

Das folgende Untersuchungsziel und dessen Detaillierung definieren zusammen mit dem Untersuchungsobjekt die dieser Arbeit zugrunde liegende Problemstellung.

**Untersuchungsziel**

(Z) *Ziel dieser Arbeit ist die Konzeption eines modellbasierten Ansatzes zur Entwicklung und Ausführung von Prozessen innerhalb von dezentral organisierten Systemen, deren Komponenten verteilt sind und keiner zentralen Koordination unterliegen.*

Damit wird der Leitfrage nachgegangen, *wie* autonome Komponenten von dezentral organisierten Systemen unter Beherrschung der zunehmenden Komplexität und Verteilung in Kooperation Prozesse entwerfen und ausführen können.

**Untersuchungsobjekt**

Dezentral organisierte Objektsysteme werden dem Ziel nach auf Modellsysteme abgebildet, die von den jeweiligen Stakeholdern in Kooperation verteilt entworfen werden und durch deren Zusammenwirken verteilt zur Ausführung kommen. Das Untersuchungsobjekt

(O) *Dezentrales System*

untergliedert sich in die innerhalb des Gestaltungsprozesses erforderlichen

(O.1) *Beschreibungsmittel*,

(O.2) *Koordinationsmethoden* sowie

(O.3) *Entwicklungs- und Ausführungsplattformen*

soweit sie der Realisierung des Untersuchungsziels dienlich sind.

**Detaillierung des Untersuchungsziels**

Aus dem übergeordneten Untersuchungsziel leiten sich im Hinblick auf die modellbasierten Beschreibungsmittel und Koordinationsmethoden sowie hinsichtlich der Plattformen eine Reihe von Unterzielen ab:

(Z.1) *Die Abbildung von interorganisationalen und auf Kooperation beruhenden Prozessen dezentral organisierter Objektsysteme auf Modellsysteme.*

(Z.2) *Die modellbasierte Koordination der prozessorientierten Gestaltung dezentral organisierter Systeme, ausgehend von zusammenwirkenden Systemkomponenten.*

(Z.3) *Die modellbasierte Koordination der Ausführung von Prozessen innerhalb von dezentral organisierten Systemen durch deren Komponenten.*

(Z.4) *Die Unterstützung der Herausbildung von Informationssystemen bei dezentraler Organisation anhand von Basistechnologien verteilter Systeme sowie anhand von Distributed-Ledger- und Blockchain-Technologien.*

Mit dieser Problemstellung soll die Gestaltung von dezentralen soziotechnischen Systemen und ihren Informationssystemen untersucht werden.

## 1.3 Hypothesen

Betriebliche Informationssysteme sind heute als Komponenten eines globalen Systems nicht voneinander isoliert. Sie stellen autonom agierende Teilsysteme dar, die als Komponenten eines Wertschöpfungsnetzes in Kooperation betriebliche Leistungen erstellen. Gleichzeitig stehen einzelne Teilsysteme dabei zueinander in Interaktionsbeziehungen; sie sind jedoch prinzipiell in ihren Handlungsentscheidungen voneinander unabhängig.

Diese Betrachtung führt vor dem Hintergrund des Untersuchungsproblems zu den folgenden Hypothesen:

(H.1) *Ein dezentral organisiertes System erfordert eine Entwicklung, die innerhalb von Teilsystemen autonom und teilsystemübergreifend in Kooperation durchgeführt wird.*

(H.2) *Die Betrachtung und Abbildung eines einzelnen globalen Objektsystems, oder aber die Betrachtung und Abbildung voneinander isolierter Teilsysteme, sind für die Entwicklung und Ausführung von Prozessen in dezentral organisierten Systemen nicht ausreichend.*

(H.3) *Zur Unterstützung der Gestaltung von Prozessen in dezentral organisierten Systemen ist eine Integration des Entwurfs und der Ausführung erforderlich.*

(H.4) *Blockchain-Systeme ermöglichen eine dezentrale Modellbildung und eine transparente Ausführung von Prozessen, die in Kombination die Entstehung von dezentral organisierten Systemen befördern.*

(H.5) *Eine auf Blockchain-Systemen basierende Dezentralisierung erhöht die Skalierbarkeit von organisationalen Systemen. Hierdurch wird die Entwicklung von komplexen und hochverteilten soziotechnischen Systemen beherrschbar.*

## 1.4 Strategie

Die Ziele zur dezentralen Gestaltung von Systemen und der Entwurf eines Ansatzes werden anhand von konzeptionell-deduktiven Methoden verfolgt (Wilde und Hess 2006, 2007). Die Ergebnistypen der Arbeit sollen Modelle, konzeptuelle Bezugsrahmen und eine Software-Implementierung umfassen (Frank 2016; Österle, Winter et al. 2010).

### Vorgehen

Das Vorgehen umfasst (a.) die Entwicklung *theoretischer Grundlagen* und *technischer Grundlagen*, (b.) die Untersuchung *dezentraler Koordinationsmethoden* anhand von Blockchain-Systemen, (c.) die *Konzeption eines Architektur-Rahmens* zur Beschreibung des Ansatzes und die *Implementierung des Ansatzes* sowie (d.) die *Evaluierung* anhand einer Fallstudie zusammen mit einer Software-Implementierung.

(a) Die Entwicklung theoretischer Grundlagen zur geschäftsprozessorientierten Systementwicklung und die Entwicklung technischer Grundlagen zu Blockchain-Systemen sind die Voraussetzung der Konstruktion des Ansatzes; sie betreffen dort die Aufgaben- und die Aufgabenträgerebene.

(b) Das Problem der dezentralen Koordination wird mit Methoden von Blockchain-Systemen adressiert. Hierzu gehört insbesondere die Abstimmung von Prozessen zwischen autonomen und verteilten Systemkomponenten sowie die Übereinkunft hinsichtlich der durchzuführenden Prozesse.

(c) Konzeption und Implementierung zielen auf den Entwurf eines Ansatzes ab, der Prozesse zur Leistungserstellung zwischen verteilten und in Kooperation zusammenwirkenden Teilnehmern anhand von Modellen koordiniert; insbesondere Modelle (1.) des zugrunde liegenden Netzes, (2.) der kooperativ gestalteten Prozesse, (3.) der dort enthaltenen Vorgangstypen und (4.) der Instanziierung. Dieses Vorgehen soll eine dezentrale Wertschöpfung unterstützen.

(d) Die Evaluierung soll (1.) die Anwendbarkeit des Ansatzes zeigen, (2.) die Implementierbarkeit anhand eines Software-Prototyps demonstrieren und (3.) limitierende Faktoren aufzeigen.

Ausgehend von der Systematisierung theoretischer Grundlagen soll der Ansatz durch einen Architektur-Rahmen beschrieben, mit Modellierungssprachen instanziiert und anhand eines Werkzeugs implementiert werden.

### Methodische Einordnung

Das mit dieser Arbeit verfolgte Ziel betrifft in erster Linie die Gestaltung von Systemen und die damit verbundene Untersuchung eines Konstruktionsproblems (Österle, Winter et al. 2010; Sinz 2010b). Die erkenntnistheoretisch dem konstruktiven Paradigma zuordenbare Ausrichtung unterstützt die Anwendung der konzeptionell-deduktiven Analyse (Wilde und Hess 2006, 2007). Die Resultate dieser Arbeit in Form von Modellen, konzeptuellen Bezugsrahmen und Software-Prototypen sind mit diesem Paradigma verbunden (Frank 2016; Österle, Winter et al. 2010).

## 1.5 Aufbau der Arbeit

Die Strukturierung der vorliegenden Arbeit folgt dem Vorgehen zur Lösung des Untersuchungsproblems. Die Entwicklung theoretischer und technischer Grundlagen der Aufgaben- und Aufgabenträgerebene ist Gegenstand der Kapitel 2 bzw. 3. Das zuletzt genannte Kapitel vertieft anschließend auf Blockchain-Systemen beruhende Koordinationsmethoden. Die Konzeption und Implementierung des Ansatzes bespricht Kapitel 4. Kapitel 5 evaluiert den Ansatz. Kapitel 6 schließt die Arbeit ab.

**Kapitel 2** diskutiert die geschäftsprozessorientierte Systementwicklung der Aufgabenebene von organisationstheoretischen Grundlagen hin zu auf Kooperation beruhenden Prozessen. Das Kapitel untergliedert sich in die Teile:

(2.1) **Organisationstheoretische Grundlagen** zur Analyse und Synthese von Aufgaben, zum Dezentralisationsbegriff und der objektorientierten Organisation betrieblicher Aufgaben. Hiermit werden verteilte und lose gekoppelte Objekte als elementare Systemkomponenten zur Beschreibung von Prozessen und Workflows etabliert.

(2.2) **Modellierung von Informationssystemen**, insbesondere von Prozessen und Workflows. Mit den Grundlagen des objektorientierten Aufgabenmodells bespricht das Kapitel die objektorientierte Abbildung von Prozessen anhand der SOM-Methodik und die Abbildung von Workflows anhand von BPMN.

(2.3) **Kooperative Geschäftsprozesse**. Vor dem Hintergrund der Dezentralisation und der Modellierung verteilter Objekte bespricht das Kapitel Grundlagen für die Gestaltung von Prozessen und Workflows zwischen kooperierenden Objekten. Ausgehend von Basiskonzepten werden die Kooperation in der Wertschöpfung und die damit verbundene Entwicklung des Wertschöpfungsbegriffs hin zu einem Verständnis einer dezentralen Wertschöpfung diskutiert.

(2.4) **Verwandte Ansätze und Methoden**. Dieser Abschnitt diskutiert existierende Ansätze zur Abbildung von auf Kooperation beruhenden Prozessen auf Modelle. Gegenstand der Diskussion ist einerseits die Aufarbeitung konzeptuell verwandter Strömungen, wie andererseits der Vergleich konkreter Ansätze mit dem hier auf Aufgabenebene vorgeschlagenen Ansatz. Weitergehend betrachtet das Kapitel elementare Methoden, die in der Literatur und der Praxis zur kooperativen Modellierung herangezogen werden.

**Kapitel 3** geht auf die technischen Grundlagen von verteilten Systemen und Blockchain-Systemen ein, die Gegenstand der Umsetzung der Aufgabenträgerebene des Ansatzes sind.

(3.1) **Blockchain-Systeme**. Das Kapitel führt zunächst in die Grundlagen von Blockchain-Systemen ein, um diese schließlich verallgemeinernd im Kontext der Themen der Dezentralisierung und der dezentralen Koordination einzuordnen.

(3.2) **Grundlagen zu verteilten Systemen und zur Anwendung von Kryptografie**. Die für Blockchain-Systeme wesentlichen Grundlagen aus den Bereichen verteilte Systeme und Kryptografie sind Gegenstand dieses Kapitels.

(3.3) **Architektur von Blockchain-Systemen**. Aufbauend auf der Datenstruktur Blockchain werden die Komponenten von Blockchain-Systemen systematisiert. In Bezug auf den State-of-the-Art verbreiteter Systeme ordnet das Kapitel Merkmale und Verfahren vor dem Hintergrund von verteilten Systemen ein.

(3.4) **Anwendungsklassen von Blockchain-Systemen**. Die Anwendung und Anwendbarkeit von Blockchain-Systemen greift dieses Kapitel anhand einer Klassifikation sowie in Beiträgen zur Anwendbarkeit und Integration von Blockchain-Systemen auf.

**Kapitel 4** umfasst die Konzeption und Implementierung des Ansatzes.

(4.1) **Dezentrale Systementwicklung**. Die Zielarchitektur ist ein dezentrales System, das ein Netzwerk zur Erstellung von Leistungen darstellt.

(4.2) **Architektur des Ansatzes**. Die Architektur umfasst drei Subsysteme zur Modellierung, Kooperation und Ausführung. Diese werden anhand von Modellen und Sichten zur Entwicklung und Ausführung von Prozessen innerhalb von dezentralen Systemen beschrieben.

(4.3) **Modellsystem**. Das Modellsystem definiert die gemeinsame Sprache zur modellbasierten Entwicklung des Systems. Es wird zunächst als Rahmen allgemein beschrieben und anschließend anhand von Metamodellen instanziiert. Hiermit werden die System-Gestaltung, die Prozess-Gestaltung und die Prozess-Ausführung auf Aufgaben- und Aufgabenträgerebene spezifiziert.

(4.4) **Kooperationssystem**. Das Kooperationssystem koordiniert die modellbasierte Systementwicklung basierend auf einem transaktionalen Modellierungskonzept zur verteilten Modellierung. Ziel ist das Management verteilter Modelle

durch Versionierung und die verteilte Vereinbarung des Modellsystems durch verteilte Commit-Verfahren und Smart Contracts.

(4.5) **Ausführungssystem**. Zur Abbildung der Ausführung auf Modelle verfolgt diese Komponente einen Monitoring-Ansatz, der die fachliche Validierung der Ausführung durch verteilte Instanz-Modelle zulässt. Technisch stützt sich das System auf Smart Contracts und die Infrastruktur des Kooperationssystems.

**Kapitel 5** führt eine Fallstudie zur Entwicklung und Ausführung von Geschäftsprozessen durch.

(5.1) **Betriebliche Prozesse in dezentralen Systemen**. Die Fallstudie greift interorganisationale Prozesse als typischen Anwendungsfall heraus und bespricht in diesem Zusammenhang ein Supply-Chain-Szenario.

(5.2) **Anwendung des Ansatzes zur System-Gestaltung**. Die System-Gestaltung zeigt die von den Komponenten des Systems ausgehende Bildung eines Netzwerks zur Erstellung von Leistungen.

(5.3) **Anwendung des Ansatzes zur Prozess-Gestaltung**. Die Prozess-Gestaltung zeigt die verteilte Erfassung von Modellen der Gestaltungszeit für die Aufgaben- und Aufgabenträgerebene.

(5.4) **Anwendung des Ansatzes zur Prozess-Ausführung**. Die Prozess-Ausführung zeigt die verteilte Bildung von Instanz-Modellen.

(5.5) **Veränderung des Prozess-Schemas**. Eine Diskussion zur Anwendung des Ansatzes als Teil eines sich verändernden Systems schließt die Fallstudie ab.

**Kapitel 6** fasst die Arbeit zusammen und diskutiert deren Ergebnisse.

# Kapitel 2

# Geschäftsprozessorientierte Entwicklung betrieblicher Systeme

## 2.1 Organisationstheoretische Grundlagen

Der in diesem Kapitel vorliegende erste Teil der Arbeit betrachtet die Entwicklung betrieblicher Systeme aus der Perspektive der in und zwischen Systemen ablaufenden Prozesse. Das Kapitel diskutiert zunächst organisationstheoretische Grundlagen (Kapitel 2.1) und die Modellierung von Geschäftsprozessen und Workflows als Teil von Informationssystemen (Kapitel 2.2). Diese Betrachtung umfasst in Abhängigkeit der Abgrenzung der beteiligten Systeme sowohl innerbetriebliche wie auch überbetriebliche Prozesse. Mit dieser Sichtweise werden, dem Untersuchungsproblem gemäß, elementare Beschreibungsmittel für prozessorientierte Systeme etabliert. Nachfolgend steht das Zusammenwirken der Systemkomponenten im Vordergrund, das zu kooperativen Geschäftsprozessen (Kapitel 2.3) führt. Diese sind die Grundlage des in Kapitel 4 entwickelten Ansatzes. Hierzu verwandte Ansätze (Kapitel 2.4) für die Modellierung von auf Kooperation beruhenden Prozessen sowie für die kooperative Modellierung werden schließlich betrachtet.

### 2.1.1 Organisation betrieblicher Aufgaben

#### 2.1.1.1 Analyse und Synthese betrieblicher Aufgaben

Gegenstand der Analyse und Synthese ist die Herausbildung von Aufgaben in einer Organisation, wie sie von Grochla (1972, S. 37) beschrieben wurde. Eine Aufgabe definiert sich anhand von Zielsetzungen, die menschliche Akteure als Maßgabe für ihr Handeln heranziehen (Kosiol 1976, S. 43). Handlungen sind damit an der Zielerreichung ausgerichtet und hinsichtlich des Ziels zweckbezogen.

Das Konzept der Analyse und Synthese von Aufgaben (Kosiol 1976, S. 189) wird im Rahmen der Aufbauorganisation zur Bildung von Einheiten durch Separation und zur anschließenden organisationalen Integration herangezogen. Die Aufgabenanalyse entspricht einer Gliederung in Aufgabenteile durch die Separation organisationaler Strukturen, deren Sub-Strukturen jeweils eigenständige Aufgabenteile konstituieren. Die Aufgabensynthese bezeichnet die Integration einzelner Aufgabenteile, sodass eine Bildung und Zuordnung von Stellen für zusammengehörige Aufgaben erreicht wird (Bleicher 1991, S. 48 f.; Vahs 2015, S. 58). Im Falle einer zu automatisierenden Aufgabe erfolgt eine auf IT-Systeme bezogene Zuordnung von Anwendungssystemen (Ferstl und Sinz 2013, S. 110). Anhand der Zuordnungen ergibt sich eine explizite Trennung zwischen Aufgaben und den als Aufgabenträgern zugeordneten Stellen oder Anwendungssystemen.

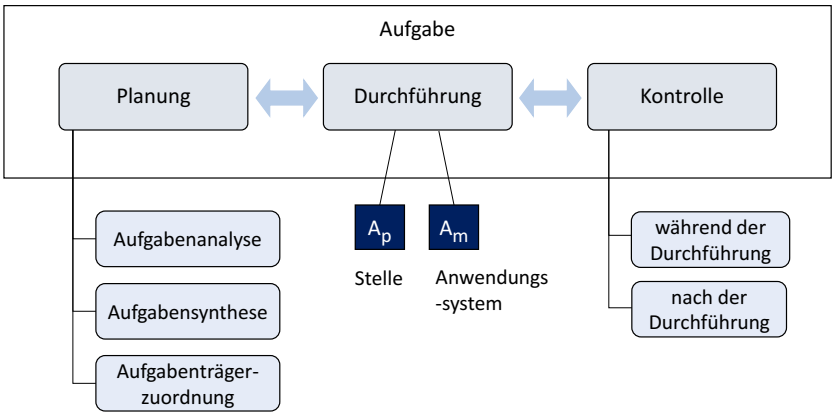



ABBILDUNG 2.1: Phasen einer Aufgabe (nach Ferstl und Sinz (2013, S. 110))

Die Gliederung einer Aufgabe nach elementaren Phasen bildet ein dreiteiliges Phasenkonzept (Ferstl und Sinz 2013, S. 110). Eine Aufgabe umfasst (1.) die Phase der *Planung*, bestehend aus der Aufgabenanalyse, der Aufgabensynthese und der Zuordnung von Stellen oder Anwendungssystemen als Aufgabenträger, (2.) die Phase der *Durchführung* durch Stellen oder Anwendungssysteme sowie (3.) die Phase der *Kontrolle*, die während der Durchführung oder auch nach der Durchführung erfolgen kann. Der Zusammenhang ist in Abbildung 2.1 dargestellt.

#### 2.1.1.2 Durchführung und Kontrolle von Aufgaben

Bezogen auf die Ablauforganisation führen Synthese- und Analysevorgänge zur Gliederung einer Aufgabe hinsichtlich des Ablaufs als Arbeitselemente (Bleicher

1991, S. 49), Arbeitsgänge (Vahs 2015, S. 58) oder Vorgänge. Diese können typisiert und als Vorgangstypen erfasst werden. Ein Vorgang eines spezifischen Typs entspricht einer Vorgangs-Instanz (Ferstl und Sinz 2013, S. 99).

Die aus der Zweckorientierung folgende Beschreibung von Sach- und Formalzielen betrieblicher Aufgaben kann zunächst unabhängig von deren Durchführung erfolgen. Die Beschreibung einer Aufgabe anhand von Zielen ist durch verschiedenartige Beschreibungsmittel vorstellbar, z.B. in natürlicher Sprache. Über die Fragestellung, *wie* die Erreichung des Ziels einer Aufgabe anhand von Methoden oder Verfahren ablaufen kann, ist hiermit noch keine Aussage getroffen. Bezüglich der Wahl geeigneter Beschreibungsmittel besteht eine Abhängigkeit zum Modelltyp der Aufgabe (Ferstl und Sinz 2013, S. 104).

Menschliche Akteure oder IT-Systeme realisieren einzelne Aufgaben in Abhängigkeit einer Zuordnung, die sich durch die Differenzierung beider Aspekte in separate Abstraktionsebenen ergibt (Ferstl und Sinz 2013, S. 4). Die Durchführung einer Aufgabe entspricht einem Vorgang, der einmalig oder wiederholt zur Ausführung kommt. Der Vorgang ist damit eine Instanz eines auf Typ-Ebene spezifizierten Vorgangstyps. Einzelne Vorgangsinstanzen sind, dem Phasenkonzept folgend, während oder nach ihrer Ausführung hinsichtlich der aus der Synthese hervorgehenden Ziele kontrollierbar.

#### 2.1.1.3 Organisationsprinzipien

Bezogen auf den Aufgabenträger einer Aufgabe sind zwei grundlegend verschiedene Prinzipien der Organisation unterscheidbar. So können die Planung und die Durchführung von Aufgaben entweder durch verschiedene Akteure erfolgen, oder durch dieselben, die in diesem Fall gleichzeitig Aufgabenträger sind. Im ersten Fall liegt *Fremdorganisation* vor, bei der die Organisation aus der Sicht der Aufgabenträger von fremden Akteuren zentral koordiniert wird (Schreyögg und Geiger 2016, S. 15 f.; Ferstl und Sinz 2013, S. 111; Bodendorf et al. 2012). Im zweiten Fall liegt *Selbstorganisation* vor, wobei die gesamte Planung durch diejenigen Akteure erfolgt, welche die Aufgabe durchführen. Die Organisation kann damit zudem *ad-hoc* erfolgen, unter der Voraussetzung, dass die beteiligten Akteure ein zur Planung und Durchführung ausreichendes Wissen sowie die notwendige Motivation mitbringen (Ferstl und Sinz 2013, S. 111). Sind diese Voraussetzungen erfüllt, kann die Organisation an Flexibilität gewinnen und durch die agierenden Akteure fortlaufend an den Aufgabenzielen ausgerichtet werden. Da die Koordination von den beteiligten

Akteuren ausgeht, ist das Organisationsprinzip der Selbstorganisation inhärent dezentral. Es wird daher in dieser Arbeit für die dezentrale Entwicklung von Systemen herangezogen.

Dabei ist anzumerken, dass anhand dieser Betrachtung an dieser Stelle noch keine Unterscheidung von personellen und maschinellen Akteuren oder Aufgabenträgern getroffen werden muss. Liegt eine zu automatisierende Aufgabe vor, wird die Phase der Planung durch die Systementwicklung übernommen und dort gemeinhin unter der Beteiligung von personellen Aufgabenträgern ausgeführt, während die Durchführung IT-Systemen überlassen wird.

### 2.1.2 Objektbasierte Aufgabenabgrenzung

#### 2.1.2.1 Objektprinzip und Verrichtungsprinzip

In der Organisationstheorie werden im Rahmen der Aufbauorganisation Kriterien für die Zusammenfassung oder Trennung von Einheiten unterschieden. Sie sind damit während den Aufgabenphasen *Analyse* und *Synthese* relevant. Zwei grundlegende Organisationsprinzipien fassen Kriterien der Aufgabenverrichtung als Verrichtungsprinzip sowie Kriterien der Bildung von einzelnen Objekten als Objektprinzip zusammen (Ferstl und Sinz 2013; Vahs 2015, S. 97 f.).

Das Verrichtungsprinzip unterteilt den Aufbau von Einheiten nach der Art der Durchführung der Aufgabe. Gleichartige Verrichtungsarten werden dabei zusammengefasst und in einer Organisationseinheit, beispielsweise einer Abteilung, gruppiert. Die klassische Aufteilung anhand von betrieblichen Funktionsbereichen ähnlicher Verrichtungsarten hat in der Praxis eine hohe Verbreitung erlangt.

Demgegenüber steht das Objektprinzip, mit dem zunächst Organisationseinheiten in Form von einzelnen Objekten abgegrenzt werden. Ein Abgrenzungskriterium bestimmt dabei eine Gliederung in Objekte, die sich in der Zuordnung von Aufgaben zu Objekten manifestiert. Das gewählte Kriterium bezieht sich auf die Homogenität der zugeordneten Aufgaben, ist prinzipiell aber beliebig wählbar. Es ist somit von der Verrichtungsart entkoppelt und ermöglicht eine den Unternehmens- und Aufgabenzielen folgende Gliederung der Organisation. Beispiele sind die an Regionen, Kundengruppen oder Produktgruppen ausgerichteten Bildungen von Abteilungen. Für die Gliederung anhand von Objekten wird auch der Begriff der Objektzentralisierung (Vahs 2015, S. 98) verwendet, der auf eine zentral von den Objekten ausgehende Koordination Bezug nimmt.

Aufgrund der angestrebten Homogenität werden durch eine geeignete Wahl des Abgrenzungskriteriums Aufgaben mit einer Vielzahl von Interdependenzen zusammengefasst, sodass den Aufgaben zugeordnete Organisationseinheiten weitgehend unabhängig voneinander agieren können. Damit sind die anhand des Objektprinzips zugeordneten Aufgaben für die *Selbstorganisation* prädestiniert. Der Analyse und Synthese von Aufgaben in dezentral organisierten Systemen liegt daher das Objektprinzip zugrunde.

Eine mehrfache Gliederung oder Zerlegung ist bei beiden Organisationsprinzipien nicht ausgeschlossen. In der Praxis sind Mischformen verbreitet (Ferstl und Sinz 2013, S. 73). Häufig werden nach dem Objektprinzip zerlegte Einheiten erneut anhand des Verrichtungsprinzips gegliedert.

Durch das Objektprinzip ergibt sich somit die Grundlage zur Zerlegung von Aufgaben, die unter Hinzunahme des Konzepts der Objektorientierung zur mehrstufigen Zerlegung und zur Kapselung von Struktur- und Verhaltensmerkmalen erweitert werden kann (siehe Kapitel 2.1.3.1).

#### 2.1.2.2 Dezentralisation in der Organisationstheorie

Im Allgemeinen beschreiben die Begriffe Dezentralisierung und Dezentralisation das Gegenteil einer Konzentration von Merkmalsausprägungen um ein Zentrum, d.h. die Verteilung von Merkmalsausprägungen vom Zentrum weg (Frese et al. 2012, S. 214 ff.). Im Kontext von dezentral organisierten Systemen sind die genannten Begriffe in zweierlei Hinsicht relevant, wobei verschiedene Bedeutungsverständnisse auf unterschiedlichen Abstraktionsebenen zugrunde liegen. Zum einen ist der Begriff Dezentralisation in der Organisationstheorie von hoher Bedeutung, zum anderen beschreibt die Dezentralisierung auf technischer Ebene Merkmale verteilter Systeme (siehe Kapitel 3.1.2).

Dem organisationstheoretischen Begriff liegen unterschiedliche Interpretationen (Frese et al. 2012, S. 213 f., 437) zugrunde.

- Entscheidungsdelegation: Simon et al. (1954) definieren den Begriff über die Delegation von Entscheidungen, weg von zentralen Stellen einer Aufbauorganisation. In diesem Verständnis steht der Begriff der Entscheidungsdezentralisation für eine Delegation in untergeordnete Hierarchieebenen (Thommen et al. 2017, S. 463).
- Systemsynthese: Bleicher (1991, S. 48 ff.) beschreibt mit der „Zentralisation und Dezentralisation bei der Systemsynthese“ die Verteilung von Aufgaben

anhand der Merkmale einer sachlichen, formalen und persönlichen Dezentralisation sowie einer Mittel- und Raumdezentralisation.

- Organisationsprinzip: Die Verrichtungsdezentralisation und die Organisationsdezentralisation beschreiben die Bildung der Aufbauorganisation unter Anwendung des Verrichtungs- bzw. Objektprinzips (Thommen et al. 2017, S. 460).
- Teilsystembildung: Ferstl und Sinz (2013, S. 44 f.) beschreiben die Bildung von Teilsystemen als Dezentralisierung, bei der die entstehenden Teilsysteme selbstständig sind und über Flussbeziehungen miteinander kommunizieren. Die Verteilung der Systemkomponenten definiert sich durch die Zerlegung einzelner Aufgaben.
- Spartenbildung: Weiterhin wird der Begriff der Dezentralisation teilweise mit der Bildung einer Spartenorganisation gleichgesetzt (Frese et al. 2012, S. 437), die auf die Struktur der Aufbauorganisation abzielt. Ziel ist die Bildung mehrerer Sparten oder Divisionen, anhand derer die strategische Führung potenziell von operativer Überforderung entlastet werden kann.

Im Folgenden wird der Begriff auf die Bildung von betrieblichen Systemen und ihre Aufgaben bezogen (Bleicher 1991; Ferstl und Sinz 2013). Für die Ablauf- und Aufbauorganisation wird das nachfolgende Verständnis unterstellt.

- In der Ablauforganisation erfolgt zunächst eine Aufteilung von Aufgaben in einzelne Elemente, Arbeitselemente oder Aktionen. Diese können im Rahmen der Synthese hinsichtlich der Merkmale Person, Raum und Zeit dezentral getrennt oder zentral, im Sinne einer Einheit, synthetisiert werden.
- Die Dezentralisation in der Aufbauorganisation wird mit einer Trennung gleichgesetzt, sodass sich im Rahmen der Aufgabenanalyse eine Trennung in Teilaufgaben ergibt, denen bei einer anschließenden Synthese Stellen zugeordnet werden. Hinsichtlich der strategischen Ausrichtung der Aufbauorganisation ist die Divisionalisierung damit ein mögliches Beispiel für eine Dezentralisation der Aufbauorganisation (Bleicher 1991, S. 77 f.). Hinsichtlich der Aufbauorganisation sind hybride Formen möglich, die z.B. auf höheren Hierarchieebenen zentral und darunter dezentral konfiguriert sind (Bleicher 1991, S. 80).

#### 2.1.2.3 Begriffsbestimmung dezentral organisierter Systeme

Für die dezentrale Planung und Durchführung von Aufgaben im Sinne dieser Arbeit stellt die Dezentralisierung der Organisation eine notwendige Voraussetzung dar. Die weitere Betrachtung stützt sich auf die Interpretation von Bleicher (1991), deren Charakteristikum die Verteilung von Aufgaben ist, sowie auf Ferstl und Sinz (2013), hinsichtlich der holistischen Betrachtung von Informationssystemen und der Merkmale selbstständiger und miteinander kommunizierender Teilsysteme. Kongruent zu diesen Darstellungen wird hier nicht notwendigerweise ein hierarchisches System unterstellt. Die Folge ist die Entstehung eines Netzes, in dem Organisationseinheiten hierarchisch oder nicht-hierarchisch koordiniert interagieren. Im Falle des Netzes verlaufen Beziehungen zwischen beliebigen Knoten, während im anderen Fall stets hierarchische Beziehungen zwischen den Einheiten (E) vorliegen. Der Zusammenhang ist in Abbildung 2.2 mit einer beispielhaften Zerlegung für die Strukturierungsformen *Hierarchie* und *Netz* dargestellt. Nimmt man eine Gliederung der Knoten des Netzes in Teilsysteme an, ergeben sich im abgebildeten Beispiel drei Teilsysteme, in denen jeweils die Möglichkeit zur Bildung einer Hierarchie besteht. Werden die abgebildeten Knoten entsprechend aggregiert, ergibt sich die links dargestellte Hierarchie.

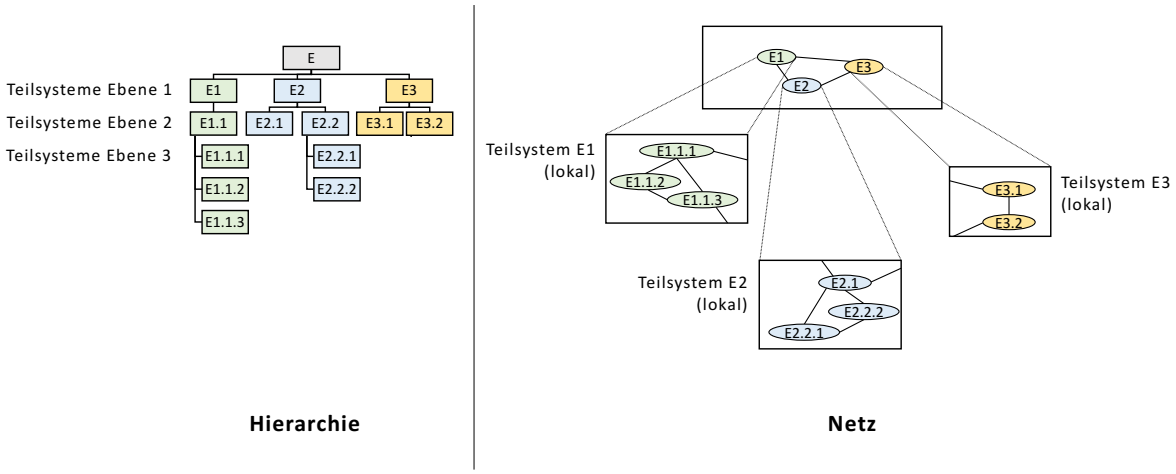



ABBILDUNG 2.2: Strukturierungsformen

Im Falle eines Netzes liegt im Vergleich mit einer Hierarchie eine schwächere und ebenso allgemeinere Strukturierungsform vor. Handelt es sich bei den Knoten des Netzes um verteilte Aufgaben, deren Aufgabenträger kooperativ Aufgaben planen und ausführen, wird durch das Netz eine höhere Strukturflexibilität ermöglicht. Gleichzeitig bedingt dies den Verzicht der einer Hierarchie inhärenten Eindeutigkeit

der Zerlegung von Einheiten. Wird die allgemeinere Strukturierungsform des Netzes eingesetzt, geht die Koordination und die Interaktion zwischen Einheiten über einzelne Hierarchiestufen hinaus. Dabei ergeben sich Vorteile hinsichtlich der Skalierung der Struktur. Bei einer nicht-synthetischen Herausbildung von Netzen ergeben sich mit größeren Knotenzahlen häufig skalenfreie Netze (Barabási und Albert 1999). In diesen folgt die Anzahl der Kanten je Knoten einer Power-Law-Verteilung. Dies kann in Kommunikationsnetzen zu skalierbaren, robusten und ausfallsicheren Architekturen führen (Barabási, Albert und Jeong 2000). Daher wird der Ansatz verfolgt, die Vorteile der Skalenfreiheit in Kommunikationsnetzen auf organisationale Strukturen zu übertragen, um Flexibilität und eine skalierbare Organisation mit einem geringeren Ausmaß an Intermediation zu realisieren.

### 2.1.3 Objektorientierte Organisation betrieblicher Aufgaben

#### 2.1.3.1 Objektorientierung

Die Verwendung von Objekten zur Modularisierung geht auf die Programmiersprache Simula 67 zurück (Dahl und Nygaard 1966), in der Quellcode in Form von Klassen strukturiert werden kann. Der Begriff der Objektorientierung wurde von Alan Kay geprägt (Plattner 2014), der in der objektorientierten Programmiersprache Smalltalk die Kommunikation von Objekten per Nachrichten (Ingalls 1978) nach dem Vorbild zellulärer Systeme der Biologie gestaltete (Chamond Liu 2000). Die daraus hervorgehende objektorientierte Programmierung (OOP) ist das heute vorherrschende Paradigma von imperativen Programmiersprachen. Weitere Ansätze zur objektorientierten Modellierung werden am Ende dieses Abschnitts aufgegriffen.

**Erfassung von Objekten der Realwelt und Bildung von Klassen**

Objektorientierte Ansätze erfassen gleichartige Objekte anhand ihrer Struktur und ihres Verhaltens in je einer *Klasse*, die den *Objekttyp* (Typ) repräsentiert. Ein *Objekt* entspricht einer *Instanz* einer Klasse, die konkrete Ausprägungen der definierten Struktur enthält und die Manipulation der Ausprägungen anhand des definierten Verhaltens erlaubt. Ein Objekt repräsentiert den genannten ursprünglichen Ansätzen zufolge meist ein Objekt oder ein Konzept der Realwelt, das anhand des Typs klassifiziert wird. Zur Definition einer Klasse wird die Struktur in Form von Attributen angegeben, deren Ausprägungen Daten sind. Im Falle einer statischen Typisierung wird die Festlegung konkreter Typen je Attribut impliziert. Die Definition des Verhaltens geschieht durch die Angabe von Operatoren oder Methoden, welche die

Ausprägungen der Attribute manipulieren. Beispielsweise durch die Anwendung einer Funktion auf einen Attributwert. Je Klasse existieren beliebig viele Objekte des Typs der Klasse, die sich jeweils durch eine Objektidentität definieren.

**Kommunikation lose gekoppelter Objekte**

Die Kommunikation zwischen Objekten wird durch den Austausch von Nachrichten unterstützt. Je Klasse werden daher die empfangbaren Nachrichten angegeben, meist in Form der per Nachricht aufrufbaren Operatoren. Damit kommunizieren einzelne Objekte untereinander, ohne Struktur- und Verhaltensdefinitionen zu exponieren. Hierdurch ergibt sich eine *Kapselung* von Struktur und Verhaltensmerkmalen, die für andere Objekte nicht sichtbar sind. Die Kommunikation zwischen Objekten basiert damit allein auf dem Austausch von Nachrichten, ohne Kenntnis der vollständigen Struktur- und Verhaltensdefinitionen der beteiligten Objekte, die somit *lose gekoppelt* sind.

**Typhierarchien und Polymorphie**

Weiterhin können Beziehungen zwischen Klassen etabliert werden, um Typhierarchien abzubilden. Eine Vererbungsbeziehung zwischen zwei Klassen übernimmt die Struktur- und Verhaltensmerkmale von einer Super-Klasse, um diese zu spezialisieren. Durch die Übernahme der Merkmale entspricht diese Spezialisierung in der Sub-Klasse einer Erweiterung der Super-Klasse. Bei umgekehrter Betrachtungsweise kann von einer Generalisierung der Sub-Klasse gesprochen werden. Im Falle der Vererbung liegt stets Polymorphie vor, da ein Objekt der Sub-Klasse auch den Typ der allgemeineren Super-Klasse und ggf. die Typen aller in der Vererbungshierarchie übergeordneten Super-Klassen besitzt. Zudem kann Mehrfachvererbung vorliegen, sofern von einer Sub-Klasse mehrere Vererbungsbeziehungen zu verschiedenen Super-Klassen führen. Dies erfordert die Handhabung konfliktärer Struktur- und Verhaltensdefinitionen, beispielsweise im Falle von gleich benannten Operatoren in zwei Super-Klassen.

**Objektorientierte Modellierung**

Aus der Objektorientierung entstandene Ansätze zur Gestaltung von Software-Systemen sind beispielsweise: objektorientierte Analyse (OOA) und objektorientiertes Design (OOD) von Coad und Yourdon (1991) sowie Ansätze von Rumbaugh, Blaha et al. (1991), Jacobson et al. (1992) und Booch (1993) aus denen die Unified Modeling Language UML und UML 2 (Rumbaugh, Jacobson et al. 2004) zur objektorientierten Modellierung von Software und IT-Systemkomponenten entstanden.

Weiterhin entstanden ist die SysML, die einige Teile der UML 2 sowie Erweiterungen für die allgemeine Entwicklung technischer Systeme enthält. UML und SysML werden durch die Object Management Group standardisiert und liegen aktuell in den Versionen 2.5.1 (OMG 2017b) bzw. 1.5 (OMG 2017a) vor. Dort werden Diagramme zur Abbildung der Statik und der Dynamik unterschieden, etwa Klassen- bzw. Sequenzdiagramme zur objektorientierten Darstellung der Struktur und des Verhaltens. Beispiele für nicht-objektorientierte Diagramme sind Anwendungsfalldiagramme (UML) oder Anforderungsdiagramme (SysML). In den genannten Ansätzen beginnt die Modellierung eines Systems mit der Entwicklungsphase der Anforderungsdefinition und Anforderungsanalyse (Balzert et al. 2009, S. 19 ff.). Im Folgenden werden daher objektorientierte Betrachtungen des Unternehmens und der betrieblichen Aufgabe diskutiert, um die anschließende objektorientierte Modellierung von Geschäftsprozessen zu motivieren.

#### 2.1.3.2 Objektorientiertes Modell der Unternehmung

Die Charakteristika objektorientierter Ansätze für die Gestaltung von Software- und IT-Systemen lassen sich auf die Gestaltung von betrieblichen Informationssystemen übertragen. Ferstl und Sinz (2013, S. 45 f.) etablieren ein objektorientiertes Modell der Unternehmung, in dem diese als System miteinander kommunizierender Objekte beschrieben wird. Analog zu den diskutierten Charakteristika der Objektorientierung werden im Folgenden die Bildung von Objekten, die Kommunikation zwischen diesen und die Herausbildung von Typhierarchien diskutiert.

**Erfassung von Objekten als Teilsysteme der Unternehmung**

Das gesamte System der Unternehmung wird als Objekt betrachtet, aus dem Teilsysteme als untergeordnete Objekte gebildet werden. In Analogie zur Bildung von Klassen entsteht damit eine Struktur- und Verhaltensdefinition der Teilsysteme der Unternehmung. Die Abgrenzung und Erfassung verschiedener Teile der Unternehmung orientiert sich an der betrieblichen Realität oder einem angestrebten Sollzustand. Hinsichtlich der Objekte wird zwischen Leistungs- und Lenkungsobjekten unterschieden, die betriebliche Leistungen erstellen oder diese koordinieren. Struktur und Verhalten sind auch hier innerhalb von Objekten gekapselt, sodass die Interaktion mit anderen Objekten anhand von definierten Kommunikationsbeziehungen ohne Kenntnis der inneren Struktur- und Verhaltensmerkmale erfolgen kann. Das rein auf Teilsystembildung basierende Objektprinzip wird damit erweitert und um Kommunikationsbeziehungen anhand von Flüssen ergänzt.

**Kommunikation anhand von Lenkungs- und Leistungsflüssen**

Kommunikationsbeziehungen werden analog zum Austausch von Nachrichten in Form von Leistungs- und Lenkungsflüssen zwischen Objekten realisiert. Leistungsflüsse beschreiben die Übertragung von Gütern, einschließlich materiellen Gütern und Energie, Zahlungen und Dienstleistungen (Ferstl und Sinz 2013, S. 48). Aus Steuerungsgründen findet die Übertragung nicht in kontinuierlicher Form statt, sondern zeitdiskret in Paketen. Lenkungsflüsse beschreiben die Übertragung von Nachrichten, die Leistungsflüsse steuern und Informationen über diese enthalten. Dabei ist ein Lenkungsfluss Auslöser eines in entgegengesetzter Richtung verlaufenden Leistungsflusses. Mit dem Leistungsfluss verläuft zudem in gleicher Richtung ein Lenkungsfluss zur Übertragung von Informationen über den Transfer der Leistung. Flussbeziehungen sind damit stets notwendig, wenn zwei Objekte voneinander abgegrenzt beschrieben werden. Beide Objekte sind dann separat zu betrachten; sie sind jeweils selbstständig und in ihrer internen Organisation voneinander unabhängig. Miteinander kommunizierende Objekte sind damit lose gekoppelt. Sie werden anhand der Bildung untergeordneter Objekte durch hierarchische Zerlegung erzeugt.

**Hierarchische Zerlegung**

Zunächst wird bei der Erfassung von Objekten die gesamte Unternehmung als einzelnes Objekt repräsentiert, welches in weitere Objekte zerlegt werden kann, die den Teilsystemen des Unternehmens entsprechen. Die mehrfache Anwendung dieser hierarchischen Zerlegung führt zu einer von einem einzelnen Objekt ausgehenden Baumstruktur. Eine Zerlegung eines Objekts in mehrere Zerlegungsprodukte gleicht damit dem Herstellen von je einer Erweiterungsbeziehung ausgehend von einem Zerlegungsprodukt hin zum ursächlichen Objekt. Die Semantik des Vererbungskonzepts der Objektorientierung wird gewahrt, da die Zerlegung hier ebenso eine Spezialisierung eines Objekts zu den untergeordneten Objekten darstellt, und umgekehrt eine Generalisierung. Die entstehende Hierarchie entspricht einer Typhierarchie von Klassen, die zueinander in Vererbungsbeziehungen stehen.

Neben der Klassenbildung liegt hinsichtlich der betrieblichen Organisation das Lenkungsebenenmodell (Ferstl und Sinz 2013, S. 39 ff.; Kirsch und Klein 1977) zugrunde. Die bereits erwähnte Differenzierung von Lenkungs- und Leistungsobjekten erfordert im Falle einer mehrstufigen Zerlegung definierte Regeln hinsichtlich der als Zerlegungsprodukte entstehenden Objekte. Leistungsobjekte, wie das initial vorliegende Objekt der gesamten Unternehmung, lassen sich in mehrere Leistungsobjekte

oder alternativ in mindestens ein Leistungs- sowie mindestens ein Lenkungsobjekt zerlegen. Lenkungsobjekte führen bei Zerlegung stets zu mehreren untergeordneten Lenkungsobjekten. Leistungsobjekte besitzen stets auch einen Lenkungsanteil, der die interne Leistungserstellung koordiniert und die nachrichtenbasierte Kommunikation zu anderen Objekten über Lenkungsflüsse herstellt.

**Objektorientierte Modellierung**

Das Objektprinzip der Organisationstheorie entspricht für sich genommen noch nicht einem objektorientierten Ansatz. So existieren diverse Möglichkeiten zur Modellierung der Strukturorganisation, die allerdings auf den Aspekt der Bildung des Unternehmensaufbaus ausgerichtet sind. Ein Beispiel hierfür ist ein einfaches Organigramm, das die hierarchische Zerlegung ohne die diskutierten Charakteristika der Objektorientierung umfasst. Hingegen fehlt den genannten Ansätzen der objektorientierten Modellierung von IT-Systemen und Software der Bezug zur Organisationstheorie. Das im folgenden Abschnitt diskutierte objektorientierte Aufgabenmodell integriert beide Konzepte.

#### 2.1.3.3 Objektorientiertes Aufgabenmodell

Das objektorientierte Aufgabenmodell schlägt eine Repräsentation betrieblicher Aufgaben (siehe Kapitel 2.1.1) als Bestandteil von Objekten vor. Diese lassen sich aus Außensicht darstellen, hinsichtlich der Ziele und der Kommunikation mit weiteren Objekten planen, sowie aus Innensicht, bezüglich der Verfahren zur Durchführung der Aufgabe, implementieren. Den Phasen einer Aufgabe folgend, werden Aspekte der Spezifikation, Durchführung und Kontrolle abgebildet.

**Betriebliche Objekte**

Ein betriebliches Objekt enthält eine Menge von betrieblichen Aufgaben sowie ein gemeinsames Aufgabenobjekt; der Zusammenhang ist in Abbildung 2.3 dargestellt. Das Aufgabenobjekt entspricht einem objektinternen Speicher, der den Zustand des Objekts im Sinne der Objektorientierung kapselt. Die Verknüpfung der enthaltenen Aufgaben (hier >A.1, A.2>, >A.3) und ihre Lösungsverfahren definieren das Verhalten des Objekts, das zweckgerichtet hinsichtlich der Ziele ist. Sie operieren auf dem gemeinsamen Speicher und verändern damit den Zustand des Objekts.

Die Kommunikation mehrerer Objekte erfolgt anhand von Nachrichten, die über Nachrichtenkanäle in Form von Transaktionen zwischen Objekten ausgetauscht werden. Eine Transaktion schafft damit eine Infrastruktur zum Austausch von Nachrichten zwischen zwei Objekten.

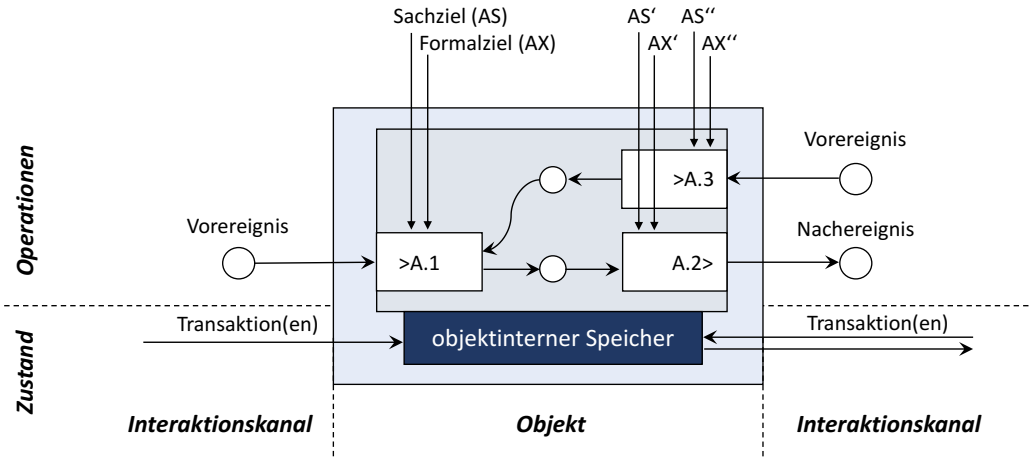



ABBILDUNG 2.3: Objektorientiertes Aufgabenmodell (nach Ferstl und Sinz (2013, S. 202) und Härer, Steffan et al. (2016, S. 94))

Eine Nachrichtendefinition definiert die anhand von Nachrichten aufrufbaren Operatoren mit ihren Parametern. Die Auslösung der Durchführung einer Aufgabe infolge einer eingehenden Nachricht wird anhand von Ereignissen beschrieben, die als Vor- und Nachereignisse auftreten.

**Außensicht der Aufgabe**

Die Aufgabenspezifikation geht von Sachzielen (AS) und Aufgaben-externen Formalzielen (AX) aus, welche die Aufgabe konstituieren. Ereignisse (Events) lösen Aufgaben aus und markieren deren abgeschlossene Durchführung. Dabei werden Vor- und Nachereignisse unterschieden, die einerseits eine Event-basierte Auslösung und andererseits, durch Verkettung, Reihenfolgebeziehungen definieren. Ein zentraler Bestandteil einer Aufgabe ist weiterhin das Aufgabenobjekt, auf dem das Lösungsverfahren zur Realisierung der Ziele operiert. Die Aufgabenbeschreibung ist damit unabhängig von der Innensicht und unabhängig von den Aufgabenträgern, die die Aufgabe durchführen. Aufgabenträger umfassen sowohl personelle Aufgabenträger wie auch Anwendungssysteme. Die Kommunikation und Interaktion zwischen Objekten ist auf dieser Abstraktionsebene bereits ableitbar.

**Innensicht der Aufgabe**

Eine als Aufgabenträger separat beschreibbare Abstraktionsebene ordnet nach den Phasen der Aufgabenanalyse und Aufgabensynthese Aufgabenträger zu. Diese Zuordnung sowie die folgende Durchführung betreffen die Innensicht der Aufgabe. Diese wird durch das Lösungsverfahren anhand von auszuführenden Aktionen

sowie einer Aktionensteuerung beschrieben. Das Lösungsverfahren definiert damit, wie die aus Außensicht formulierten Sach- und Formalziele erreicht werden können. Die Aktionensteuerung wird von den Vorereignissen der Aufgabe initiiert, steuert die Auslösung und die Abfolge der elementaren Aktionen und kann anschließend Nachereignisse auslösen. Einzelne Aktionen des Lösungsverfahrens werden anhand der Aktionensteuerung ausgelöst und liefern Aktionenergebnisse, ggf. iterativ. Die Aktion operiert dabei auf dem Aufgabenobjekt und verändert damit den objektinternen Speicher des Objekts. Abbildung 2.4 visualisiert das Konzept.

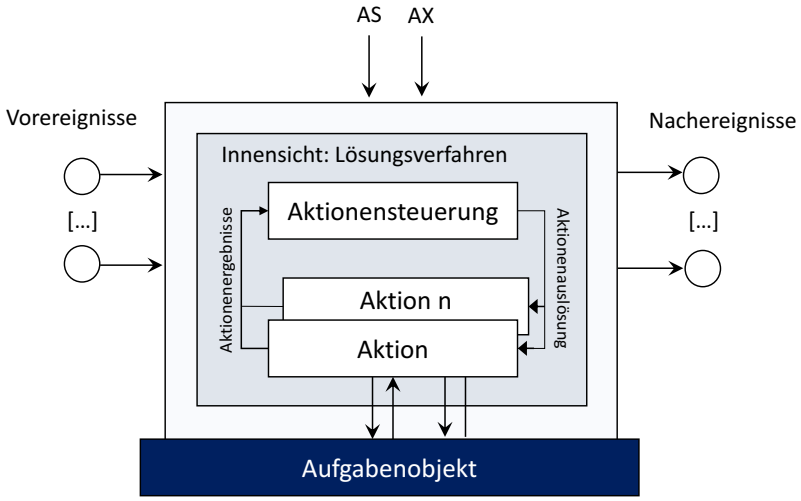



ABBILDUNG 2.4: Betriebliche Aufgabe (nach Ferstl und Sinz (2013, S. 103))

Die Aktionensteuerung kann, für sich genommen, einer beliebig komplexen Ausgestaltung unterliegen und beispielsweise mit Beschreibungsmitteln der deklarativen oder imperativen Programmierung formuliert sein. Die konkrete Ausgestaltung hängt vom Modelltyp und dem Lösungsverfahren der Aufgabe ab. Ein Lösungsverfahren geht dabei exakt, approximierend, heuristisch oder lernend vor (Ferstl 1979). Der Einsatz lernender Verfahren bezieht sich auf konnektionistische Modelltypen von Aufgaben, beispielsweise neuronale Netze, während alle vier Typen für analytische und wissensbasierte Modelltypen zum Einsatz kommen.

Von der genauen Spezifikation des Lösungsverfahrens bleibt die hierarchische Zerlegung von Aufgaben in Form von Objekten unberührt bestehen. Sie wird zur Modellierung von Geschäftsprozessen im nachfolgenden Kapitel wieder aufgegriffen.

## 2.1.4 Geschäftsprozesse und Workflows

### 2.1.4.1 Geschäftsprozess

Geschäftsprozesse erlauben die Planung, Durchführung und Kontrolle von Aufgaben, die zueinander in zeitlich-logischen Beziehungen stehen. In dieser Funktion liegen sie hier als grundlegendes Gestaltungsparadigma für Informationssysteme zugrunde. Im Folgenden wird das Begriffsverständnis aus der Perspektive der Organisationstheorie eingeführt.

**Begriffsverständnis**

Die aus der Ablauforganisation folgende Verknüpfung von Aufgaben kann anhand von Geschäftsprozessen beschrieben werden. Hess definiert die Elemente eines Prozesses als „Aufgaben, Aufgabenträger und Sachmittel“ sowie die Beziehungen als Ablaufbeziehungen zwischen den Elementen (Hess 1996, S. 13). Vossen und Becker (1996, S. 18 f.) definieren einen Geschäftsprozess als „inhaltlich abgeschlossene zeitliche und sachlogische Abfolge von Funktionen, die zur Bearbeitung eines betriebswirtschaftlich relevanten Objekts notwendig sind“. Inhaltlich zusammengehörige und verkettete Aktivitäten werden zur Leistungserstellung und Leistungsverwertung vollzogen und führen zu einem abgeschlossenen Ergebnis, das zum Unternehmenserfolg beiträgt (Vahs 2015, S. 221). Der Ablauf durchzuführender Aktivitäten ist nicht auf sequenzielle Abfolgen beschränkt. Bartmann et al. (2011, S. 3) nehmen (a.) auf Aktivitäten Bezug, deren Zusammenhang über ein gemeinsames Ziel definiert ist, (b.) auf deren Ablauf anhand einer Ereignis-basierten Steuerung, (c.) auf die Leistungserstellung durch die „Übernahme von Inputs [...] sowie die Erzeugung und Bereitstellung von Outputs [...]“, sowie (d.) schließlich auf die Zuordnung und Nutzung von Ressourcen. Ferstl und Sinz (2013, S. 4 ff.) definieren auf Basis des zuletzt genannten Merkmals eine Trennung der in Aktivitäten durchzuführenden Aufgaben und der Aufgabenträger. Zusammenfassend besitzt ein Geschäftsprozess:

- **M1**: zielgerichtet durchzuführende Aktivitäten,
- **M2**: einen von Ereignissen oder auch Abfolgebeziehungen determinierten Ablauf,
- **M3**: leistungsverwertende Inputs und leistungserstellende Outputs sowie
- **M4**: eine Zuordnung einzusetzender Ressourcen.

Die in M1 implizierte Zielorientierung kann sich auf zeitliche und sachlogische Funktionsabgrenzungen sowie zusammengehörige Aktivitäten beziehen. Anhand von M2 wird ein Ablauf unterstellt, der hinsichtlich des Ziels determiniert, aber nicht notwendigerweise deterministisch ist. M3 betrifft die Wertschöpfung, die in dieser Form keine Aussage über den Unternehmenserfolg trifft. In M4 wird die Definition von Ressourcen zur Gestaltungszeit und deren Einsatz zur Laufzeit impliziert. Ressourcen beinhalten personelle Aufgabenträger und Anwendungssysteme (Ferstl und Sinz 2013, S. 197), die Teile eines Prozesses zur Laufzeit durchführen. Zur Gestaltungszeit des Geschäftsprozesses kann der Prozess damit zunächst unabhängig von der Zuordnung der Aufgabenträger erfolgen, sodass sich die beiden Abstraktionsebenen Aufgabenebene und Aufgabenträgerebene ergeben. Die Definition eines Geschäftsprozesses wird anhand der Aufgabenebene gebildet. Die Aufgabenträgerebene wird im weiteren Verlauf zur Zuordnung von Aufgabenträgern und für die Definition von Lösungsverfahren betrachtet, z.B. bei der Definition detaillierter Prozess-Teile in Form von Workflows.

**Einordnung des Geschäftsprozess-Begriffs**

Die Aufgabenebene enthält die diskutierte Abfolge einzelner Elemente in Form von in Beziehung stehenden Aufgaben, um Geschäftsprozesse zu definieren. Bezogen auf das objektorientierte Aufgabenmodell umfasst ein Geschäftsprozess Aufgaben in ihrer Außensicht (Ferstl und Sinz 2013, S. 202 f.). Je Aufgabe sind somit Sach- und Formalziele, Vor- und Nachereignisse sowie das Aufgabenobjekt zu erfassen. Das auf dem Aufgabenobjekt operierende Lösungsverfahren bleibt zunächst unbestimmt. Zur Verknüpfung von Aufgaben kommen die Vor- und Nachereignisse zur Anwendung, wobei ein Nachereignis gleichzeitig als Vorereignis zur Auslösung weiterer Aufgaben gesetzt werden kann. Damit ist anhand des Ablaufs ein Systemverhalten festgelegt und, bei Betrachtung der Komponenten einzelner Aufgaben, eine Systemstruktur. Unabhängig davon ist als Konsequenz der Objektorientierung eine Zerlegung von Aufgaben möglich, die sich aus den in der Objektorientierung bestehenden Typhierarchien ergibt.

**Betriebliches Informationssystem**

Betriebliche Informationssysteme als die informationsverarbeitenden Teilsysteme eines Unternehmens (Ferstl und Sinz 2013, S. 4) umfassen der Objektart *Information* zugeordnete Aufgaben von Prozessen zur Lenkung der Leistungserstellung und zur Durchführung der Leistungserstellung (Ferstl und Sinz 2013, S. 6 f.). Das hiermit abgegrenzte Informationssystem wird durch das Basissystem ergänzt, das

die Objektart Nicht-Information betrifft und damit physische Flussbeziehungen wie „materielle Güterflüsse, Energieflüsse, Zahlungsflüsse und Flüsse von physischen Dienstleistungen (z. B. ärztliche Behandlung)“ (Ferstl und Sinz 2013, S. 7) enthält.

Dieses Begriffsverständnis orientiert sich damit an der vorliegenden Objektart und beschreibt ein soziotechnisches System. Das Informationssystem schließt Aufgaben ein, die automatisiert oder nichtautomatisiert durchgeführt werden können, sowie personelle und maschinelle Aufgabenträger; der Begriff schließt damit IT-Systeme ein und ist nicht auf diese beschränkt. Dieses Verständnis ist z.B. bei Krcmar (2015, S. 22), Ferstl und Sinz (2013, S. 6 ff.), Becker et al. (2012, S. 20) oder Fink et al. (2005, S. 3) zu finden. Davon abzugrenzen sind die Begriffe Führungsinformationssystem (Executive Information System, EIS), Management-Informationssystem sowie „operative“ und „analytische“ Informationssysteme (Mertens et al. 2012, S. 60; Gluchowski und Chamoni 2016, S. 6; Kurbel 2016, S. 206 ff.). Diese Begriffe werden im Kontext von IT-Systemen und deren betrieblichen Anwendungen verwendet und sind darauf fokussiert.

Aufbauend auf dem Begriffsverständnis nach Ferstl und Sinz kann eine weitere Abgrenzung hinsichtlich der Lenkung und der Leistungserstellung vorgenommen werden. Das Teilsystem zur Lenkung umfasst die Aufgabenphasen Planung, Steuerung und Kontrolle anhand von entsprechenden Informationsflüssen. Das Lenkungssystem betrifft somit ausschließlich das Informationssystem. Das Leistungssystem umfasst die Phase der Durchführung zur Ausführungszeit und betrifft das Informations- und das Basissystems, je nach Objektart der enthaltenen Aufgaben. Die Zuordnung des Aufgabenträgertyps erfolgt in Abhängigkeit der Automatisierung einer Aufgabe sowie in Abhängigkeit der Objektart. Automatisierten Aufgaben der Objektart Information kann der Aufgabenträgertyp Anwendungssystem zugeordnet werden, während nichtautomatisierten Aufgaben dieser Objektart personelle Aufgabenträger zugeordnet werden. Beiden Aufgabenträgertypen können Aufgaben des Lenkungs- oder des Leistungssystems zugeordnet werden.

**Wertschöpfungsketten und Wertschöpfungsnetze**

Mehrere Geschäftsprozesse bilden in ihrer Verbindung eine Wertschöpfungskette aus miteinander verknüpften Prozessen, die in Output-Input-Beziehungen zueinander stehen. Nach Porter fasst eine Wertschöpfungskette Tätigkeiten zusammen, mit denen ein Produkt entworfen, hergestellt, vertrieben, ausgeliefert und unterstützt

wird (Porter 2014, S. 65). Entlang der Kette entstehen folglich Wertschöpfungsstufen, die durch Prozesse realisiert sind. Die nicht notwendigerweise sequenzielle Verkettung führt zu Wertschöpfungsnetzen.

Geschäftsprozesse als Teil von Wertschöpfungsnetzen umfassen intra- und interorganisationale Geschäftsprozesse, die an einer mehrstufigen Erstellung von Leistungen in betrieblichen und überbetrieblichen Organisationsstrukturen beteiligt sind (Ferstl und Sinz 1993).

#### 2.1.4.2 Workflow

Der Begriff des Geschäftsprozesses bezieht sich im Allgemeinen auf eine Beschreibung eines Prozessablaufs, ohne zwischen verschiedenen Abstraktionsebenen zu differenzieren. Ein Workflow weist gegenüber einem Geschäftsprozess eine höhere Spezifität auf, die durch eine Detaillierung der im Geschäftsprozess definierten Aufgaben ausgedrückt werden kann. Die Aufgaben eines Geschäftsprozesses entsprechen damit den Detaillierungen in Workflows auf einer höheren Abstraktionsebene und umfassen diese.

**Begriffsbestimmung**

Der Begriff des Geschäftsprozesses bezieht sich im Allgemeinen auf alle Aspekte des Prozesssystems, einschließlich Struktur- und Verhalten. Ein Workflow kann als Detaillierung von Prozessen auf operativer Ebene verstanden werden (Mertens et al. 2012, S. 75), um Teile von Prozessen und deren einzelne Aktivitäten weiter zu detaillieren. Er bezieht sich dabei auf die Ablaufsicht des Prozesses und erfasst anhand von zeitlich-logisch geordneten Arbeitsschritten oder Activities primär die Verhaltensaspekte des ursächlichen Systems. Activities können einzelnen personellen oder maschinellen Aufgabenträgern direkt zugeordnet werden, sodass die Workflow-Ebene als separate Abstraktionsebene von der Geschäftsprozessebene unterschieden werden kann.

Workflows weisen damit eine höhere Spezifität auf, die durch eine Detaillierung der in den zugehörigen Geschäftsprozessen definierten Aufgaben ausgedrückt werden kann (Ferstl und Sinz 2013, S. 103 f.). Die Ebenen werden in Abbildung 2.5 gezeigt.

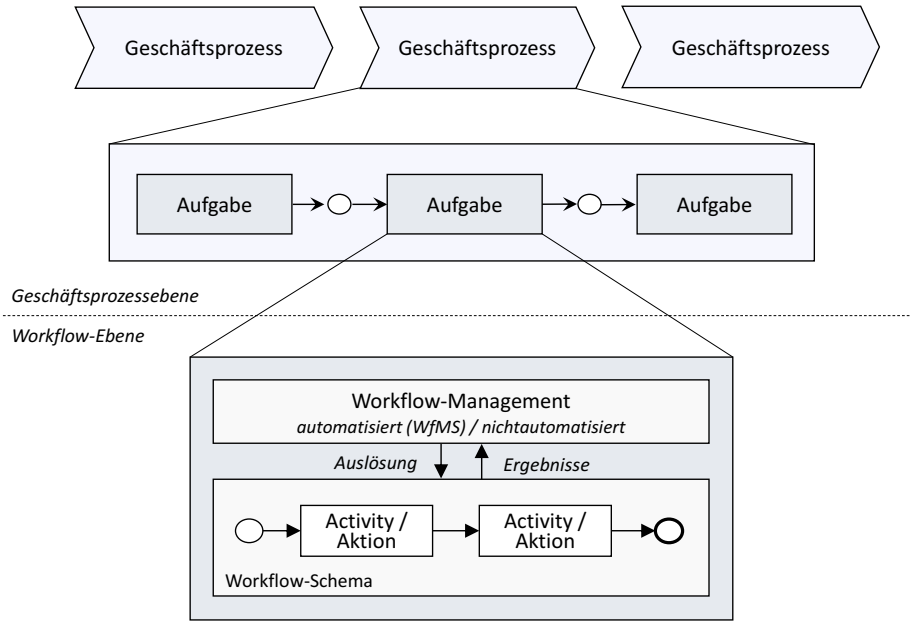



ABBILDUNG 2.5: Geschäftsprozess- und Workflow-Ebene

Zur Wahrung eines einheitlichen Begriffsverständnisses werden folgende Termini unterschieden:

- **Workflow**:
  Beschreibung einer zeitlich-logischen Abfolge aus einzelnen Aktionen oder Activities, die einen Vorgang bilden und als Vorgangs-Instanz ausgeführt werden.
- **Workflow-Schema** oder **Workflow-Definition**:
  Der als Schema vorliegende Workflow zur Realisierung der Aufgabenphase Planung. Das Schema liegt typischerweise in einem definierten Format vor, z.B. XPDL (WfMC 2012) oder BPMN (OMG 2014).
  - Nichtausführbare Workflow-Schemata dienen der Planung und ermöglichen zur Ausführungszeit eine nichtautomatisierte Ausrichtung an der Planung.
  - Ausführbare Workflow-Schemata erlauben eine Automatisierung des Workflows. Je nach Automatisierungsgrad wird die Auslösung von Aktionen oder auch die automatisierte Durchführung von Aktionen ermöglicht.

- **Workflow Management System (WfMS)**:
  Ein WfMS ist ein IT-System zur Instanziierung ausführbarer Workflow-Schemata. Der Begriff des Workflow-Managements bezieht sich auf die Auslösung und Kontrolle von Aktionen des Workflows. WfMS sind oft service-orientiert ausgelegt, d.h. sie exponieren eine aufrufbare Schnittstelle für das Workflow-Management, z.B. eine REST-Schnittstelle zur Annahme von GET-HTTP-Requests.
- **Workflow Engine**:
  Die Workflow Engine ist die zentrale Komponente der Architektur eines WfMS. Sie führt die Ausführung des Workflow-Managements durch.

**Standardisierung von Workflow-Technologien**

Eine wichtige Rolle bei der Standardisierung von Workflow-Technologien spielen die Workflow Management Coalition (WfMC) und die Object Management Group (OMG). Die meist industriellen Mitglieder der WfMC standardisieren seit 1993 Begriffe und Technologien, z.B. in Form eines Referenzmodells (WfMC 2006) und den Dateiformaten Wf-XML (WfMC 2004) und XPDL (WfMC 2012) für ausführbare Workflow-Schemata. Ein neueres Format zur Spezifikation von Workflow-Schemata standardisiert die OMG als Teil von BPMN 2.0. Das XML-basierte Format gewinnt zunehmend an Bedeutung (OMG 2014).

**Einordnung des Workflow-Begriffs**

Der Begriff des Workflows lässt sich zudem anhand des diskutierten Aufgabenmodells beschreiben. Die Darstellung von Aufgaben ist damit rein konzeptuell und betrachtet keine Lösungsverfahren, die zur Durchführung der Aufgabe in Form von einzelnen Aktionen ausgeführt werden müssen. Auf der Abstraktionsebene von Workflows erfolgt die detaillierte Spezifikation der Lösungsverfahren, welche die für die Durchführung notwendigen Aktionen der Aufgabe mit ihrer Ansteuerung angeben.

Hinsichtlich des objektorientierten Aufgabenmodells beschreibt ein Workflow die Innensicht einer Aufgabe anhand des Lösungsverfahrens (Ferstl und Sinz 2013, S. 103), bestehend aus Aktionensteuerung und Aktionen, die auf dem Aufgabenobjekt operieren. Die Aktionensteuerung entspricht dem Workflow-Management, das jede Aktion in Form einer Activity koordiniert. Ein Workflow-Schema enthält die Planung der Auslösung der Activities, sowie im Falle ausführbarer Workflow-Schemata eine Beschreibung der Activities in der Sprache einer Workflow-Engine.

**Automatisierungsgrade von Workflows**

Der Grad der Automatisierung einzelner Activities (1.), der Aktionensteuerung (2.) und der Auslösung des Workflows (3.) kann für (1.), (2.) und (3.) jeweils die Ausprägungen maschinell (m) und personell (p) besitzen (Ferstl und Sinz 2013, S. 113 f.).

Ein Workflow-System besitzt den Automatisierungsgrad (_, m, _) zur Automatisierung des Workflow-Managements, wobei keine Aussage über die Automatisierung der Aktionen oder der Auslösung des Workflows getroffen wird. Ein dokumentenorientierter Workflow, der die Bearbeitung von Dokumenten zwischen verschiedenen Aufgabenträgern koordiniert, besitzt die Ausprägung (p, m, _). Kommt eine Workflow-Engine zur Ausführung der Aktionen zum Einsatz, liegt (m, m, _) vor. Bei webbasierten und serviceorientierten Systemen wird meist die Auslösung in Form von (_, m, m) automatisiert, etwa durch den Aufruf eines Web-Service, z.B. mittels WS-BPEL oder BPMN (Ouyang et al. 2006), bei der ein Webserver eine Workflow-Engine aufruft, etwa zur Übergabe von Aufträgen oder für Berechnungen. Für die Beschreibung von Workflows kann beispielsweise BPMN 2.0 zum Einsatz kommen, welches im Falle von (m, m, m) eine Ausführungsbeschreibung im BPMN-eigenen XML-Format vorsieht.

## 2.2 Modellierung von Informationssystemen

Modelle von Informationssystemen erlauben die Beherrschung der bei der Entwicklung und der Ausführung von Prozessen erwachsenden Komplexität. So ergeben sich mit zunehmender Detaillierung eines Systems feingliedrige Komponenten, die zudem aus unterschiedlichen Perspektiven betrachtet werden können. Dieses Kapitel führt zunächst allgemeine Grundlagen zu Modellen und der Modellierung ein (Kapitel 2.2.1), geht auf die Modellierung von betrieblichen Systemen und Prozessen ein (Kapitel 2.2.2) und diskutiert die Modelltransformation (Kapitel 2.2.3). Diese Weiterführung und Anwendung dieser Themen führt zur objektorientierten Modellierung von Geschäftsprozessen anhand der SOM-Methodik (Kapitel 2.2.4) und zur Workflow-Modellierung unter Nutzung von BPMN 2.0 (Kapitel 2.2.5).

### 2.2.1 Grundlagen der Modellierung

#### 2.2.1.1 Modelltheorie und Modellbegriff

Eine Modellierung erfasst in der Realwelt vorhandene oder auch konstruierte Zusammenhänge in abstrakter Form als Modell. Das Begriffsverständnis beruht in der Wirtschaftsinformatik auf dem Abbildungscharakter der Modellierung, der in den folgenden beiden Definitionen zum Ausdruck kommt. Das anschließend betrachtete konstruktivistische Modellierungsverständnis bezieht zudem die subjektive Wahrnehmung während der Erstellung von Modellen ein.

Der allgemeinen Modelltheorie von Stachowiak folgend (Stachowiak 1973), lassen sich Modelle anhand von drei Merkmalen charakterisieren.

1. Das Abbildungsmerkmal beschreibt die Repräsentation eines Originals anhand des Modells. Ein Original muss dabei nicht notwendigerweise in der Realität existieren, sondern kann synthetisch oder als Planung verstanden werden.
2. Das Verkürzungsmerkmal bezieht sich auf die Abbildung in einer abstrakten Form derart, dass ein Original anhand des Modells verkürzt wiedergegeben wird. Das Original enthält präterierte Attribute, die nicht Teil des Modells sind. Die Abbildung führt zudem zu abundanten Attributen im Modell, die nicht Teil des Originals sind.
3. Das pragmatische Merkmal bezieht den Zweck oder das Ziel der Modellierung in die Abbildung und die Verkürzung ein. Letztere muss so gewählt

werden, dass sich das Modell für seinen vorgesehenen Zweck anstelle des Originals einsetzen lässt.

Die Definition stützt sich damit auf die Abbildung, die in allen Merkmalen wiederzufinden ist. Dem abbildungsorientierten Modellbegriff (Vom Brocke und Grob 2015, S. 10; Thomas 2005, S. 13 ff.) lässt sich auch die Definition des Modellbegriffs von Ferstl und Sinz (2013, S. 22) zuordnen.

Informal handelt es sich bei einem Modell um ein „System, das ein anderes System zielorientiert abbildet“ (Ferstl und Sinz 2013, S. 22). Das abgebildete System wird als Objektsystem bezeichnet, das anhand der Abbildung in ein Modellsystem überführt wird.

Formal kann ein Modell damit als 3-Tupel

$$M = (S_{\mathrm{O}}, S_{\mathrm{M}}, f)$$

mit dem Objektsystem $S_O$, dem Modellsystem $S_M$ und der Modellabbildung f definiert werden. Die Modellabbildung bildet die Systemkomponenten $V_O$ des Objektsystems auf Systemkomponenten $V_M$ des Modellsystems ab. Bei der Modellierung von Informationssystemen handelt es sich bei $S_O$ typischerweise um einen Ausschnitt eines realen betrieblichen Systems, der in Abhängigkeit des Ziels auf $S_M$ anhand einer homomorphen oder isomorphen Abbildung erstellt wird. Eine homomorphe Abbildung wird gefordert, wenn das Ziel die Anforderung der Strukturtreue nach sich zieht. Isomorphie wird gefordert, sofern Verhaltenstreue als Anforderung auf Basis des Ziels folgt.

Besteht das Ziel beispielsweise in der Implementierung von Geschäftsprozessen, können diese hinsichtlich ihrer betrieblichen Aufgaben in einem $S_M$ erfasst und damit anstelle von $S_O$ analysiert werden. Eine direkte Folge ist die bessere Beherrschbarkeit der Komplexität bei der Analyse von Systemen, da diese auf die in $S_M$ abgebildeten Merkmale reduziert wird. Für die Gestaltung von Systemen trifft dies ebenso zu, wobei hinzukommt, dass $S_M$ wiederum als Modell abgebildet werden kann. Die in diesem Fall vorliegende Modelltransformation bildet die Grundlage einer modellgetriebenen Entwicklung.

Für die Abbildung spielt keine Rolle, ob $S_O$ ein reales System ist. Die Definition umfasst deskriptive Abbildungen, wie im Falle von Informationssystemen beispielsweise die Erfassung von Ist- oder Soll-Zuständen einer Aufbauorganisation. Andererseits fällt auch die präskriptive Erfassung zur zukünftigen Konstruktion eines Originals unter die Definition, etwa bei der Erstellung von Modellen für den

Software-Entwurf eines zu entwickelnden Anwendungssystems.

#### 2.2.1.2 Konstruktivistischer Modellbegriff

Das konstruktivistische Modellierungsverständnis geht auf den radikalen Konstruktivismus (Glasersfeld 1995) zurück. Diese Art des Konstruktivismus bezeichnet die erkenntnistheoretische Auffassung einer nicht objektiv abbildbaren Realität, die durch den Menschen subjektiv als Konstruktion seiner selbst wahrgenommen wird. Aufbauend auf dem bisherigen Verständnis des Modellbegriffs bezieht ein konstruktivistisches Begriffsverständnis die Subjektivität bei der Konstruktion von Modellen mit ein (Thomas 2005, S. 17 ff.; Ferstl und Sinz 2013, S. 135 f.). Die folgenden Merkmale charakterisieren das Begriffsverständnis:

- Perzeption: Der Ersteller eines Modells, das Subjekt, nimmt das abzubildende Objektsystem über die Systemgrenzen hinaus als Teil der Realität wahr.
- Interpretation: Das Subjekt interpretiert das Objektsystem basierend auf der Perzeption und steht dabei in einer Kontextbeziehung zur Realität.
- Konstruktion: In der Kontextbeziehung zur Realität formuliert das Subjekt Modellziele hinsichtlich der Realität und führt basierend auf Interpretation und Perzeption eine Konstruktion des Modellsystems durch.

Dem Abbildungscharakter der vorher genannten Definitionen steht dies nicht direkt entgegen, wobei die Erstellung der Modellabbildung hier indirekt anhand des Subjekts charakterisiert wird.

#### 2.2.1.3 Metamodellierung

Anhand eines Metamodells werden Modell-Elemente und Regeln zur Herstellung von Beziehungen zwischen diesen zusammen mit einem Begriffssystem beschrieben (Ferstl und Sinz 2013, S. 137), um damit Modelle zu gestalten (Vom Brocke und Grob 2015, S. 83). Ein Metamodell legt damit die zur Konstruktion von Modellen zur Verfügung stehende Syntax sowie eine inhärente Semantik fest. Die anhand des Metamodells erstellten Modelle stehen in Extensionsbeziehungen zum Metamodell, d.h., sie erweitern die vorgegebenen Meta-Elemente um konkrete Modellelemente unter Einhaltung der vorgegebenen Meta-Beziehungen. Im Sinne der Objektorientierung ist jedes Element eines Modells anhand des jeweils erweiterten Meta-Elementes typisiert.

Die Verwendung des Begriffs kann rekursiv erfolgen und mehrere Modell-Ebenen unterscheiden. Verbreitet ist eine Gliederung in vier Ebenen (Sinz 1996, S. 123-143;

OMG 2016b) als Meta-Metaebene (Ebene 3), Metaebene (Ebene 2), Schemaebene oder Modellebene (Ebene 1) sowie Ausprägungs- oder Instanzebene (Ebene 0). In der Terminologie der OMG besitzt die Ziffer ein M als Präfix.

Die Meta-Metaebene erlaubt eine Extension des dort definierten Modells anhand von Metamodellen für die Definition einer Sprache, die zur Formulierung von Metamodellen genutzt wird. Das Meta-Metamodell macht damit syntaktische und semantische Vorgaben zur Definition von Metamodellen. Es stellt selbst keine Extension dar und definiert sich anhand der inhärenten Semantik eigenständig. Das hier zugrunde liegende Meta-Metamodell besitzt die in Abbildung 2.6 dargestellten Meta-Objekte und Meta-Beziehungen, sowie die spezialisierten Meta-Beziehungen *has*, *connects* und *is_a*, die als Basistypen zur Verfügung stehen (Sinz 1996, S. 129). Meta-Beziehungen verbinden hier stets zwei Meta-Objekte und besitzen zwei Kardinalitätsangaben. Die generische Abbildung von Beziehungen wird unterstützt durch die Meta-Beziehung *has*, für Attribut-Zuordnungen, *connects*, für miteinander assoziierte Objekte sowie *is_a* zur Angabe einer Generalisierung zwischen zwei Objekten. Die Metaebene formuliert anhand von diesen Ausdrucksmitteln eine Modellierungssprache, bei der es sich im Sinne der konzeptuellen Modellierung um eine domänenspezifische Modellierungssprache zur Beschreibung von Konzepten handelt. Die Schemaebene umfasst zugehörige Modelle oder Schemata. Das erfasste Schema, beispielsweise ein Prozess, betrifft hinsichtlich des Zeitbezugs die Gestaltungszeit. Auf Ausprägungsebene liegen die zur Laufzeit des Modells entstehenden Daten vor; in Abhängigkeit der Domäne beispielsweise Tupel eines Datenschemas oder Ausführungsergebnisse einzelner Vorgänge eines Workflow-Modells.

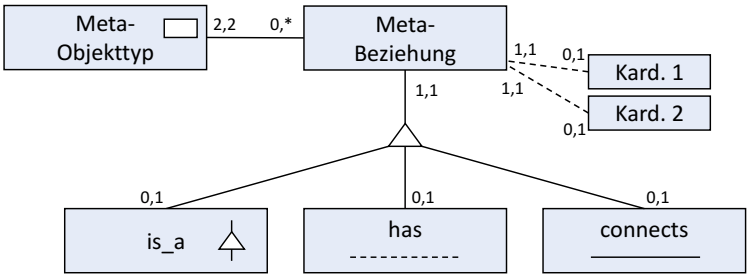



ABBILDUNG 2.6: Modell der Meta-Metaebene (nach Sinz (1996, S. 129))

Der Begriff der Konformität (OMG 2003) beschreibt in diesem Zusammenhang die Validität der Erweiterung eines Metamodells, sodass bei einer validen Erweiterung

eine Beziehung „konform zu“ zwischen Modell und Metamodell besteht. Die Validität ist gegeben, sofern die syntaktischen und semantischen Vorgaben des Metamodells eingehalten werden. Dies ist der Fall, sofern die definierten Modell-Elemente regelkonform in Beziehung stehen und der anhand des Begriffssystems inhärent definierten Semantik folgen.

#### 2.2.1.4 Modellierungsansätze und Modellierungsmethodiken

Ansätze und Methodiken zur Modellierung gliedern zusammengehörige Modelle im Kontext eines konzeptuellen Rahmens. Die Zusammengehörigkeit definiert sich anhand der pragmatischen Merkmale einzelner Modelle, die hinsichtlich ihrer Modellierungszwecke Relevanz im Kontext des Rahmens besitzen. Als Strukturierungsmittel zur Gliederung von Modellen und Abstraktionsebenen werden beispielsweise Modellsichten und Modellebenen (Ferstl und Sinz 2013, S. 142) herangezogen.

Diese Strukturierungsmittel werden zudem für die generische Beschreibung von Informationssystemarchitekturen innerhalb des generischen Architekturrahmens herangezogen (Sinz 2002, S. 1056 ff.). Anhand der Struktur definierte Modelle werden anhand von existierenden oder neu geschaffenen Modellierungssprachen beschrieben, beispielsweise unter Nutzung von Metamodellen.

Über die genannten Merkmale hinaus beziehen Methodiken gegenüber Ansätzen zusätzlich Aspekte des methodischen Vorgehens zur Erstellung von Modellen mit ein, die auf ein Architekturmodell Bezug nehmen. Eine mögliche Definition kann anhand eines Vorgehensmodells erfolgen, bei dem das Vorgehen Gegenstand einer Modellierung ist (Sinz 2016). Neben diesem Artefakt kann die Konzeptualisierung der Realwelt durch eine Metapher unterstützt werden (Sinz 2016). Karagiannis und Kühn (2002) führen unter dem Begriff der Modeling Method ebenso Aspekte des Vorgehens als „Modeling Procedure“ und, zusammen mit der „Modeling Language“, als „Modeling Technique“. Zudem werden „Mechanisms & Algorithms“ einbezogen. Das Begriffsverständnis bezieht sich damit primär auf Aspekte der Konstruktion ausgehend von Modellierungssprachen.

Ein Beispiel zur Vorgehensdefinition ist das V-Modell der SOM-Methodik (Ferstl und Sinz 2013, S. 197 f.), welches ein parallelisiertes Vorgehen entlang von Struktur- und Verhaltenssichten über verschiedene Modellebenen der Methodik vorgibt. Die hiermit gegebenen Strukturierungsmittel Modellsicht und Modellebene werden im Folgenden erläutert.

**Modellsichten**

Eine Sicht erlaubt im Kontext der Informatik die Betrachtung verschiedener Aspekte eines Datenbestandes. Zugrunde liegende Daten sind dabei nicht Teil der Sicht, sodass im Falle mehrerer Sichten unterschiedliche Repräsentationen einer gemeinsamen Basis bestehen. Änderungen wirken sich daher unmittelbar auf alle Sichten aus und führen nicht zu Inkonsistenzen. Beispiele sind im Software Engineering zu finden, etwa das Model-View-Controller-Pattern (Balzert 2011, S. 62 ff.) zur Separation einer oder mehrerer grafischer Oberflächen vom Programmzustand, oder im Bereich relationaler Datenbanken bei der Definition von Views (Kemper und Eickler 2015, S. 139).

Das Verständnis einer Sicht als Projektion auf zugrunde liegende Ausprägungen ist auf die Modellierung unmittelbar übertragbar. So können die Sichten eines Modells unter Bezugnahme auf das zugehörige Metamodell formal als Projektion definiert werden (Sinz 2002, S. 1056 ff.), die anhand von Meta-Objekten und Meta-Beziehungen parametrisiert wird. Bei Anwendung der Projektion auf der Ausprägungsebene bleiben diejenigen Elemente und Beziehungen sichtbar, welche die bei der Definition angegebenen Meta-Objekte und Meta-Beziehungen erweitern. Unterschiedliche Sichten auf ein Modell stellen damit verschiedene Aspekte des Modells dar, beispielsweise Struktur- oder Verhaltensmerkmale. Dies impliziert einerseits eine assoziative Beziehung zwischen Modell und Sicht, und andererseits ein Verständnis einer nicht-materialisierten Sicht, die selbst keine Ausprägungen enthält.

Der Begriff der Sicht wird neben der Begriffsverwendung im Sinne der projektionsbasierten Definition mitunter auch im Sinne einer partitionsbasierten Definition genutzt. Sichten entstehen in diesem Zusammenhang durch Bildung verschiedener Teilmodelle als Ergebnis einer Zerlegung des Modells (Allweyer 2005, S. 140 ff.). Unterschiedliche Sichten stellen auch hier verschiedene Aspekte des zugrunde liegenden Modells dar, allerdings wird zwischen Modell und Sicht eine aggregierende Beziehung (is_part_of-Beziehung) impliziert sowie ein Verständnis einer materialisierten Sicht, die Ausprägungen vorhält. Infolge dessen können Inkonsistenzen entstehen, da die Ausprägungen unterschiedlicher Sichten zur Wahrung der Konsistenz untereinander abgeglichen werden müssen (Karagiannis, Mayr et al. 2016), sofern Änderungen einer Sicht die Ausprägungen einer anderen beeinflussen. Zudem setzt eine partitionsbasierte Sichtenbildung keine Sichtendefinition auf der Metaebene voraus, sodass Sichten nicht unabhängig von Modellen der Ausprägungsebene definiert werden. Im Weiteren wird daher ein projektionsbasiertes Begriffsverständnis unterstellt.

**Modellebenen**

Modellebenen erlauben eine Unterscheidung unterschiedlicher Abstraktionsebenen, die anhand von separaten Modellen beschrieben werden. Zwischen Modellebenen können Beziehungen definiert werden, die Relationen zwischen Modellen oder Modellelementen der Ebenen entsprechen. Dem sprachbasierten Modellbegriff folgend, handelt es sich bei den Ebenen um Beschreibungsebenen, die das abzubildende Objektsystem dem Abstraktionsgrad nach gliedern. Die Modelldefinition kann je Modellebene in Syntax und Semantik beispielsweise anhand eines Metamodells erfolgen. Die Definition von Beziehungen kann in diesem Fall anhand eines Beziehungsmetamodells angegeben werden (Sinz 2002, S. 1056). Dies erlaubt die Angabe von Beziehungen unabhängig von konkreten Ausprägungen der Modellebene und gibt Hinweise auf die Transformation von Modellen zwischen den betrachteten Ebenen. In bestimmten Fällen werden Transformationen anhand des Beziehungsmetamodells syntaktisch vollständig definiert.

Die Modellebenen konkreter Ansätze und Methodiken gliedern Informationssystemarchitekturen beispielsweise in fachliche und technische Ebenen (Ferstl und Sinz 1995; Scheer et al. 2005). Fachliche Ebenen unterscheiden z.B. Geschäftsprozesse hinsichtlich ihrer betrieblichen Aufgaben und ihrer Aufgabenträger. Technische Ebenen umfassen z.B. softwaretechnische Modelle und Quellcode. Die Abstimmung zwischen Ebenen kann anhand von Modellen unterstützt werden, beispielsweise für das Business-IT-Alignment.

#### 2.2.1.5 Modellierungssprachen

Der Begriff der Modellierungssprache nimmt auf den sprachbasierten Modellbegriff Bezug (Strahringer 1998, S. 3). Der Aufbau von Modellen wird damit analog zum Aufbau von Sprachen anhand der Begriffe Syntax und Semantik beschrieben. Syntaktische und semantische Merkmale definieren Sprachen hinsichtlich ihres elementaren Aufbaus und ihrer Bedeutungszusammenhänge. Der vorliegende Abschnitt beschreibt die Begriffe Syntax, Notation und Semantik im Kontext der Modellierung vor dem Hintergrund der Formalisierungsgrade von Modellierungssprachen.

**Formalisierungsgrade von Modellierungssprachen**

Hinsichtlich der Formalisierungsgrade von Modellierungssprachen können die damit beschreibbaren Modellabbildungen als informal, formal oder semi-formal klassifiziert werden (Vom Brocke und Grob 2015, S. 66; Bork und Fill 2014, S. 3401 f.; Ferstl und Sinz 2013, S. 143).

- Formale Modelle sind als formale Sprachen hinsichtlich ihrer Syntax und Semantik formal definiert. Sie können eine textuelle oder auch eine grafische Notation besitzen.
- Semi-formale Modelle sind hinsichtlich ihrer Semantik nicht formal definiert. Die Syntax kann teilweise oder vollständig formalisiert vorliegen. Sie wird zur Beschreibung formulierbarer Modelle insoweit formalisiert, als es für die Erstellung von Modellen erforderlich ist – beispielsweise anhand eines Metamodells zur Beschreibung einer grafischen Notation.
- Informale Modelle umfassen textuelle Beschreibungen oder Skizzen, denen keine formale Syntax oder Semantik zugrunde liegt.

Syntax, Notation und Semantik werden im Folgenden hinsichtlich der Formalisierungsgrade genauer erläutert.

**Syntax und Notation**

Die Syntax gibt anhand von Syntax-Elementen und Syntax-Regeln an, welche Kombinationen von Elementen unter Anwendung der Regeln zur Formulierung von Modellen herangezogen werden können. Wird zwischen abstrakter und konkreter Syntax unterschieden, entspricht die erläuterte Syntaxdefinition einer abstrakten Syntax, mit der die in einer Sprache formulierbaren Modelle beschrieben werden, während die konkrete Syntax zur Formulierung von einzelnen Modellen herangezogen wird. Die abstrakte Syntax beschreibt damit ein Metamodell und die konkrete Syntax ein einzelnes Modell.

Formale Modelle können rein textuell notiert sein, wobei Elemente als Wörter definiert sind, die anhand von Regeln in Form einer Grammatik zu Sätzen zusammengesetzt werden. Ein Beispiel sind Spezifikationen für Software, die in der Backus Naur Form (BNF) oder einer ihrer Erweiterungen formuliert sind (McCracken und Reilly 2003). Zudem kann eine grafische Notation für die Syntax definiert werden. Dies ist beispielsweise bei Petri-Netzen der Fall (Reisig 2010), die hinsichtlich ihrer Syntax rein formal definierbar sind, aber neben der Syntax für sämtliche Syntax-Elemente Stelle, Transition, Kante, Kantengewicht und Markierung jeweils ein grafisches Notationselement besitzen.

Semi-formale Modelle können in Syntax und Notation rein grafisch definiert werden, wobei die Definition von Syntax und Notation unabhängig oder zusammenhängend erfolgen kann. Syntax und Notation können beispielsweise in einem grafischen Metamodell festgelegt werden, das die Elemente und Regeln anhand von

grafischen Symbolen und Beziehungen zwischen diesen festlegt (Ferstl und Sinz 2013, S. 143). Semi-formale Modelle können bei der Modellierung komplexer Systeme von Vorteil sein, da die formale Erfassung aller Komponenten eines Systems oft nicht beherrschbar ist. Der Umfang der Formalisierung lässt sich somit anhand des Ziels der Modellierung festlegen. Semi-formale Modelle oder Spezifikationen sind u.a. in der Datenmodellierung, Unternehmensmodellierung, der Modellierung von Geschäftsprozessen (SOM) und Workflows (BPMN), aber auch in der Software-Modellierung (UML) verbreitet.

Informale Modelle sind beispielsweise im Requirements Engineering verbreitet. Anforderungsdefinitionen liegen zumeist erst in natürlicher Sprache vor (siehe z.B. Sommerville (2011, S. 95)), die zwar sprachlichen Konventionen folgen, aber manuell in semi-formale oder z.T. auch formale Spezifikationen überführt werden müssen. Hierfür werden formale Sprachen zur Spezifikation von Software entwickelt, z.B. Z in ISO/IEC 13568:2002 (ISO 2002) oder TLA+ (Lamport 2002).

Innerhalb des formal definierten Teils einer Sprache können Beweise anhand von formalen Verifikationen geführt werden, beispielsweise durch Model Checking (Clarke et al. 1999). Verifikationen sind begrenzt auf das formale System der Sprache und erlauben beispielsweise Aussagen über erreichbare oder nicht erreichbare Zustände des Systems, etwa bei Petri-Netzen. Sie erlauben keine Aussagen über semantische Aspekte außerhalb des definierten Systems, etwa über die inhärente Semantik, die sich aus den Bezeichnungen von Stellen und Transitionen eines Petri-Netzes ergibt.

**Semantik**

Die Semantik definiert die Bedeutung von syntaktisch korrekt formulierten Modellen. Um ein gemeinsames Verständnis der Konzepte einer Domäne sicherzustellen, können die Konzepte der Domäne hinsichtlich ihrer Semantik definiert werden (Harel und Rumpe 2004). Eine semantische Domäne (Semantic Domain) beschreibt das gemeinsame Bedeutungsverständnis, welches den beteiligten Akteuren zugrunde liegt. Dieses ist eine Grundvoraussetzung, um Konzepte der Domäne formal auszudrücken.

Syntaktische Elemente einer Modellierungssprache lassen sich Domänen-Konzepten, den semantischen Konzepten einer Domäne, zuordnen. Jedes Syntax-Element, das unter Anwendung von syntaktischen Regeln zur Modellierung herangezogen werden kann, wird einem Domänen-Konzept zugeordnet. Nach Harel und Rumpe (2004) kann diese Zuordnung einer Sprach-Syntax L zu

einer semantischen Beschreibung der Domäne S (Semantic Mapping) als Abbildung definiert werden:

$$M : L \rightarrow S$$

Die formale Spezifikation von S hängt auch von der Beschreibungsform der Syntax ab und ist daher nicht immer einheitlich formulierbar. Verbreitete Typen von Beschreibungsformen sind operationale, denotationale und axiomatische Beschreibungsformen (H. R. Nielson und F. Nielson 2007, S. 19 ff., 91 ff., 205 ff.). Im Falle von Petri-Netzen kann eine operationale Semantik festgelegt werden, welche die Voraussetzungen für das Schalten von Transitionen und die damit verbundene neue Zuordnung von Markierungen zu Stellen angibt. Die operationale Semantik legt fest, welche Berechnungen, Einzelschritte oder Zustände bei der Ausführung des Modells ablaufen. Eine denotationale Semantik gibt eine mathematische Modellierung des Ergebnisses der Ausführung an, ohne darauf einzugehen, über welche Zwischenschritte das Ergebnis erreicht wird. Eine axiomatische Semantik beschreibt zur Laufzeit auftretende Eigenschaften als Assertions, wobei einzelne Eigenschaften herausgegriffen werden und nicht notwendigerweise eine vollständige Beschreibung vorliegt.

### 2.2.2 Modellierung von betrieblichen Systemen und Geschäftsprozessen

#### 2.2.2.1 Konzeptuelle Modellierung

Aus dem abbildungsorientierten Modellbegriff ergibt sich, dass Modelle als „zweckgerichtet konstruierte Abstraktionen“ (Frank et al. 2014, S. 49) die Ergebnisse einer durch den Modellierer erbrachten Abstraktionsleistung sind. Die Leistung besteht darin, abzubildende Objekte hinsichtlich ihrer für den Modellierungszweck wesentlichen Eigenschaften zu erfassen. Die Erfassung von Domänen-Konzepten eines Fachbereichs ist Gegenstand der konzeptuellen Modellierung, die in Abhängigkeit des Modellierungszwecks wesentliche Eigenschaften bedeutender Konzepte der Domäne in Form von Modellen erfasst. Die Realwelt der betrachteten Domäne ist damit der Gegenstand der Modellierung. Eine Modellierungssprache hilft dabei, die Realwelt zu erfassen und als Modell zu formulieren (Sinz 2019). Für die konzeptuelle Modellierung werden häufig semi-formale Modellierungssprachen mit grafischer Notation herangezogen (Frank et al. 2014, S. 49 ff.; Ferstl und Sinz 2013, S.

143; Fill und Karagiannis 2013, S. 11). Im Gegensatz zu formalen Modellierungssprachen sind die Abstraktion und die Präzision der Modellabbildung dabei nicht an den formalen Regeln einer Modellierungssprache orientiert, sondern an den jeweiligen Domänenkonzepten. Der Begriff der domänenspezifischen Modellierungssprache bezeichnet eine Sprache, die für die konzeptuelle Modellierung einer bestimmten Domäne ausgelegt ist (Frank 2013). Im Gegensatz dazu sind Sprachen für das „General-Purpose Modeling" domänenunabhängig und verwenden keine Konstrukte der Domäne zur Formulierung von Modellen. Unter den domänenspezifischen Begriff fallen Sprachen, die Domänen-Konzepte in ihren Elementen und Beziehungen abbilden. Modelle der Domäne werden innerhalb des Begriffsverständnisses der Domäne formuliert. Dies erlaubt die Repräsentation von komplexen Modellen auf einer gezielt ausgewählten Abstraktionsebene, die in Abhängigkeit von der Domäne, den Anforderungen und den beteiligten Akteuren festgelegt werden kann. Beteiligte Akteure sind insbesondere Fachanwender, Domänen-Experten, die durch die Ausdrückbarkeit von Modellen anhand von Domänen-Konzepten in die Lage versetzt werden, Modelle zu analysieren und zu gestalten. Damit wird das Ziel verfolgt, komplexe Sachverhalte in beherrschbarer Form mit Begriffen der Domäne abzubilden. Domänenspezifische Sprachen definieren sich über die Spezifität der abgebildeten Domäne. Im weiteren Sinne fallen Sprachen für die konzeptuelle Modellierung darunter, etwa ER- oder SER-Diagramme in der Domäne des Datenmanagements, und SOM- oder BPMN-Modelle in der Domäne des Geschäftsprozessmanagements (Karagiannis, Mayr et al. 2016, S. 4; Ferstl und Sinz 2013, S. 143 ff., S. 189 ff.). Im engeren Sinne liegt eine domänenspezifische Sprache nur vor, wenn die Spezifität hinsichtlich der Domäne derart hoch ist, dass die Konstrukte der Sprache in Syntax und Semantik direkten Bezug auf eine fachliche Anwendung nehmen (Karagiannis, Mayr et al. 2016). Eine Sprache muss damit für eine spezifische fachliche Anwendung konzipiert werden. Für die Konzeptualisierung domänenspezifischer Modellierungssprachen kann die Metamodellierung herangezogen werden. Metamodelle definieren in diesem Fall die fachlichen Sprachkonstrukte, anhand derer domänenspezifische Modelle formuliert werden können.

#### 2.2.2.2 Ansätze und Methodiken

Ansätze zur Analyse und Gestaltung von Informationssystemen in Organisationen sind im Bereich der Modellierung betrieblicher Informationssysteme, der Unternehmensmodellierung (Enterprise Modeling) sowie im Speziellen in Ansätzen und Methodiken zur Geschäftsprozess- und Workflow-Modellierung zu finden.

Die Unternehmensmodellierung betrachtet die Gesamtheit einer Organisation aus verschiedenen Perspektiven und setzt diese zueinander in Relation (Frank et al. 2014; Sandkuhl et al. 2013). Die Betrachtung kann einzelne Perspektiven separat erfassen und verknüpfen, oder integrativ als Teil einer Unternehmensarchitektur (Enterprise Architecture) einordnen. Beispiele abzubildender Perspektiven sind die Organisation, die Geschäftsprozesse, die Produkte und Dienstleistungen oder die IT- und Anwendungssysteme einer Organisation.

Zur Modellierung betrieblicher Informationssysteme sind eigenständige Ansätze und Methodiken verbreitet. Diese stellen einen konzeptuellen Rahmen zur Erfassung betrieblicher Systeme zur Verfügung, der zur Abbildung des Systems Modellierungssprachen einsetzt. Methodiken umfassen weitere Komponenten, anhand derer das Vorgehen zur Anwendung von Modellierungssprachen innerhalb des vorgesehenen Rahmens definiert wird. Im weiteren Sinne relevant sind zudem Rahmenwerke. Der Begriff des Rahmenwerks (Framework) bezeichnet beliebige Strukturierungshilfen, anhand derer Unternehmen mit ihren Architekturen abgebildet werden. Rahmenwerke sind nicht notwendigerweise auf die Modellierung ausgerichtet.

- Beispiele für Ansätze und Methodiken sind das Semantische Objektmodell (SOM) (Ferstl und Sinz 1990), die Architektur integrierter Informationssysteme (ARIS) (Scheer 1991), die Multiperspektivische Unternehmensmodellierung mit der Modellierungskomponente Multi Purpose Enterprise Modelling (MEMO) (Frank 1994) und der zum Business Engineering gehörende Ansatz PROMET (Österle 1995).
- Beispiele für Rahmenwerke sind ArchiMate (The Open Group 2017), GERAM (Bernus und Nemetz 1994), TOGAF (The Open Group 2018) und das Zachman EA Framework (Matthes 2011, 210 ff.).

Modellierungssprachen bilden Struktur- und Verhaltensaspekte von betrieblichen Systemen beispielsweise in Prozess- und Workflow-Modellen, fallbasierten Modellen, Anwendungssystemspezifikationen und Datenschemata ab. Ihr Einsatz kann als Bestandteil von Ansätzen und Methodiken oder unabhängig davon stattfinden. Ein unabhängiger Einsatz im Rahmen der Modellierung von beliebigen Systemen und Prozessen bezieht auch Sprachen ein, die hinsichtlich ihrer Syntax und Notation definiert sind, ohne semantisch Bezug auf eine über das Diagramm hinausgehende Unternehmensabbildung zu nehmen. Modellierungssprachen besitzen nicht notwendigerweise die Merkmale eines Ansatzes oder einer Methodik, d.h. die bei

einem Ansatz vorhandene strukturierte Einordnung in einen konzeptuellen Rahmen und die im Falle einer Methodik vorausgesetzte Ausrichtung an einem Vorgehensmodell werden nicht notwendigerweise berücksichtigt.

- Beispiele zur Prozessmodellierung sind Petri-Netze (Brauer und Reisig 1996; Reisig 2010) und interorganizational Workflows (van der Aalst 2000; van der Aalst und Weske 2001), ereignisgesteuerte Prozessketten (EPK) (Keller et al. 1992) und erweiterte EPK (eEPK) (Gadatsch 2015, S. 18), Interaktions- und Vorgangsereignisschemata (IAS bzw. VES) (Ferstl und Sinz 2013, S. 198 ff.), Aktivitätsdiagramme der Unified Modeling Language (UML) 2.5 (OMG 2017b) und Process- und Collaboration-Diagramme der Business Process Model and Notation (BPMN) 2.0 (OMG 2014). Letztere kann Bezug auf deklarativ modellierte Geschäftsregeln der Decision Model and Notation (DMN) 1.1 (OMG 2019) nehmen.
- Beispiele zur fallbasierten Modellierung bilden deklarative Ansätze wie die Case Management Model and Notation (CMMN) (OMG 2016a), die Geschäftsvorfälle schematisiert erfassen, und hybride Ansätze, die miteinander kombinierte Prozessfragmente auf neue Vorfälle anwenden (Hewelt und Weske 2016). Die Ansätze sind dem Adaptive Case Management (ACM) zuzuordnen.
- Beispiele zur Modellierung von Anwendungssystemen sind das Konzeptuelle Objektschema und das Vorgangsobjektschema (KOS bzw. VOS) (Ferstl und Sinz 2013, S. 223 ff.) auf fachlicher Ebene sowie statische und dynamische Diagramme der UML auf technischer Ebene (OMG 2017b), wie etwa Klassen- bzw. Sequenzdiagramme.
- Beispiele zur Datenmodellierung sind das Entity-Relationship-Modell (ERM) (Chen 1976), das Strukturierte Entity-Relationship-Modell (SERM) (Sinz 1988) sowie zur objektorientierten Datenmodellierung die auf der UML basierende Martin-Notation (Krähenfußnotation) (Martin und Odell 1992) und das konzeptuelle Objektschema (KOS) (Ferstl und Sinz 2013, S. 223 ff.).

Die für die Modellierung betrieblicher Informationssysteme im engeren Sinne relevanten Ansätze und Methodiken enthalten einen konzeptuellen Rahmen zur Erfassung der Unternehmensarchitektur anhand von organisationalen Strukturen und

Geschäftsprozessen. Unterschiede bestehen hinsichtlich der Definition der Modellierungssprachen sowie der Integration der Sprachen mit dem konzeptuellen Rahmen. Die Definition hinsichtlich der Syntax, Semantik und Notation kann beispielsweise anhand von Metamodellen erfolgen. Der Grad der Integration steigt notwendigerweise mit zunehmender Spezifität der Sprachendefinition an.

SOM definiert eine Unternehmensarchitektur, die von einem Unternehmensplan ausgehend Geschäftsprozesse und Ressourcen ableitet (Ferstl und Sinz 2013). Die Modellabbildung von Geschäftsprozessen erfasst betriebliche Aufgaben im Sinne des objektorientierten Aufgabenmodells in struktur- und verhaltensorientierten Sichten einer Aufgaben-Modellebene. Aus diesen leiten sich objektorientierte Modelle von Anwendungssystemen ab, die als Ressourcen die separate Modellebene der Aufgabenträger bilden. Metamodelle definieren die Modellierungssprachen in beiden Modellebenen. SOM wird aufgrund der Sprachdefinition als objektorientiert sowie aufgrund der von den Geschäftsprozessen ausgehenden Modellierung als prozessorientiert und modellgetrieben charakterisiert.

ARIS sieht eine Gliederung in Aspekte der Organisation, Steuerung, Funktion und Daten vor. Zur Geschäftsprozessmodellierung werden v.a. EPK und BPMN 2.0 eingesetzt (Scheer 1991; Seidlmeier 2015, S. 159). Die Syntax der EPK und der eEPK bilden Prozesse ablauforientiert ab, indem Ereignis- und Funktionselemente über Kontrollflüsse und Konnektoren miteinander verknüpft werden. Die erweiterte Syntax sieht neben Organisationseinheiten u.a. Informationsobjekte zur Datenerfassung vor. Die verwendete BPMN-Syntax umfasst v.a. Elemente der Process Diagrams (OMG 2014), die Diagramme anhand von Swimlanes partitionieren und Abläufe durch Activity- und Sequence-Flow-Elemente angeben. BPMN kann als Workflow-Sprache charakterisiert werden (siehe Kapitel 2.1.4.2). Der ARIS-Ansatz verwendet mit eEPK und BPMN ablauforientierte Diagramme und ist inhärent daten- und funktionsorientiert.

MEMO wird hinsichtlich der zugrunde liegenden Modellierungssprachen anhand der MOF und der UML unter Nutzung eines Meta-Metamodells definiert (Frank 2011, S. 34; Rohloff 2009, S. 91 ff.). Es sind Möglichkeiten zur methodischen Integration von Modellierungssprachen vorgesehen, die als Erweiterungen des Meta-Metamodells in eine Spracharchitektur eingeordnet werden. Aufgrund der freien Definition der Meta-Meta-Ebene ist eine Festlegung auf bestimmte Modellierungssprachen nicht gegeben. Die objektorientierte Definition des Meta-Metamodells mit Mitteln der UML erlaubt die Implementierung objektorientierter Sprachen auf der Meta-Ebene, ohne diese zu bedingen.

PROMET nutzt zur Darstellung von Geschäftsprozessen Netze und Schemata, die Zusammenhänge und Gewichte abbilden; ein Metamodell definiert grundlegende Strukturen (Österle 1995; A. Bach 2009, S. 270; Scheer 1998, S. 19). Das Metamodell definiert Begriffszusammenhänge konzeptuell im Sinne eines semantischen Netzes, ohne auf die Syntax einer Modellierungssprache Bezug zu nehmen. Prozesse entsprechen hier Aggregationen von Funktionen, die zueinander in sequenziellen und parallelen Ablaufbeziehungen stehen. Ein Aufgabenkettendiagramm (Gadatsch 2012, S. 81) setzt Aufgaben und Ereignisse über sequenzielle und parallele Kontrollflüsse in Beziehung und sieht z.B. „Datenbankzugriff“ als Syntaxelement vor. Nachrichtenflüsse und eine objektorientierte Typisierung und Kapselung der Funktionen sind nicht Teil des daten- und funktionsorientierten Ansatzes.

#### 2.2.2.3 Modellebenen zur Modellierung von Geschäftsprozessen und Workflows

Der Begriff der Geschäftsprozessmodellierung wird in den besprochenen Ansätzen und Methodiken als allgemeine Bezeichnung zur Erstellung von Modellen interpretiert, welche die Leistungserstellung von Prozessen in betrieblichen Systemen abbilden. Hierzu wird innerhalb einer Abstraktionsebene das Verhalten des Systems hinsichtlich der ablaufenden Vorgänge und, im Falle von SOM, zudem die Struktur abgebildet. Eine zunehmende Detaillierung der Prozessmodelle zieht aufgrund der steigenden Komplexität Bedarf nach einer zweiten Abstraktionsebene nach sich, welche die Modellierung einzelner Vorgänge der Geschäftsprozessebene in Form von Workflows erlaubt. Die Workflow-Ebene detailliert die aus betrieblichen Aufgaben resultierenden Vorgänge so weit, dass deren einzelne Aktionen jeweils einem Aufgabenträger als elementare Aktion zu Durchführung übergeben werden können (Pütz und Sinz 2010). Die damit definierten Workflows lassen sich in Abhängigkeit der bestehenden Aufgaben automatisiert auslösen oder ausführen (Kapitel 2.1.4.2).

Neben der Geschäftsprozess- und Workflowebene können weitere Abstraktionsebenen unterschieden werden. Jablonski differenziert zwischen Prozesslandkarten, Prozessen, Workflows und technischen Abläufen (Jablonski et al. 1997). Prozesslandkarten (F. Bayer und Kühn 2013, S. 37 ff.) bilden eine Zusammenfassung der für die Leistungserstellung relevanten Prozesse ab, beispielsweise gegliedert nach Haupt- und Unterstützungsprozessen. Zudem werden Relationen zwischen Prozessen definiert, die Abhängigkeiten oder Ablaufbeziehungen angeben. Anschließend an die Definition der Geschäftsprozesse und Workflows beschreiben technische Abläufe im Sinne von Jablonski die technische Implementierung von Workflows, z.B.

als Programm, das mit einem Workflow-Management-System ausführbar ist (Jablonski et al. 1997, S. 196; Ferstl und Sinz 2013, S. 102 f.).

Aus der vorgestellten Untergliederung sowie aus der Diskussion um Automatisierungsgrade folgt, dass sich die Workflow-Ebene und die Workflow-Spezifikation nicht durch ihre Automatisierbarkeit definieren, sondern durch die Detaillierung zugrunde liegender Vorgänge in Aktionen. Eine konzeptuelle Modellierung von Workflows ist damit auch ohne Workflow-Management-System zweckmäßig. Gründe hierfür sind die geringere Komplexität der Geschäftsprozessmodelle und die Unabhängigkeit vom Geschäftsprozessmodell bei der Anpassung der Vorgangsspezifikation. Im Folgenden werden daher die Geschäftsprozess- und die Workflow-Ebene zur Modellierung betrieblicher Systeme unterschieden.

## 2.2.3 Modelltransformation

### 2.2.3.1 Einführung

Die Abbildung von Systemen auf Modelle verschiedener Abstraktionsebenen führt unmittelbar zu einem Bedarf nach Methoden, die eine syntaktische und eine semantische Überführung von Modellen zwischen Ebenen gestatten. Eine Modelltransformation spezifiziert eine Transformation zwischen zwei syntaktisch und semantisch korrekten Modellen einer Quell- und einer Zielsprache, etwa anhand von Zuordnungsbeziehungen zur Definition eines Model Mappings (Brambilla et al. 2017, S. 18). Das Ziel besteht in der syntaktischen Überführung in die Zielsprache sowie in der Überwindung der semantischen Lücke. Grundlegend wird zwischen den Transformationsarten Modell-zu-Modell und Modell-zu-Text unterschieden (Kahani et al. 2018, S. 6; Brambilla et al. 2017, S. 123; Gaevi et al. 2009, S. 149). Gegenstand von Modell-zu-Modell-Transformationen im Kontext betrieblicher Systeme sind zum einen fachliche Modelle, etwa von Geschäftsprozessen, Workflows und Anwendungssystemen, und zum anderen technische Modelle, wie z.B. von Software-Architekturen, Klassen und Methodenaufrufen. Im Falle von Modell-zu-Text-Transformationen verwendet die Ziel-Sprache eine textuelle Notation. Die Transformationsart wird in erster Linie im modellgetriebenen Software Engineering für die Generierung von Quellcode aus technischen Modellen genutzt.

### Modellgetriebene Ansätze

Modellgetriebene Ansätze aller Anwendungsdomänen werden häufig als MD* (Model Driven *) bezeichnet (Brambilla et al. 2017, S. 9). Das Charakteristikum eines modellgetriebenen Ansatzes ist die Verwendung von Modellen als primäres Entwicklungsartefakt. Hinsichtlich der Anwendungsdomäne werden beispielsweise Model Driven Engineering (MDE), Model Driven Software Engineering (MDSE) und Model Driven Development (MDD) unterschieden. MDE bezieht sich als Oberbegriff auf die ingenieurmäßige Entwicklung, z.B. von Systemen, Prozessen oder Produkten. MDSE und MDD sind Spezialisierungen des MDE-Begriffs für die Domäne des Software Engineering bzw. der Software-Entwicklung. Neben modellgetriebenen Ansätzen werden Ansätze teilweise als modellbasiert bezeichnet und anhand des Präfixes MB in gleicher Form nach Anwendungsdomäne spezialisiert. Modellbasierte Ansätze verwenden Modelle als wichtige Entwicklungsartefakte, die Entwicklung wird allerdings nicht allein auf ihrer Basis vorangetrieben.

### Metamodellbasierte Transformation

Die Spezifikation einer metamodellbasierten Transformation wird anhand der Metaebene der Quell- und der Ziel-Sprache festgelegt. Die Syntax des Metamodells der Quellsprache wird beispielsweise anhand einer Abbildung mit der Syntax des Metamodells der Zielsprache in Beziehung gesetzt. Damit sind alle zum Metamodell der Quellsprache konformen Modelle transformierbar. Der Zusammenhang ist in Abbildung 2.7 dargestellt.

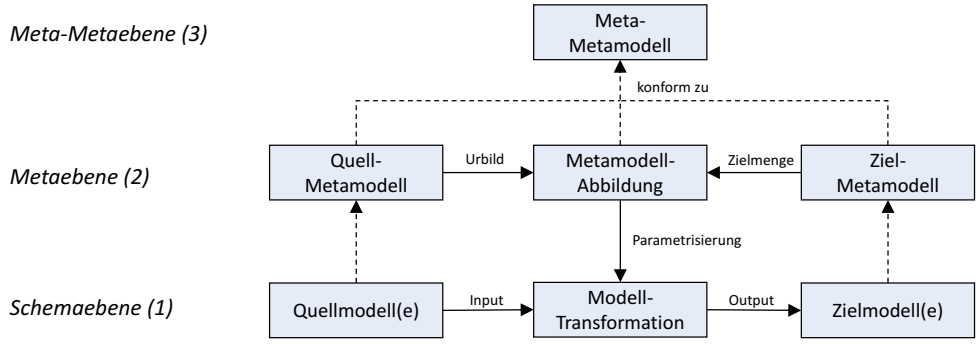



ABBILDUNG 2.7: Metamodellbasierte Transformation

Das Konzept metamodellbasierter Beziehungen zur Ableitung oder Transformation von Modellen ist Teil des generischen Architekturrahmens (Sinz 1995) und die

Grundlage der Model Driven Architecture (MDA) der OMG (OMG 2001). Beide Ansätze werden in den nachfolgenden Abschnitten beschrieben.

#### 2.2.3.2 Generischer Architekturrahmen

Zur generischen Beschreibung von Informationssystemarchitekturen schlägt der generische Architekturrahmen (Sinz 1995, 1997, 2002) eine Gliederung des Architektur-Entwurfs anhand von Modellebenen und Sichten vor. Die Architektur konkreter Klassen von Informationssystemen ist damit hinsichtlich ihrer Konstruktionsregeln in Metamodellen mehrerer Abstraktionsebenen beschreibbar. Anhand des Rahmens wird eine Standardisierung des Architektur-Entwurfs von Informationssystemen hinsichtlich der Gesamtarchitektur, der Beschreibung verschiedener Abstraktionen, der Bildung von Perspektiven auf Abstraktionen, der Spezifikation von Beziehungen zur Transformation von Abstraktionen und der Wiederverwendung von Modellen und Beziehungen geschaffen. Eine zentrale Anwendung des Architekturrahmens ist daher die Konzeptualisierung modellgetriebener Ansätze und Methodiken.

Der Entwurf der Gesamtarchitektur definiert sich anhand von vier Merkmalen entlang des folgenden Vorgehens (Sinz 1997):

- Zunächst ist die Anzahl der Modellebenen festzulegen (1. Merkmal). Eine Modellebene bildet jeweils eine Abstraktion anhand von Modellen ab.
- Anschließend folgt die Festlegung von Modellierungskonzepten und Metamodellen je Ebene (2. Merkmal).
- Je Modellebene und Metamodell werden eine Reihe von Sichten definiert (3. Merkmal). Sichten beschreiben verschiedene Perspektiven auf Modelle, die spezifische Aspekte der jeweiligen Abstraktion darstellen.
- Zwischen zwei benachbarten Ebenen definiert je ein Beziehungsmetamodell eine Verknüpfung (4. Merkmal). Diese wird anhand der Beziehungen zwischen den jeweils enthaltenen Syntaxelementen auf der Metaebene angegeben.

Das Konzept ist in Abbildung 2.8 dargestellt.

Weiterhin besteht die Möglichkeit zur Beschreibung wiederkehrender struktureller Bestandteile von Metamodellen durch die Angabe von Strukturmustern. Die in

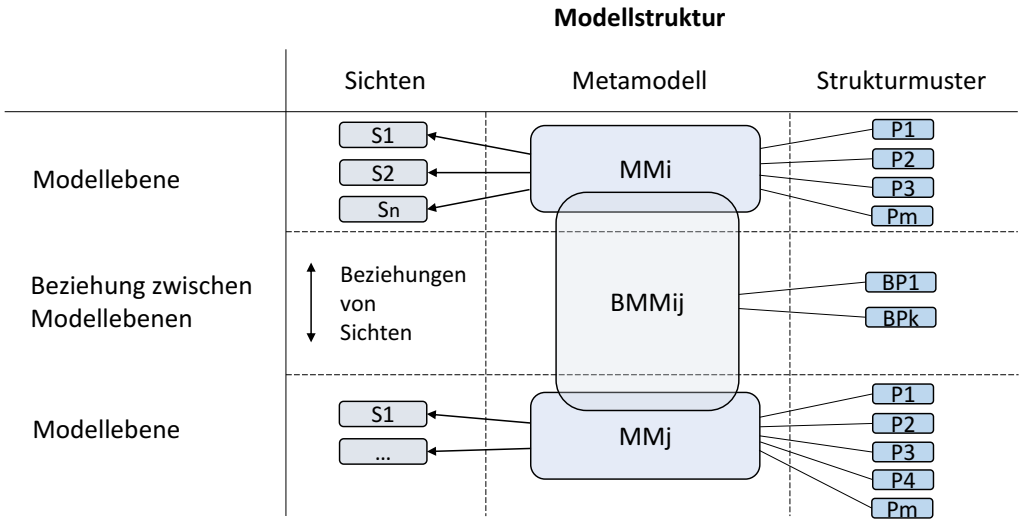



ABBILDUNG 2.8: Generischer Architekturrahmen (Sinz 1995, 1997)

einem Strukturmuster hinterlegte Syntax in Form von Meta-Elementen und Meta-Beziehungen kann auf mehreren Modellebenen zur Formulierung von Metamodellen wiederverwendet werden.

Bei den Beziehungen der Beziehungsmetamodelle handelt es sich um Zuordnungsbeziehungen zwischen den Syntaxelementen der benachbarten Ebenen, die für eine Modelltransformation herangezogen werden können. Zudem können zwischen Sichten benachbarter Modellebenen Beziehungen bestehen.

#### 2.2.3.3 Model Driven Architecture

Die Model Driven Architecture der OMG ist ein konkreter Ansatz für MDE, der ausgehend von fachlichen Modellen plattformunabhängige und plattformspezifische technische Modelle ableitet (OMG 2001, S. 1). Ziel der MDA ist die Standardisierung der modellgetriebenen Entwicklung, einerseits anhand einer mehrstufigen Architektur zur Modellierung und Transformation, sowie andererseits anhand von technischen Standards, mit denen die Kompatibilität von Software-Tools zur Modellierung und Transformation gewährleistet werden soll. Die MDA eignet sich damit für die Spezifikation von Transformationen in modellgetriebenen Ansätzen und Methodiken sowie für die technische Spezifikation, insbesondere im Bereich des Software Engineering anhand von MDSE. Die Nutzenpotenziale der MDA sind eine höhere Portabilität, Interoperabilität und Wiederverwendbarkeit (OMG 2003, S. 2-2).

### Architektur des Ansatzes

Fachliche Modelle werden innerhalb eines Computation Independent Model (CIM) erfasst und in zwei Computational Models, das Platform Independent Model (PIM) und das Platform Specific Model (PSM), überführt. Zunächst wird innerhalb des PIM eine plattformunabhängige Beschreibung einer Architektur der Implementierung modelliert. Eine Modelltransformation definiert den Übergang zu einem PSM, welches die Implementierung für eine spezifische technische Plattform darstellt. Die Abbildung eines PIM kann auf mehrere PSM unterschiedlicher Plattformen erfolgen. Abbildung 2.9 ordnet die Architektur des Ansatzes im Kontext der Systementwicklung (Ferstl und Sinz 2013, S. 499) ein.

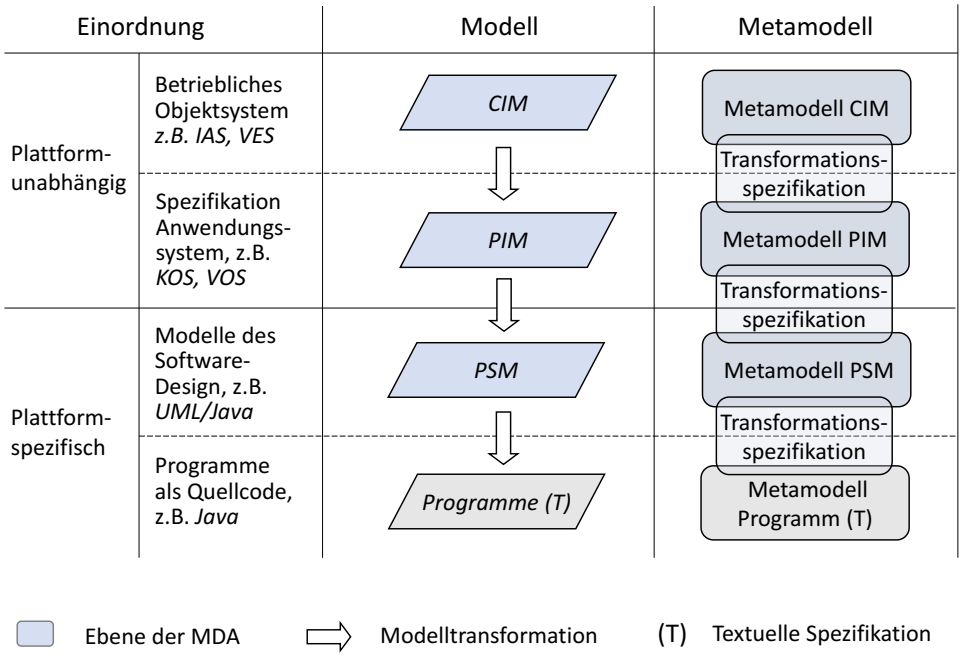



ABBILDUNG 2.9: Einordnung der Model Driven Architecture (vgl. OMG (2003), Ferstl und Sinz (2013, S. 499))

- Das *CIM* ist ein Modell zur Beschreibung der Fachlichkeit der abzubildenden Domäne im Sinne eines konzeptuellen Modells. Es erfasst beispielsweise Geschäftsprozesse mit ihren fachlichen Anforderungen, etwa in Form von IAS und VES der SOM-Methodik.
- Das *PIM* ist ein Computational Model, das Strukturen und auszuführende Operationen als abstrakte technische Architektur festlegt, ohne auf eine konkrete Plattform Bezug zu nehmen. Damit werden Komponenten abgegrenzt, die Objekte der fachlichen Domäne abbilden, etwa Organisationseinheiten,

Dokumente und Produkte. Die genannten Objekte können beispielsweise in einem KOS mit Attributen und Methoden definiert werden, deren Aufrufbeziehungen in einem VOS hinterlegt sind.

- Das *PSM* ist ein Computational Model, das sich auf eine konkrete technische Plattform bezieht, z.B. UML-Klassen- und Sequenzdiagramme, die eine Anwendung anhand von erweiterten Klassen eines software-technischen Frameworks beschreiben. Eine Plattform kann in einem Platform Model festgehalten werden, das beispielsweise Extension Points eines Frameworks mit Klassen und Parametern in UML modelliert. Mehrere technische Plattformen können involviert werden, um beispielsweise ein PIM in Anwendungssoftware verschiedener Betriebssysteme, etwa für PCs oder Smartphones, zu überführen.

**Konzepte zur Abbildung von Beziehungen zwischen PIM und PSM**

Die Abbildung von Beziehungen zwischen PIM und PSM ist die Voraussetzung für Modelltransformationen. Die OMG sieht hierfür verschiedene Mapping-Konzepte vor (OMG 2003, S. 3-2 ff.), anhand derer Zuordnungsbeziehungen definierbar sind:

- Ein *Metamodel Mapping* ist eine Zuordnung von Elementen des Metamodells des PIM zu dem des PSM. Anhand der Zuordnung ist jede Instanz eines PIM auf ein PSM abbildbar. Zur Generierung von Java-Klassen beispielsweise, indem die Meta-Objekte Attribut und Operator des KOS den Meta-Objekten Attribut und Methode eines Metamodells der Java-Plattform zugeordnet werden. Den Technologien der OMG folgend, ist eine Spezifikation von Metamodellen unter Nutzung der Meta Object Facility (MOF) (OMG 2016b) als Meta-Metamodell möglich.
- Ein *Model Type Mapping* stellt Zuordnungen zwischen typisierten Elementen eines PIM und eines PSM auf Schemaebene her. Ein Beispiel ist die Angabe einer Zuordnung zwischen einem generalisierten Objekttyp „Customer“ und einer Java-Klasse „Fragment“ eines Frameworks für Android-Apps. Ein Subtyp des Objekttyps, z.B. „Enterprise Customer“ wird in diesem Fall als Erweiterung der „Fragment“-Klasse abgebildet, welche Kundendaten von Unternehmenskunden darstellt.
- Ein *Model Instance Mapping* beschreibt eine Zuordnung zwischen beliebigen Elementen eines PIM und eines PSM auf Schemaebene, indem innerhalb des PIM spezifische Eigenschaften des PSM anhand von Markierungen (Marks)

hinterlegt werden. Das Setzen einer bestimmten Markierung (Marking) in Modellelementen des PIM determiniert deren Transformation zu Modellelementen des PSM, die in Abhängigkeit der Markierung aus vordefinierten Templates stammen können. Ein Beispiel ist eine Markierung derjenigen konzeptuellen Objekttypen eines PIM, die anhand des PSM in Form eines SQL-Code-Modells persistiert werden sollen.

Zudem sind Erweiterungen der Konzepte definierbar, etwa zur Zuordnung zwischen nicht aus dem PSM oder dem PIM hervorgehenden Typen, zur Kombination von Mapping-Konzepten, zur Transformation zwischen Patterns von PIM und PSM sowie zur Zusammenführung (Merging) mehrerer PIM in einem PSM (OMG 2003, S. 3-3 - 3-14).

#### 2.2.3.4 Implementierung von Modelltransformationen

Zur Beschreibung der Implementierung einer Modelltransformation können die folgenden Spezifikationsformen herangezogen werden. Ferner sind für textuelle Quell- und Ziel-Modelle Formen der Modell-zu-Text- sowie Text-zu-Modell-Transformation unterscheidbar (Kahani et al. 2018).

Spezifikationsformen im Bereich der Modell-zu-Modell-Transformation sind (Kahani et al. 2018):

- Relational-deklarative Beschreibungssprachen wie QVT Relational (QVTr), die Relationen anhand von Prädikaten und Constraints beschreiben, welche sich auf Quell- und Zielmodellmetamodell beziehen.
- Imperativ-operative Beschreibungssprachen wie QVT Operational (QVTo), in denen Zuordnungsfunktionen zwischen Quell- und Zielmetamodell für das Erstellen einer Abbildung beschrieben werden.
- Graph-basierte Spezifikationen anhand von Regeln, die auf einen Graphen angewendet werden. Eine Regel besteht aus einem Left-Hand- und Right-Hand-Side-Graphen (LHS, RHS), sowie ggf. einem Negative Application Condition (NAC) Graphen. Eine weitere Form sind Triple Graph Grammars (TGGs).
- Hybride Spezifikationsformen, wie beispielsweise innerhalb der Atlas Transformation Language (ATL), die relationalen und imperativen Spezifikationsformen kombiniert.

QVT ist eine Familie standardisierter Beschreibungssprachen (OMG 2011), die als Teil der MOF 2.0 spezifiziert ist und von mehreren Software-Tools implementiert

wird. Zu den Sprachen gehören QVTr, QVTo sowie QVT Core, wobei letztere als Basis der QVTr und QVTo definiert ist. Die Spezifikation der Sprachen erlaubt die Abfrage von Modellelementen (Query), die Bildung von Sichten (View) und die Durchführung von Modelltransformationen (Transformation). Implementierungen der Spezifikation von QVTr und QVTo sind beispielsweise als Teil des Eclipse Modeling Framework (EMF) verfügbar und unterstützen die Transformation von Modellen des Ecore-Formats (Gronback 2009, S. 231-243).

Spezifikationsformen im Bereich der Modell-zu-Text-Transformation sind (Kahani et al. 2018):

- Visitor-basierte Ansätze wie Melange (Czarnecki und Helsen 2003; Melange 2018), die einzelne Elemente des Quellmodells „besuchen“ und währenddessen kontinuierlich Quellcode in Form eines Streams generieren.
- Template-basierte Ansätze wie Xtend (Voelter 2013, S. 274-278), die Quellcode-Ausschnitte aus Vorlagen übernehmen und darin festgelegte Felder mit Instanzwerten aus den Quellmodellen versehen.
- Hybride Ansätze wie etwa MERL in MetaEdit+ (MetaCase 2018), die Visitor-basierte Ansätze nutzen und statische Quellcode-Ausschnitte aus Templates beziehen.

Eine Anwendung ist die Spezifikation von Anwendungssystemen anhand von konzeptuellen Objekten und Vorgangsobjekten für verschiedene Zielarchitekturen (Malischewski 1997). Software-Tools zur Generierung von Quellcode greifen häufig auf Template-basierte Sprachen wie Xtend zurück (Kahani et al. 2018; Voelter 2013, S. 274-278). Xtend ist eine auf Java basierende Programmiersprache, die JVM-kompatiblen Code erzeugt. Templates sind reguläre Klassen, die anhand von Template Expressions zu ersetzende Felder definieren. In der Umgebung des EMF (Steinberg et al. 2008) ist eine Modell-zu-Text-Transformation etwa anhand einer Template Expression für Modelle im Ecore-Format definierbar. Ein Beispiel hierfür ist die Template-basierte Generierung von Java-Methoden aus ECore-Modellen (Härer 2014). Im Falle einer Transformation der KOT eines KOS wird eine Template Expressen für jedes Element des Typs KOT aufgerufen. Im einfachsten Fall enthält die Template Expression eine vordefinierte Methodensignatur, deren Methodenname durch die Bezeichnung des Operators ersetzt wird.

### 2.2.4 Modellierung von Geschäftsprozessen am Beispiel des Semantischen Objektmodells (SOM)

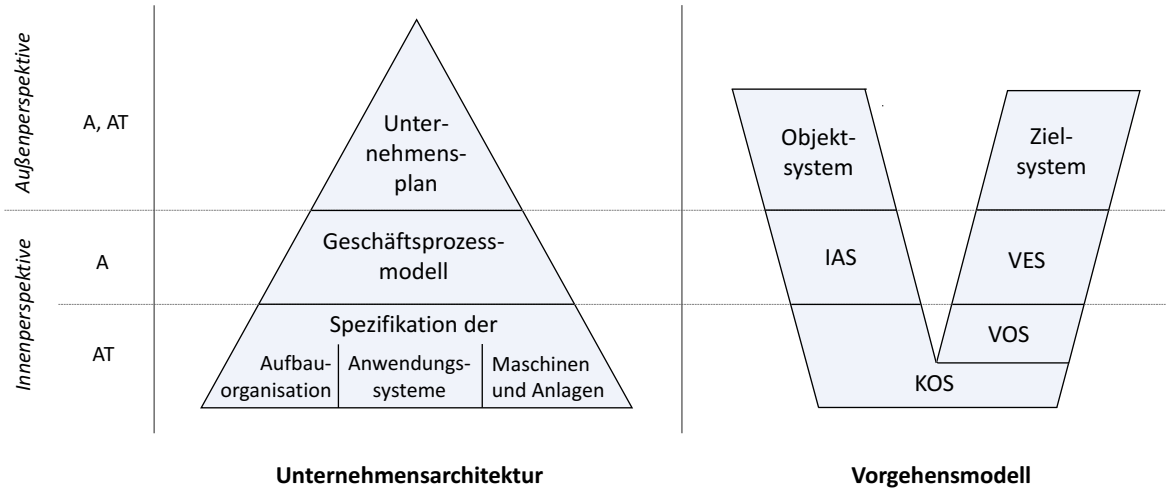
Das Semantische Objektmodell umfasst als Methodik zur Modellierung von Unternehmensarchitekturen die Unternehmensplanung, die Gestaltung von Geschäftsprozessen und die Festlegung von Ressourcen (Ferstl und Sinz 1990, 1993, 1995, 2013). Ein betriebliches System wird entlang dieser drei Ebenen einer Unternehmensarchitektur anhand von Modellen beschrieben. Gegenstände der Modellabbildungen sind das Objektsystem einer Unternehmung zusammen mit Zielen (1. Ebene), die betriebliche Leistungserstellung außerhalb und innerhalb des Informationssystems (2. Ebene) und die Aufgabenträger in Bezug auf die Aufbauorganisation, die Anwendungssysteme sowie die Maschinen und Anlagen (3. Ebene). Das Vorgehensmodell der Methodik sieht eine durchgehend auf Modellen beruhende Ableitung vom Unternehmensplan über Geschäftsprozesse hin zur Spezifikation von Anwendungssystemen vor.

#### 2.2.4.1 Unternehmensplan

Die Darstellung der Unternehmensarchitektur in Abbildung 2.10 enthält zunächst den Unternehmensplan zur Beschreibung eines betrieblichen Systems aus Außenperspektive. Das nebenstehende V-Modell sieht hierfür Angaben zum Objektsystem vor, die Teil der links notierten strukturorientierten Sichten sind, sowie Angaben zum Zielsystem, das zu den rechts notierten verhaltensorientierten Sichten zählt.

Angaben zum Objektsystem beschreiben innerhalb des Unternehmensplans, welche Leistungen durch die abzugrenzende Diskurswelt gegenüber der Umwelt erstellt werden. Dies entspricht der Gesamtaufgabe des Unternehmens und betrifft in der Gliederung des Unternehmensplans die Aufgabenebene. Hierfür wird die Struktur der Diskurswelt und der Umwelt festgehalten, indem betriebliche Objekte der Diskurs- und Umwelt notiert und gemäß der Leistungserstellungen in Beziehung gesetzt werden. Betriebliche Objekte werden im Sinne des objektorientierten Aufgabenmodells (siehe Abschnitt 2.1.3) verstanden.

Angaben zum Zielsystem umfassen die initialen Sach- und Formalziele, die Strategien und die Rahmenbedingungen zu deren Umsetzung sowie die dafür benötigten Ressourcen. Damit wird auf das Verhalten des betrieblichen Systems Bezug genommen. Die Angaben zu den Zielen der globalen Unternehmensaufgabe und zu Ressourcen beziehen sich im Unternehmensplan auf die Aufgaben- und die Aufgabenträgerebene.




*A: Aufgaben, AT: Aufgabenträger*

ABBILDUNG 2.10: Unternehmensarchitektur und Vorgehensmodell des Semantischen Objektmodells (Ferstl und Sinz (2013, S. 195, 198))

#### 2.2.4.2 Geschäftsprozessmodell

Das Geschäftsprozessmodell beschreibt in der Unternehmensarchitektur die Innenperspektive des betrieblichen Systems auf der Aufgabenebene. Geschäftsprozesse geben damit an, wie die Gesamtaufgabe der Unternehmung hinsichtlich der Leistungserstellung, der Lenkung und des Ablaufs gelöst werden kann. Diese drei Sichten werden innerhalb des V-Modells durch die Angabe zweier Schemata repräsentiert. Die strukturorientierten Leistungs- und Lenkungssichten werden in einem Interaktionsschema (IAS) erfasst, die verhaltensorientierte Ablaufsicht in einem Vorgangs-Ereignis-Schema (VES). Abbildung 2.11 definiert das Metamodell für Geschäftsprozessmodelle mit der in IAS und VES zur Verfügung stehenden Syntax.

**Abbildung der Struktur von Geschäftsprozessen**

Zur Erfassung der Struktur innerhalb des IAS werden die für die Abgrenzung der Diskurswelt angegebenen betrieblichen Objekte und Beziehungen in der aus dem Metamodell hervorgehenden Syntax und Notation modelliert. Die Modellierung wird anhand einer mehrstufigen Zerlegung der Objekte und Beziehungen detailliert, um die Leistungs- und Lenkungssicht anhand der in Objekten enthaltenen betrieblichen Aufgaben anzugeben (siehe Abschnitt 2.1.3). Weiterhin unterschieden werden Objekte der Diskurswelt in ihrer Notation als Rechteck sowie jeweils als Ellipse notierte Objekte der Umwelt. Gerichtete Beziehungen zwischen Objekten modellieren betriebliche Transaktionen, die initial als Durchführungstransaktion ($T_d$)

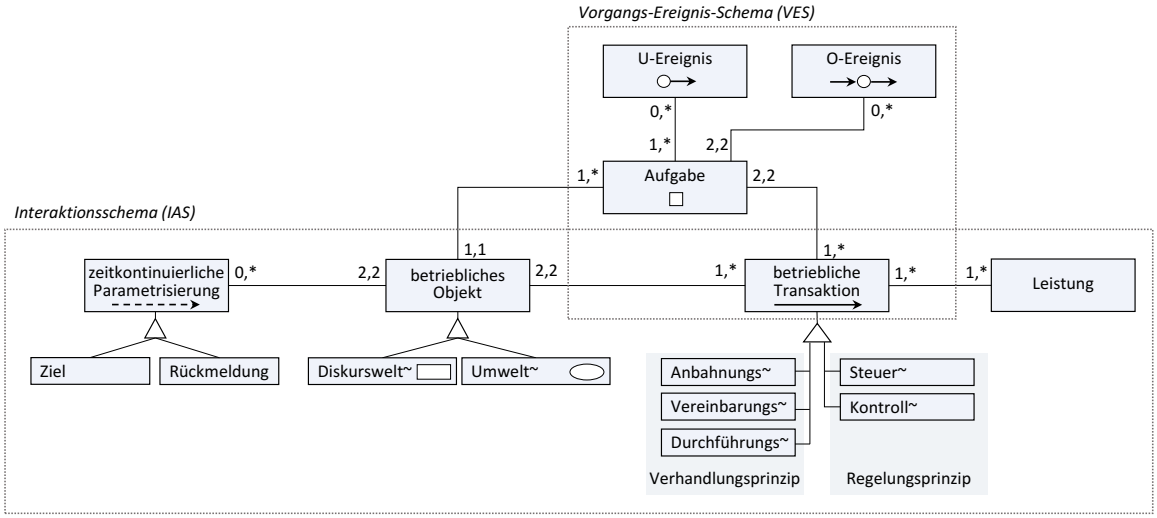



ABBILDUNG 2.11: Metamodell für Geschäftsprozesse in SOM (vgl. Ferstl und Sinz 2013, S. 219)

einer Leistungserstellung typisiert sind. Eine Transaktion bedingt jeweils zwei Aufgaben, die Bestandteile der in Beziehung stehenden betrieblichen Objekte sind. Initial wird damit die Leistungssicht anhand der Leistungserstellungen der abzubildenden Prozesse gebildet, beispielsweise zur Darstellung relevanter Haupt- und Service-Prozesse. Die Aufdeckung der Leistungs- und Lenkungssicht innerhalb von Prozessen kommt anhand einer Zerlegung von Transaktionen in weitere Transaktionen verschiedener Subtypen und anhand einer Zerlegung von Objekten in Objekte und Transaktionen hinzu. Anhand dieser wird die transaktionsbasierte Koordination betrieblicher Objekte angegeben sowie ggf. eine zeitkontinuierliche Parametrisierung von Objekten. Eine Zerlegung orientiert sich an den Sach- und Formalzielen der enthaltenen betrieblichen Aufgaben. Objekte und Transaktionen sind hinreichend detailliert, wenn mit ihnen die Umsetzung der enthaltenen Sach- und Formalziele auf der angestrebten Abstraktionsebene planbar, durchführbar und kontrollierbar (siehe Abschnitt 2.1.1.1) ist. Die Zerlegungsprodukte von Objekten und Transaktionen der initialen angegebenen Diskurswelt bilden jeweils Hierarchien. Jedes initiale Objekt und jede Transaktion kann damit in einer Baumstruktur als Zerlegungsbaum angegeben werden.

*Zeitdiskrete Koordination betrieblicher Objekte*

Die Koordination der Leistungserstellung betrieblicher Objekte erfolgt anhand von Transaktionen, die ein hierarchisches oder ein nicht-hierarchisches Grundmuster implementieren. Zur Angabe der Koordination werden vorhandene Objekte oder Transaktionen zerlegt.

Die Zerlegungsregeln sind formal definiert (Ferstl und Sinz 2013, S. 208):

- Die hierarchisch koordinierte Leistungserstellung folgt dem Regelungsprinzip. Ein gegebenes betriebliches Objekt O wird in ein Reglerobjekt O‘ und ein Regelstreckenobjekt O“ sowie eine Steuertransaktion ($T_s$) von O‘ zu O“ und ggf. eine Kontrolltransaktion ($T_k$) von O“ nach O‘ zerlegt. Die Lenkung erfolgt im Sinne eines Regelkreises durch O‘, das als Regler Lenkungsnachrichten über $T_s$ an die Regelstrecke O“ übermittelt. O“ entspricht dem Leistungssystem, welches über $T_k$ Nachrichten an den Regler zurückmeldet. Die Zerlegungsregel in BNF lautet formal:

$$O ::= \{O‘, O“, T_s(O‘, O“), [T_k(O“, O‘)]\} \qquad \text{(Regel 1)}$$

  Die Objektzerlegung ist anhand einer Ersetzungsregel rekursiv anwendbar:

$$O‘|O“ ::= O \qquad \text{(Regel 4)}$$

- Die nicht-hierarchisch koordinierte Leistungserstellung folgt dem Verhandlungsprinzip. Eine Transaktion T zwischen zwei Objekten O und O‘ wird entlang der Phasen Anbahnung, Vereinbarung und Durchführung in entsprechende Transaktionen zerlegt. So entstehen eine Anbahnungstransaktion ($T_a$) von O nach O‘, eine Vereinbarungstransaktion ($T_v$) von O‘ nach O sowie eine Durchführungstransaktion ($T_d$) von O nach O‘. Sofern eine Anbahnung fachlich nicht notwendig ist, entfällt $T_a$. Falls nur eine Leistungserstellung ohne Anbahnung und Vereinbarung stattfindet, entfallen $T_a$ und $T_d$. Die Zerlegung ist mehrstufig anwendbar. Die Zerlegungsregeln in BNF lauten formal:

$$T(O, O‘) ::= [[T_a(O, O‘)seq]T_v(O‘, O)seq]T_d(O, O‘) \qquad \text{(Regel 6)}$$

  Die Transaktionszerlegung ist durch eine Ersetzungsregel rekursiv anwendbar:

$$T_a|T_v|T_d ::= T \qquad \text{(Regel 9)}$$

Weitere Zerlegungsregeln beschreiben die Zerlegung eines Objekts in zwei Objekte sowie eine zwischen ihnen verlaufende Transaktion (Regel 2), die Spezialisierung

von Objekten oder Transaktionen in 1 bis $n$ Objekte bzw. Transaktionen des gleichen Typs (Regel 3 bzw. 8), die Zerlegung einer Transaktion in 2 bis $n$ sequenzielle (seq) oder alternativ in 2 bis $n$ parallele (par) Transaktionen (Regel 7), sowie die Objektzerlegung in zwei bis $n$ Objekte zwischen denen ein Parameterfluss eines Ziels $P_Z$ zur zeitkontinuierlichen Parametrisierung und ggf. ein gegenläufiger Parameterfluss einer Rückmeldung $P_R$ verlaufen (Regel 5).

*Zeitkontinuierliche Parametrisierung betrieblicher Aufgaben*

Neben der zeitdiskreten Koordination anhand von Transaktionen können mehrere Objekte eine zeitkontinuierliche Verfolgung von Zielen anhand eines fortlaufenden Abgleichs von Parameterwerten durchführen. Die zeitkontinuierliche Parametrisierung ist mit der hierarchischen und nicht-hierarchischen Koordination kombinierbar und entspricht einer Zielvorgabe bzw. Zielvereinbarung zwischen den beteiligten Objekten. Eine Anwendung ist im Falle von Entscheidungsaufgaben möglich, deren Formalziele als Parameter herangezogen werden können. Ein Ziel wird anhand eines Parameterwertes quantifiziert, der anhand eines kontinuierlichen Parameterflusses $P_Z$ von einem Objekt zu 1 bis n weiteren Objekten verläuft. Die in diesen Objekten tatsächlich vorliegenden Parameterwerte können dem ursprünglichen Objekt optional über den gegenläufigen Parameterfluss $P_R$ zurückgemeldet werden. Da $P_Z$ und $P_R$ auf Objekte bezogen und entkoppelt von einzelnen Aufgaben sind, ist die zeitkontinuierliche Parametrisierung Teil des IAS, nicht aber des VES.

**Abbildung des Verhaltens von Geschäftsprozessen**

Das V-Modell sieht auf der Ebene der Geschäftsprozessmodelle neben der strukturorientierten Darstellung innerhalb des IAS eine verhaltensorientierte Darstellung der Ablaufsicht von Geschäftsprozessen innerhalb des VES vor. Das Schema beschreibt Vorgänge einzelner Aufgaben in einer ablauforientierten Darstellung auf Typebene. Die strukturell bereits erfassten betrieblichen Transaktionen bedingen die Existenz von Aufgaben in den miteinander verbundenen Objekten. Eine Transaktion wird von einer Aufgabe innerhalb des ursächlichen Objekts und einer Aufgabe innerhalb des zweiten Objekts gemeinsam als Vorgang durchgeführt. Innerhalb des VES werden entsprechend der Syntax und der Notation des Metamodells die Aufgaben einer betrieblichen Transaktion gegenüberstehend notiert und anhand einer Transaktion verbunden. Hiermit wird ein geplanter Vorgang als instanziierbarer Vorgangstyp abgebildet.

Abbildung 2.12 stellt IAS und VES beispielhaft für einen Online-Shop dar. Tabelle 2.1 zeigt die zugehörigen Zerlegungsbäume.

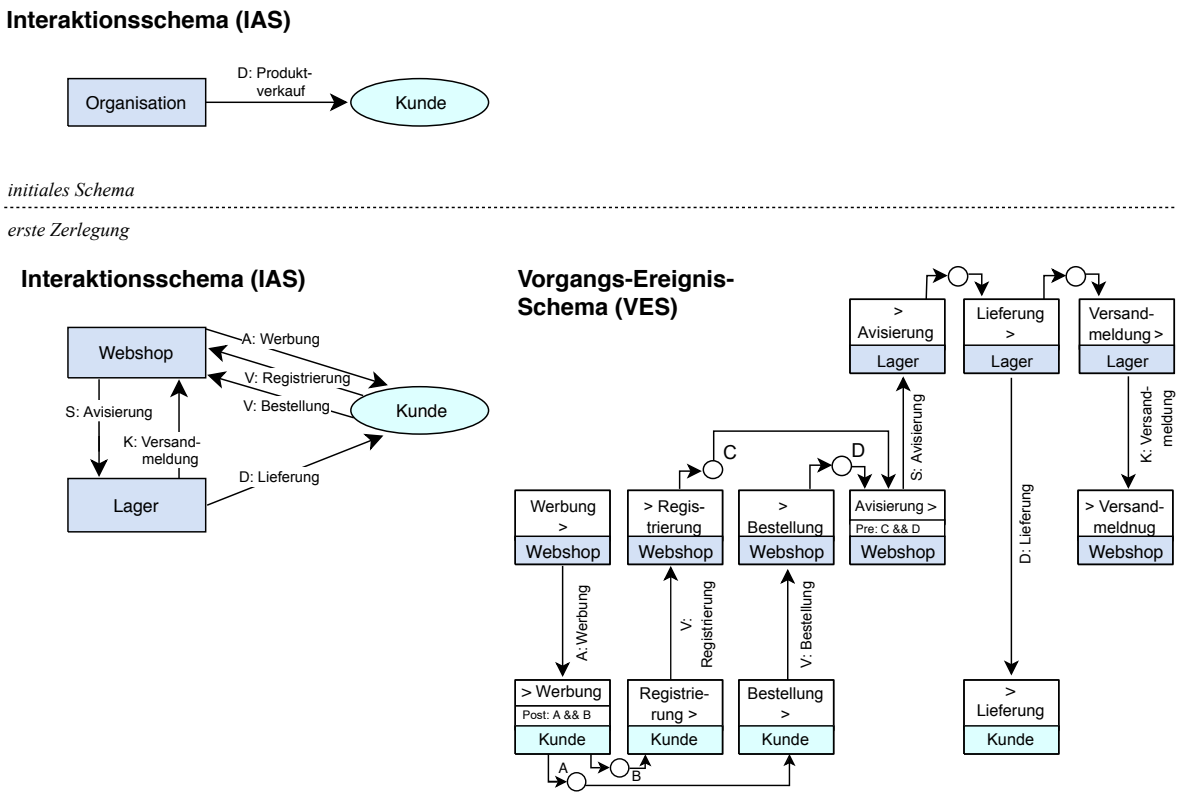



ABBILDUNG 2.12: Beispiel zu IAS und VES anhand eines Online-Shops (vereinfacht)

| Objektzerlegung | Transaktionszerlegung |
|---|---|
| **Organisation** | **D: Produktverkauf** |
| **Webshop** | A: Werbung |
| **Lager** | V: Auftrag |
| S: Avisierung | V (par): Registrierung |
| K: Versandmeldung | V (par): Bestellung |
| **Kunde** | D: Lieferung |

TABELLE 2.1: Beispiel zur Objekt- und Transaktionszerlegung

*Ereignisgesteuerte Vorgangsauslösung*

Die Auslösung eines Vorgangs geschieht mit dem Eintreten der Vorereignisse einer Aufgabe, welche die betriebliche Transaktion des Vorgangs initiiert. Nach der Durchführung treten die Nachereignisse der über die Transaktion in Beziehung stehenden zweiten Aufgabe ein. Das Metamodell des VES sieht zur Modellierung von Ereignissen ein objektinternes Ereignis (O-Ereignis) sowie ein Umweltereignis (U-Ereignis) vor.

O-Ereignisse legen den Ablauf der mit ihnen verknüpften Aufgaben fest. Sie implizieren eine Reihenfolgebeziehung zwischen den verbundenen Aufgaben. Ein O-Ereignis, das über eine gerichtete Beziehung hin zu einer Aufgabe verbunden ist, stellt ein Vorereignis dar, während eine gegenläufige Beziehung ein Nachereignis repräsentiert. Eine Reihenfolgebeziehung ergibt sich durch die Festlegung eines O-Ereignisses als Vorereignis einer Aufgabe und gleichzeitig als Nachereignis einer zweiten Aufgabe. Die Verknüpfung der Vorgänge einzelner Aufgaben führt zu einem Vorgangsnetz.

U-Ereignisse lösen eine Aufgabe aufgrund eines Ereignisses der Umwelt aus. Der Eintritt von objektexternen U-Ereignissen wird nicht von betrieblichen Objekten und Transaktionen beeinflusst. U-Ereignisse modellieren beliebige objektexterne Ereignisse, die per Definition nicht als Teil der Diskurswelt abbildbar sind. Beispiele sind das Eintreten eines Datums oder einer Uhrzeit, die Verfügbarkeit von Ressourcen, das Erreichen eines Ortes, oder auch das Über- oder Unterschreiten von Schwellwerten, wie etwa per Sensor erfasste Temperaturwerte.

*Petri-Netz-Semantik*

Ein VES lässt sich als ein spezielles Petri-Netz interpretieren, in dem Ereignisse als Stellen und Aufgaben mit Transaktionen als Transitionen notiert sind (Ferstl und Sinz 2013, S. 206). Damit wird der Ausführungszustand von Vorgängen durch eine Markierung der Stellen von Vor- und Nachereignissen erfassbar. Aufgrund der Abbildung fachlicher Transaktionen in Vorgängen wird dabei eine synchrone Vorgangsdurchführung unter Beteiligung der jeweils in Beziehung stehenden Aufgaben unterstellt. Der Zustand während der Vorgangsdurchführung ist hier fachlich undefiniert und wird zur Wahrung der Konsistenz nicht abgebildet.

Aufgaben können zudem Pre- und Post-Conditions enthalten, die Vor- bzw. Nachbedingungen für die Auslösung und den Abschluss von Aufgaben darstellen. Pre- und Post-Conditions verwenden logische Operatoren, die auf Ereignisse oder Objektzustände bezogene Operanden verknüpfen (Ferstl und Sinz 2013, S. 207). Ein Beispiel sind die in Abbildung 2.12 dargestellten Post- und Pre-Conditions der Aufgaben „> Werbung“ bzw. „Avisierung >“. In diesem Beispiel wird die nebenläufige Durchführung einer Registrierung und einer Bestellung durch einen Kunden angenommen. Eine zeitliche Überlappung, z.B. durch Nutzung verschiedener Browser und Geräte, ist hiermit zulässig.

#### 2.2.4.3 Spezifikation von Anwendungssystemen

Die Innenperspektive der Unternehmensarchitektur beinhaltet neben dem Geschäftsprozessmodell auf der Aufgabenebene die Spezifikation von Ressourcen auf der Aufgabenträgerebene. Die Ebene der Ressourcen spezifiziert die Aufbauorganisation, die Anwendungssysteme sowie die Maschinen und Anlagen. Für die Gestaltung des Informationssystems sind insbesondere Anwendungssysteme von Belang, deren fachliche Spezifikation von der Geschäftsprozessebene ausgehend modellgetrieben ableitbar ist. Für die folgende Spezifikation werden ausschließlich automatisierbare Aufgaben betrachtet. Dem V-Modell folgend, werden IAS und VES in das struktur- und verhaltensorientierte konzeptuelle Objektschema (KOS) und in das verhaltensorientierte Vorgangsobjektschema (VOS) überführt.

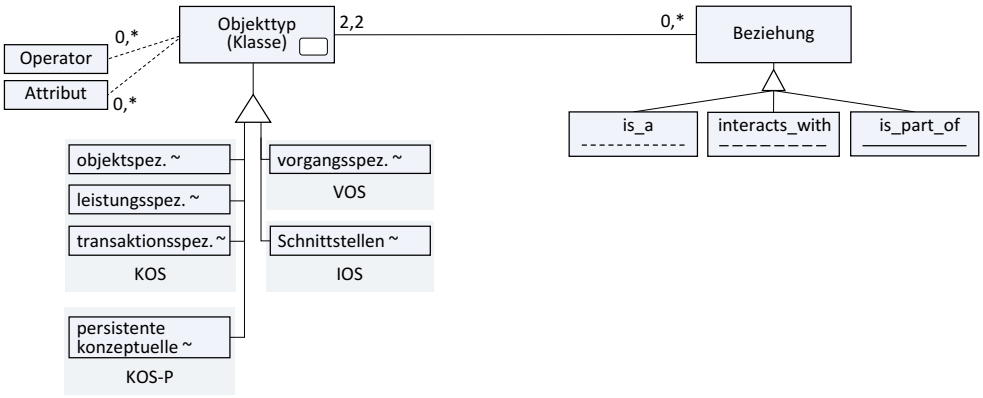



ABBILDUNG 2.13: Metamodell für die Spezifikation von Anwendungssystemen in SOM nach Ferstl und Sinz (2013, S. 233)

Das Metamodell zur Spezifikation von Anwendungssystemen in Abbildung 2.13 zeigt die Syntax und Notation der spezialisierten konzeptuellen Objekttypen (KOT) des KOS und der Vorgangsobjekttypen (VOT) des VOS. Darüber hinaus sind Angaben der zu persistierenden KOT in einem KOS-P-Schema und zu Schnittstellenobjekttypen in einem Interface-Objektschema (IOS) möglich (Ferstl und Sinz 2013, S. 234).

Abbildung 2.14 zeigt das KOS und VOS für die automatisierten Aufgaben des zuvor eingeführten Beispiels.

**Konzeptuelles Objektschema**

Das KOS entspricht einer Abbildung konzeptueller Klassen im Sinne der Objektorientierung. Klassen werden durch die drei Subtypen des KOT abgebildet. Einem KOT sind dementsprechend Attribute und Operatoren (Methoden) zuzuordnen. In

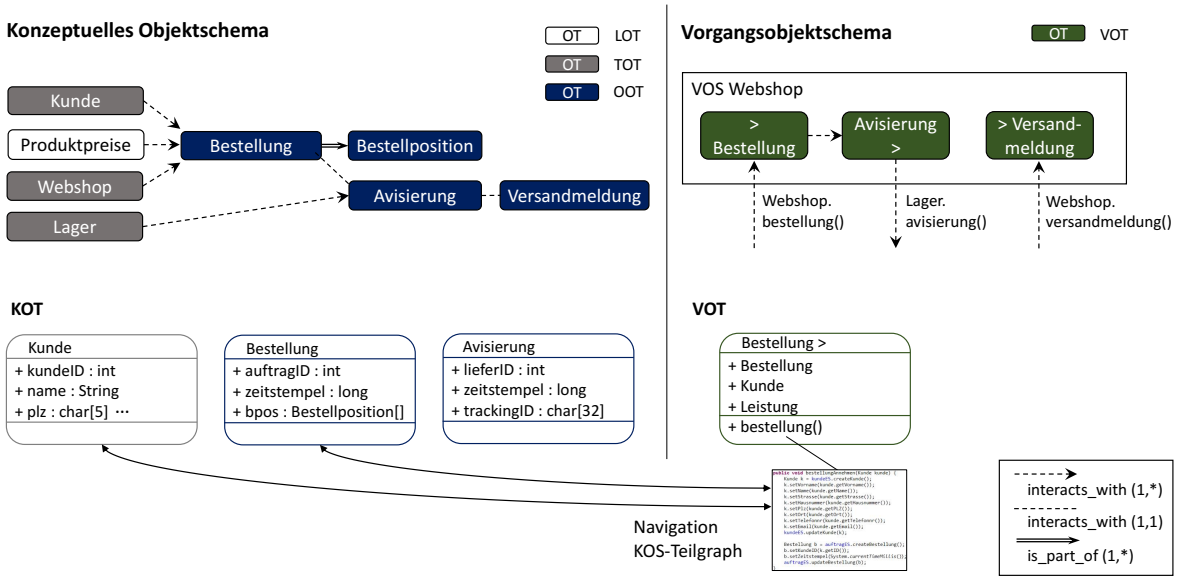



ABBILDUNG 2.14: Beispiel zu KOS und VOS anhand eines Online-Shops (Ausschnitt)

das konzeptuelle Objektschema werden aus der innerhalb des IAS spezifizierten Zerlegung (a.) objektspezifische Objekttypen (OOT) je betrieblichem Objekt, (b.) leistungsspezifische Objekttypen (LOT) je betrieblicher Leistung und (c.) transaktionsspezifische Objekttypen (TOT) je betrieblicher Transaktion aufgenommen. Die KOT nichtautomatisierter Aufgaben werden nicht erfasst.

Zur Darstellung der Abhängigkeiten werden zunächst OOT und LOT als unabhängige Elemente notiert. TOT schließen sich in Reihenfolge des Ablaufs der Transaktionen bzw. der Abhängigkeiten gemäß VES an und werden anhand von gerichteten interacts_with-Beziehungen verknüpft. Ein TOT ist Ziel der interacts_with-Beziehungen von genau zwei OOT, die aus den betrieblichen Objekten der Transaktion des TOT hervorgehen, und ggf. von genau einem LOT, der aus der betrieblichen Leistung heraus der Transaktion zugehörig ist. Ein sequenziell nachfolgender TOT ist Ziel der interacts_with-Beziehungen der vorhergehenden TOT. Die sich ergebenden transitiven Abhängigkeiten werden nicht notwendigerweise notiert.

Eine interacts_with-Beziehung gibt an, dass Operator- und Attributaufrufe in Richtung der Beziehung entsprechend einer festzulegenden Kardinalität erfolgen. Zudem sind is_part_of- und is_a-Beziehungen definiert. Zur Auflösung von 1-zu-n-Beziehungen zwischen den Attributen eines KOT kann eine Aufteilung der Attribute auf zwei KOT erfolgen, die durch eine is_part_of-Beziehung verbunden sind. Generalisierungen von KOT können anhand von is_a-Beziehungen erfolgen.

**Vorgangsobjektschema**

Das VOS entspricht einer verhaltensorientierten Abbildung der je Vorgang aufzurufenden Operatoren (Methoden) und Nachrichtendefinitionen. Ein Vorgangsobjekttyp (VOT) beschreibt die Aufrufbeziehungen einer vollautomatisierten Aufgabe eines Vorgangs anhand der in den KOT definierten Operatoren. Innerhalb eines VOS stehen mehrere VOT über interacts_with-Beziehungen in Verbindung, um Abläufe über mehrere Vorgänge abzubilden.

Die Basis zur Ableitung eines VOS bilden die Aufgaben des Geschäftsprozessmodells. Jede automatisierte Aufgabe des abzubildenden Teils eines VES wird in einen VOT überführt. Ereignisse und Transaktionen werden auf interacts_with-Beziehungen abgebildet. Ein VOT besitzt Attribute und Operatoren, wobei Attribute diejenigen KOT umfassen, die zur Vorgangsdurchführung zugegriffen werden müssen, während Operatoren von anderen VOT aufrufbare Methoden darstellen, um eine Vorgangsdurchführung auszulösen.

Eine Kommunikation zwischen VOT erfolgt entlang der interacts_with-Beziehungen durch den Austausch von Nachrichten. Die von jedem VOT interpretierbaren Nachrichten und die damit aufzurufenden Operatoren sind als Nachrichtendefinitionen anzugeben. Innerhalb eines VOT werden Aufrufbeziehungen durch Angaben zu den jeweils zugegriffenen KOT und den auf ihnen aufgerufenen Operatoren hinterlegt.

**Implementierung der Anwendungssystemspezifikation**

Die fachliche Spezifikation des KOS und VOS bildet den Ausgangspunkt für technische Spezifikationen und kann insbesondere für die Ableitung von Software-Artefakten herangezogen werden.

Aufgrund der Prozessorientierung unter Angabe einer Verhaltenssicht ist eine Implementierung von ablauforientierten Systemen auf Basis des VOS durchführbar, z.B. für ein WfMS. Zudem ermöglicht die Struktursicht der Prozesse innerhalb eines betrieblichen Systems in Kombination mit der Verhaltenssicht die Spezifikation von verteilten Anwendungssystemen, die aufgrund des objektorientierten Aufgabenmodells inhärent objektorientiert und objektintegriert sind (Ferstl und Sinz 2013, vgl. S. 223).

Im Rahmen von MDSE eignet sich das KOS für die Überführung in Klassen objektorientierter Programmiersprachen, deren Daten in ebenso aus dem KOS hervorgehenden Datenschemata oder serviceorientierten Datenschnittstellen abgelegt

werden. Das VOS eignet sich zur Ableitung von Services oder, je VOT, Microservices (Malischewski 1997, 2016, S. 225), die Schnittstellen und Aufrufbeziehungen in den aus dem KOS hervorgehenden Software-Artefakten definieren. Weitere Beispiele sind die Generierung von Quellcode für JavaEE-Services (Härer 2014) und die Ableitung von REST-Schnittstellen serviceorientierter Architekturen (Wolf 2015).

### 2.2.5 Modellierung von Workflows am Beispiel von Business Process Model and Notation (BPMN)

Workflows detaillieren die in Geschäftsprozessen festgelegten Vorgangsabläufe auf der Ebene der Aufgabenträger (siehe Abschnitt 2.1.4.2).

Betriebliche Aufgaben einzelner Vorgänge werden in ihrer Innensicht so weit detailliert, dass die zur Durchführung von einem bestimmten Aufgabenträger auszuführenden Schritte in elementaren Aktionen erfasst sind. Die Abbildung wird in Workflow-Spezifikationen und, im Falle von Modellen, in Workflow-Schemata notiert. Die Steuerung des Ablaufs der Aktionen durch eine Aktionensteuerung betrifft das Workflow-Management, das in einem Workflow-Management-System (WfMS) automatisiert wird (siehe Abschnitt 2.1.4.2).

Die Business Process Model and Notation (BPMN) ist eine Modellierungssprache, deren Notation inner- und zwischenbetriebliche Abläufe anhand von verschiedenen Diagrammen abbildet. Die Spezifikation der von der OMG standardisierten Sprache schlägt in Version 2.0.2 (OMG 2014) ein Process Diagram, ein Collaboration Diagram und ein Choreography Diagram (siehe Abbildung 2.15) sowie darauf aufbauende Diagramme vor.

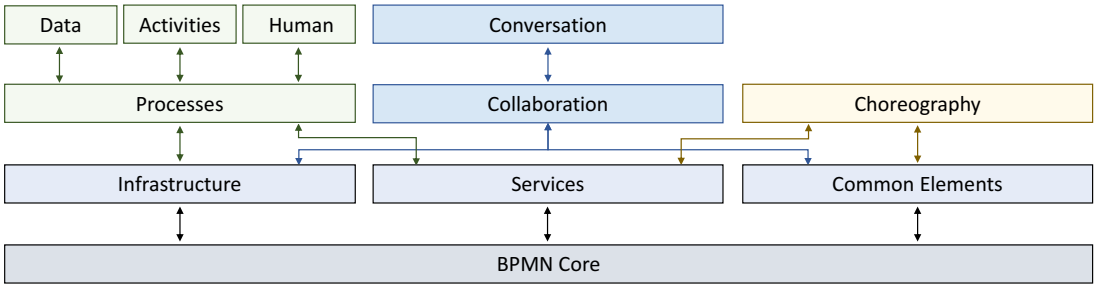



ABBILDUNG 2.15: Komponenten von BPMN Core (eigene Darstellung nach OMG (2014, S. 47))

#### 2.2.5.1 BPMN Core

Die Kern-Komponenten der Sprache setzen sich aus der Infrastructure als Definition einer abstrakten Syntax anhand eines technischen Metamodells (OMG 2014, S. 49 ff.), den Core Elements als Definition der gemeinsamen Syntax in Elementen und Beziehungen von Diagrammen sowie den Services als Definition einer technischen und implementierungsnahen Syntax für Services anhand von Service-Interfaces, Endpoints und Messages (OMG 2014, S. 101 ff.) zusammen.

Die grundlegende Syntax der Common Elements erfasst die Verhaltenssicht eines betrieblichen Systems in Activity-Elementen, die entlang eines Kontrollflusses in Form von Sequence-Flow-Beziehungen verknüpft werden. Eine Activity entspricht einer Aktion, die ein Aufgabenträger zur Durchführung einer betrieblichen Aufgabe ausführt. Die Elemente sind Teil des Process Diagram und des Collaboration Diagram. Sie sind die Basis für die Beschreibung zwischenbetrieblicher Beziehungen anhand des Choreography Diagram.

Das in Abbildung 2.16 dargestellte Metamodell fasst die Syntax der Komponenten zur Abbildung von Workflows in Process Diagrams und Collaboration Diagrams zusammen.

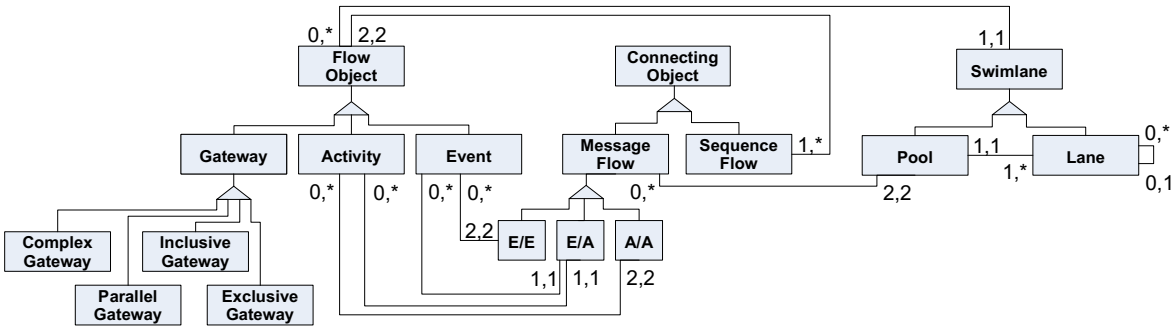



ABBILDUNG 2.16: BPMN-Metamodell (erweiterte Darstellung von Pütz und Sinz (2010, S. 49))

#### Process Diagram

Ein Process Diagram beschreibt durch Sequence Flows verknüpfte Activities innerhalb einer Organisation (OMG 2014, S. 143 ff.). Ein Diagramm kann dabei einen abgegrenzten Ausschnitt der Organisation beschreiben, der beispielsweise einer Organisationseinheit entspricht. Die Beschreibung kann damit als intraorganisational

und verhaltensorientiert charakterisiert werden. Mehrere Diagramme sind hinsichtlich ihrer Interaktionsbeziehungen zunächst nicht verknüpft. Eine Abgrenzung unter Angabe mehrerer interagierender betrieblicher Objekte erfolgt zunächst nicht.

**Collaboration Diagram**

Die Abläufe während der Interaktionen mehrerer Teilnehmer werden in einem Collaboration Diagram beschrieben (OMG 2014, S. 107 ff.). Ein Teilnehmer ist ein Participant, der beispielsweise einer Organisation oder Organisationseinheit zugehörig ist. Das Diagramm kann als interorganisationale und intraorganisationale Beschreibung des Verhaltens charakterisiert werden.

Organisationseinheiten bilden die für die gemeinsame Durchführung des Ablaufs notwendigen Aktivitäten in Flow Objekts eines Pools ab. Dem Metamodell nach ist ein Pool ein Strukturierungskonstrukt, das weiterhin in Lanes unterteilbar ist und zusammengehörige Flow Objects anhand des Connecting Object Sequence Flow verbindet. Der Austausch zwischen Pools basiert auf Nachrichten, die entlang von Beziehungen in Form von Message Flows übermittelt werden. Pools können aus Außensicht als Black Box (OMG 2014, S. 111) ohne Inhalt dargestellt werden.

Ein Subtyp des Collaboration Diagram ist das Conversation Diagram, welches den Austausch von Nachrichten zwischen Participants aggregiert als Conversation oder anhand einzelner Message Flows darstellt (OMG 2014, S. 123 ff.). Ein Anwendungsfall ist die Angabe zusammengehöriger Nachrichten in einem komplexen und potenziell lange andauernden Nachrichtenaustausch zwischen zwei Instanzen. Das Collaboration Diagram ist strukturorientiert, bietet allerdings keine Möglichkeit einer typerhaltenden mehrstufigen Darstellung.

**Choreography Diagram**

Die Choreographie modelliert den Austausch derjenigen Nachrichten zwischen mehreren Teilnehmern, die für den Gesamtablauf aus Sicht aller Teilnehmer zur Koordination der Interaktionen relevant sind (OMG 2014, S. 315). Im Gegensatz dazu stellt das Process Diagram eine Orchestrierung dar, die sich auf die Innensicht einer teilnehmenden Organisationseinheit bezieht. Die Notation einer Choreographie verwendet als primäres Syntax-Element Choreography Activities, die anhand von Sequence Flows in Beziehung stehen und außerdem auf eine eingeschränkte Event- und Gateway-Syntax (OMG 2014, S. 339, 344) zurückgreifen.

Eine Choreography Activity bildet einen Nachrichtenaustausch zwischen mindestens zwei Teilnehmern in Form eines Message Exchange Patterns (MEP) ab. Ein

grundlegendes Beispiel ist die Interaktion zwischen einem dienstanfragenden und einem dienstbereitstellenden Teilnehmer, die anhand eines Request-Response-MEP in einer Choreography Activity angegeben wird. Darüber hinaus sind komplexe MEP definierbar, die beispielsweise Streams abbilden.

#### 2.2.5.2 Syntax und Notation

**Connecting Objects**

Die innerhalb des Metamodells dargestellten Connecting Objects Sequence Flow und Message Flow repräsentieren Kontrollflussbeziehungen bzw. Nachrichtenflüsse. Ein Sequence Flow ist eine gerichtete Beziehung zur Angabe des Kontrollflusses zwischen den im Metamodell vermerkten Flow Objects. Im Sinne von Tokens, die in Analogie zu Petri-Netzen für die Definition der Ausführungssemantik herangezogen werden, bilden Sequence Flows den sequenziellen Transport von Tokens ab. Ein Message Flow ist eine gerichtete Beziehung zur Übertragung von Nachrichten zwischen Elementen, die diese senden oder empfangen können. Hierzu zählen Pools, oder aber eigenständig notierte Flow Objects wie Events des Typs Message Event oder Activities mit Event Handling.

**Flow Objects**

Flow Objects umfassen entsprechend des angegebenen Metamodells die Subtypen Activity, Event und Gateway. Diese sind jeweils Basistypen für spezialisierte Subtypen.

*Activity*

Activities sind entsprechend des Metamodells Flow Objects, zu denen außerdem Events zur Auslösung des Kontrollflusses für eine Instanz und Gateways für die Spezifikation von Verzweigungen und Zusammenführungen des Kontrollflusses gehören. Die Auslösung des Ablaufs wird durch ein Start Event initiiert und mit einem End Event abgeschlossen.

Eine Activity untergliedert sich in die Subtypen Task und Sub Process. Task ist als Subtyp von Activity eine elementar auszuführende Aktion, die zudem typisierbar ist. Zur hierarchischen Gliederung von Prozess-Teilen gliedert ein Sub Process einen Prozessteil innerhalb einer Activity ein. Marker können in Tasks und Sub Processes Wiederholungen oder eine Mehrfachinstanziierung angeben. Die Notation von Start- und End-Events kann im Falle eines Sub Process entfallen.

*Event*

Neben Start Event und End Event werden weiterhin dazwischenliegende Intermediate Events unterschieden. Weiterhin besteht die Möglichkeit ein Event Handling als Teil einer Activity ohne separat notiertes Event anzugeben, z.B. zur Auslösung eines Sub Process. Events besitzen einen Typ, der die Art des Ereignisses beschreibt, z.B. Message Events zur Definition nachrichtenbasierter Ereignisse oder Timer Events für zeitbasierte Ereignisse. Die Auslösung eines Ereignisses erfolgt beispielsweise zeitbasiert oder durch ausgetauschte Nachrichten. Dabei werden Catch Events für die Behandlung eintretender Ereignisse, z.B. für den Nachrichtenempfang, sowie Throw Events für die Auslösung von Ereignissen unterschieden, z.B. für den Nachrichtenversand. Events können zudem als Interrupting bezeichnet werden, wenn sie die Ausführung des Kontrollflusses unterbrechen, oder andernfalls als Non-Interrupting.

*Gateway*

Verzweigungen und Zusammenführungen des Kontrollflusses werden anhand von Gateways fünf verschiedener Subtypen realisiert. Deren Ausführungssemantik wird anhand von Tokens analog zu Petri-Netzen beschrieben. Ein Exclusive Gateway (1.) definiert eine XOR-basierte Ausführungssemantik, bei der ein Token genau einen Sequence Flow entsprechend einer definierten Bedingung wählt bzw. in einem von mehreren Zweigen zusammengeführt wird. Ein Parallel Gateway (2.) entspricht einem logischen AND, bei dem Tokens parallel in alle verknüpften Beziehungen verzweigt bzw. aus allen verknüpften Beziehungen synchronisiert werden. Ein Inclusive Gateway (3.) realisiert ein logisches OR, bei dem Tokens parallel in alle Verzweigungen fortgeführt werden, für die eine jeweils anzugebende Bedingung zutrifft. Im Falle einer Zusammenführung können die Bedingungen ein synchrones oder nicht-synchrones Verhalten definieren. Ein Complex Gateway (4.) spezifiziert ein beliebiges Synchronisationsverhalten des Kontrollflusses durch Angabe der zur Aktivierung notwendigen Tokens in eingehenden Sequence Flows. Bedingungen in ausgehenden Sequence Flows bestimmen analog zum Inclusive Gateway die anschließend verfolgten Flussbeziehungen. Alternativ zu explizit formulierten Bedingungen können Default Flows definiert werden, die aktiviert werden, sofern keine der anderen Bedingungen zutrifft. Ein Event-Based Gateway (5.) beschreibt eine Aktivierung eines Kontrollflusses durch das Eintreten von Events, wobei zudem entsprechend des Subtyps des Gateways zwischen exklusiv und parallel erwarteten Nachrichten unterschieden wird. Der Basistyp Event-Based Gateway spezifiziert

eine Verzweigung in mehrere Events, die z.B. unterschiedliche Nachrichten empfangen und die ggf. die damit zusammenhängenden Kontrollflüsse aktivieren. Der Subtyp Exclusive Event-Based Gateway erfordert ein eintretendes Event zur Aktivierung, während der Subtyp Parallel Event-Based Gateway das Eintreten mehrerer Events zur Aktivierung benötigt.

#### 2.2.5.3 Transformation von Geschäftsprozessmodellen des SOM in BPMN-Workflow-Schemata

Die Modellierung von Geschäftsprozessen betrifft die Modellebene betrieblicher Aufgaben, die zur Überführung in Workflow-Schemata hinsichtlich ihrer Innensicht beschrieben, zu Vorgängen verbunden und in Vorgangsnetzen verknüpft werden. Ein Workflow-Schema legt auf der Ebene der Aufgabenträger fest, welche elementaren Aktionen in welcher Abfolge von personellen oder maschinellen Aufgabenträgern durchzuführen sind, um die zu einem Workflow gehörenden Vorgänge auszuführen (siehe Kapitel 2.1.4.2).

Ein SOM-Geschäftsprozessmodell bildet die zuvor strukturell erfassten betrieblichen Aufgaben als Vorgänge unter Beteiligung mehrerer betrieblicher Objekte innerhalb des VES ablauforientiert ab. Die BPMN definiert ablauforientierte Diagramme zur Erfassung des Zusammenwirkens mehrerer Teilnehmer (Participants) als Collaboration Diagram, das eine Spezifikation elementar auszuführender Activities erlaubt.

Eine Metamodelltransformation von SOM-Geschäftsprozessmodellen in BPMN-Workflowschemata (Pütz und Sinz 2010) überführt alle auf das VES bezogene Syntaxelemente des Metamodells für SOM-Geschäftsprozessmodelle in Elemente des Collaboration Diagram entsprechend eines BPMN-Metamodells. Das Ziel ist die Erstellung syntaktisch und semantisch korrekter BPMN-Schemata, deren Ausführungssemantik sich von der eines VES unterscheidet und daher je Element sowie aus Sicht des Kontrollflusses betrachtet wird.

Die Transformation betrieblicher Objekte führt zu Pools verschiedener Participants, die den öffentlichen Teil des Prozesses (Public Process) je Diskursweltobjekt enthalten oder im Falle von Umweltobjekten ohne Inhalt als Black-Box-Pool dargestellt sind. Die Aufgaben eines Objekts werden in ein Start Event je Pool sowie in je eine Activity überführt. Zur semantisch korrekten Überführung ist das VES zuvor so weit zu detaillieren, dass eine Aufgabe einer elementar beschreibbaren Activity entspricht (Pütz und Sinz 2010). Betriebliche Transaktionen modellieren Interaktionen

zwischen betrieblichen Objekten, die analog zur nachrichtenbasierten Kommunikation in SOM als Message Flow zwischen den Pools der entsprechenden Participants ausgetauscht werden.

Die Transformation des Kontrollflusses überführt zunächst die Ereignisbeziehungen von O-Ereignissen in Sequence Flows. Auf O-Ereignisse bezogene Bedingungen von Pre- und Post-Conditions determinieren zudem eine Abbildung auf Gateways. Post-Conditions mit IF-ELSE-Semantik führen zu verzweigenden Exclusive Gateways, die anhand von Pre-Conditions mit OR-verknüpften Teilausdrücken in einem oder mehreren sequenziellen Exclusive Gateways wieder zusammengeführt werden. AND-Verknüpfte Teilausdrücke führen stets zu verzweigenden oder zusammenführenden Parallel Gateways (Pütz und Sinz 2010). Die Transformation des Kontrollflusses wird im Folgenden verallgemeinert und um weitere Gateway-Elemente der BPMN ergänzt.

Boole'sche Logik in den Bedingungen von Post-Conditions bestimmt die Abbildung auf verzweigende Gateways, während diese in ebensolchen Bedingungen von Pre-Conditions die Abbildung auf zusammenführende Gateways determiniert. Die Bedingungen von IF-ELSE- und IF-ELSEIF-Ausdrücken werden als Disjunktion dargestellt. Eine weitere Unterscheidung zwischen Zusammenführungen und Verzweigungen ist nicht erforderlich.

Die Transformation von Bedingungen umfasst:

- OR-verknüpfte disjunktive Ausdrücke werden auf Inclusive Gateways abgebildet. Die Operanden führen zu je einer Bedingung in den Sequence Flows aller Zweige.
- AND-verknüpfte konjunktive Ausdrücke führen zu Parallel Gateways.
- XOR-verknüpfte Literale werden auf Exclusive Gateways abgebildet.

Die Transformationen sind unter Beibehaltung der Evaluationsreihenfolge logischer Ausdrücke mehrfach anwendbar.

## 2.3 Kooperative Geschäftsprozesse

Das vorliegende Kapitel diskutiert Grundlagen und Methoden zur Kooperation in Geschäftsprozessen hinsichtlich der Planung und Durchführung unter Mitwirkung mehrerer verteilter Teilnehmer. Neben Konzepten zur Kooperation in Prozessen werden Ansätze für eine auf Kooperation beruhende Entwicklung anhand von Modellen eingeführt.

### 2.3.1 Grundlagen zur Kooperation in Geschäftsprozessen

#### 2.3.1.1 Einführung

In einem Geschäftsprozess werden Leistungen anhand eines determinierten Ablaufs zielgerichteter Aktivitäten unter dem Einsatz zugeordneter Ressourcen erstellt (siehe Abschnitt 2.1.4). Die Entwicklung und Ausführung von Prozessen in dezentral organisierten betrieblichen Systemen betrifft Geschäftsprozesse, in denen das Zusammenwirken verteilter Organisationseinheiten zur Erstellung von Leistungen im Vordergrund steht.

Kooperative Geschäftsprozesse werden innerhalb des vorliegenden Abschnitts auf Grundlage des Geschäftsprozessbegriffs entwickelt. Definitorisch werden damit spezielle Geschäftsprozesse abgebildet, welche auf dem Begriff des Geschäftsprozesses beruhen und diesen einschränken. Das Merkmal der Leistungserstellung und die durch Inputs und Outputs definierbare Wertschöpfung eines Prozesses werden hier in Kooperationen gemeinsam von mehreren beteiligten Aufgabenträgern und weiteren Ressourcen erbracht. Die Zuordnung einzusetzender Aufgabenträger und Ressourcen muss im Falle interorganisationaler Kooperationen eine entsprechende Differenzierung, z.B. nach Unternehmensgrenzen, vornehmen.

Der vorliegende Abschnitt führt das Thema basierend auf dem Kooperationsbegriff ein. Die aus verteilten Umgebungen hervorgehenden Implikationen für die Wertschöpfung in kooperativen Prozessen bespricht Abschnitt 2.3.2, um die Anforderungen an eine spätere Konstruktion von dezentralen Systemen zu erfassen. Die Modellierung von kooperativen Prozessen ist Gegenstand von Abschnitt 2.3.3. Zur Zusammenführung der Modellierung und der Anforderungen im Kontext der Verteilung bespricht Abschnitt 2.3.4 Ansätze zur Entwicklung und Anpassung der Prozess-Struktur. Abschnitt 2.3.5 fasst abschließend existierende Konzepte für den Austausch und die Verteilung von Modellen zusammen.

#### 2.3.1.2 Kooperationsbegriff

Kooperationen sind Unternehmensverbindungen, zu deren Zielen das interne und externe Wachstum, das Erzielen von Synergieeffekten und die Risikostreuung durch Diversifikation zählen (Thommen et al. 2017, S. 32 f.). Ein erweiterter Kooperationsbegriff schließt im Kontext der Wertschöpfung neben Unternehmen zudem Kunden ein (Piller et al. 2017, S. 9).

**Klassifikation**

Klassifiziert nach Produktionsstufen ergibt sich die folgende Gliederung verschiedener Kooperationsarten (Thommen et al. 2017, S. 32 f.; Hungenberg 1999, S. 6):

- Horizontale Kooperationen werden innerhalb einer Produktionsstufe geschlossen, wie etwa im Falle von Unternehmensverbindungen zwischen produzierenden Automobilherstellern, die innerhalb einer Branche gleiche Produkte herstellen. Teilnehmende Unternehmen schließen sich auf strategischer Ebene zusammen oder bündeln auf operativer Ebene Ressourcen.
- Vertikale Kooperationen realisieren sequenziell aufeinander folgende Produktionsstufen durch unterschiedliche Teilnehmer der Kooperation. Ein Beispiel sind Kooperationen zwischen Automobilzulieferern und Automobilherstellern. Es wird weiterhin zwischen einer Rückwärtsintegration zur Angliederung vorgelagerter Produktionsstufen, sowie einer Vorwärtsintegration zur Angliederung nachgelagerter Produktionsstufen unterschieden.
- Laterale Kooperationen liegen vor, wenn Unternehmensverbindungen mit Teilnehmern geschlossen werden, die nicht in einer Wertschöpfungsbeziehung zueinander stehen und typischerweise unterschiedlichen Branchen angehören. Ein Beispiel ist ein gemeinsames Produktangebot von Automobil- und Smartphone-Herstellern.

Hungenberg (1999, S. 6) bezeichnet laterale Kooperationen als konglomerate Kooperationen und weist auf Nutzeffekte der gemeinsamen Vermarktung von Produkten dieser Kooperationsform hin.

### 2.3.2 Kooperation im Kontext der Wertschöpfung

Auf Kooperationen basierende Geschäftsprozesse realisieren innerhalb von Netzwerken eine in mehreren Stufen stattfindende Wertschöpfung. Die als Ergebnis vorliegende Leistungserstellung wird durch das Zusammenwirken der Beteiligten erbracht. Beteiligte sind sämtliche Teilnehmer von Geschäftsprozessen in intra- und inter-organisationalen Kooperationen sowie in daraus entstehenden Netzwerken zur Wertschöpfung (Piller et al. 2017, S. 8; Ferstl und Sinz 2013, S. 91 f.; Thommen et al. 2017, S. 470 f.).

#### 2.3.2.1 Grundlagen

Teilnehmer eines Geschäftsprozesses sind klassischerweise Unternehmen, die intra- und interorganisational kooperieren, und unter dem Einfluss zunehmender Vernetzung globale interorganisationale Wertschöpfungsnetze bilden. Die marktkoordinierte Organisation der Leistungserstellung in Netzwerken beschreibt ein Leitbild der Wertschöpfung, zu dessen Auswirkungen auch die Auflösung der Grenzen betrieblicher Systeme und die damit entstehende globale Skalierbarkeit zählen (Picot et al. 2003). Im Gegensatz zur hierarchischen Organisation des Taylorismus beruhen marktkoordinierte Netzwerke auf beliebigen Verknüpfungen von Zulieferund Abnehmerbeziehungen. Weiterentwicklungen des Wertschöpfungsbegriffs beziehen Kunden oder Nutzer einer Leistung kooperativ und interaktiv in die Leistungserstellung ein (Piller et al. 2017, S. 9 f.). Eine Manifestierung ist die Individualisierung, die individuelle Produktanpassungen unter Beteiligung von Kunden in der Endphase des Produktentwicklungsprozesses gestattet.

#### 2.3.2.2 Value Co-Creation

Eine Grundlage der Entwicklung hin zu partizipativen und interaktiven Prozessen der Wertschöpfung ist eine Erweiterung des Wertschöpfungsbegriffs, oder Value-Creation-Begriffs, zur Value Co-Creation. Value Co-Creation bezieht Merkmale der gemeinsamen Leistungserstellung in den Wertschöpfungsprozess ein (Prahalad und Ramaswamy 2004; Redlich, Moritz und Wulfsberg 2019).

Unternehmen agieren als Plattform zur Organisation der Leistungserstellung (Prahalad und Ramaswamy 2004). Merkmale der entstehenden Systeme sind Offenheit, Interaktion, Kooperation und Zusammenwirken hinsichtlich der Leistungserstellung. Eine weitergehende Diskussion von Merkmalen findet sich bei Vorbach et al. (2016, S. 287).

Auf dieser Basis bilden sich nunmehr Prozesse heraus, in denen die klassischen Rollen von Unternehmen und Kunden nicht mehr ausschließlich anhand der Leistungserstellung unterschieden werden können. Die Integration von Kunden in die Leistungserstellung wird in geringem Maße in Trends wie Mass Customization von Produkten und Customer Self Service sichtbar, sowie in hohem Maße in digitalen Produkt-Service-Kombinationen der Sharing Economy (z.B. Uber, AirBnB, Zipcar) und einer bis zur „Losgröße 1“ fortschreitenden Individualisierung (Bogner et al. 2018).

Insbesondere die Leistungserstellung der Internetökonomie ist durch diese Konzepte abbildbar. Die Rollenzuordnung von Kunden und Unternehmen verschwimmt zunehmend, beispielsweise in Plattformen wie Quirky (Redlich und Moritz 2016) oder Crowdfunding-Unternehmen wie Kickstarter, die „Kunden“ in die Produktentwicklung einbeziehen, oder in offene Produktionsprozesse (Gershenfeld 2007; Troxler 2016).

#### 2.3.2.3 Wertschöpfung im Kontext zunehmender Vernetzung

Die zuvor betrachtete Wertschöpfungsform setzt voraus, dass beteiligte Unternehmen und Kunden über Kommunikationstechnologien vernetzt sind. Grundlegende Strömungen der auf Kooperation und Interaktion basierenden Leistungserstellung unter Ausnutzung der zunehmenden Vernetzung anhand von Kommunikationstechnologien sind bei von von Hippel (2005) und Benkler (2002) zu finden.

- von Hippel (2005) sowie Baldwin und von Hippel (2010) beschreiben eine durch Zusammenarbeit entstehende und durch Offenheit geprägte Innovation, die in erster Linie in Herstellungsprozessen von Software- und Informationsprodukten, aber auch in physischen Produkten vorzufinden ist. Prozesse dieser Art werden von Communities getragen. Von Kunden oder Nutzern ausgehende Innovationen fließen in offenen Innovationsprozessen zurück in Unternehmen und gehen in die Produktion von materiellen oder immateriellen Gütern ein.
- Benkler (2002) beschreibt eine auf der Basis von Informationsnetzwerken koordinierte „Information Economy“, in der selbstorganisierte Teilnehmer in offenen Prozessen zur Produktion von Informationen zusammenwirken (Benkler und Nissenbaum 2006). Der als „Commons-Based Peer Production“ (CBPP) bezeichnete Produktionsprozess stellt miteinander kommunizierende Teilnehmer in den Vordergrund. Hilgers et al. (2010) untersuchen die Anwendbarkeit von CBPP auf die Entwicklung von materiellen Gütern, die für

die drei dort untersuchten Fallstudien angenommen wird. Kreiss et al. (2011) weisen auf Grenzen der Effizienz sowie auf soziale und ethische Implikationen hin.

Beide Strömungen manifestieren sich in der Entwicklung von Open-Source-Software und der Entstehung von offenen, partizipativen und interaktiven Projekten wie Wikipedia. Die tradierten Rollen von Unternehmen und Kunden treten dort in den Hintergrund.

#### 2.3.2.4 Dezentrale Wertschöpfung

Redlich, Wulfsberg und Bruhns (2010) definieren eine auf Co-Creation basierende Theorie einer dezentralen und vernetzten Wertschöpfung als „Bottom-up Economics“ (Redlich, Moritz und Wulf 2017; Redlich, Moritz und Wulfsberg 2019; Redlich und Wulfsberg 2011). Diese greift Offenheit als zentrales Merkmal erneut auf, welches die Theorie hinsichtlich der Dimensionen „Architektur des Wertschöpfungsartefakts“, „Prozess“ und „Systemstruktur“ adressiert (Redlich, Moritz und Wulf 2017, S. 164):

- Architektur des Wertschöpfungsartefakts: Artefakte sind (a.) Private Güter exklusiver Verfügungsrechte sowie (b.) öffentliche Güter unter Open-Source-Verfügungsrechten.
- Prozess: Aktivitäten des Prozesses werden als Co-Aktivitäten gemeinsam ausgeführt. Die Co-Aktivität umfasst in ihrer Tiefe (a.) die Koordination als Integration der Beteiligten, (b.) die Kooperation bei zusätzlich vorliegender Partizipation sowie (c.) die Collaboration bei zusätzlich vorliegender Interaktion der Beteiligten.
- Systemstruktur: Die Struktur des Systems ist interorganisational und unterstützt Organisationsstrukturen, die (a.) hierarchisch und (b.) „adhokratisch“ selbst-organisierend sind.

Die dezentrale Wertschöpfung betrifft offene Organisationsnetzwerke, deren verteilte Teilnehmer ohne Intermediation direkt untereinander vernetzt sind, um Leistungen als Co-Creation oder Co-Aktivität zu erstellen. Die Beziehungen der Leistungserstellung wandeln sich von marktorientierten Erstellung-Abnahme-Beziehungen zu interaktiven und kooperativen Erstellung-Erstellung-Beziehungen der gemeinsamen Erstellung. Die korrespondierenden Architekturformen der hierfür geeigneten Informationssysteme gehen hinsichtlich ihrer Verteilung und Rollenzuordnung von Client-Server zu Peer-to-Peer über. Diese Architektur wird durch

Blockchain-Systeme (siehe Kapitel 3) anhand von Protokollen zur dezentralen Koordination unterstützt. Die Verknüpfung der global vernetzten Wertschöpfung auf Grundlage der Vernetzung und der dezentralen Wertschöpfung unter Hinzunahme von Peer-to-Peer- und Blockchain-Technologien führt zu einem System aus unmittelbaren Leistungsbeziehungen innerhalb eines globalen Netzwerks. Eine Konsequenz ist eine geringere Bedeutung zentraler Intermediationen durch einzelne Teilnehmer oder Plattformen.

### 2.3.3 Modellierung von kooperativen Geschäftsprozessen

#### 2.3.3.1 Einordnung

Gegenstand dieses Abschnitts ist die Modellierung kooperativer Geschäftsprozesse anhand von etablierten Ansätzen zur Geschäftsprozess- und Workflow-Modellierung. Dabei wird zunächst die Kooperation oder *Collaboration* in der Leistungserstellung betrachtet, die in Prozessen und ihren Modellen erfassbar ist. Ansätze zur Erstellung von Modellen in Kooperation sind als *Collaborative Modeling* den zu dem Ansatz dieser Arbeit verwandten Arbeiten in Kapitel 2.4 zugeordnet.

#### 2.3.3.2 Charakteristika der Modellierung kooperativer Geschäftsprozesse

Ein kooperativer Geschäftsprozess begründet sich durch die kooperative Zusammenarbeit mehrerer Objekte. Für die Modellierung ergeben sich spezifische Charakteristika, die auf die Durchführung des Prozesses in Zusammenarbeit und den interorganisationalen Charakter Bezug nehmen. Die Modellierung kooperativer Geschäftsprozesse wird im Folgenden hinsichtlich der verwendeten Modellarten, des Vorgehens zur Modellbildung, der Teilnehmer sowie notwendigerweise zu erfüllenden Kriterien diskutiert. Tabelle 2.2 zeigt hierfür relevante Ausprägungen.

| **Dimension** | **Ausprägungen** | | |
|---|---|---|---|
| **Abstraktionsebene** | Intra-Organisational | Inter-Organisational | |
| **Prozess** | Private Process | Public Process | |
| **Modell** | Private Model | Public Model | Choreography Model |
| **Abgrenzung** | Unternehmen | Kooperation | Netzwerk |
| **Objekt** | Organisationseinheit | Unternehmung | Kooperation |

TABELLE 2.2: Merkmale von kooperativen Geschäftsprozessen

Für die auf Kooperation beruhenden Prozesse wird häufig der Begriff des *Collaborative Business Process* (CBP) herangezogen (Fdhila et al. 2015; Chengfei Liu et al. 2009; Niehaves und Plattfaut 2011). Die Modelle eines CBP gliedern sich je Teilnehmer in (Fdhila et al. 2015) die Modellarten:

1. Privates Modell (Private Model) zur Abbildung der internen und privaten Geschäftslogik (business logic) und der mit Partnern auszutauschenden Nachrichten,
2. Öffentliches Modell (Public Model), für die Beschreibung nach außen sichtbarer Aktivitäten und auszutauschender Nachrichten mit Abfolgebeziehungen und
3. Choreographie-Modell (Choreography Model), um die nachrichtenbasierten Interaktionen zwischen allen Prozessteilnehmern abzubilden.

Das Vorgehen zur Modellbildung verläuft Top Down oder Bottom Up. In einer Top Down Collaboration geht die Modellbildung von einer Choreographie aus, die öffentliche Modelle und, hieran anschließend, dazu kompatible private Modelle ableitet. In einer Bottom Up Collaboration beginnt die Entwicklung mit privaten Prozessen und öffentlichen Sichten je Teilnehmer, um anschließend im Austausch mit anderen Teilnehmern ein öffentliches Modell zu beschreiben.

Die Teilnehmer eines CBP wirken zweckbezogen zusammen, um den Prozess und die damit verbundene Leistungserstellung durchzuführen. Die Bestimmung von Teilnehmern kann zur Gestaltungszeit oder zur Laufzeit erfolgen (Huemer et al. 2008). Die Teilnehmerauswahl zur Laufzeit wird in Prozessen mit erhöhten Flexibilitätsanforderungen benötigt, z.B. hGP. Möglichkeiten des Hinzukommens neuer Teilnehmer und des Ausscheidens existierender Teilnehmer zur Laufzeit werden im Falle eines hohen Strukturierungsgrades aufgrund von Abhängigkeiten zwischen den Aktivitäten unterschiedlicher Teilnehmer beeinträchtigt.

Von einem CBP sind drei Kriterien notwendigerweise zu erfüllen (Fdhila et al. 2015):

1. Consistency: Die in einem privaten Modell beschriebene interne Implementierung eines Prozesses stimmt mit dem von außen beobachtbaren Verhalten des öffentlichen Modells überein.
2. Compatibility: Prozesse unterschiedlicher Teilnehmer sind in ihrer Struktur und in ihrem Verhalten aufeinander abgestimmt.
    - Hinsichtlich der Struktur bestehen Möglichkeiten zum gegenseitigen Austausch von Nachrichten zwischen den Teilnehmern.

- Hinsichtlich des Verhaltens ermöglichen die anhand von Kontrollflüssen festgelegten Abfolgen aufeinander abgestimmte Nachrichtenaustausche.

3. Realizability: Für jeden Teilnehmer ist ein Prozess formulierbar, der mit der Choreographie übereinstimmt.

Die genannten Kriterien fordern die Abstimmung der öffentlichen und privaten Modelle von Teilnehmern und zwischen Teilnehmern, sowie deren Realisierbarkeit.

#### 2.3.3.3 Modelle und Schemata

Existierende Modellierungssprachen müssen die Erstellung der im vorhergehenden Abschnitt genannten Modelle unterstützen. Für die Sprachen von BPMN (Abschnitt 2.2.5) und SOM (Abschnitt 2.2.4) können die in Tabelle 2.3 angegebenen Schemata zur Erfassung der im vorherigen Abschnitt diskutierten Modelle herangezogen werden.

| | | BPMN | SOM |
|---|---|---|---|
| **Private Model** | **Struktur** | - | IAS in Zerlegungsstufe m |
| | **Verhalten** | Process Diagram, Collaboration Diagram | VES in Zerlegungsstufe m |
| **Public Model** | **Struktur** | - | IAS in Zerlegungsstufe n mit n < m |
| | **Verhalten** | Collaboration Diagram | VES in Zerlegungsstufe n mit n < m |
| **Choreography Model** | **Struktur** | Conversation Diagram | IAS in Zerlegungsstufe n mit n < m |
| | **Verhalten** | Choreography Diagram | VOS der Zerlegungsstufe n |

TABELLE 2.3: Modelle und Schemata kooperativer Prozesse

**Business Process Model and Notation**

BPMN erfasst Prozesse innerhalb des Private Model und des Public Model ausschließlich hinsichtlich des Verhaltens, da die hierfür verwendeten Schemata Abläufe anhand von Sequence Flows je Pool abbilden. Ein Process Diagram (OMG 2014, S. 143) modelliert den Ablauf eines Teilnehmers in einem Private Model. Ein Collaboration Diagram stellt entweder einen Teilnehmer zusammen mit den öffentlich bekannten Prozessteilen weiterer Teilnehmer als Private Model dar oder zeigt die öffentlichen Prozesse aller Teilnehmer als Public Model (siehe z.B. Fdhila et al.

2015). Eine Choreographie kann in ihrer Struktur als Conversation Diagram erfasst werden, das die Beziehungen für den Austausch von Nachrichten (OMG 2014, S. 124) als Conversation beschreibt. Eine Conversation aggregiert Message Flows, die in disaggregierter Form ebenfalls Teil der Darstellung sein können. Das Verhalten einer Choreographie bildet das Choreography Diagram ab, indem Reihenfolgebeziehungen zwischen auszutauschenden Nachrichten als Sequence Flows bzw. Choreography Tasks abgebildet werden (OMG 2014, S. 317, 323). Die Beziehungen nehmen auf Nachrichtenaustauschmuster Bezug.

**Semantisches Objektmodell**

SOM bildet Prozesse innerhalb des Private Model und des Public Model jeweils hinsichtlich der Systemmerkmale Struktur und Verhalten ab. Die Darstellung der Struktur als Private Model kann durch ein IAS einer Zerlegungsstufe $m$ beschrieben werden, die aus einer $m$-Mal durchgeführten hierarchischen Zerlegung hervorgeht (Ferstl und Sinz 2013, S. 207 ff.). Die Zerlegungsstufe $m$ ist erreicht, wenn die betrieblichen Objekte des IAS die Innensicht eines Prozessteilnehmers abbilden. Das analog erstellte VES der Zerlegungsstufe $m$ beschreibt das Verhalten der Prozesse des Teilnehmers anhand von betrieblichen Aufgaben, Transaktionen und Ereignissen. Aufgrund der hierarchischen Zerlegung können die Schemata IAS und VES ebenso für die Struktur- und Verhaltensabbildung des Public Model herangezogen werden. Ein Public Model kann damit als Private Model einer vorhergehenden Zerlegungsstufe $n$, mit $n < m$, interpretiert werden. Zur Abbildung der Struktur einer Choreographie anhand der für den Austausch von Nachrichten erforderlichen Beziehungen kann auf das IAS der Zerlegungsstufe $n$ zurückgegriffen werden. Dort werden Nachrichtenaustausche abgebildet. Die innerhalb des IAS definierten betrieblichen Transaktionen beschreiben eine Choreographie durch Kommunikationskanäle und die Typen übertragbarer Nachrichten (Ferstl und Sinz 2013, S. 203 f.). Abhängigkeiten und Reihenfolgebeziehungen gehen aus den Typisierungen und aus den während der Zerlegung gewählten Koordinationsformen hervor. Die Verhaltenssicht einer Choreographie leitet sich aus dem VES der Zerlegungsstufe $n$ ab und wird als Teil der Anwendungssystemspezifikation innerhalb des VOS erfasst. Das VOS beschreibt die Nachrichtenaustausche der Objekte von Stufe $n$ mit deren Nachrichtendefinitionen, die auf Operatoren der Objekte abgebildet werden (Ferstl und Sinz 2013, S. 233).

#### 2.3.3.4 Elemente der Modell-Syntax

Die in den zuvor besprochenen BPMN-Diagrammen und SOM-Schemata zur Verfügung stehende Syntax für die Modellierung kooperativer Prozesse ist in Tabelle 2.4 dargestellt. Die hierfür spezifischen Charakteristika (siehe Abschnitt 2.3.3) erfordern grundlegend eine unterscheidbare Abbildung öffentlicher und privater Prozessarten sowie eine Darstellung von Interprozess-Beziehungen anhand von Nachrichtenaustauschen und Nachrichtenaustauschmustern. Hierzu zu unterscheiden sind die Kontrollflüsse innerhalb eines Prozesses.

| | | BPMN | SOM |
|---|---|---|---|
| **Prozessart** | **Public Process** | Pool, Lane | Diskurswelttobjekt und Zerlegungsprodukte |
| | **Private Process** | Black Box Pool | Umwelttobjekt |
| **Interprozess-Beziehung** | **Nachrichten-austausch** | Message Flow | Transaktion, interacts_with |
| | **Nachrichten-austauschmuster** | Choreography Task | Vorgangsobjekttyp (VOT) |
| **Kontrollfluss-Beziehung** | | Sequence Flow | Aufgabe-Ereignis-Beziehung |

TABELLE 2.4: Syntax-Elemente zur Abbildung von kooperativen Prozessen

**Business Process Model and Notation**

Die in BPMN verwendete Syntax des Process Diagram und Collaboration Diagram greift auf das Pool-Element zurück, das Prozesse eines abgegrenzten Teilnehmers enthält und weiterhin anhand von verschachtelten Lane-Elementen gegliedert werden kann. Zur Modellierung eines Public Process werden Pools und Lanes mit den dort enthaltenen Flow Objects als „White Box“ gezeigt. Die Modellierung eines „Private Process“ verwendet nur das Syntax-Element des Pools, dessen Innensicht als „Black Box“ verborgen ist. Zwischen zwei Prozessen repräsentiert das Beziehungselement Message Flow eine Nachrichtenaustauschbeziehung. Message Flows können zudem Teil des Conversation Diagram sein. Weiterhin werden Nachrichtenaustauschmuster anhand des Elements Choreography Task abgebildet, das Teil des Choreography Diagram ist. Kontrollfluss-Beziehungen innerhalb von Pools verwenden das Sequence-Flow-Element.

**Semantisches Objektmodell**

Das SOM sieht anhand des Metamodells für Geschäftsprozessmodelle eine syntaktische Trennung zwischen betrieblichen Objekten der Diskurswelt und der Umwelt vor. Ein Public Process kann initial anhand eines Diskursweltobjekt-Elements modelliert werden, das hinsichtlich der darin öffentlich sichtbaren Aufgaben typerhaltend zerlegt wird. Die entstehenden Zerlegungsprodukte bilden etwa Organisationseinheiten oder Teilnehmer des Prozesses als Diskursweltobjekt-Elemente ab. Die in einem Private Process verborgenen betrieblichen Aufgaben sind nicht Teil der betrachteten Prozessmodellierung und werden in Umweltobjekt-Elementen dargestellt, die keine weitere Zerlegung erfordern. Einen Kanal für den Nachrichtenaustausch stellt das Syntax-Element der betrieblichen Transaktion dar. Transaktionen verlaufen zwischen den in betrieblichen Objekten abgebildeten Prozessen. Kontrollfluss-Beziehungen innerhalb von betrieblichen Objekten werden ereignisbasiert anhand einer Verknüpfung der Syntax-Elemente U-Ereignis und O-Ereignis mit dem Element der betrieblichen Aufgabe festgelegt. Nachrichtenaustauschmuster sind als Teil der Anwendungssystemspezifikation anhand von Vorgangsobjekttypen (VOT) abbildbar (Teusch 2016; Teusch und Sinz 2012). Mehrere durch die Beziehungsart *interacts_with* verknüpfte VOT bilden ein Nachrichtenaustauschmuster ab.

#### 2.3.3.5 Abstraktionsebenen kooperativer Prozesse

Public Process und Private Process sind unter Berücksichtigung der diskutierten Syntax-Elemente anhand der Modellierungssprachen von BPMN und SOM prinzipiell abbildbar. Zur Abbildung kooperativer Prozesse werden zudem öffentliche Modelle (Public Model) einer Kooperation herangezogen, die öffentlich sichtbare intraorganisationale und interorganisationale Prozesse enthalten und ggf. weitere nicht-öffentlich sichtbare Prozessteile umfassen. Für die Abgrenzung von öffentlich und nicht-öffentlich sichtbaren inter- und intraorganisationalen Prozessen stehen unterschiedliche Strukturierungsmöglichkeiten zur Bildung von Abstraktionsebenen zur Verfügung, die in Tabelle 2.5 zusammengefasst sind.

**Business Process Model and Notation**

Die Erfassung von Geschäftsprozessen orientiert sich in BPMN an einer Reihe einzelner Teilnehmer (Participant), die jeweils anhand eines Pools zur Darstellung von Prozessabläufen notiert und in Lanes untergliedert werden. Syntaktische Elemente zur Gliederung zusammengehöriger Teilnehmer-Elemente (Pools) sind nicht vorhanden. Neben einer Unterteilung in Lanes können Sub-Prozesse (Sub Process) als

| | | BPMN | SOM |
|---|---|---|---|
| **Systemabgrenzung** | **Kriterium** | Teilnehmer | Objekt |
| **Abstraktionsebene** | **Inter-Organisational** | Pool, Black Box Pool | Diskurswelt, Umwelt |
| | **Intra-Organisational** | Lane (mehrstufig), Sub-Process | Diskursweltobjekte als Zerlegungsprodukte |

TABELLE 2.5: Abstraktionsebenen und Abgrenzung kooperativer Prozesse

hierarchische Gliederung des Ablaufs herangezogen werden, von denen die Strukturierung der Pools unberührt bleibt. Die Darstellung inter- und intraorganisationaler Prozesse in einem Public Model erfolgt damit innerhalb einer Abstraktionsebene, die durch die Festlegung der Teilnehmer bestimmt ist.

**Semantisches Objektmodell**

SOM geht von einer an organisationalen Koordinationsformen ausgerichteten Zerlegung der Diskurswelt in betriebliche Objekte aus. Letztere repräsentieren in Abhängigkeit der betrachteten Zerlegungsstufe etwa Unternehmungen oder Organisationseinheiten, sodass eine inter- bzw. intraorganisationale Abbildung der Prozesse zwischen den jeweiligen betrieblichen Objekten entsteht. Die Auswahl geeigneter Zerlegungsstufen orientiert sich an der für die Diskurswelt gewählten Abgrenzung.

## 2.3.4 Merkmale von kooperativen Geschäftsprozessen in Netzwerken

Geschäftsprozesse sind als Teil von Netzen miteinander verwoben. Der Strukturierungsgrad der beteiligten Prozesse wirkt sich dabei unmittelbar auf die gegenseitigen Abhängigkeiten zwischen den Prozessen und ihren Teilnehmern aus.

### 2.3.4.1 Strukturierungsgrade von Geschäftsprozessen und Workflows

Zur Unterscheidung von Strukturierungsgraden wird eine Klassifikation von (Nastansky und Hilpert 1994, S. 2) herangezogen, die zwischen Ad-hoc-Prozessen, schwach strukturierten Prozessen und strukturierten Prozessen unterscheidet. Die genannten Typen sind Teil eines Kontinuums.

**Ad-hoc-Prozesse**

Ad-hoc-Prozesse sind kurzfristig entstehende Prozesse, die ohne Vorab-Planung zur Laufzeit geplant und ausgeführt werden. In erster Linie werden hiermit Ausnahmefälle abgefangen oder einmal auftretende Prozesse für genau eine Instanz gebildet.

**Schwach-strukturierte Prozesse**

Schwach-strukturierte Prozesse sind vorwiegend für kooperative Abläufe von Belang, deren gemeinsame Planung durch mehrere Beteiligte keine vollständig strukturierte Beschreibung zulässt. Die geringe Strukturierung weisen offene Teamprozesse auf, in denen beliebig hinzutretende und ausscheidende Teilnehmer Einfluss auf eine durchzuführende Gesamtaufgabe nehmen, ohne deren Ablauf in detaillierten Einzelaktionen festzulegen. Ein integrierter Teamprozess geht von definierten Start- und Endpunkten aus, innerhalb derer eine festgelegte Teilnehmerzahl den Prozess und detaillierte Ablauffestlegung durchführt. Integrierte kooperative Aktivitäten sind Teil eines ansonsten strukturell definierten Prozesses. Werden solche Aktivitäten im Ablauf des Prozesses erreicht, kann deren Durchführung von einer beliebigen Anzahl von Akteuren selbst koordiniert und ausgeführt werden.

**Strukturierte Prozesse**

Strukturierte Prozesse lassen im Falle von Ad-hoc-Ausnahmen bei Vorliegen von Verhaltensflexibilität auf der Schema-Ebene alternative Ablaufvarianten zu, die als Teil eines Verhaltensrepertoires vordefiniert sind. Strukturflexibilität ist dabei keine Voraussetzung. Wohlstrukturierte Prozesse sind hinsichtlich ihrer Struktur vollständig spezifiziert. Sie bilden typischerweise Normalfälle der Haupt- und Unterstützungsprozesse eines Unternehmens ab.

#### 2.3.4.2 Flexibilität in Geschäftsprozessen

Ein Geschäftsprozess umfasst einen Ablauf von Aktivitäten, der als determiniert charakterisiert werden kann. Dieses Merkmal des determinierten Ablaufs (M1 und M2, Abschnitt 2.1.4.1) bezieht sich dabei nicht auf eine deterministische und starre Durchführung einer festgelegten Abfolge von Aktivitäten, sondern auf die Ausrichtung der Durchführung anhand des gegebenen Ziels. Im Falle eines nichtdeterministischen Ablaufs bestehen hinsichtlich der Zielerreichung Freiheitsgrade, die Potenziale für Flexibilität bedingen.

Der Begriff der Flexibilität steht für die „Fähigkeit eines Systems zur Änderung seines Verhaltens oder seiner Struktur“ (Sinz 2012, S. 9; Bartmann et al. 2011, S.2; Wagner, Suchan et al. 2011, S. 88). Eine Implikation dieser Definition ist die Differenzierung zwischen Verhaltens- und Strukturflexibilität, die im Kontext von Geschäftsprozessen unterschieden werden (Bartmann et al. 2011, S. 2). Verhaltensflexibilität bezeichnet eine Veränderung des Ablaufs zur Gestaltungs- oder Laufzeit, während Strukturflexibilität für eine Veränderung der Aktivitäten oder deren struktureller Ablaufdetermination steht, z.B. in Ereignissen und Beziehungen.

**Hochflexible Geschäftsprozesse**

Hochflexible Geschäftsprozesse (hGP) (Sinz et al. 2011) definieren sich durch Ausprägungen der Dimensionen Planbarkeit, Zeitbezug (Planung und Ausführung) und Kontextsensitivität.

| **Dimension** | **Ausprägungen** | |
|---|---|---|
| **Planbarkeit** | vollständig | unvollständig |
| **Zeitbezug** | Planung und Ausführung seriell | Planung und Ausführung überlappend |
| **Kontextsensitivität** | ja | nein |

TABELLE 2.6: Merkmale hochflexibler Geschäftsprozesse

Die in Tabelle 2.6 hervorgehobenen Ausprägungen charakterisieren einen hochflexiblen Geschäftsprozess in folgenden Fällen (Bartmann et al. 2011, S. 2):

- Ein hGP ist gegeben, sofern unvollständige Planbarkeit und ein überlappender Zeitbezug zu Planung und Ausführung bestehen.
- Ein hGP ist zudem gegeben, sofern Kontextsensitivität vorliegt. Dies ist der Fall, wenn Struktur- und Verhaltensänderungen in Abhängigkeit von prozessexternen Einflussgrößen vorgenommen werden (Wagner und Ferstl 2011, S. 178).

Ein Beispiel ist die Einführung von erhöhten Compliance-Anforderungen im Supplier Relationship Management, die sich in zusätzlichen Überprüfungen laufender Zulieferprozesse äußert und Schemaänderungen nach sich zieht (vgl. Fallstudie in Kapitel 5.5). Ein hGP liegt in diesem Fall vor, da die Änderung der Planung des Prozesses zur Laufzeit stattfindet und der Prozess unvollständig geplant ist.

#### 2.3.4.3 Evolution von Geschäftsprozess-Schemata

In Zusammenhang mit der Veränderung von Geschäftsprozessen untersuchen Ansätze zur Schema-Evolution und zur Co-Evolution, wie die Planung eines Prozesses in Form eines Schemas verwaltet und auf neue Schema-Versionen überführt werden kann.

**Schema-Evolution**

Bei der Aktualisierung eines vorhandenen Schemas treten auf Instanz- und Schemaebenen zwei grundlegende Problemstellungen zutage, die mit den folgenden Flexibilitätskonzepten adressiert werden (Rinderle und Dadam 2003):

- Instanz-Flexibilität bezeichnet die Veränderung einzelner in Ausführung befindlicher Instanzen. Dazugehörige Änderungen einzelner Instanzen zur Laufzeit werden als Ad-hoc-Änderungen bezeichnet. Eine einzelne Instanz kann damit auf durch Ausnahmen hervorgerufene Flexibilitätsbedarfe reagieren, wobei nicht notwendigerweise eine Anpassung des Schemas des Prozess-Normalverhaltens erfolgen muss. Im Falle eines aktualisierten Schemas besteht hiermit zunächst die Möglichkeit, jede Prozess-Instanz eigenständig auf das aktualisierte Schema zu überführen.
- Schema-Flexibilität bezeichnet die Veränderung des Prozess-Schemas, die sich auf in Ausführung befindliche Instanzen auswirken kann. Schema-Evolution bezeichnet im engeren Sinne die Überführung des Schemas und der dazugehörigen Instanzen auf eine aktualisierte Version. Hierbei besteht das Problem, Instanzen möglichst automatisiert auf das aktualisierte Schema zu überführen, indem die Veränderungen gegenüber der vorherigen Version identifiziert, in Bezug auf den Ausführungszustand jeder Instanz lokalisiert und anhand von Änderungsoperationen durchgeführt werden.

Der Begriff bezieht sich je nach betrachteter Abstraktionsebene auf Geschäftsprozesse oder Workflows. Ein Beispiel ist die Implementierung von Schema- und Instanz-Flexibilität für hGP sowie daraus abgeleitete Workflow-Schemata für WfMS (Härer 2012).

Das Management von Modellen kann in Modell-Repositories oder auch in Verbindung mit Versionierungsansätzen erfolgen (Brambilla et al. 2017). Diese werden in Abschnitt 2.3.5 besprochen.

**Co-Evolution**

Wird die Problemstellung der Schema-Evolution auf eine Änderung mehrerer Schemata bezogen, in der Änderungen oder Aktualisierungen gleichermaßen abzubilden sind, liegt eine Co-Evolution der Schemata vor (Brambilla et al. 2017). Eine ursächliche Schema-Änderung muss dabei beispielsweise in Modellen unterschiedlicher Kooperationsteilnehmer entsprechend der jeweils verwendeten Modellierungsmethoden und Modellierungssprachen umgesetzt werden. Hierfür eigenen sich metamodellbasierte Ansätze, die anhand einer Metamodell-Abbildung die Syntax beider zu verändernder Schemata in Beziehung setzen. Die Veränderung der Modell-Elemente auf der Schemaebene kann dann unter Einsatz von Ansätzen der Schema-Evolution, wie etwa innerhalb des Beispiels zu AristaFlow durchgeführt werden.

#### 2.3.4.4 Konzepte der Service-Orientierung

Interaktionen zwischen Organisationseinheiten in verteilten betrieblichen Systemen, insbesondere in interorganisationalen Kooperationen (Fdhila et al. 2015), basieren auf definierten Nachrichten, die zur Laufzeit zwischen den teilnehmenden Organisationseinheiten ausgetauscht werden.

**Choreographie**

Dem serviceorientierten Paradigma folgend, handelt es sich bei dem Koordinationstyp der zur Interaktion notwendigen Nachrichtenaustausche um eine Choreographie (Peltz 2003), die eine nicht-hierarchische Koordination umsetzt (Ferstl und Sinz 2013, S. 67 f.). Eine Choreographie definiert Interaktionen anhand von Nachrichten und Nachrichtenaustauschmustern (Message Exchange Pattern) zwischen öffentlichen Service-Schnittstellen der Beteiligten. Eine Zuordnung (Mapping) verknüpft dort enthaltene Service-Methoden mit der Auslösung von Prozessen und Workflows.

**Orchestrierung**

Aus der Innensicht einzelner Teilnehmer besteht die Notwendigkeit, die intern auszulösenden Prozesse und Workflows zu koordinieren. In serviceorientierten Systemen bezeichnet der Koordinationstyp Orchestrierung die auf Teilnehmer bezogene interne Verknüpfung der Abläufe in Prozessen und Workflows, die Reihenfolgebeziehungen zwischen Aufgaben und Aktivitäten herstellen (Peltz 2003). Eine Orchestrierung bezieht sich im Falle von Kooperationen auf die Prozesse innerhalb eines teilnehmenden Unternehmens.

**Öffentliche Prozesse**

Ein Prozess eines Teilnehmers, dessen Aktivitäten und Abläufe als lokale Bestandteile einer globalen Choreographie von anderen Teilnehmern benötigt werden, wird als öffentlicher Prozess (Public Process) bezeichnet (Fdhila et al. 2015). Die in öffentlichen Prozessen definierten Aktivitäten und Abläufe besitzen eine öffentliche Sichtbarkeit und sind damit von anderen Teilnehmern einsehbar. Eine Kooperation legt die Systemabgrenzung der globalen Choreographie fest und impliziert kooperationsintern eine öffentliche Sichtbarkeit.

**Private Prozesse**

Ein privater Prozess (Private Process) eines Teilnehmers beschreibt die Implementierung eines öffentlichen Prozesses aus der internen Sicht des Teilnehmers. Der öffentliche Prozess kann als Sicht auf den privaten Prozess interpretiert werden (Fdhila et al. 2015). Aktivitäten und Abläufe des privaten Prozesses besitzen eine nichtöffentliche Sichtbarkeit. Im Falle von interorganisationalen Kooperationen können private Prozesse genau einer Organisation zugeordnet werden. Ein privater Prozess kann beispielsweise anhand einer hierarchischen Zerlegung eines öffentlichen Prozesses erstellt werden, sodass der öffentliche Prozess unter Hinzunahme der Zerlegungsprodukte einen privaten Prozess definiert (van der Aalst und Weske 2001). Werden öffentliche und private Prozesse unabhängig voneinander definiert, ist hingegen eine Zuordnung öffentlicher Aktivitäten zu privaten Aktivitäten erforderlich.

**Öffentliche und Private Workflows**

Wird zwischen Prozessen und Workflows anhand von mehreren Abstraktionsebenen unterschieden, werden auf Ebene der Workflows die Begriffe öffentlicher Workflow (Public Workflow) und privater Workflow (Private Workflow) herangezogen. Wird keine Unterscheidung vorgenommen, subsumieren die mit Geschäftsprozessen verbundenen Begriffe die hinsichtlich der Abstraktion untergeordneten Workflow-Begriffe.

**Öffentliche und Private Prozesse in Kooperationen**

Im Falle von vertikalen Kooperationen definieren öffentliche Prozesse Schnittstellen zwischen Produktionsstufen, während horizontale Kooperationen ein Offenlegen gleichartiger operativer oder strategischer Prozesse erfordern, die für ein Zusammenwirken innerhalb einer Produktionsstufe benötigt werden. In lateralen Kooperationen legen öffentliche Prozesse die Abstimmung des gemeinsamen Leistungsangebots unterschiedlicher Branchen fest.

#### 2.3.4.5 Merkmale von kooperativen Geschäftsprozessen in Netzwerken

Zur Bestimmung und Abgrenzung von Geschäftsprozessen im Kontext der Netzwerkorganisation ergeben sich aus den besprochenen Grundlagenthemen die nachfolgenden Merkmale einer Arbeitsdefinition. Ein kooperativen Geschäftsprozess der Netzwerkorganisation beschrieben durch:

- **M1**: Zusammenwirken von Teilnehmern zur gemeinsamen Leistungserstellung,
- **M2**: Abgrenzung der öffentlichen und privaten Ablauforganisation der Teilnehmer,
- **M3**: auf interorganisationalen Netzwerken beruhende Koordination,
- **M4**: Autonomie und Selbstorganisation der Teilnehmer,
- **M5**: Co-Evolution vollständig oder unvollständig strukturierter Prozessaktivitäten,
- **M6**: Zeitbezug zu Gestaltungszeit und Laufzeit.

M1 geht aus dem Merkmal der Leistungserstellung von Geschäftsprozessen und dem Kooperationsbegriff unmittelbar hervor (Abschnitt 2.3.1.2). M2 fordert eine Unterscheidbarkeit der Beteiligten hinsichtlich des Ablaufs der Prozessaktivitäten in öffentlichen und privaten Prozessen (Abschnitt 2.3.3). M3 beschreibt die netzwerkbasierte Koordination von Beteiligten zur Leistungserstellung über Unternehmensgrenzen hinaus (Abschnitt 2.3.2). M4 nimmt auf die aus der Selbstorganisation erwachsende Flexibilität Bezug (Abschnitt 2.3.4.2). M5 unterstellt die Notwendigkeit der gemeinsamen Fortentwicklung von Prozessen (Abschnitt 2.3.4.3). M6 geht auf die mögliche Parallelität der Planung und Durchführung von Aufgaben in Netzwerken ein und bezieht sich dabei auf das Merkmal der zeitlichen Überlappung in hGP (Abschnitt 2.3.4.2). Ein hGP liegt demnach vor, wenn neben diesem stets erfüllten Merkmal zudem unvollständige Planung gegeben ist. Dies ist der Fall, wenn Prozessaktivitäten unvollständig strukturiert sind (M5).

### 2.3.5 Methoden zur Verteilung der Modellierung von Geschäftsprozessen

#### 2.3.5.1 Einordnung

Die Erstellung von Modellen durch das Zusammenwirken beteiligter Modellierer kann auf Methoden der kooperativen Modellierung zurückgreifen. Diese umfassen Verfahren, die eine Verteilung und einen Abgleich von Modellen unter Beteiligung mehrerer Akteure ermöglichen. Im Folgenden werden zunächst Verfahren des Modell-Managements betrachtet, die anschließend entsprechend des im vorherigen Abschnitt eingeführten Kriteriums der asynchronen Kooperation diskutiert werden.

#### 2.3.5.2 Modell-Management

Zur Identifikation geeigneter Verfahren wird auf Methoden des Modell-Managements zurückgegriffen. Dieses betrifft die Verwaltung erstellter Modellartefakte einzelner oder mehrerer Modellierer. Hierzu gehören u.a. Methoden für den Austausch, die persistente Speicherung in Modell-Repositories sowie den Vergleich und die Versionierung von Modellen (Brambilla et al. 2017). Die genannten Verfahren werden als Grundlage zur kooperativen Modellierung diskutiert.

**Modellaustausch**

Ein Austausch von Modellen (Model Interchange) verlangt eine über Modellierungswerkzeuge hinweg konsistente Implementierung der Syntax und Semantik von Standards und Austauschformaten (Kurz 2016). Austauschformate für BPMN, UML und MOF werden durch die OMG anhand von XMI-basierte Dateiformaten definiert, deren Ziel die Interoperabilität verschiedener Modellierungswerkzeuge und Workflow-Engines ist (OMG 2015). Eine Übersicht zur Unterstützung der genannten Formate mit einer Untersuchung der Kompatibilität verschiedener Workflow-Engines liegt hierzu vor (OMG 2018).

**Vergleich von Modellen**

Der Vergleich von Modellen ermitteltet Unterschiede zwischen Elementen der Ausprägungsebene mehrerer Modelle. Eine mögliche Strategie ist die Durchführung eines Matchings identischer Modellelemente, um anschließend anhand eines Differencings Unterschiede in Form von hinzugekommenen, fehlenden und abgeänderten Elementen zu detektieren (Stephan und Cordy 2013). Unterschiede werden anhand von Modellen, Graphen, Datenbanken sowie text- und operations-basierten

Repräsentationen festgehalten (Kuryazov et al. 2018). Für Modellierungssprachen bestehen Metamodell-unabhängige Verfahren (Cicchetti et al. 2007) sowie auf Modellierungssprachen spezialisierte Verfahren, z.B. für UML (Xing und Stroulia 2005), BPMN (Gerth et al. 2010) und SOM (Wolf 2015).

#### 2.3.5.3 Modell-Repositories

Die persistente Speicherung von Modellen erfolgt typischerweise in Form von Dateien oder in Modell-Repositories. Die Ablage von Dateien greift auf lokale oder verteilte Dateisysteme zurück. Auf einer höheren Abstraktionsebene erlauben Modell-Repositories die Ablage von Modellen unter Nutzung von Dateien oder Datenbanksystemen, deren Einsatz für den Nutzer transparent ist. Relationale Datenbanksysteme werden von Modellierungswerkzeugen wie ADOxx zur Speicherung und Verteilung herangezogen (Fill, Eberhart et al. 2011). NoSQL-Datenbanksysteme für Modelle sind für EMF verfügbar, z.B. NeoEMF (Daniel et al. 2017). Connected Data Objects (CDO) ist ein Repository, das unterschiedliche relationale und nicht-relationale Datenbanksysteme unterstützt (Brambilla et al. 2017, S. 160).

**Versionierung**

Verfahren zur Versionierung von Modellen basieren auf Versionskontrollsystemen, die einzelne, durch Versionskennungen identifizierte Artefakte in einem zentralisierten Repository oder in mehreren verteilten Repositories hinterlegen und bereitstellen. Subversion (Collins-Sussman et al. 2004) und Git (Chacon und Straub 2014) sind Beispiele für Systeme, deren Repositories zentralisiert bzw. verteilt sind. Im Kontext von Modellen stellt ein Versionskontrollsystem ein spezielles Modell-Repository dar. Die Systeme sind dabei in erster Linie auf die in Kooperation stattfindende Entwicklung von Software-Systemen ausgerichtet und operieren auf der Ebene einzelner Dateien. Eine Versionierung auf dieser Basis betrifft somit typischerweise die Ebene der Modelle, nicht aber einzelne Modell-Elemente der Ausprägungsebene.

#### 2.3.5.4 Asynchrone Versionierung von Modellen

Spezialisierte Modellversionierungsansätze behandeln die Bildung von Versionen auf dieser Ebene. Verfahren von Kaufmann et al. (2010) und Taentzer et al. (2014) sind auf Ebene der konkreten und abstrakten Syntax allgemein definiert und für EMF implementiert. Weitere produktspezifische Verfahren existieren, z.B. für die Metamodellierungsumgebung MetaEdit+ (Kelly und Tolvanen 2018). Verfahren zur Einbeziehung einer formalisierten Semantik werden diskutiert (Brosch, Egly et al.

2012). Sofern keine formale Beschreibung der Semantik vorliegt, ist die Anwendung potenziell beliebiger Operatoren in Abhängigkeit der Fachlichkeit der Modellierung durch einen Domänenexperten vorzunehmen.

Modellversionierungsverfahren umfassen die Erkennung und Repräsentation von Änderungen in Modellen, die Behandlung von Konflikten und Verfahren der Zusammenführung von Modellen (Altmanninger et al. 2009; Brosch, Kappel et al. 2012).

**Erkennung und Repräsentation von Änderungen**

Die Erkennung von Änderungen nutzt die Methoden des Modellvergleichs, um durch Matching und Differencing syntaktische und semantische Änderungen der Schema- oder Instanz-Ebene zu erkennen und zu repräsentieren. Eine Version definiert sich durch die per Differencing ermittelten Änderungen oder alternativ, im Falle von operations-basierten Verfahren, anhand der zur Herbeiführung der Änderungen durchgeführten Modell-Operationen (Herrmannsdoerfer und Koegel 2010).

**Behandlung von Konflikten**

Optimistische Versionierungsverfahren erlauben die parallele Bearbeitung von Modellen, während pessimistische Verfahren die Sperrung von in Bearbeitung stehenden Artefakten erfordern. Eine Konsequenz optimistischer Verfahren sind Konflikte, die durch die parallele Anwendung nicht-austauschbarer und damit konfliktärer Operationen auf ein Artefakt entstehen. Mehrere in Konflikt stehende Artefakte werden infolge eines Differencings erkannt und erfordern eine Auflösung (Conflict Resolution) durch Zusammenführung (Merge) der Artefakte.

**Verfahren der Zusammenführung**

Drei Verfahren zur paarweisen Zusammenführung (Merge) von Artefakten sind prinzipiell unterscheidbar (Brambilla et al. 2017).

- **Raw Merge**: Ein Raw Merge wendet eine Reihe von Operationen sequenziell auf eines der zusammenzuführenden Artefakte an, um die gegenüber dem anderen Artefakt bestehenden Änderungen in einer neuen Version zu integrieren.
- **Two-way Merge**: Ein Two-way Merge ermittelt basierend auf einem Differencing Unterschiede zwischen den zusammenzuführenden Artefakten und vereinigt diese. Dabei werden nur Unterschiede zwischen den zusammenzuführenden Artefakten betrachtet. Hierdurch sind unterschiedliche Änderungen

nicht erkennbar, die sich auf eine gemeinsame Vorgängerversion beziehen.

- **Three-way Merge**: Ein Three-way Merge ermittelt für beide zusammenzuführenden Artefakte Unterschiede gegenüber der letzten gemeinsamen Vorgängerversion. Die Änderungen der beiden Artefakte gegenüber ihrer Vorgängerversion werden vereinigt, sodass insgesamt Unterschiede zwischen drei Artefakten zusammengeführt werden. Hierdurch sind etwa unterschiedliche Änderungsoperationen gegenüber der Vorgängerversion erkennbar.

Aktuelle Implementierungen von Versionskontrollsystemen wie Git unterstützen die erläuterten Formen und wenden im Normalfall bei Ausführung einer Merge-Operation einen Three-way Merge an (Chacon und Straub 2014).

## 2.4 Verwandte Ansätze und Methoden

### 2.4.1 Einordnung

Die in diesem Abschnitt diskutierten Ansätze sind verwandte Arbeiten, die sich innerhalb des übergeordneten Themas der Prozesse in dezentral organisierten betrieblichen Systemen auf Abbildungen der zu erstellen Prozesse anhand von Modellen beziehen. Im Weiteren sind verwandte Ansätze nach den Themengebieten Adaptive Case Management, Workflow-Modellierung und Geschäftsprozessmodellierung gegliedert.

### 2.4.2 Ansätze des Adaptive Case Management

Adaptive-Case-Management- oder Case-Handling-Ansätze betrachten Daten eines Geschäftsvorfalls, der von Aufgabenträgern stets holistisch eingesehen und bearbeitet wird. Das Paradigma des Case Handling kann als daten- und prozessorientiert charakterisiert werden (Weske 2012, S. 361). Das Case Handling nutzt nicht notwendigerweise grafische Modelle. Ein holistisch betrachteter Fall (Case) ist mit Daten verknüpft, deren Attribute während der Bearbeitung verändert werden. Die Objektart zugehöriger Aufgabenobjekte ist Information, wodurch hohe fachliche Anforderungen an Aufgabenträger gestellt werden. Abläufe, z.B. nacheinander zu bearbeitende Attribute, werden jeweils in einzelnen Prozessfragmenten hinterlegt. Durch die Auswahl geeigneter Fragmente besteht Verhaltensflexibilität auf Instanzebene.

**Erweiterungen und verwandte Ansätze**

Erweiterungen und verwandte Ansätze des Adaptive Case Management erfassen spezifische Domänen, z.B. Production Case Handling (PCM), oder verknüpfen Case Handling mit anderen Paradigmen, wie dem Geschäftsprozessmanagement (Motahari-Nezhad und Swenson 2013).

Arbeiten von Kurz und Fleischmann (2011) und Huber et al. (2013) schlagen Ansätze für das Adaptive Case Management vor, die in erster Linie die Implementierung von Instanzflexibilität und die Kooperation bei der Erstellung von Cases behandeln. Zum Thema der Kooperation schlagen Huber et al. eine Erfassung von Cases anhand von Templates auf Schemaebene vor, die als Teil von Ad-Hoc- und Team-Workflows abgelegt werden. Dabei wird zwischen Instanz- und Schemaebene unterschieden, indem Templates zur Laufzeit instanziiert werden.

Hewelt und Weske (2016) beschreiben einen hybriden Ansatz unter Nutzung von BPMN-basierten Prozessfragmenten und Cases, die gemeinsam bearbeitet werden können. Dabei werden die Planung und die Ausführungssemantik erfasst. Im Unterschied zu dem in dieser Arbeit vorgeschlagenen Ansatz wird aufgrund des Paradigmas hier kein Gesamtprozess unterstellt.

**Abgrenzung**

Hinsichtlich der Prozessorientierung des im Rahmen dieser Arbeit entwickelten Ansatzes bestehen zwei grundlegende Unterschiede.

1. Die auszuführenden Schritte beziehen sich auf einzelne Aktionen, etwa Formulare oder Attributwerte, anhand derer ein Lösungsverfahren einer Aufgabe definiert wird; somit lässt sich die Ablaufbeschreibung als Workflow-Fragment einordnen.
2. Weiterhin ergibt sich aufgrund des zugrunde liegenden Paradigmas keine Planung und Steuerung des Gesamtablaufs eines übergeordneten Prozesses. Eine Modellierung des Case Handling wird anhand der Case Management and Notation (CMMN) von der OMG standardisiert (OMG 2016a).

### 2.4.3 Ansätze zur Workflow-Modellierung

Verwandte Ansätze aus dem Bereich des Workflow-Managements behandeln die Modellierung sowie die Ausführung, im Kontext von WfMS, über Unternehmensgrenzen hinweg.

Hierzu gehören Ansätze aus den Bereichen Workflow Evolution (Casati und Discenza 2000) und Cross-Organisational Workflows (Ludwig und Whittingham 1999). Die Ansätze gehen von einer Client-Server-Architektur aus. Die Autoren thematisieren bereits die Übereinkunft bei der Erstellung von Modellen anhand von Schnittstellen zur Annahme oder Ablehnung von Workflow-Modellen.

Workflow-Modelle im Kontext der virtuellen Unternehmung untersuchen beispielsweise D.-R. Liu und Shen (2003). Sie schlagen zudem die Erstellung von Prozess-Sichten vor. Diese werden für Teilnehmer unterschiedlicher Unternehmen unter Erhaltung von Reihenfolgebeziehungen entwickelt.

**Interorganizational Workflows**

Interorganizational Workflows (van der Aalst 2000; van der Aalst und Weske 2001) ist ein Ansatz für die Modellierung zwischenbetrieblicher Workflows, der das zwischen Unternehmen entstehende Netzwerk hinsichtlich der dort realisierbaren Abläufe zum Austausch von Informationen beschreibt. Die Darstellung in Interorganizational Workflow Nets (IOWF-Nets) nutzt eine Petri-Netz-Syntax zur Erstellung ablauforientierter Modelle, die einen potenziellen Nachrichtenaustausch abbilden. Damit wird eine Choreographie modelliert, die in öffentliche und private Prozesse einer Orchestrierung überführbar ist. Public-To-Private (P2P) bezeichnet den Ansatz zur Überführung basierend auf Interorganizational Workflow Nets (IOWF-Nets). Der Ansatz sieht ein in drei Schritten angegebenes Vorgehen vor, nach dem

1. unter Maßgabe eines gemeinsamen Verständnisses ein gemeinsam zu nutzender, öffentlicher Workflow erstellt,
2. nach den Grenzen der beteiligten Organisationen partitioniert und
3. je Organisation in private Workflows überführt wird, die eine Subklasse des öffentlichen Workflows bilden.

Die formalisierte Abbildung in Petri-Netzen erlaubt die Durchführung von Simulationen und Verifikationen (Fernández Venero und Corrêa Da Silva 2017).

**Workflow-Management-Systeme am Beispiel von AristaFlow**

Zur Workflow-Automatisierung in WfMS lieft das ADEPT-Projekt (Dadam und Reichert 1998) und die daraus entstandene Software AristaFlow (Dadam, Reichert, Rinderle-Ma et al. 2009) einen grundlegenden Beitrag.

Das WfMS enthält Komponenten für die Modellierung von Workflows und Daten, das Management von Schemata in Repositories, die Vergabe von Rollen und die Ausführung von Schemata. Daten werden ohne ein zugrunde liegendes Schema anhand von Attributen mit Syntax-Elementen verknüpft.

Die Modellierungssprache von AristaFlow nutzt verschachtelte Blöcke zur Spezifikation des Kontrollflusses und sieht hierfür grundlegende Aktivitätselemente, Verzweigungen, Zusammenführungen sowie Synchronisations- und Entscheidungselemente vor. Das Konzept Correctness-by-Construction stellt zur Gestaltungszeit die Formulierung syntaktisch korrekter Schemata sicher, indem die Einfügeoperationen auf Elemente beschränkt werden, die kompatibel zu vorhandenen Elementen sind.

AristaFlow unterstützt Konzepte zur Erhöhung der Flexibilität (Dadam, Reichert und Rinderle-Ma 2011), die Instanz-Flexibilität während der Ausführung anhand von Ad-hoc-Änderungen des Schemas erlauben. Hinsichtlich der Schema-Evolution unter Einbeziehung aller Instanzen bestehen Einschränkungen; so müssen je Instanz alle in Ausführung befindlichen Aktivitäten in Relation zu allen veränderten Aktivitäten hinsichtlich der durchgeführten Änderungsoperationen analysiert werden.

Auch andere WfMS wie Tibco iProcess oder Camunda (Camunda 2018; Tibco 2017) lassen Ad-hoc-Änderungen zu und beschränken sich hinsichtlich der Schema-Evolution auf Batch-Operationen, die eine Anwendung von Änderungsoperationen auf mehrere manuell zu bestimmende Instanzen ermöglicht.

**Abgrenzung**

Zu dem in dieser Arbeit beschriebenen Ansatz bestehen hinsichtlich der zugrunde liegenden Paradigmen und der Positionierung der Ansätze wesentliche Unterschiede.

1. Workflow-Netze enthalten ablauforientierte Darstellungen, während der Ansatz dieser Arbeit eine ablauforientierte Verhaltenssicht neben einer Strukturansicht vorsieht.
2. Ausführbare Ansätze auf der Basis von Petri-Netzen können hinsichtlich ihrer Ausführungssemantik vollständig formalisiert werden. Die Ausführungssemantik wird in dieser Arbeit in semi-formalen Modellen untersucht, die teilweise in Petri-Netze überführbar sind (siehe Kapitel 4.2).
3. Die besprochenen Ansätze treffen keine Unterscheidung zwischen Modellebenen und differenzieren nicht zwischen Geschäftsprozessen und Workflows. Beispielsweise betrifft die Subklassenbildung in IOWF eine Modellebene, während diese Arbeit mehrere Modellebenen und Transformationen vorschlägt.

Die Arbeit übernimmt das Grundkonzept interorganisationaler Abläufe und das grundlegende Public-To-Private-Konzept zur gemeinsamen Erstellung privater Prozesse auf der Basis von öffentlichen Prozessen.

### 2.4.4 Ansätze zur Geschäftsprozessmodellierung

Ansätze zur Modellierung kooperativer Geschäftsprozesse greifen entweder auf bestehende Modellierungssprachen zurück, erweitern bestehende Sprachen oder entwerfen spezialisierte Sprachen.

**Modellierung unter Nutzung bestehender Modellierungssprachen**

Die folgenden Ansätze der Geschäftsprozessmodellierung werden als verwandt klassifiziert, da sie (1.) die Abbildung interorganisationaler Prozesse durch eine Abgrenzung beteiligter Organisationen und (2.) das Herstellen von Beziehungen zwischen diesen erlauben:

- SOM (siehe Abschnitt 2.2.4) gestattet eine hierarchische Strukturierung anhand von betrieblichen Objekten der Diskurswelt und Umwelt (1.), zwischen denen eine nachrichtenbasierte Kommunikation über Transaktion stattfindet (2.).
- BPMN (siehe Abschnitt 2.2.5) erlaubt eine flache Strukturierung anhand von Pools (1.), die Nachrichten durch Message Flows austauschen (2.).

Die Sprachen EPK, eEPK und Aufgabenkettendiagramm (siehe Abschnitt 2.2) besitzen eine Syntax zur Gliederung von Organisationseinheiten, die keine explizite Abgrenzung von Organisationen ermöglichen. Beziehungen sind anhand von Kontrollflüssen definiert, die eine Steuerung des Ablaufs von Geschäftsprozessaktivitäten abbilden. Kommunikationsbeziehungen zwischen separat gesteuerten Abläufen mehrerer Organisationen werden davon nicht erfasst.

Eine weitergehende Untersuchung zur Nutzung bestehender Modelle und Syntax-Elemente für die Modellierung von kooperativen Geschäftsprozessen anhand der Sprachen von BPMN und SOM ist Gegenstand des Kapitels 2.3.3.

**Modellierungssprachen zur kooperativen Modellierung**

Erweiterungen lassen sich für die Modellierungssprachen EPK, Petri-Netz und UML identifizieren, die aus Literaturrecherchen sowie aus Arbeiten von Aleem et al. (2012), Hermann et al. (2017), Niehaves und Plattfaut (2011) und Oppl (2017) hervorgehen. Weiterhin können spezialisierte Modellierungssprachen unterschieden werden, von denen nur sehr wenige bekannt sind. Grundlage der nachfolgenden Klassifikation ist die aus Sicht des Modellierers verwendete Sprache, in der kooperative Prozesse fachlich erfasst werden. Dies ist relevant, sofern ein Ansatz mehrere bestehende Sprachen erweitert.

Neben den im vorherigen Abschnitt angeführten Kriterien zur Kooperation werden die in Tabelle 2.7 gegebenen Kriterien zu System und Zeitbezug herangezogen. Hinsichtlich der Kooperation wird die Abbildbarkeit abgegrenzter Organisationen mit ihren Kommunikationsbeziehungen untersucht. Das Kriterium Organisationen-Abgrenzung ist gegeben, sofern Organisationen und Organisationseinheiten mehrstufig voneinander abgrenzbar erfasst werden. Eine Kommunikationsbeziehung ist gegeben, wenn über die Modellierung von sequenziellen Ablaufbeziehungen hinaus fachliche Austausche darstellbar sind, die keine zentrale Steuerung erfordern, z.B. in Form von Nachrichten. Die Abbildbarkeit des Verhaltens ist im Falle von Ablaufbeziehungen gegeben, sofern diese in Form von Relationen zwischen Objekten oder anderen strukturellen Elementen bestehen. Die Gestaltungszeit ist abbildbar, wenn die Modelle eine Planung der Ausführung von Prozessen erlauben. Die Laufzeit ist abbildbar, sofern der Zustand einzelner Instanzen während der Ausführung abbildbar ist. Für alle Kriterien wird im Falle der Ausprägung *teilweise zutreffend* (siehe Legende) ein Stichwort als Verweis auf den Erläuterungstext gegeben.

| | **Kooperation** | | **Systemmerkmal Prozess** | | **Zeitbezug** | |
|---|---|---|---|---|---|---|
| | **Organisationen-Abgrenzung** | **Kommunikations-beziehungen** | **Verhalten** | **Zustand** | **Gestaltungszeit (Schema)** | **Laufzeit (Instanz)** |
| **Barjis 2009** | ◐ (Stufen: 1) | ◐ (Transition) | ● | ○ | ● | ◐ (Zustand) |
| **Bauer et al. 2005** | ○ (Modelle) | ○ (Sequenz) | ● | ○ | ● | ○ |
| **Lee et al. 2010** | ◐ (Stufen: 2) | ◐ (Konnektor) | ● | ○ | ● | ● |
| **Mevius und Oberweis 2005** | ◐ (Stufen: 1) | ◐ (Transition) | ● | ○ | ● | ◐ (Zustand) |
| **Ryu und Yücesan 2007** | ◐ (Stufen: 2) | ◐ (Konnektor) | ● | ○ | ● | ○ |
| **Villarreal et al. 2010** | ◐ (Stufen: 2) | ○ (Sequenz) | ● | ◐ (Rollen) | ● | ○ |

● zutreffend ◐ teilweise zutreffend ○ nicht zutreffend

TABELLE 2.7: Verwandte Ansätze bestehender Modellierungssprachen

*EPK-basierte Modellierung*

Bauer et al. (2005) beschreiben einen MDA-basierten Ansatz, der CIM-Modelle der EPK in BPDM-basierte PIM-Modelle überführt. Zur Darstellung der Abgrenzung von Organisationen werden unterschiedliche EPK-Modelle herangezogen, die anhand von UML-Sequenzdiagrammen in Beziehung stehen. Eine ein- oder mehrstufige Abgrenzung innerhalb eines Modells ist nicht vorgesehen. Beziehungen werden in Form von Aufrufen ohne fachliche Kommunikationsbeziehungen dargestellt.

*UML-basierte Modellierung*

UML-basierte Ansätze erweitern die in UML-Diagrammen vorhandene Syntax typischerweise unter Nutzung von UML-Profilen. Villarreal et al. (2010) schlagen einen Ansatz vor, der das „UML Profile for Collaborative Business Processes based on Interaction Protocols“ (UP-ColBPIP) als Ausgangspunkt für MDA-basierte Modelltransformationen zu gefärbten Petri-Netzen vorschlägt. Das Profil basiert auf Sequenz-Diagrammen der UML 2 und sieht eine Reihe von Sichten für Rollen in Beziehung stehender Teilnehmer (1.), Geschäftsprozesse (2.), Interaktionsprotokolle (3.), Geschäftsdokumente (4.) und Schnittstellen (5.) vor. Strukturorientiert ist nur die erste Sicht (1.), die Teilnehmer anhand von Rollen identifiziert. Die Sichten zur Abbildung von Geschäftsprozessen und Interaktionsprotokollen verwenden die Notation von Sequenz-Diagrammen und sind verhaltensorientiert. Der Ansatz unterscheidet Organisationen und Rollen in zwei Stufen. Aus Sicht der Geschäftsprozessmodellierung bildet der Ansatz damit das Systemverhalten während der Gestaltungszeit ab.

*Petri-Netz-basierte Modellierung*

Mevius und Oberweis (2005) nutzen Petri-Netze, die Abläufe im Sinne von Interorganizational Workflows zwischen den Grenzen beteiligter Organisationen sowie in ihrer Innensicht auf einer Stufe darstellen. Weitere Stufen zur Abgrenzung von Organisationseinheiten werden nicht herangezogen. Eine Ausführbarkeit ist durch die zugrunde liegende Petri-Netz-Semantik grundsätzlich gegeben, allerdings erfasst das Modell keine unterscheidbaren Instanzen. Die Modellierung ist damit auf die Abbildung eines Zustands beschränkt.

Barjis (2009) verwendet modifizierte Petri-Netze, die um abgrenzbare Organisationen innerhalb einer Stufe erweitert sind. Beziehungen bestehen in Form von Transitionen und Kanten. Transitionen und Kanten implementieren Geschäftstransaktionen, die das Muster Order-Execution-Result zur Vereinbarung, Durchführung und Bestätigung von Transaktionen umsetzen. Damit wird das Systemverhalten auf

Schemaebene abgebildet. Die Ausführbarkeit ist auch hier ohne eine Möglichkeit der Unterscheidung verschiedener Instanzen gegeben.

*Spezialisierte Modellierungssprachen*

Ryu und Yücesan (2007) entwerfen die Modellierungssprache Collaborative Process Modeling for Cooperative Manufacturers (CPM) für die Abbildung von intra- und interorganisationalen Prozessen in produzierenden Unternehmen. CPM unterscheidet Organisationen und Organisationseinheiten in zwei Stufen. Die Syntax und Notation von CPM sieht Elemente für normale, intra- und interorganisationale Prozesse, Entscheidungen sowie Beziehungen vor. Entscheidungen und Konnektor-Beziehungen sind an die Notation von UML-Aktivitätsdiagrammen angelehnt, basieren semantisch allerdings auf abweichenden Konzepten. Beziehungen repräsentieren Prozess-Übergänge oder Synchronisationen von Prozessen. Konzepte zur Kommunikation, etwa durch Nachrichten, sind nicht vorhanden. Ferner werden Ressourcen und Referenz-Annotationen dargestellt. Zur Definition der Ausführungssemantik wird eine Transformation in Petri-Netze festgelegt. Der Ansatz modelliert das Systemverhalten zur Gestaltungszeit. Eine Erweiterung (Lee et al. 2010), exCPM (Extended CPM), schlägt Color Tokens und State Tokens vor, um den Zustand während der Ausführungszeit zu erfassen. Eine Abgrenzung von Organisationen erfolgt nicht explizit, stattdessen wird eine Zuordnung von Unternehmen und Organisationseinheiten zu Prozess-Elementen vorgenommen.

**Methoden zur kooperativen Modellierung**

Bislang nicht adressiert wird die Verteilung von Entwicklung und Ausführung, deren Betrachtung über das im vorherigen Abschnitt betrachtete Untersuchungsobjekt *Beschreibungsmittel* hinausgeht. Einige wenige Ansätze verwandter Arbeiten diskutieren hierfür Methoden, die geeignet sind, Modelle kooperativ zu erstellen. Zur Erstellung der Modellabbildung wirken in diesem Fall mehrere Akteure zusammen. Tabelle 2.8 zeigt die hier evaluierten Ansätze.

Die nachfolgend betrachteten Kriterien betreffen die Verteilung bezüglich der Entwicklung und Ausführung, sowie die Kooperation. Hinsichtlich der Entwicklung wird die Abbildung globaler Modelle und lokaler Modelle unterschieden. Globale Modelle bilden Geschäftsprozesse ab. Lokale Modelle oder lokale Sichten detaillieren Ausschnitte der globalen Modellierung. Hinsichtlich der Ausführung wird untersucht, ob eine Ausführungsüberwachung (Monitoring) anhand des Ansatzes durch die beteiligten Akteure möglich ist.

| | Verteilung | | | Kooperation | |
|---|---|---|---|---|---|
| | Entwicklung | | Ausführung | Operationenausführung | |
| | Globale Abbildung | Lokale Abbildung | Monitoring | Synchron | Asynchron |
| **Boaro et al. 2011** | ● (Prozess) | ○ | ◐ (BPEL) | ○ | ● |
| **Dollmann et al. 2011** | ● (Prozess) | ○ | ○ | ● | ● |
| **Forster und Pinggera 2012** | ● (Prozess) | ○ | ○ | ● | ○ |
| **Huber 2014** | ● (Fragmente) | ○ | ○ | ○ | ● |
| **Rittgen 2009** | ● (Prozess) | ○ | ○ | ○ | ● |
| **Kurz 2010** | ● (Prozess) | ○ | ○ | ○ | ● |

● zutreffend ◐ teilweise zutreffend ○ nicht zutreffend

TABELLE 2.8: Verwandte Ansätze zur kooperativen Modellierung

Die Methoden zur Kooperation werden bezüglich der Operatorenausführung betrachtet, die unter synchroner Kommunikation der Beteiligten, oder unter asynchroner Kommunikation erfolgen kann. Für alle Kriterien wird im Falle der Ausprägung *teilweise zutreffend* (siehe Legende) ein Stichwort als Verweis auf den Erläuterungstext gegeben.

Boaro et al. (2011) entwerfen ein gemeinsam nutzbares Modellierungswerkzeug, mit dem Geschäftsprozesse gemeinsam modelliert und für eine technische Umsetzung in BPEL vorbereitet werden. Somit führt der Ansatz in Richtung einer ausführbaren Geschäftsprozessdefinition, deren Instanziierung nicht Teil des Ansatzes ist. Die Ausführungsüberwachung (Monitoring) ist kein Bestandteil der Plattform, jedoch erlaubt die Ausführung der BPEL-Prozessdefinition ein Monitoring durch eine Workflow-Engine.

Dollmann et al. (2011) stellen ein Tool zur gemeinsamen Erstellung von EPK-Modellen und Petri-Netzen vor. Die genannten Modelle werden getrennt voneinander erstellt und eigenen sich damit nicht für die Erstellung lokaler Abbildungen in Form von Modellen oder Sichten. Das Werkzeug erlaubt ein synchrones Zusammenwirken während der Modellerstellung oder eine asynchrone Modellierung unter Nutzung von Modell-Repositories.

Forster et al. (2012) tragen einen explorativen Forschungsansatz bei, der eine Plattform mit Basiselementen für die Modellierung anhand einer Block-basierten Prozessmodellierungssprache enthält. Die Plattform unterstützt ein Replay-Konzept, das auf Basis einer Verfolgung der Modelloperationen im Zeitverlauf eine Rücksetzung des gemeinsamen Modells erlaubt. Das Zusammenwirken während der Modellierung erfolgt synchron.

Huber (2014) konzipiert anhand des CoCaMa-Ansatzes für ein Collaborative Case Management eine Methode sowie ein dazugehöriges Tool. Teil des Werkzeugs ist die Planung von Prozess-Fragmenten, die in BPMN modelliert für einzelne Akteure zugreifbar sind. Aufgrund des Case-Management-Paradigmas werden keine vollumfänglichen und potenziell interorganisationalen Prozesse betrachtet. Die Ausführung und Überwachung von Prozessen ist daher nicht Teil des Ansatzes.

Rittgen (2009) betrachten anhand einer Collaborative Modeling Architecture (COMA) die Erstellung von Modellen sowie den Prozess während der Erstellung. Teil des Ansatzes ist ein Prozess, der die Akzeptanz erstellter Modelle durch eine Reihe von Verhandlungsmustern sicherstellt. Ein Modell kann zunächst vorgeschlagen und anschließend in mehreren Schritten angenommen oder abgelehnt werden. Eine Konsequenz dieses Vorgehens ist eine asynchrone Modellbildung.

Kurz (2010) beschreibt einen auf SharePoint basierten Modelleditor, der asynchron von einzelnen Mitarbeitern zugegriffen wird. Das Werkzeug sieht die Planung von Prozessen vor. Weitere Arbeiten im Kontext von ACM (Kurz und Fleischmann 2011) nehmen auf Task-Listen Bezug, ohne die Prozessmodellierung im engeren Sinne zu betreffen. Laufzeitaspekte sind daher nicht Teil des Ansatzes.

**Business Process Management**

Im weiteren Sinne verwandt ist das Management von kooperativen Geschäftsprozessen. Collaborative Business Process Management ist ein Oberbegriff, der Ansätze und Methoden interorganisationaler Geschäftsprozesse umfasst und als Teilbereich des Business Process Management (BPM) eingeordnet werden kann (Hermann et al.

2017; Chengfei Liu et al. 2009). Hierzu gehören Ansätze mit Bezug zum BPM Lifecycle (Dumas et al. 2018, S. 16). Der Großteil existierender Ansätze behandelt Kooperationen vor dem Hintergrund transaktionaler und vertraglicher Beziehungen zwischen den Teilnehmern; vertrauensbasierte Ansätze spielen dabei keine Rolle (Niehaves und Plattfaut 2011).

**Abgrenzung**

Die Ansätze zur Modellierung kooperativer Geschäftsprozesse unterscheiden sich in ihrer Ausrichtung von dem in dieser Arbeit vorgeschlagenen Ansatz:

1. Die Modellierung von Organisationen und ihren Kommunikationsbeziehungen wird nicht auf einer fachlichen Ebene betrachtet.
   (a) Eine Abgrenzung von Organisationen ist möglich, allerdings nicht im Sinne einer mehrstufigen Abbildung, die nicht-hierarchisch oder hierarchisch strukturierte Abstraktionsebenen umfasst. Eine mehrstufige Abbildung erlaubt in erster Linie die Darstellung lokaler Sichten für einzelne Organisationen, die damit in ihrer Innensicht abbildbar werden.
   (b) Beziehungen zur Kommunikation verwenden Sequenzen zur Darstellung von Aufrufbeziehungen oder Kontrollflussbeziehungen in Form von Transitionen oder Konnektoren.
2. Die Modellierung der Systemstruktur wird nicht erfasst. Ein Ansatz (Villarreal et al. 2010) greift auf eine Darstellung der Struktur zur Identifikation der Rollen von Teilnehmern zurück, ohne auf den Geschäftsprozess bezogene Beziehungen in ihrer Struktur zu erfassen.
3. Die Modellierung betrachtet nur die Gestaltungszeit. Während die Ausführungssemantik häufig durch Petri-Netze definiert wird, besteht nur in einem Fall die Möglichkeit zur separaten Abbildung einzelner Instanzen (Lee et al. 2010).
4. Lokale Modellabbildungen, die als Projektion auf eine globale Modellabbildung die Darstellung einer für das jeweilige Objekt privaten Innensicht erlauben, sind in bestehenden Ansätzen nicht vorzufinden.
5. Die Ausführungsüberwachung (Monitoring) von Prozessen ist typischerweise kein Bestandteil von Prozessmodellierungsansätzen. Diese wird, wie etwa bei (Boaro et al. 2011) von einer Workflow-Engine übernommen.

Die in dieser Arbeit betrachtete zeitliche Überlappung von Planung und Ausführung im Kontext von dezentral organisierten Systemen zieht eine integrierte Entwicklung und Ausführung nach sich. Die Arbeit übernimmt die häufig vorzufindende Darstellung von Petri-Netzen zur Angabe von Zuständen während der Ausführung, die eine Überwachung der Instanziierung zulassen. Der Grundgedanke der gemeinsamen Modellierung im Sinne der diskutierten Plattformen ist Gegenstand des in dieser Arbeit entwickelten Werkzeugs.

## 2.5 Ergebnisdiskussion

Der Dezentralisationsbegriff nimmt in der Ablauforganisation auf die Verteilung von personellen, räumlichen und zeitlichen Merkmalen während der Phase der Aufgabensynthese Bezug. Eine Dezentralisation der Aufbauorganisation führt während der Aufgabenträgerzuordnung zu einer verteilten Stellenbildung. Aus beiden Aspekten folgt die Bildung von autonomen und lose gekoppelten Teilsystemen, die einer nicht-hierarchischen Strukturierung eines Netzes entsprechen. Das objektorientierte Aufgabenmodell bildet diese Struktur unmittelbar ab.

Die Abbildung der Aufgaben in Geschäftsprozessmodellen und die daraus hervorgehenden Abbildungen von Workflows eines Informationssystems gliedern sich in separate Modellebenen. Vor dem Hintergrund unterschiedlicher Paradigmen der Prozessmodellierung und der Anforderung der Abbildung von autonomen und lose gekoppelten Teilsystemen wird die Modellierung von dezentral organisierten Systemen durch das semantische Objektmodell unterstützt. Die Abbildung der Workflow-Ebene kann hiervon unmittelbar abgeleitet und ggf. separat in BPMN erfasst werden. Die Transformation zwischen den Ebenen wird durch den generischen Architekturrahmen, die MDA und die Implementierungstechnologien der OMG unterstützt.

Eine dezentrale Organisation über mehrere Unternehmen hinweg wirkt sich auf die Prozesse und Paradigmen der Wertschöpfung aus und betrifft insbesondere interorganisationale Prozesse, die in Kooperation unter Beteiligung mehrerer verteilter Akteure ablaufen. Prozesse einer dezentralen Wertschöpfung weisen Merkmale einer interaktiven und kooperativen Gestaltung auf und sind Teil von betrieblichen Systemen, die als offene Plattformen charakterisierbar sind. Die Abbildung dieser Prozesse erfordert die Unterstützung kooperativer Merkmale hinsichtlich (a.) der Modellabbildung und (b.) der Interaktion während einer verteilten Modellbildung.

(a.) Die Modellabbildung muss die Merkmale kooperativer Prozesse (Collaborative Processes) in privaten und öffentlichen Prozessen unter Wahrung ausreichender Strukturflexibilität unterstützten.

(b.) Die Interaktion während der Modellbildung erfordert Methoden zur verteilten Modellierung, wie z.B. Modell-Repositories und Versionierungssysteme.

Für die Abgrenzung von Organisationen und die Bildung von Teilmodellen kann die Zerlegung von Objekten und Transaktionen in der Struktursicht des SOM herangezogen werden, wobei eine Untergliederung und Abgrenzung für Aspekte der

Sichtbarkeit und des Nachrichtenaustausches innerhalb und außerhalb von kooperierenden Objekten erforderlich ist. Weiterhin ist eine Integration des Modells mit dessen Erstellung zur Adressierung der Interaktion während einer verteilten Modellbildung notwendig, um insbesondere lokal abweichende Sichten auf öffentliche Prozesse eines dezentral organisierten, offenen Systems zu unterstützen.

Verwandte Ansätze zur Abbildung von kooperativen Prozessen sind nicht auf Prozesse einer dezentralen Wertschöpfung ausgerichtet. Insbesondere die Abbildung der von einzelnen Prozessen ausgehenden Systemstruktur und die mehrstufige Abgrenzung beliebig vieler Organisationen über beliebig viele Ebenen werden damit nicht erfasst. Verwandte Ansätze zur verteilten Modellierung berücksichtigen die Integration der Modellierung mit der Bildung von Modellen aufgrund ihrer Ausrichtung bisher nicht, so dass die Bildung von lokalen Abbildungen oder Sichten derzeit nicht realisierbar ist.

Die Integration der Modellbildung mit dem Entwurf und der Ausführung von Prozessen einer dezentralen Wertschöpfung kann durch Technologien zur Verteilung von Systemen ohne zentrale Koordination unterstützt werden. Das nachfolgende Kapitel behandelt in diesem Kontext Blockchain-Technologien als mögliche Grundlage der technischen Realisierung von dezentral organisierten Systemen.

# Kapitel 3

# Blockchain-Technologien

## 3.1 Blockchain-Systeme

Auf Grundlage des 2008 veröffentlichten Bitcoin-Whitepapers (Nakamoto 2008a) funktionieren Blockchain-Systeme heute als Distributed Ledger zur verteilten und konsistenten Speicherung von Transaktionshistorien, sowie als Smart-Contract-Plattformen zur Ausführung von Programmen. Das zentrale Merkmal dieser Technologien ist das Erzielen einer Übereinkunft (Consensus) über einen globalen Systemzustand, der einen für alle Beteiligten vertrauenswürdigen Single Point of Truth darstellt. Damit wird die Durchführung von Transaktionen zwischen einander nicht vertrauenden Teilnehmern eines Netzwerks ohne vertrauenswürdige Dritte in der Rolle von Intermediären möglich.

### 3.1.1 Komponenten und Merkmale

Die folgende Begriffsbestimmung diskutiert Arbeitsdefinitionen unter Berücksichtigung aktueller Entwicklungen von Blockchain-Systemen.

#### 3.1.1.1 Begriffsbestimmung

Das in der Literatur und in der Praxis vorherrschende Begriffsverständnis des Blockchain-Begriffs bezieht sich auf systemspezifische Architekturmerkmale (a.), die Datenstruktur (b.), die in der Transaktionshistorie gespeicherten Informationen (c.) oder wird in verallgemeinerter Form anhand von Systemkomponenten und Merkmalen (d.) beschrieben.

Der Begriff Blockchain wird aufgrund der Entwicklungshistorie häufig unter Angabe spezifischer Architekturmerkmale bestehender Systeme wie Bitcoin (Nakamoto 2008a) und Ethereum (Buterin 2013) definiert (a.). Diesen Systemen liegt das

Consensus-Verfahren Proof-of-Work zugrunde. Eine Blockchain kann hier verstanden werden als „sequence of blocks validated by the proof-of-work system, each linking to its predecessor all the way to the genesis block“ (Antonopoulos 2018, S. xxvi). Die Orientierung an bestehenden Systemen führt zur Verwendung des Begriffs als „loose umbrella term [...] to refer to systems that bear varying levels of resemblance to bitcoin and its ledger“ (Narayanan und Clark 2017). Eine ausschließliche Bezugnahme auf den Transfer von Geld oder Werteinheiten wird damit nicht unterstellt (Narayanan, Bonneau et al. 2016).

Ein an den Daten und der Datenstruktur (b.) ausgerichtetes Verständnis bezieht den Begriff auf den innerhalb eines Systems vorliegenden Datenbestand und die Datenstruktur Blockchain. Die Datenstruktur wird durch mehrere Blöcke beschrieben, die bis zum ersten Block jeweils auf einen Vorgänger verweisen (Antonopoulos 2017; Nofer et al. 2017). Blockchain „is used to refer to a data structure and occasionally to a network or system“ (Xu, Weber, Staples et al. 2017).

Bezogen auf die Informationen der Transaktionshistorie (c.) bezeichnet Blockchain im Kontext des Begriffs Distributed Ledger ein „dezentrales Hauptbuch“ (Sixt 2017) zur Speicherung von Transaktionen (Swan 2015). Ein Distributed Ledger wird im Folgenden als verteilte Transaktionshistorie bezeichnet. Über den Blockchain-Begriff hinausgehende Technologien zur verteilten Speicherung einer Transaktionshistorie werden als Distributed Ledger Technology bezeichnet (Böhme und Pesch 2017).

Ein allgemeineres Verständnis des Begriffs anhand von Systemkomponenten und Merkmalen (d.) verbreitet sich zunehmend (Böhme und Pesch 2017; Glaser und Bezzenberger 2015; Härer und Fill 2019a; Nofer et al. 2017; Urbach 2017; Xu, Weber, Staples et al. 2017). Als Arbeitsdefinition werden die folgenden Komponenten zur Beschreibung des systemischen Charakters herangezogen und in ein Architekturmodell überführt (Abschnitt 3.3).

#### 3.1.1.2 Begriff und Komponenten von Blockchain-Systemen

Ein Blockchain-System basiert auf einer spezifischen Datenstruktur rückwärts verketteter Blöcke zusammen mit Protokollen für die validierbar konsistente Speicherung von signierten Transaktionen innerhalb der verteilten Infrastruktur eines Peer-to-Peer-Netzwerks.

**Systemkomponenten:**

- **Datenstruktur**: Die Datenstruktur umfasst eine in Blöcken gespeicherte Transaktionshistorie. Blöcke enthalten, mit Ausnahme des ersten Blocks, jeweils einen an die hinterlegten Transaktionen gebundenen Verweis auf einen Vorgänger.
- **Protokoll**: Ein Protokoll beschreibt Anwendungsfunktionen zur Verteilung der Datenstruktur sowie für das Erzielen einer Übereinkunft (Consensus-Verfahren) hinsichtlich der enthaltenen Transaktionen.
- **Netzwerk**: Ein Peer-to-Peer-Netzwerk realisiert die direkte Kommunikation zwischen Teilnehmern und führt die Verteilung und das Consensus-Verfahren des Protokolls aus.

Dabei geht die Ausführung des Protokolls von den Komponenten des Systems aus, die das Verfahren als *Peers* (p) des zugrunde liegenden Netzwerks ausführen (Abbildung 3.1) und Transaktionen tätigen.

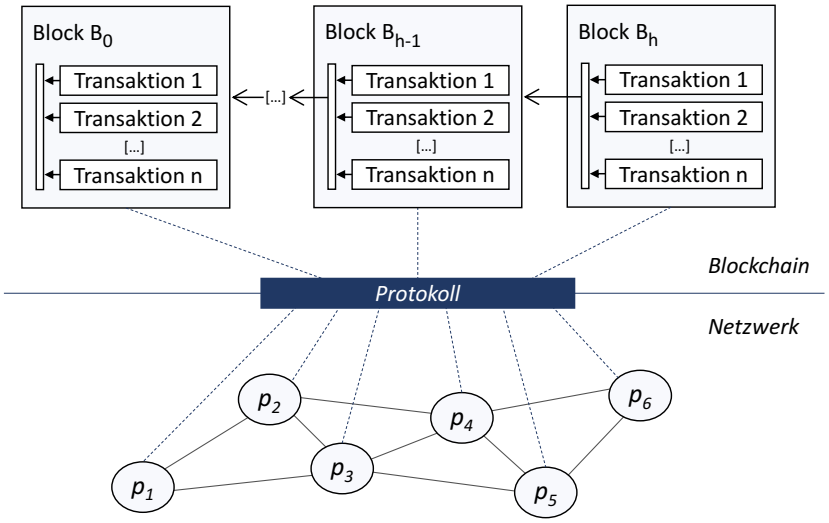



ABBILDUNG 3.1: Blockchain-System

Weiterhin wird auf Basis der im vorhergehenden Abschnitt genannten Quellen das nachfolgende Begriffsverständnis unterstellt:

**Blockchain** bezeichnet die Datenstruktur eines Blockchain-Systems mit einer darin enthaltenen Datenbasis, die mindestens aus Transaktionen besteht.

**Transaktion** bezieht sich in Blockchain-Systemen auf signierte und persistent zu speichernde Nachrichten, die Werte oder Daten zur Ausführung von Operationen auf der Datenbasis enthalten.

**Smart Contracts** sind anhand von Transaktionen gespeicherte Daten und Instruktionen eines annähernd Turing-mächtigen Programmiersystems, das anhand des Protokolls eines Smart-Contract-(Blockchain-)Systems ausgeführt wird.

**Smart-Contract-Systeme** besitzen die Eigenschaften von Blockchain-Systemen und sehen Transaktionen für die Speicherung und die Ausführung von Smart Contracts vor. Ein Smart-Contract-System kann als Universalrechenmaschine (Ferstl und Sinz 2013, 279 ff.) interpretiert werden, die verteilt ist.

**Distributed-Ledger-Systeme** bezeichnen Systeme, die eine verteilte Transaktionshistorie ausgehend von verteilten Transaktionen in einer Datenstruktur, wie etwa Blockchain, verwalten.

**Vertrauen** (Trust) bezieht sich auf eine Sache oder Person, die als verlässlich und zuverlässig betrachtet wird (Eckert 2018, S. 14). Die Bildung von vertrauenswürdigen Beziehungen (Trust Relationships) zwischen Personen und ihren digitalen Identitäten in verteilten Systemen betrifft das Thema Trust Management (Blaze et al. 1996). Eine Bildung von vertrauenswürdigen Beziehungen kann über Anwendungssysteme zustande kommen, wenn eine Verbindlichkeit der Zuordnung zwischen den Beteiligten und ihren digitalen Identitäten angenommen werden kann. In zentralisierten Systemen entsprechen die beteiligten Anwendungssysteme einer Trusted Third Party; z.B. einer Certificate Authority, die eine verbindliche Zuordnung durch ein zentral ausgestelltes Zertifikat bestätigt.

#### 3.1.1.3 Merkmale von Blockchain-Systemen

1. **Ordnung**: Das System stellt eine Ordnung (Notheisen et al. 2017) als partielle oder totale Ordnung über den Elementen der Datenstruktur her.
2. **Integrität**: Als Merkmal der Informationssicherheit verhindert die Sicherung der Integrität unautorisierte und unbemerkte Modifikationen von Daten (Böhme und Pesch 2017; Xu, Weber, Staples et al. 2017), z.B. durch kryptografische Hash-Funktionen. Die Merkmale Ordnung und Integrität erlauben einen Nachweis der syntaktischen Validität von Elementen der Datenstruktur.
3. **Verbindlichkeit**: Die Verbindlichkeit oder Nichtabstreitbarkeit (non-repudiation) (Xu, Weber, Staples et al. 2017) erstellter Transaktionen bezeichnet als Merkmal der Informationssicherheit (siehe Abschnitt 3.2.2) die stets mögliche Zuordnung von Transaktionen zu digitalen Identitäten von Teilnehmern. Eine Identität ist nicht notwendigerweise personenbezogen.

4. **Verteilung**: Das System ist verteilt (Notheisen et al. 2017) und besitzt die Merkmale von verteilten Systemen. Ein wesentliches Merkmal ist die Kommunikation anhand von Nachrichten zwischen lose gekoppelten Teilnehmern oder Knoten. Eine Verteilung erlaubt die Validierung der Integrität und der Verbindlichkeit, die anhand des Protokolls von Teilnehmern des Netzwerks durchgeführt werden kann. Diese Merkmale können in Kombination für den Nachweis einer ungefähren Erstellungszeit von Daten herangezogen werden (Gipp et al. 2015; Haber und Stornetta 1991).

5. **Transparenz**: Die Datenstruktur ist von den Teilnehmern des Systems einsehbar (Böhme und Pesch 2017; Urbach 2017; Xu, Weber, Staples et al. 2017), d.h. die Teilnehmer können Lese-Operationen auf die Datenstruktur anwenden. Diesem Merkmal steht eine Speicherung verschlüsselter Daten und eine Beschränkung des Teilnehmerkreises nicht entgegen.

6. **Unveränderlichkeit**: Die Datenstruktur speichert Daten unveränderlich (Böhme und Pesch 2017; Nofer et al. 2017; Notheisen et al. 2017; Urbach 2017; Xu, Weber, Staples et al. 2017). Das System speichert Daten dauerhaft und sieht als Schreib-Operation nur „Append“ für das Anfügen von Daten oder Blöcken vor, jedoch keine Operationen für das Ändern oder Löschen. Die Ausführung der Operation und die Verteilung der Daten erfolgen als Teil des Protokolls unter Beteiligung von weiteren Teilnehmern des Netzwerks.

7. **Trustless** oder **Trust-free**: Das System erlaubt die Durchführung vertrauenswürdiger Transaktionen, deren Integrität nachweisbar gewährleistet ist, ohne eine zentrale Instanz (Böhme und Pesch 2017), unter einander nicht vertrauenden Teilnehmern eines Netzwerks (Greiner und Wang 2015). Der Zustand des Systems wird nicht von einzelnen Teilnehmern kontrolliert. Ein Verfahren des Protokolls erzielt eine Übereinkunft (Consensus) hinsichtlich des Systemzustandes (Nofer et al. 2017; Notheisen et al. 2017). Im Falle von dezentralen Blockchain-Systemen (Abschnitt 3.1.2.4) bestehen bezüglich der Kontrolle des Systems gleiche Zugriffsrechte für alle Teilnehmer (Xu, Weber, Staples et al. 2017).

Die Merkmale verteilter Systeme und der Informationssicherheit sowie die Transparenz sind für sich genommen bereits in einer Vielzahl von Anwendungssystemen implementiert. Die Kombination der Merkmale führt unter Einbeziehung der genannten Komponenten zu Unveränderlichkeit und, in Kombination mit diesem Merkmal, zu Trustless. Die genannten Merkmale werden von heute implementierten Blockchain-Systemen unterstützt (Notheisen et al. 2017; Xu, Pautasso et al.

2016), sie sind jedoch nicht notwendigerweise in allen Systemen gegeben. Die Merkmale Unveränderlichkeit und Trustless sind unter Verwendung von Consensus-Verfahren wie Proof-of-Work (Abschnitt 3.3.4.3) von weiteren Faktoren abhängig, insbesondere von der Aufwendung von Ressourcen (siehe 3.3.5.3).

**Blockchain-Systeme unter Einschränkung von Zugriffen und Zugriffsrechten**

Blockchain-Systeme können Einschränkungen bezüglich des Zugriffs und der Zugriffsrechte vorsehen (Buterin 2015; Wüst und Gervais 2017; Xu, Weber, Staples et al. 2017).

- Einschränkung des Zugriffs
  - **Öffentlich** (Public): Ein öffentliches Blockchain-System ist ein offenes System, dessen Zugriff nicht auf bestimmte Teilnehmer beschränkt ist.
  - **Privat** (Private): Ein privates Blockchain-System schränkt den Zugriff auf ausgewählte Teilnehmer ein, die z.B. mehreren Unternehmen angehören. Aus Sicht des Netzwerks kann eine bekannte Menge von Knoten angenommen werden. Private Blockchain-Systeme mit mehreren Knoten werden auch als Consortium Blockchain (Buterin 2015) bezeichnet.
- Einschränkung der Zugriffsrechte
  - **Permissionless**: Ein permissionless Blockchain-System trifft keine Einschränkungen der Zugriffsrechte.
  - **Permissioned**: Ein permissioned Blockchain-System schränkt die Zugriffsrechte bestimmter Operationen für ausgewählte Teilnehmer ein.

Die genannten Systemtypen werden nachfolgend im Kontext dezentraler Systeme diskutiert.

## 3.1.2 Dezentrale Systeme

### 3.1.2.1 Einordnung

Der Begriff der Dezentralisierung besitzt auf der technischen Ebene eine von der organisationstheoretischen Dezentralisation abweichende Bedeutung (vgl. Kapitel 2.1.2.2). Die Dezentralisation beschreibt eine Entwicklung von einem Zentrum weg, z.B. anhand der Delegation einer Entscheidungsbefugnis an einen oder mehrere Aufgabenträger. Im Kontext von verteilten Systemen kann die Dezentralisierung anhand der Verteilung des Systems und der Koordination definiert werden.

**Topologischer Dezentralisierungsbegriff**

In Zusammenhang mit dem Dezentralisierungsbegriff werden mitunter die von Baran (1964) beschriebenen Grundlagen der paketvermittelten Kommunikation in verteilten Kommunikationsnetzen angeführt. Hierzu zählen drei Verteilungsarten. Zentrale Netzwerke (1.) besitzen eine Stern-Topologie, ausgehend von einem zentralen Knoten. Verteilte Netzwerke (2.) bilden eine Kombination von Teilnetzwerken, die Stern- und Mesh-Topologien aufweisen. Dezentrale Netzwerke (3.) besitzen eine Topologie, die sich aus einer Kombination von mehreren Netzwerken mit Stern-Topologien ergibt. Die zentralen Knoten der Stern-Topologien sind miteinander verbunden, wobei keine Verbindungen mit weiteren Knoten bestehen. In dieser Definition beschreibt die Dezentralisierung eine hierarchische Struktur zur Kommunikation. Die über die Kommunikation hinausgehende Verteilung in Blockchain-Systemen wird mit dieser Definition nicht erfasst. Anstelle des topologischen Begriffsverständnisses wird daher die nachfolgende Abgrenzung von verteilten und dezentralen Systemen herangezogen.

#### 3.1.2.2 Merkmale dezentraler Systeme

Basierend auf den Charakteristika von Peer-to-Peer-Netzwerken (Abschnitt 3.2.1.2) kann der Begriff des dezentralen Systems als Kombination der folgenden Merkmale verstanden werden:

- **Verteilung des Systems**: Das System besitzt die Merkmale eines verteilten Systems (Abschnitt 3.2.1) und unterstützt eine Verteilung der Systemkomponenten unabhängig von der Topologie des Netzwerks (Steen und Tanenbaum 2017, S. 80). Bei Betrachtung dieses Merkmals allein erfolgt keine Unterscheidung zwischen dezentralen und verteilten Systemen.
- **Verteilung der Koordination**: Protokolle in Peer-to-Peer-Netzwerken (Steinmetz und Wehrle 2005, S. 10) erlauben zusammen mit Verfahren für das Erzielen einer Übereinkunft (Consensus) eine Verteilung der Koordination (Abschnitte 3.2.1.2 bzw. 3.2.1.7). Die Regeln der Koordination werden durch ein Protokoll repräsentiert, das unter Beteiligung aller Systemkomponenten zur Ausführung kommt. Die Korrektheit der Ausführung ist von allen Systemkomponenten validierbar.

Ein dezentrales System ist ein verteiltes System, in dem eine verteilte Koordination durch eine validierbare Ausführung eines Protokolls unter Beteiligung der Systemkomponenten erreicht wird.

#### 3.1.2.3 Dezentrale Koordination

Die Koordination in einem dezentralen System anhand eines Protokolls wird im Folgenden als dezentrale Koordination bezeichnet. Daneben bestehen von Objekten ausgehende Koordinationsformen in zentralen oder verteilten Systemen, die hierarchische und nicht-hierarchische Ausprägungen besitzen (Ferstl und Sinz 2013, S. 216). Tabelle 3.1 stellt die Begriffe im Zusammenhang dar.

Die Regeln des Protokolls legen die globale Koordination des Systems fest, z.B. in Form eines Consensus-Verfahrens in einem dezentralen Blockchain-System. Sofern ein Protokoll die Bildung von Extensionen zur Laufzeit unterstützt, sind spezifische Regeln zur Festlegung der lokalen Koordination einzelner Komponenten definierbar, z.B. in Form von Smart Contracts.

| | | **Systemarchitektur** | |
|---|---|---|---|
| | | **Zentral** | **Verteilt** |
| **System-koordination** | **Objektzentriert** | Zentrales System | Verteiltes System |
| | **Protokollzentriert** | - | Dezentrales System |

TABELLE 3.1: Abgrenzung von dezentralen Systemen

Ein Smart Contract legt beispielsweise aufeinanderfolgende Blockchain-Transaktionen fest, die einer Vereinbarung und Durchführung nicht-hierarchisch koordinierter betrieblicher Transaktionen entsprechen. Anwendungen betreffen den Verkauf von digitalen Gütern, Prognosemärkte oder dezentrale autonome Organisationen (DAO) (siehe 3.4). Der in Kapitel 4 vorgestellte Ansatz wendet die dezentrale Koordination auf die Entwicklung von Informationssystemen an.

#### 3.1.2.4 Dezentrale Blockchain-Systeme

Dezentrale Blockchain-Systeme verwenden ein Protokoll zur Ausführung eines Consensus-Verfahrens (Xu, Weber, Staples et al. 2017), das eine dezentrale Koordination realisiert. Die hiermit realisierbaren Merkmale Trustless und Unveränderlichkeit definieren sich durch dieses Protokoll und sind nur dann gegeben, sofern das Blockchain-System hinsichtlich des Protokolls operational ist. Dies ist der Fall, wenn anhand des Consensus-Verfahrens ein Systemzustand bestimmt werden kann. Die Merkmale können von dezentralen Blockchain-Systemen sowie, unter Einschränkungen, von partiell dezentralen Blockchain-Systemen erfüllt werden.

**Dezentrale Blockchain-Systeme**

Dezentrale Blockchain-Systeme sind öffentlich und permissionless. Diese Systeme stellen den Ausgangspunkt der Entwicklung von Blockchain-Systemen dar (Buterin 2013; Nakamoto 2008a; Wood 2014). Das Protokoll wird verteilt und validierbar ausgeführt. Der öffentliche Zugriff erlaubt eine Validierung der Integrität der Datenstruktur und eine Validierung der Verbindlichkeit der Transaktionen. Systeme dieser Art werden als *dezentralisiert* (Wüst und Gervais 2017) oder *vollständig dezentralisiert* (Buterin 2015; Xu, Weber, Staples et al. 2017) bezeichnet.

Ein dezentrales Blockchain-System bleibt operational, solange anhand des Consensus-Verfahrens ein Systemzustand bestimmt werden kann. Hierfür müssen fehlerfrei operierende Systemkomponenten typischerweise mehr als 50% der insgesamt innerhalb des Netzwerks vorhandenen Ressourcen bereitstellen (Abschnitte 3.3.4, 3.3.5.3). Die verteilte Speicherung aller Daten sichert die Verfügbarkeit (Abschnitt 3.3.5).

**Partiell dezentrale Blockchain-Systeme**

Partiell dezentrale Blockchain-Systeme sind permissioned und erlauben eine Validierung der Ausführung des Protokolls durch mehrere ausgewählte Teilnehmer des Systems, z.B. Unternehmen eines Konsortiums (Buterin 2015; Xu, Weber, Staples et al. 2017). Ein System dieser Art ist privat und kann einen öffentlichen Zugriff für diejenigen lesenden Zugriffsrechte gestatten, die eine öffentliche Validierung der Ausführung des Protokolls erlauben. Die Validierung bezieht sich auch hier auf die Überprüfung der Integrität und der Verbindlichkeit. Lesende und schreibende Zugriffsrechte können in Abhängigkeit der Anwendung feingliedrig vergeben werden, z.B. Operationen zum Anlegen oder Löschen von Modellen und Modellelementen (siehe Abschnitt 3.4.5).

Ein partiell dezentrales Blockchain-System kann daher neben den Consensus-Verfahren öffentlicher Systeme auf weitere BFT-Verfahren wie PBFT zurückgreifen (Abschnitte 3.3.4). Diese Systeme sind typischerweise dann operational, wenn der Anteil fehlerfrei operierender Knoten mehr als 2/3 beträgt. Die Fehlertoleranz entspricht dem State of the Art von Consensus-Verfahren aus verteilten Systemen (Abschnitt 3.2.1.7). Gegenüber öffentlichen Blockchain-Systemen besteht eine höhere Skalierbarkeit, allerdings wird die Fehlertoleranz mit einer sinkenden Anzahl an Knoten eingeschränkt. Dies betrifft die geringere Verfügbarkeit aufgrund der eingeschränkten Verteilung der Daten sowie die geringere Absicherung gegenüber fehlerhaft operierenden Knoten bei der Bestimmung des Systemzustandes anhand des

Consensus-Verfahrens. Ein öffentlicher Zugriff mit bestimmten Zugriffsrechten zur Validierung kann die Fehlertoleranz erhöhen.

**Zentralisierte Blockchain-Systeme**

Vollständig private Blockchain-Systeme (Buterin 2015) unter der Kontrolle einer einzelnen Instanz sind zentralisierte Systeme. Diese Systeme entsprechen Datenbanksystemen, die Anwendungsfunktionen zur Sicherung der Integrität und Verbindlichkeit bereitstellen. In Abgrenzung zu Datenbanksystemen werden in dieser Arbeit dezentrale und partiell dezentrale Blockchain-Systeme betrachtet.

#### 3.1.2.5 Smart Contracts zur Repräsentation von Vertragsbeziehungen in dezentralen Systemen

Die dezentrale Koordination wird einerseits durch das Consensus-Verfahren realisiert, das als inhärenter Bestandteil des Protokolls übereinstimmende Systemzustände verteilt festlegt. Darüber hinaus kann das Protokoll die Ausführung von Smart Contracts vorsehen, die als Extension des Protokolls zur Laufzeit verstanden werden kann. Die Festlegung der Regeln eines Smart Contracts eignet sich daher zur Repräsentation von Vertragsbeziehungen innerhalb von dezentralen Systemen.

In Analogie zur Festlegung von Geschäftsbeziehungen anhand von Verträgen entwirft Szabo (1994, 1997) Smart Contracts als Konzept zur Formalisierung und Absicherung von Beziehungen in offenen Kommunikationsnetzwerken. Die Implementierung dieses Konzeptes durch eine in Integrität und Verbindlichkeit abgesicherte Ablage und Ausführung von Smart Contracts wird nunmehr in Blockchain-Systemen wie Ethereum (Buterin 2013; Buterin et al. 2014; Wood 2014) unterstützt. Damit wird die Frage aufgeworfen, wie formalisierte Beziehungen in Smart Contracts und privatrechtliche Verträge in Relation zu rechtlichen Rahmenbedingungen stehen (siehe z.B. Filippi und Hassan 2016; Goldenfein und Leiter 2018; Levy 2017).

In der Spezifikation des Blockchain-Systems Ethereum gibt Wood (2014) im Kontext des in den 1990er-Jahren beschriebenen Smart-Contract-Konzepts und der damit verbundenen algorithmischen Durchsetzung von Vereinbarungen an: „Though no specific system was proposed to implement such a system, it was proposed that the future of law would be heavily affected by such systems. In this light, Ethereum may be seen as a general implementation of such a crypto-law system.“

Als „Lex Cryptographia“ bezeichnen Wright und De Filippi (2015) anhand von Smart Contracts festgelegte Regelungen, die nicht mehr vollständig durch das bestehende Rechtssystem erfasst werden. Die Operationalisierung der Regelungen basiert vollständig auf Daten, die autonom zur Ausführung kommen. Zur Einordnung von privatrechtlichen Verträgen als Smart Contracts eines „crypto-law“-Systems und einer Plattform eines „Lex Cryptographia“ stellt Tabelle 3.2 die Repräsentation und Operationalisierung von Verträgen und Smart Contracts gegenüber.

| | | **Vertrag** | **Smart Contract** |
|---|---|---|---|
| **Repräsentation** | **Beschreibungsform** | Wet Code | Dry Code |
| | **Beschreibungsmittel** | Natürliche Sprache | Quellcode, Bytecode |
| **Operationalisierung** | **Inferenz** | Wissensbasiert (u.a. explizites Wissen, Erfahrungswissen, Fallbasis) | Daten- und informationsbasiert |
| | **Ausführung** | Heteronom, Nichtdeterminiert | Autonom, Determiniert |

TABELLE 3.2: Verträge und Smart Contracts (Beschreibungsform nach Szabo (2018))

### Repräsentation

Wet Code und Dry Code (Szabo 2008) nehmen auf die Beschreibungsform von Verträgen und Smart Contracts Bezug. Wet Code ist eine für den Menschen interpretierbare Form der Beschreibung, während Dry Code eine von Computern interpretierbare Form darstellt. Sprachliche Beschreibungen liegen als Wet Code vor. In Ausnahmefällen kann Sprache Dry Code darstellen, z.B. im Falle von Domain-Namen (Szabo 2008). In dieser Klassifikation liegen Verträge als Wet Code vor und unterliegen somit der menschlichen Interpretation. Smart Contracts sind in Form von Dry Code erfasst, so dass eine stets gleichartige Interpretation angenommen werden kann.

Die Beschreibungsmittel von Verträgen basieren auf natürlicher Sprache, während Smart Contracts auf Quellcode zurückgreifen. Rechtsnormen von privatrechtlichen Verträgen beziehen sich auf allgemeine Gesetze des Privatrechts, sowie spezielle Gesetzte in Abhängigkeit der Vertragsart. Eine unmittelbar auf Smart Contracts bezogene spezielle Rechtsnorm existiert bislang nicht, so dass allgemeine Gesetze in

Verbindung mit den von spezifischen Smart Contracts berührten speziellen Gesetzen zur Anwendung kommen. Dem nicht entgegenstehend ist eine Einbeziehung von Smart Contracts in Verträge.

Ein in einer Blockchain abgelegter Smart-Contract-Code kann dort innerhalb einer Laufzeitumgebung zur Ausführung gebracht werden, z.B. in der als Stack-Maschine konzipierten Ethereum Virtual Machine (EVM) (Wood 2014). Der abgelegte Code liegt typischerweise als kompilierter Bytecode vor, der ausgehend von einem Quellcode einer Programmiersprache übersetzt wurde. Im Falle von Ethereum erfolgt die Entwicklung in einer Programmiersprache wie Solidity oder Serpent, deren Quellcode vor der Ablage in der Blockchain in Bytecode für die EVM übersetzt wird. Die Ausführung wird von Dritten nicht beeinflusst.

**Operationalisierung**

Das Wissensmanagement unterscheidet grundlegend zwischen Daten, Information und Wissen (siehe z.B. Bodendorf 2006, S. 1). Weiterhin werden in Organisationen mehrere Formen der Umwandlung (Nonaka und Takeuchi 1995) zwischen taziten und expliziten Wissensarten unterschieden. Die Inferenz ist im Falle von Verträgen wissensbasiert, da sich Schlussfolgerungen direkt auf rechtliche Rahmenbedingungen beziehen. Verträge stellen explizites Wissen dar, dessen Umsetzung und Ausführung in Form von Handlungen heteronom durch die beteiligten Vertragspartner erfolgen. Die Verarbeitung von explizitem Wissen in Verträgen führt aufgrund der Auslegung und Interpretation und des Hinzukommens von neuen Fällen im Zeitverlauf zu nichtdeterminierten Ergebnissen. Demgegenüber ist die Ausführung eines Smart Contracts autonom und determiniert, da der in einer Blockchain hinterlegte Bytecode validierbar zur Ausführung kommt. Der Output wird eindeutig durch die Input-Parameter bestimmt.

**Diskussion**

„Lex Cryptographia“ geht davon aus, dass die in Smart Contracts autonom stattfindende und algorithmisch definierte Ausführung maßgeblich für eine Vertragsbeziehung ist. Wenn der Vertragsgegenstand der Objektart Information entspricht und Auswirkungen der Objektart Information nach sich zieht, ist eine Blockchain-interne Umsetzung und Durchsetzung des Smart Contracts unmittelbar möglich. Ein Beispiel sind Nutzungsverträge von digitalen Produkten oder Dienstleistungen, die eine Nutzung in Abhängigkeit einer vereinbarten Gebühr, z.B. pro Zeit, in Einheiten virtueller Währung (Tokens) innerhalb eines Blockchain-Systems abrechnen (siehe z.B. Crosby 2016). Im Falle von vernetzten physischen Objekten kann ein

Smart Contract mit diesen interagieren. Solche Objekte verändern als „Smart Property“ (Szabo 1997) Parameter in Abhängigkeit des Smart Contracts, z.B. Zugang (physisch), verfügbare Services oder Zugriffsrechte. Limitationen bestehen hinsichtlich der nicht-garantierten Zustandskonsistenz bezüglich der Abbildung der Realwelt. Eine aktuelle Entwicklung in diesem Kontext ist die Übertragung von Konzepten des bestehenden Rechtssystems auf analoge Konzepte für Smart Contracts (Kostecki und Sharma 2018; Werbach 2018).

## 3.2 Grundlagen zu verteilten Systemen und zur Anwendung von Kryptografie

Dieser Abschnitt bespricht technische und anwendungsorientierte Grundlagen von Blockchain-Technologien und deren Anwendungen zur Realisierung der verteilten und gemeinsamen Historisierung von Transaktionen in einem dezentralen System.

### 3.2.1 Verteilte Systeme

#### 3.2.1.1 Einführung

Ein verteiltes System besteht aus autonomen Systemkomponenten, die zur Erreichung gemeinsamer Ziele zusammenwirken. Aus Sicht eines Anwenders tritt das System als einzelnes und einheitliches System in Erscheinung (Steen und Tanenbaum 2017, S. 2). Die miteinander kooperierenden Komponenten des Systems sind lose gekoppelt und interagieren anhand von Nachrichten (Ferstl und Sinz 2013, S. 223). Eine Systemkomponente wird unter Bezugnahme auf Kommunikationsnetzwerke im Folgenden als Knoten oder Node bezeichnet. Der im Kontext von verteilten Systemen etablierte Begriff (Steen und Tanenbaum 2017, S. 2) bezieht Software- und Hardware-Systeme ein, z.B.

- auf Client-Server-Architekturen beruhende Web-Client- und Web-Server-Software,
- auf Peer-to-Peer-Architekturen beruhende Node-Software in Blockchain-Systemen sowie
- Hardware-Systeme, die verteilte Recheneinheiten zur Prozessausführung koppeln.

Client-Server- sowie Peer-to-Peer-Architekturen sind Gegenstand des nachfolgenden Abschnitts. Nicht näher betrachtet werden Themen zu Hardware-Systemen und Rechenarchitekturen, die z.B. die Verteilung von Ressourcen wie CPUs oder Speichern zur Ausführung von parallelen Prozessen betreffen (siehe z.B. Solihin 2015). Weiterhin betrachtet werden grundlegende Problemstellungen in verteilten Systemen, welche die Trade-Offs des CAP-Theorems, die Ordnung von Ereignissen und die Fehlertoleranz während des Erzielens einer Übereinkunft zwischen Knoten (Consensus) betreffen. Eine Diskussion entsprechender Verfahren verteilter Systeme im Kontext von Blockchain-Systemen bildet den Abschluss des Abschnitts.

#### 3.2.1.2 Client-Server- und Peer-to-Peer-Architekturen

Die Architekturformen Client-Server und Peer-to-Peer beschreiben den Aufbau und die Kommunikation von verteilten Anwendungssystemen, die über lokale und globale Netzwerke interagieren. Von der Hardware und der physischen Topologie des Netzwerks abstrahierend wird damit ein Kommunikationssystem beschrieben (Kurose und Ross 2017), dessen Knoten oder Nodes einzelne Software-Systeme darstellen, während Kanten als Links Kommunikationsbeziehungen beschreiben.

**Client-Server-Architekturen**

In einer Client-Server-Architektur stehen die Rollen der beteiligten Systeme ex ante fest. Dienstanfragen (Requests) gehen von Client-Systemen aus, werden in Server-Systemen verarbeitet, und ggf. anhand von Dienstantworten (Responses) beantwortet. Server-Systeme können als Client-System agieren, um in mehrschichtigen Architekturen verteilter Anwendungssysteme weitere Server-Systeme einzubinden. Ausprägungen von mehrschichtigen Client-Server-Architekturen bilden sich beispielsweise durch zwei bis vier Schichten von Server-Systemen, die funktional anhand des ADK-Strukturmodells abgrenzbar sind (Ferstl und Sinz 2013, S. 321, 340, 494). Eine typische 3-Schichtenarchitektur umfasst Kommunikation (K), z.B. Web-Server, (2.) Anwendungsfunktionen (A), z.B. Applikationsserver und Ausführungsumgebungen, sowie Datenhaltung (D), etwa in Form von Datenbanksystemen. Einzelne Komponenten können als Cluster zusammenhängender Server-Systeme auftreten, das verteilt Dienstanfragen beantwortet, oder in Form von Partitionierungen und Replikationen zur Lastverteilung und Steigerungen der Ausfallsicherheit.

Mehrschichtige Kommunikationsprotokolle definieren die Abfolge und den Aufbau von Request- und Response-Nachrichten. Auf der Ebene einer Anwendung können im Kontext von service-orientierten Architekturen Nachrichtenaustauschmuster (Message Exchange Pattern) angegeben werden, z.B. In-Out von SOAP-Nachrichten (W3C 2007a,b). Das Protokoll ist Bestandteil einer Protokollhierarchie. In der Klassifikation des OSI-Referenzmodells sind für die angegebenen Beispiele die Schichten 3, 4 und 7 relevant, die z.B. in Form von IP (3), TCP oder UDP (4) und HTTP (7) realisiert werden.

Ein verteiltes Anwendungssystem der beschriebenen Architektur unterstellt, dass sich die beteiligten Server-Systeme gegenseitig vertrauen. Innerhalb des Beispiels ist es erforderlich, Dienstanfragen über mehrere Schichten hinweg zu stellen. Während der Übertragung von Daten über Links des Netzes ist die Integrität durch die Erkennung von Übertragungsfehlern in Transportprotokollen wie TCP sowie durch

Verschlüsselungsprotokolle wie TLS gesichert. Während der Verarbeitung von Requests innerhalb eines Nodes besteht typischerweise kein Mechanismus zur Gewährleistung der Integrität und keine Möglichkeit einer von außen nachvollziehbaren Validierung der Verarbeitung. Aus der Sicht eines Clients ist die Verarbeitung eines Requests in einem Server hinsichtlich der Integrität der Verarbeitung nicht bewertbar.

**Peer-to-Peer-Architekturen**

Eine Peer-to-Peer-Architektur implementiert eine direkte und dezentrale Kommunikation zwischen Knoten eines Kommunikationsnetzes. Etablierte Ansätze wie verteilte Hash-Tabellen erlauben die direkte Kommunikation zwischen Knoten, die als Netzwerkteilnehmer zur Laufzeit mit einer vorab nicht festgelegten Menge weiterer bekannter Knoten vernetzt werden (Steen und Tanenbaum 2017, S. 82 ff.). Eine Unterscheidung zwischen dienstanfragenden Clients und diensterbringenden Servern wird auf der Ebene des Netzes nicht getroffen. Peers kommen und gehen dabei zur Laufzeit. Ein System dieser Architekturform ist ein selbstorganisiertes System von ebenbürtigen und autonomen Einheiten (Peers), das auf die gemeinsame Nutzung verteilter Ressourcen in einem Netzwerk abzielt, ohne auf zentrale Server zurückzugreifen (Steinmetz und Wehrle 2005, S. 10). Anwendungen umfassen z.B. den verteilten Datenaustausch, die Verteilung von Software-Updates und beliebige Blockchain-Transaktionen (siehe Kapitel 3.3).

Ein Protokoll regelt die Kommunikation und Koordination der Knoten, die innerhalb des Netzes jeweils mit einer begrenzten Anzahl weiterer Knoten direkt in Verbindung stehen. Die Verteilung über mehrere Knoten hinweg betrifft die (a.) die Verteilung von Daten an alle Knoten oder (b.) Anfragen (Lookups) nach bestimmten Daten, die von einzelnen Knoten an das Netzwerk gerichtet sind. Die Verteilung von Daten (a.) realisieren Gossip-Protokolle (Birman 2007). Ein Gossip-Protokoll leitet eingehende Nachrichten an alle direkt verbundenen Knoten weiter. Die Ausführung des Protokolls in allen Knoten propagiert Daten parallel zwischen Knoten des Netzes. Protokolle zur Anfrage nach bestimmten Daten (b.) sehen zunächst die Propagation von Lookup-Nachrichten zur Ermittlung derjenigen Knoten vor, denen die gesuchten Daten vorliegen (1.), sowie anschließend die Herstellung einer direkten Verbindung und die Übertragung der Daten von Peer zu Peer (2.). Ein Routing-Verfahren leitet Anfragen eines gegebenen Lookups (1.) an den jeweiligen Zielknoten weiter. Durch Routing ermittelte und miteinander verbundene Knoten bilden ein Overlay-Netzwerk aus logischen Kommunikationsbeziehungen (Lua et al. 2005).

Eine erste Generation unstrukturierter Peer-to-Peer-Ansätze stellen Verbindungen zu bekannten Knoten ohne das Konzept der Overlay-Netzwerke ad-hoc her (Steen und Tanenbaum 2017, S. 84). Diese Ansätze sind aufgrund ihrer Funktionsweise hinsichtlich der Skalierbarkeit eingeschränkt und greifen als hybride Verfahren z.T. auf zentrale Server zurück (Steinmetz und Wehrle 2005, S. 15).

*Aufbau strukturierter Peer-to-Peer-Ansätze*

Strukturierte Ansätze sind auf den Abruf indexierter Daten spezialisiert und nutzen hierfür häufig verteilte Hash-Tabellen (DHT). Die Indexierung und Adressierung eines Datensatzes ist als Content-Addressable Data Storage durch den Inhalt des Datensatzes determiniert. Hierfür werden die zu speichernden Daten auf eine Hash-Funktion angewendet, deren Funktionswert als Identifikator verwendet wird. Die Hash-Tabelle enthält Key-Value-Paare, die für einen gegebenen Daten-Identifikator als Key eine Netzwerk-Adresse des zur Speicherung verwendeten Knotens verwaltet. Häufig werden Binärbäume oder Varianten von B-Bäumen für die Indexstruktur verwendet, die partitioniert bei Teilnehmern des Netzes vorliegt. Die Komplexitätsklasse für den Abruf von Daten liegt für DHT-basierte Ansätze typischerweise bei O(log N) für N Knoten (Steinmetz und Wehrle 2005, S. 16, 87).

Ein Protokoll verwaltet die Verteilung der Indexstruktur und der Daten und legt Anfrage- und Antwort-Nachrichten fest. Grundlegende Komponenten sind (1.) die Indexstruktur, (2.) die Adressierung und die Suche von Daten sowie (3.) die Übertragung durch Routing der Daten zum anfragenden Knoten (Steinmetz und Wehrle 2005, S. 42-45, 86–93).

Ein Beispiel ist der auf DHT basierende Kademlia-Ansatz, der einen Binärbaum zur hierarchischen Adressierung von Datensätzen in Knoten verwendet, die Blätter des Baumes bilden (1.). Für die Suche eines Knotens, bestimmt durch Anwendung der Daten auf eine Hash-Funktion, wird dessen Adresse als hierarchische Pfadangabe innerhalb des Baumes interpretiert (2.). Ein Knoten kennt ausgehend von seiner Position p innerhalb des Baumes sämtliche Teilbäume des Pfades von p, die von Knoten zwischen p und dem Wurzelknoten ausgehen. Jeder Teilbaum ist ausgehend von einem bekannten Knoten in der Lage, Anfragen an untergeordnete Zielknoten weiterzuleiten. Eine Minimierung der Distanzen zwischen Knoten wird durch Anwendung der Adressen auf eine XOR-Funktion erreicht (3.).

Kademlia liegt in verschiedenen Software-Implementierungen vor. Beispiele sind das Torrent-Protokoll für den Austausch von Dateien (Birman 2012, S. 136), global verteilte Dateisysteme wie „Swarm“ oder „InterPlanetary File System“ (IPFS)

(Tenorio-Fornés et al. 2018) und das Blockchain-System Ethereum, welches das beschriebene Adressierungsschema zum Auffinden von Knoten verwendet (Buterin et al. 2014). Die Implementierungen betreffen Schicht 7 des OSI-Modells, so dass sich für die genannten Beispiele die Protokollhierarchie IP (Schicht 3), TCP oder UDP (Schicht 4) und Peer-to-Peer-Protokoll (Schicht 7) ergibt.

**Diskussion**

Insgesamt entspricht die Koordination in Peer-to-Peer-Systemen hinsichtlich der örtlichen Verteilung und der Verteilung der Protokollausführung einer dezentralen Koordination (Abschnitt 3.1.2.4), da die Regeln der Kommunikation und Koordination systemimmanent als Teil des Peer-to-Peer-Protokolls festgelegt sind.

Im Gegensatz zu Client-Server-Architekturen impliziert eine Peer-to-Peer-Architektur keinerlei Vertrauensbeziehungen zwischen den beteiligten Knoten. Eine Integritätssicherung wird durch die Verwendung von Content-Addressable Data Storage erreicht, sofern moderne kryptografische Hash-Funktionen zum Einsatz kommen. Im Unterschied zu Client-Server-Systemen ist die Verarbeitung in erster Linie auf Operationen des Protokolls beschränkt, das vorab festgelegt und öffentlich definiert ist.

Zur Abbildung der direkten Kommunikation zwischen Knoten, die ohne Intermediär oder „Trusted Third Party“ kommunizieren, bieten Peer-to-Peer-Architekturen die Grundlage für die Verteilung der Datenbasis in Blockchain-Systemen. Eine weitergehende Verarbeitung, die nicht auf die Regeln eines Protokolls beschränkt ist, wird in Smart Contracts realisiert. Darüber hinaus wird die Nutzung von Content-Addressable Data Storage anhand von IPFS (Tenorio-Fornés et al. 2018) für die dezentrale Speicherung von Daten in Blockchain-Systemen diskutiert.

#### 3.2.1.3 CAP-Theorem

In verteilten Systemen bestehen Einschränkungen hinsichtlich der parallel durchführbaren Schreib- und Leseoperationen auf verteilt vorliegenden Daten. Die Ausführung der Operationen ist von der Zuverlässigkeit der Nachrichtenkommunikation über Knoten und Kommunikationskanäle abhängig.

Das CAP-Theorem (Brewer 2000; Fox und Brewer 1999) formuliert die sich hieraus ergebenden Implikationen anhand von drei spezifischen Merkmalen, zwischen denen infolge der genannten Problemstellung Zielkonflikte bestehen.

### Theorem

Das Theorem betrachtet drei Merkmale (Brewer 2000; Gilbert und Lynch 2002). Es besagt: *für ein System mit einer gemeinsam genutzten Datenbasis (shared-data system) können maximal zwei der Merkmale zutreffen*.

- **Konsistenz** (Consistency) der verteilt vorliegenden Daten des Systems. Konsistenz wird angenommen, wenn eine totale Ordnung von zugrunde liegenden Elementaroperationen konstruierbar ist.
- **Verfügbarkeit** (Availability) der verteilt vorliegenden Daten des Systems. Verfügbarkeit wird angenommen, wenn jeder Request an einen nicht-fehlerhaften Knoten beantwortet wird. Dabei wird unterstellt, dass die zur Erzeugung von Requests in Knoten ausgeführten Prozesse terminieren.
- **Partitionstoleranz** (Partition Tolerance) des Systems gegenüber der Bildung von Teilsystemen, zwischen denen keine Kommunikation möglich ist. Bei Vorliegen einer Partition wird der Verlust aller Nachrichten zwischen Knoten angenommen, die sich in unterschiedlichen Teilsystemen befinden.

### Merkmalskombinationen

Aufgrund der Zielkonflikte sind folgende Merkmalskombinationen möglich:

- **Konsistenz** und **Partitionstoleranz** sind beispielsweise im Falle von verteilten Datenbanksystemen gegeben. Ein charakteristisches Merkmal sind pessimistische Locking-Strategien, die verteilte Zugriffe nur bei gesetzten Lese- und Schreibsperren ermöglichen und somit die Verfügbarkeit des Systems für parallele Operationen beeinträchtigen. Konsistenz wird durch die anhand von Locks erreichte Serialisierung von Elementaroperationen gewährleistet. Im Falle einer Partitionierung operieren Knoten innerhalb der Teilsysteme, wobei die Verfügbarkeit hinsichtlich der jeweils nicht erreichbaren Knoten eingeschränkt ist.
- **Konsistenz** und **Verfügbarkeit** kann für Systeme angenommen werden, in denen einzelne Knoten zentralisiert Dienste bereitstellen, z.B. Datenbank- oder Webserver. Sofern keine Partitionierung auftritt, ist die Verfügbarkeit gewährleistet, da Nachrichten zugestellt und Requests stets beantwortet werden. Weiterhin ist die Konsistenz sichergestellt, da Elementaroperationen für Systeme in einzelnen Knoten serialisierbar sind. Das System ist nicht operational, sofern sich Partitionen herausbilden, da einzelne Knoten aus mindestens einem Teilsystem ohne Verzicht auf Konsistenz nicht mehr nutzbar sind.

- **Verfügbarkeit** und **Partitionstoleranz** trifft für Systeme zu, die vorübergehend oder dauerhaft geringere Konsistenzgarantien zugunsten der Verfügbarkeit in Kauf nehmen. Beispiele sind verteilte Caches, die Nameserver-Einträge oder Webseiten speichern, Content-Delivery-Netzwerke, z.T. NoSQL-Datenbanksysteme, sowie Blockchain-Systeme (siehe Kapitel 3.3.5). Daten sind auch in möglicherweise entstehenden Partitionen verfügbar, jedoch aufgrund von nicht-zustellbaren Nachrichten nicht notwendigerweise konsistent.

**BASE und ACID**

Systeme, die konsistente Zustände nach Partitionierungen im Zeitverlauf wiederherstellen sind Eventually Consistant und besitzen einen sich ändernden Zustand, d.h. einen Soft State. Systeme dieser Art bieten Verfügbarkeit und besitzen die Merkmale „Basically Available, Soft State, Eventually Consistent“ (BASE). Eventual Consistency ist in bestimmten Fällen konfliktfrei im Zeitverlauf herstellbar, z.B. durch Convergent oder Commutative Replicated Data Types (CRDT) (Shapiro et al. 2011). Ein CRDT kann zwei unter monotoner Veränderung entstandene Zustände zusammenführen bzw. kommutative Operationen ausführen, anhand derer sich partielle Ordnungen bilden lassen. In Kontrast zu BASE stehen transaktionsbasierte Systeme, welche die Merkmale Atomicity (1.), Consistency (2.), Isolation (3.) und Durability (4.) (ACID) erfüllen (Haerder und Reuter 1983; Ferstl und Sinz 2013, S. 403). In Systemen dieser Art wird je Transaktion Atomarität (1.), die Überführung des Datenbestandes von einem konsistenten Zustand in einen konsistenten Zustand (2.), die Isolation gegenüber anderen Transaktionen (3.) und Dauerhaftigkeit der Daten (4.) vorausgesetzt. Bezogen auf den CAP-Konsistenzbegriff kann zu jedem Zeitpunkt eine totale Ordnung über der Menge der Elementaroperationen unterstellt werden. Transaktionen dieser Art werden im Folgenden als ACID-Transaktionen bezeichnet. Eine weitergehende Diskussion zu BASE- und ACID-Eigenschaften von Blockchain-Systemen wird in Abschnitt 3.3.5 geführt.

#### 3.2.1.4 Ordnung von Ereignissen

Die Knoten asynchroner verteilter Systeme kommunizieren anhand von Nachrichten über potenziell unzuverlässige Kommunikationskanäle, die insbesondere in global verteilten Systemen über eine Vielzahl physischer Links eines Kommunikationsnetzwerks etabliert werden. Die Zuverlässigkeit der Übertragung von Nachrichten, bestimmte Laufzeiten, das nicht-mehrmalige Eintreffen oder eine definierte Reihenfolge können nur unter bestimmten Annahmen garantiert werden (Dwork, Lynch

et al. 1988). Unter diesen Bedingungen treffen einzelne Knoten Entscheidungen auf der Basis von Nachrichten. Eine grundlegende Anforderung zur Koordination mehrerer Knoten ist daher die Bestimmung von Ordnungsrelationen auf der Menge der Nachrichten.

**Logische Uhren**

Um diese zu bestimmen, etablieren Lamport-Uhren (Lamport 1978; Steen und Tanenbaum 2017, S. 311) einen Mechanismus zur Anpassung von logischen Uhren, die in Form von Zählern je Knoten verwaltet werden. Mit dem Empfang einer Nachricht wird der lokale Zählerwert durch den übermittelten Zählerwert des Absenders ersetzt, sofern letzterer den lokalen Wert übersteigt. Zählerwerte werden zudem je Event, d.h. vor dem Senden und nach dem Empfangen einer Nachricht, inkrementiert. Eine zwischen zwei Events bestehende Happend-Before-Relation wird damit anhand der Zählerwerte codiert, so dass je Knoten eine totale Ordnung über der Menge gesendeter und empfangener Nachrichten eines Knotens definiert werden kann. Eine Erweiterung hin zu Vektoruhren (Fidge 1988; Mattern 1989) ersetzt Zählerstände durch Vektoren, die einen Wert für jeden bekannten Knoten per Index verwalten. Dem bisherigen Mechanismus folgend, werden lokale Vektorwerte durch empfangene Werte übereinstimmender Indizes ersetzt, sofern letztere die lokalen Werte übersteigen. Ein Knoten inkrementiert den Wert des ihm zugewiesenen Index je Event. Dies erlaubt kausale Vergleiche der Vektor-Werte zweier Knoten, die nun eine Aussage über die Abfolge zugrunde liegender Events treffen.

**Blockchain-Systeme**

In Blockchain-Systemen entspricht die Erstellung eines Blocks einem Event. Jeder erstellte Block wird ausgehend von einem Knoten unmittelbar an alle erreichbaren Knoten propagiert, um je Knoten effektiv einen als „Block Height“ repräsentierten Zählerwert zu erhöhen. Die Block Height entspricht der Kardinalität der Menge derjenigen Blöcke, die unter Anwendung der Regeln des Consensus-Verfahrens gültig sind und den Systemzustand definieren (siehe z.B. Narayanan, Bonneau et al. 2016, S. 158). Happend-Before-Relationen sind definiert durch Verweise zwischen Blöcken, die als Teil des Protokolls z.B. anhand von Hash-Werten vorgesehen sind. Über die Zählerwerte anderer Knoten besteht dabei keine Kenntnis. Die parallele Erstellung von Blöcken gleicher Block Height wird im Zeitverlauf anhand des Protokolls aufgelöst. Happend-Before-Relationen definieren schließlich in der entstehenden Kette eine totale Ordnung über der Menge derjenigen Blöcke, die nach Maßgabe des Consensus-Verfahrens als gültig betrachtet werden.

#### 3.2.1.5 Fehlertoleranz

In verteilten Systemen kann nicht davon ausgegangen werden, dass einzelne Knoten stets fehlerfrei operieren und verfügbar sind. Die Fehlertoleranz eines verteilten Systems kann in Abhängigkeit von Fehlerklassen beschrieben werden, die für ein gegebenes System angenommen werden. Eine grundlegende Klassifikation unterscheidet die Fehlerklassen Absturzfehler (Crash Failure), Auslassungsfehler (Omission Failure), zeitabhängiger Fehler (Timing Failure), Antwortfehler (Reponse Failure) und beliebiger Fehler (Arbitrary Failure) (Steen und Tanenbaum 2017, S. 428).

**Byzantinische Fehlertoleranz**

Zur Untersuchung der Fehlertoleranz wird im Folgenden die Klasse beliebiger Fehler herangezogen, die in Anlehnung an das *Byzantine Generals Problem* (Lamport et al. 1982) als Byzantinischer Fehler (Byzantine Fault) bezeichnet wird. Eine begriffliche Unterscheidung der Ursächlichkeit eines Byzantinischen Fehlers (Byzantine Fault) von dessen Auftreten als Byzantinischer Fehlschlag (Byzantine Failure) wird in der Literatur nicht konsequent verfolgt (Cachin et al. 2011, S. 71).

Byzantinische Fehler sind beliebige Fehler, infolge derer keine Annahmen über die Vollständigkeit oder Korrektheit von Antwort-Nachrichten eines Knotens getroffen werden können (Cachin et al. 2011, S. 29). Byzantinische Fehlertoleranz (BFT, Byzantine Fault Tolerance) ist ein Merkmal eines verteilten Systems, das Fehlertoleranz gegenüber dem Auftreten von Byzantinischen Fehlschlägen innerhalb eines verteilten Systems beschreibt. BFT nimmt auf eine Klasse von Problemen Bezug, in denen für eine nichtleere Teilmenge $K_{BF}$ der Knotenmenge $K$ des Systems Byzantinische Fehler angenommen werden, während die Menge $K_{LG} = K \setminus K_{BF}$ fehlerfrei operiert. Die Elemente der Mengen sind nicht bekannt.

#### 3.2.1.6 Byzantine Generals Problem

Die Übertragung von Nachrichten ausgehend von einer bestimmten Quelle ist ein grundlegendes Problem bei der Gewährleistung der Fehlertoleranz gegenüber $K_{BF}$. Das Problem wurde von Pease et al. (1980) beschrieben und später als Byzantine Generals Problem (BGP) formuliert (Lamport et al. 1982). Das BGP entspricht einer 1:N-Übertragung einer Nachricht v ausgehend von einem $k \in K$ an jeden Knoten in $K$, wobei folgende Kriterien erfüllt werden müssen (Coulouris et al. 2012, S. 662; Garay und Kiayias 2018):

- Terminierung: Jeder Knoten in $K_{LG}$ erzeugt einen Ausgabewert.
- Einigung: Alle Ausgabewerte der Knoten in $K_{LG}$ sind identisch.

- Integrität: Falls $k \notin K_{BF}$, ist der von $k$ abgesendete Wert v der Ausgabewert für jeden Knoten in $K_{LG}$.

In der Terminologie des BGP bildet $K$ eine Menge von Generälen, die zur Koordination eines Angriffs ausgehend von einem Befehlshaber $k \in K$ einen Algorithmus nutzen, der durch Nachrichtenübermittlungen (a.) eine Festlegung auf einen Plan für loyale Generäle $K_{LG}$ ermöglicht und (b.) bei einer geringen Anzahl von Verrätern $|K_{BF}|$ die Koordination der $K_{LG}$ nicht gefährdet. Modifikationen bei der Übermittlung des Plans, z.B. von „attack“ zu „retreat“, gefährden die Koordination. Lamport et al. (1982, S. 384) formulieren zwei Bedingungen, die für das Senden eines Befehls durch den Befehlshaber $k$ an $n-1$ untergebene Generäle $K \setminus \{k\}$ erfüllt sein müssen:

1. Alle Generäle in $K_{LG}$ befolgen denselben Befehl.
2. Jeder untergebene General in $K_{LG}$ befolgt den Befehl, sofern $k \in K_{LG}$ loyal ist.

Eine Lösung, die unter Erfüllung von $|K| > 3 * |K_{BF}|$ weniger als $1/3$ der Generäle als Verräter zulässt, basiert auf den Annahmen der Auslieferung jeder abgesendeten Nachricht (1.), der Identifizierung des Absenders einer Nachricht durch den Empfänger (2.) und der Erkennbarkeit ausgelassener Nachrichten (3.) (Lamport et al. 1982, S. 387). Eine weitere Lösung erfordert digitale Signaturen unter den hinzukommenden Annahmen einer Erkennung von Fälschungen von Signaturen oder Nachrichten von $K_{LG}$ (4.) und der Möglichkeit der Authentifizierung der Signatur eines Generals (Lamport et al. 1982, S. 391).

#### 3.2.1.7 Übereinkunft (Consensus)

Das Erzielen einer Übereinkunft (Consensus), auch *Byzantine Agreement*, ist ein grundlegendes Problem unter der Annahme der nichtleeren Teilmenge $K_{BF}$. Eine Übereinkunft wird erzielt, indem jeder Knoten in $K$ einen individuell bestimmten Wert als Nachricht mit anderen Knoten in $K$ austauscht, wobei folgende Kriterien erfüllt werden müssen (Coulouris et al. 2012, S. 660; Garay und Kiayias 2018):

- Terminierung: Jeder Knoten in $K_{LG}$ erzeugt einen Ausgabewert.
- Einigung: Alle Ausgabewerte der Knoten in $K_{LG}$ sind identisch.
- Integrität: Falls alle Knoten in $K_{LG}$ einen identischen Wert v vorschlagen, ist der Ausgabewert für jeden Knoten in $K_{LG}$ v.

Die Bedingung der Integrität kann in Abhängigkeit der Anwendung abgeschwächt werden und erfordert nicht notwendigerweise eine Übereinstimmung eines identischen Wertes von allen Knoten (Coulouris et al. 2012, S. 661). Z.B. kann derjenige

Wert gewählt werden, der von der Mehrheit der Knoten $K$ vorgeschlagen wurde, oder etwa das Minimum oder Maximum ordinalskalierter Werte. Der nachfolgende Abschnitt diskutiert Ansätze zur Lösung und Implementierung des Problems.

Algorithmen zur Lösung des Problems sind, etwa durch Mehrheitsentscheidungen, in synchronen Systemen umsetzbar (Coulouris et al. 2012, S. 663–664). Für asynchrone Systeme werden Annahmen hinsichtlich der Zustellung von Nachrichten getroffen, die effektiv eine garantierte Zustellung in definierter Zeit vorsehen. Komplexere Algorithmen für partiell asynchrone Systeme treffen geringere Annahmen. Dwork, Lynch et al. (1988) zeigen Lösungen, die u.a. eine feste Zeitdauer für Zustellungen annehmen, die a priori unbekannt ist. Der Paxos-Algorithmus löst das Problem der Übereinkunft u.a. für beliebig lange dauernde Zustellungen von Nachrichten sowie für Absturzfehler (Lamport 1989). Das Verfahren sieht die Auswahl eines Leaders aus der Menge der Knoten vor, der mit der Ausführung von Zustandsübergängen die Fortführung des Verfahrens bestimmt. Der Ablauf des Verfahrens bestimmt sich effektiv anhand der Zustände einer verteilten State Machine. Konsistenz (Safety) und die Fortführung des Verfahrens (Liveness) sind sichergestellt, sofern für einen vorgesehenen Zustandsübergang mindestens ein Knoten als Leader ausgewählt werden kann (Lamport 1989). Das zunächst theoretisch beschriebene Verfahren beruht auf der Handlungsfähigkeit des Leaders. Practical Byzantine Fault Tolerance (PBFT) (Castro und Liskov 2002; Castro, Liskov et al. 1999) ist ein implementiertes Verfahren, das Nachrichtenzustellungen mit beliebig langen, endlichen Verzögerungen annimmt. Das Verfahren ist eine Form der State Machine Replication (SMR). SMR operiert analog zu Paxos (Lamport 1989; Schneider 1990) durch global gesteuerte Zustandsübergänge einer State Machine, ausgelöst durch wechselnde Knoten in der Rolle eines Leaders. Eine Übereinkunft über einen Wert wird im Sinne eines Commit hergestellt, der von einem Leader an die verbleibenden Knoten verteilt wird. Diese tauschen den erhaltenen Wert in zwei aufeinanderfolgenden Phasen *prepare* und *commit* untereinander aus. Der zweimalige Austausch verhindert Manipulationen durch einzelne Knoten. Das Verfahren erzielt eine Übereinkunft, sofern mehr als 2/3 der Knoten fehlerfrei operieren, d.h. für f fehlerhafte Knoten müssen 3f + 1 fehlerfrei operierende Knoten bestehen als $|K| > 3 * |K_{BF}|$.

Ohne jegliche Annahme über Garantien, z.B. von Nachrichtenlaufzeiten, sind Lösungen des Byzantine Generals Problem und des Problems der Übereinkunft in asynchronen Systemen nach dem „Fischer, Lynch and Patterson Impossibility Result“ (FLP) unmöglich (Fischer et al. 1985). Der Beweis zeigt die prinzipielle Möglichkeit zur Herbeiführung beliebig lange andauernder Verzögerungen anhand von

Netzwerk-Partitionierungen für beliebige Consensus-Verfahren (Birman 2012, S. 312, 361).

**Consensus-Verfahren von Blockchain-Systemen im Kontext verteilter Systeme**

Die Problemstellung der Übereinkunft ist für dezentrale Blockchain-Systeme relevant (siehe 3.1.2). Das Consensus-Verfahren ist als Teil des Protokolls eines Blockchain-Systems für die Koordination der Knoten und das Erzielen einer Übereinkunft hinsichtlich des Datenbestandes der Blockchain verantwortlich (Abschnitt 3.3.3). Wenige Blockchain-Systeme greifen direkt auf etablierte Consensus-Verfahren wie PBFT oder Paxos zurück (Burchert und Wattenhofer 2018). Gründe hierfür sind die Skalierbarkeit über mehr als einige zehn Knoten hinaus und Aspekte der praktischen Implementierbarkeit der Algorithmen in dezentralen Systemen, z.B. aufgrund der Annahme einer bekannten Menge von adressierbaren Knoten und fehlende Toleranz gegenüber Absturzfehlern (Garay und Kiayias 2018; Vukoli 2016). Gleichzeitig wird die Erweiterung etablierter Consensus-Verfahren für Blockchain-Systeme aktiv untersucht, v.a. für private Blockchain-Systeme.

Prinzipiell bestehen für private Blockchain-Systeme gegenüber öffentlichen Systemen geringere Anforderungen an Fehlertoleranz (siehe 3.2.1.7), da ein System dieser Art von einem ausgewählten Kreis bekannter Teilnehmer und mit einer bekannten Menge von Knoten genutzt wird. Beispiele für Verfahren werden in Abschnitt 3.3.4 diskutiert.

Öffentliche Blockchain-Systeme können aufgrund des unbeschränkten Zugriffs durch beliebige Teilnehmer nicht die Annahmen eines synchronen Systems treffen. Die Knoten des Systems gelten prinzipiell nicht als vertrauenswürdig. Das System muss Byzantine Fault Tolerance mindestens für die angesprochenen Annahmen partiell asynchroner Systeme unterstützen, um mit verzögerten, inkorrekten oder ausbleibenden Nachrichten umgehen zu können. Die Entwicklung hierfür geeigneter Verfahren beginnt für Blockchain-Systeme mit dem Nakamoto Consensus, der auf das BGP Bezug nimmt.

*Nakamoto Consensus und BGP*

Die Innovation des heute als Nakamoto Consensus bezeichneten Verfahrens besteht darin, eine Übereinkunft zwischen einander nicht vertrauenden Knoten unter der Aufwendung von Ressourcen herzustellen, ohne Annahmen hinsichtlich des Verhaltens der teilnehmenden Knoten zu unterstellen (Nakamoto 2008a; Narayanan und Clark 2017).

Nakamoto Consensus ist ein Proof-of-Work-Verfahren, das in Abschnitt 3.3.4.2 im Kontext ressourcenabhängiger Verfahren diskutiert wird. Nakamoto (2008b) beschreibt das Verfahren als Variante des BGP:

*„A number of Byzantine Generals each have a computer and want to attack [...]. They only have enough CPU power [...] if a majority of them attack at the same time. [...]. The problem is that the network is not instantaneous, and if two generals announce different attack times [...], some may hear one first and others hear the other first."*

*„They use a proof-of-work chain to solve the problem. Once each general receives whatever attack time he hears first, he sets his computer to solve an extremely difficult proof-of-work problem that includes the attack time in its hash. The proof-of-work is so difficult, it's expected to take 10 minutes of them all working at once before one of them finds a solution. Once one of the generals finds a proof-of-work, he broadcasts it to the network, and everyone changes their current proof-of-work computation to include that proof-of-work in the hash they're working on. If anyone was working on a different attack time, they switch to this one, because its proof-of-work chain is now longer."*

*„[...]. Every general, just by verifying the difficulty of the proof-of-work chain, can estimate how much parallel CPU power per hour was expended on it and see that it must have required the majority of the computers to produce that much proof-of-work in the allotted time. They had to all have seen it because the proof-of-work is proof that they worked on it. If the CPU power exhibited by the proof-of-work chain is sufficient [...], they can safely attack at the agreed time."*

Das Verfahren wird als Lösung einer Variante des BGP beschrieben. Im Gegensatz zu BGP geht die Variante nicht von einer 1:N-Übertragung einer Nachricht aus, sondern sieht eine Übertragung von Werten ausgehend von mehreren Knoten an mehrere Knoten vor (Garay, Kiayias und Leonardos 2015). Das Verfahren beschreibt damit einen Lösungsvorschlag des Consensus-Problems anhand von Proof-of-Work. Ein Erzielen einer Übereinkunft geschieht probabilistisch und sichert keine Garantien hinsichtlich der Konsistenz zu (siehe Abschnitt 3.3.5). Untersuchungen über die durch das Verfahren getroffenen Annahmen sind Gegenstand aktueller Diskussionen (Bano et al. 2017; Garay und Kiayias 2018; Narayanan und Clark 2017; Pass et al. 2017; Wattenhofer 2016). Diskussionen zur Konsistenz und zur Skalierung der Verfahren sind Gegenstand der Abschnitte 3.3.5 bzw. 3.3.6.

## 3.2.2 Kryptografie

### 3.2.2.1 Einführung

Blockchain-Systeme greifen neben den diskutierten Grundlagen aus dem Bereich der verteilten Systeme auf Verfahren der Kryptografie zurück. Wesentlich für die Architektur von Blockchain-Systemen sind kryptografische Hash-Verfahren sowie asymmetrische Signaturverfahren. Letztere stellen gleichzeitig Verschlüsselungsverfahren dar, die in dieser Funktion allerdings kein notwendiger Bestandteil der Architektur sind.

Im Allgemeinen verfolgen kryptografische Verfahren die Realisierung von Zielen der IT-Sicherheit (Eckert 2018, S. 7). Das übergeordnete Ziel ist dabei die Beschränkung und die Kontrolle des Zugriffs auf Informationen.

Spezifische Ziele sind:

1. **Datenintegrität** unter Verhinderung unautorisierter und unbemerkter Modifikationen von Daten,
2. **Informationsvertraulichkeit** derart, dass keine unautorisierte Informationsgewinnung erfolgt,
3. **Authentizität** im Sinne der Echtheit und Glaubwürdigkeit von beteiligten Subjekten und Objekten,
4. **Verfügbarkeit** eines Anwendungssystems für authentifizierte und autorisierte Subjekte, ohne Beeinträchtigung des Zugriffs oder der Ressourcen des Systems,
5. **Verbindlichkeit** der Zuordenbarkeit von Aktionen zu Subjekten derart, dass von Subjekten durchgeführte Aktionen nicht abstreitbar sind.

Zu Beginn der beiden folgenden Abschnitte wird jeweils kurz auf entsprechende Ziele Bezug genommen.

### 3.2.2.2 Kryptografische Hash-Verfahren

Verfahren zur Erstellung von Prüfsummen und Hash-Werten sind die wesentliche technische Basis zur Sicherung der Datenintegrität (Ziel 1.), u.a. in Blockchain-Systemen. Hash-Verfahren definieren sich algorithmisch anhand einer Hash-Funktion. Eine Hash-Funktion $H(m) = v$ erlaubt eine Zusammenfassung einer Nachricht $m$ beliebiger Länge durch einen Hash-Wert $v$ konstanter Länge, z.B. 256 bit im Falle der Hash-Funktion SHA-256 (NIST 2015). $H(m)$ ist eine nicht-injektive

Funktion, da die Länge von m die Länge von $v$ übersteigen kann. Die Zusammenfassung einer Nachricht in $v$ wird z.T. als Message Digest bezeichnet (Ferguson et al. 2010, S. 77).

Vorrangige Ziele sind (1.) die Erkennung von Speicher- und Übertragungsfehlern sowie (2.) die Erkennung beliebiger Modifikationen (Kurose und Ross 2013, S. 438, 689).

**Verfahren zur Erkennung von Speicher- und Übertragungsfehlern**

Die Verfahrensklasse umfasst neben Hash-Funktionen beliebige Verfahren zur Bildung von Prüfsummen, anhand derer das Prinzip der Integritätssicherung illustriert werden kann. Hierzu gehören Verfahren wie Cyclic Redundancy Check (CRC) oder Internet Checksum für die Protokolle UDP, TCP und IP (Kurose und Ross 2013, S. 438 ff.). Die Berechnung der Internet Checksum bildet Byte-Paare, interpretiert diese als 16-bit Integer-Werte und summiert diese. Das Einerkomplement wird als Prüfsumme während einer Übertragung im Header der genannten Protokolle hinterlegt. Nach der Übertragung wird Integrität angenommen, sofern alle Bits des Einerkomplements der Summe der empfangenen Daten, einschließlich der Checksumme, den Wert 1 besitzen (Kurose und Ross 2013, S. 442). In dieser Verfahrensklasse ist die Sicherung der Integrität im Falle von gezielten Manipulationen nicht gewährleistet. Die Integrität ist für das Beispiel verletzt, sofern Veränderungen in Byte-Paaren vorgenommen werden, die nicht zu abweichenden Einerkomplement-Summen führen.

**Verfahren zur Erkennung beliebiger Modifikationen**

Die in dieser Verfahrensklasse eingesetzten Hash-Funktionen sind resistent gegenüber zwei Arten von Manipulationen, die das Urbild (Preimage) einer Funktion bzw. eine Kollision (Collision) betreffen. Eine Kollision liegt vor, sofern Paare verschiedener Nachrichten $m, m'$ bekannt sind, die auf ein $v$ abgebildet werden, so dass $H(m) = H(m') = v$ gilt. Hash-Funktionen, die hinsichtlich dieser Manipulationsarten die nachfolgenden Merkmale erfüllen, werden als kryptografische Hashfunktionen bezeichnet (Rogaway und Shrimpton 2004):

1. Urbild: H(m) ist eine Einwegfunktion, die m effizient berechnet, jedoch nicht effizient umkehrbar ist, d.h. für ein gegebenes v ist die Bestimmung eines m mit $H(m) = v$ nicht effizient berechenbar (Preimage Resistance). Zudem ist es nicht effizient, ausgehend von einer Nachricht m und deren Hash-Wert $H(m)$ eine weitere Nachricht $m'$ zu finden, für die $m \neq m'$ und $H(m) = H(m')$ gilt (Second-Preimage Resistance).

2. Kollision: Da $H(m)$ eine nicht-injektive Funktion ist, existieren Paare von Nachrichten $m, m'$, die auf ein $v$ abgebildet werden und somit eine Kollision bedingen. Es ist jedoch nicht effizient, ein beliebiges Paar von Nachrichten $m, m'$ zu finden, für das $m \neq m'$ und $H(m) = H(m')$ gilt (Collision Resistance).

Beispiele für kryptografische Hash-Funktionen sind die von NIST standardisierten Algorithmen SHA-256, SHA-512/256, SHA-384 und SHA-512 sowie deren Nachfolger SHA3-256, SHA3-384, SHA3-512 (NIST 2015; BSI 2018, S. 40). Collision Resistance kann für die Hash-Funktionen MD5 (Xie et al. 2013) und SHA-1 (Stevens et al. 2017) nicht mehr ohne Weiteres unterstellt werden.

**Anwendung**

Die diskutierten Eigenschaften bedingen, dass gegebene Daten, die als $m$ in eine kryptografische Hash-Funktion $H(m) = v$ eingehen, zu einem nicht vorhersagbaren Funktionswert v führen. Eine Änderung von $m$ zu $m'$ ($m \neq m'$) mit Anwendung von $H(m') = v'$ führt zu einer pseudo-zufälligen Änderung von $v$ zu $v'$, d.h. zu einer durch H determinierten und nicht vorhersagbaren Änderung.

Die Überprüfung der Integrität von Daten $m$ im Zeitverlauf kann auf Grundlage einer Hash-Funktion in zwei Schritten erfolgen. Hierfür wird eine Hash-Funktion $H$ ausgewählt, $H(m) = v$ berechnet und $v$ gespeichert (1.). Integrität wird angenommen, sofern eine Anwendung der zu einem späteren Zeitpunkt vorliegenden Daten $m'$ auf $H$ den Funktionswert $v$ ergibt, so dass $H(m) = H(m') = v$ gilt (2.). In diesem Fall wird $m = m'$ angenommen. Andernfalls wird $m \neq m'$ angenommen.

#### 3.2.2.3 Asymmetrische Signatur- und Verschlüsselungsverfahren

Blockchain-Systeme setzen Signaturverfahren ein, um die Authentizität von Teilnehmern und den von ihnen abgesendeten Transaktionen zu gewährleisten. Ein Signaturverfahren wird von einem Teilnehmer herangezogen, um digitale Signaturen für eine Nachricht zu erstellen oder zu überprüfen. Signaturverfahren beziehen sich hiermit auf die Ziele Authentizität (3.) und Verbindlichkeit (5.) der IT-Sicherheit (siehe Abschnitt 3.2.2.1).

Das Ziel der Informationsvertraulichkeit (2.) wird durch den Einsatz von Verschlüsselungsverfahren verfolgt. Verschlüsselungsverfahren führen Transformationen zwischen unverschlüsselten Nachrichten und verschlüsselten Ciphertexten unter Verwendung von Schlüsseln durch. Die Übertragung einer Nachricht über einen

potenziell unsicheren Kommunikationskanal erfolgt als Ciphertext, nach Anwendung einer Verschlüsselungsfunktion unter Verwendung eines Schlüssels. Transformationsfunktionen zur Verschlüsselung und Entschlüsselung bilden ein Verschlüsselungsverfahren.

Verschlüsselungsverfahren verwenden symmetrische oder asymmetrische Verfahren der Kryptografie. Signaturverfahren basieren auf asymmetrischen Verfahren (Eckert 2018, S. 331 f.).

**Symmetrische Verfahren**

Symmetrische kryptografische Verfahren setzen vor der Übertragung eines Ciphertexts $C$ einer Nachricht M eine Vereinbarung über einen geheimen Schlüssel $K_{A,B}$ zwischen Absender $A$ und Empfänger $B$ voraus. Für eine Übertragung von $M$ wird eine Funktion E zur Verschlüsselung als $C = E(M, K_{A,B})$ vor einer Übertragung von A angewendet und eine Funktion $D$ zur Entschlüsselung als $M = D(C, K_{A,B})$ nach einer Übertragung von $B$ angewendet.

**Asymmetrische Verfahren**

Asymmetrische kryptografische Verfahren oder Public-Key-Kryptografie erlauben eine Verschlüsselung ohne vorherigen Austausch eines Pre-Shared Secret in Form eines geheimen Schlüssels (Diffie und Hellman 1976). Die mathematische Basis sind ineffizient berechenbare Probleme, wie der diskrete Logarithmus oder das Faktorisierungsproblem (Beutelspacher et al. 2010, S. 132). Verfahren wie RSA kombinieren Verschlüsselungs- und Signaturverfahren (Rivest et al. 1978). Die Verfahren sehen öffentliche Schlüssel vor, die über unsichere Kommunikationskanäle verteilbar sind, sowie private Schlüssel, die lokal bei einzelnen Teilnehmern vorliegen. In den Blockchain-Systemen Bitcoin und Ethereum kommt ein auf elliptischen Kurven basierender digitaler Signaturalgorithmus (ECDSA) (Johnson et al. 2001) zum Einsatz, welcher der Elliptic Curve Cryptography (ECC) zuzuordnen (Koblitz 1987; V. S. Miller 1986) ist.

**Generierung von Schlüsseln**

Die Teilnehmer des Systems generieren jeweils ein Schlüsselpaar ($K_P^T$, $K_S^T$) für einen Teilnehmer T mit separaten Schlüsseln $K_P^T$, zur Verschlüsselung oder Signaturprüfung, sowie $K_S^T$, zur Entschlüsselung oder Signaturerstellung. $K_P^T$ wird als Public Key oder öffentlicher Schlüssel bezeichnet und anderen Teilnehmern zugänglich gemacht, während $K_S^T$ als Secret Key, Private Key oder privater Schlüssel geheim bleibt.

Die Generierung eines Schlüsselpaares ($K_P^T$, $K_S^T$) erfolgt lokal bei T, da Informationsvertraulichkeit hinsichtlich des privaten Schlüssels $K_S^T$ an anderer Stelle nicht angenommen werden kann. Bei der lokalen Generierung wird eine Erzeugung eines systemweit eindeutigen Schlüsselpaares angestrebt, indem zunächst $K_S^T$ durch eine zufällige Auswahl aus einem ausreichend großen Schlüsselraum bestimmt wird. Im Falle von $K_S^T$ der Länge 256 bit ergibt sich ein Schlüsselraum der Größenordnung $\lfloor \log_{10} 2^{256} \rfloor = 77$. Die Blockchain-Systeme Bitcoin und Ethereum verwenden $K_S^T$ dieser Länge und greifen auf ECDSA-Verfahren unter Nutzung der standardisierten elliptische Kurve secp256k1 zurück (Bos et al. 2013; Wood 2014). Die Generierung eindeutiger Schlüssel wird bei Absenz von Implementierungsfehlern angenommen (Bos et al. 2013). Genauer betrachtet, wird der Schlüsselraum durch die Ordnung n der Kurve secp256k1 (Certicom Research 2010, S. 9) begrenzt.

$$n = \text{FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141}_{16}$$

Die Größenordnung bleibt hierdurch unverändert bei $\lfloor \log_{10} n \rfloor = 77$.

Zur Generierung des Schlüssels $K_S^T$ (Eckert 2018, S. 353; Bos et al. 2013; Wood 2014) wird eine kryptografisch sichere RNG-Funktion unterstellt, die zufällig oder pseudozufällig Werte unterhalb von n annähernd gleichverteilt auswählt:

$$K_S^T = RNG(n) \text{ mit } K_S^T \in \{z \in Z : 1 \leq z < n\}$$

Zur Berechnung des Schlüssels $K_P^T$ aus $K_S^T$ (Eckert 2018, S. 350-355) wird die auf elliptischen Kurven definierte Punkt-Multiplikation ( $\cdot$ ) aus $K_S^T$ und einem festgelegten Basispunkt G der Kurve berechnet, so dass sich $K_P^T$ als Punkt der Kurve ergibt:

$$K_P^T = K_S^T \cdot G$$

Die elliptische Kurve secp256k1 der Form $y^2 = x^3 + ax + b$ mit den Parametern $a = 0, b = 7$ ist über dem endlichen Körper $\mathbb{F}_p$ der Primzahl $p$ mit einem Basispunkt $G$ definiert (Certicom Research 2010, S. 9; Antonopoulos 2017, S. 61), so dass jeder $K_P^T$ als Punkt $(x, y)$ der Kurve die Kurvengleichung erfüllt:

$$y^2 \bmod p = (x^3 + 7) \bmod p$$

In G sind zweidimensionale Koordinaten $(x, y)$ des Basispunktes serialisiert (Antonopoulos 2018).

Der mit der Kurve definierte Basispunkt sowie die Primzahl (Certicom Research 2010, S. 9) lauten hier:

$G = 0279\text{BE}667\text{EF}9\text{DCBBAC}55\text{A}06295\text{CE}870\text{B}07029\text{BFCDB}2\text{DCE}28\text{D}959\text{F}2815\text{B}16\text{F}81798_{16}$

$$p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$$

$K_P^T$ ist anhand der Punkt-Multiplikation für secp256k1 effizient berechenbar, aufgrund des Diskreten-Logarithmus-Problems jedoch nicht effizient für die Berechnung von $K_S^T$ aus $K_P^T$ umkehrbar (Eckert 2018, S. 351 f.).

**Anwendung**

$K_P^T$ kann im Sinne einer öffentlichen Kennung von T veröffentlicht werden, so dass T von anderen Teilnehmern global identifizierbar ist. Jeder veröffentlichte $K_P^T$ ist durch andere Teilnehmer (1.) als Kennung von T, (2.) zur Verschlüsselung von Nachrichten an T sowie (3.) zur Überprüfung von Signaturen von T (3.) anwendbar. $K_S^T$ ist durch den Teilnehmer T in Fall (2.) zur Entschlüsselung von Nachrichten an T sowie in Fall (3.) zur Erstellung von Signaturen anwendbar.

*1. Identitäten*

Identitäten werden beispielhaft für die Blockchain-Systeme Bitcoin und Ethereum berechnet. Die Berechnung beider Identitäten basiert auf einem einzigen, zufällig erzeugten, privaten Schlüssel $K_S^T$:

$K_S^T = 3277\text{A}3\text{BB}4872052025\text{B}94\text{C}149659105\text{EEECBF}8\text{B}3177\text{E}4137\text{A}1\text{F}7\text{CFD}17\text{F}1\text{AA}9\text{FB}_{16}$

Nach Anwendung der Punkt-Multiplikation ergeben sich für $K_P^T$

$x = \text{FCD}570\text{FB}7\text{C}172\text{D}087\text{D}1\text{B}8\text{D}957\text{C}17\text{F}3670712101\text{EC}8\text{BB}07216106\text{A}74\text{B}2\text{F}70\text{D}885_{16}$
$y = \text{E}5\text{BB}47\text{DBA}8\text{A}1957\text{D}6717577\text{A}185\text{EB}17\text{E}8\text{CAD}3\text{BE}018\text{C}063321\text{E}27959\text{D}8\text{C}8132\text{D}0_{16}$

$K_P^T = x \,||\, y$

In Bitcoin und Ethereum werden u.a. Hash-Funktionen auf $K_P^T$ angewendet, um öffentlich adressierbare Identitäten oder Adressen zu erzeugen. Ausgehend von dem angegebenen $K_P^T$ ergeben sich für die beiden Systeme die folgenden Adressen.

$Adresse^T_{Bitcoin}$ = 1M4SBBDZNP6KFNSFCH6ZM7KOM7RPRNUDAI
$Adresse^T_{Ethereum}$ = 0xBC89A40388A368BBE3C170CADC1C7D9441FF0EC6

Die Berechnung für Bitcoin ergibt sich anhand der folgenden Definition, für die Varianten in Abhängigkeit des gewählten Adressformats existieren (Antonopoulos 2017, S. 65–80). Zur Berechnung in Ethereum siehe (Wood 2014).

$$Adresse^T_{Bitcoin} = \text{BASE58Check}(\text{VERSION}||H^T||\text{CHECKSUM}(\text{VERSION}||H^T))$$

$$H^T = \text{RIPEMD-160}(\text{SHA-256}(\text{PREFIX}||K^T_P))$$

Mit den Funktionen (Antonopoulos 2017, S. 68):

- PREFIX ist eine Kennung zur Unterscheidung zwischen einem unkomprimierten Format bei Wert $04_{16}$ sowie verschiedenen komprimierten Formaten.
- BASE58Check(x) ist eine Funktion, die x in einer modifizierten BASE58-Codierung encodiert (Antonopoulos 2017, 66 f.).
- VERSION ist eine Versionskennung als Byte mit dem Wert $00_{16}$ (nach BASE58Check-Codierung 1) oder $05_{16}$ (nach BASE58Check-Codierung 3). Die Berechnung verwendet VERSION = $00_{16}$.
- CHECKSUM(x) = TRUNCATE(SHA-256(SHA-256(x)), 32 bit) ist eine Funktion, welche die ersten 32 bit der zweifach für x angewendeten Hash-Funktion SHA-256 berechnet.
- Die Hash-Funktion RIPEMD-160 limitiert den Raum möglicher Adressen auf 160 bit.

*2. Verschlüsselung*

In Systemen wie Bitcoin und Ethereum kommt neben ECDSA kein Verschlüsselungsverfahren zum Einsatz (Nakamoto 2008a; Wood 2014). Prinzipiell eigenen sich asymmetrische Verfahren für die Verschlüsselung, z.B. anhand von RSA (Eckert 2018, S. 289, 385). Zur verschlüsselten Übertragung einer Nachricht zwischen den Teilnehmern $A$, $B$, von $A$ nach $B$, wird die Verschlüsselungsfunktion $C = E(M, K^B_P)$ vor der Übertragung durch $A$ angewendet. Nach der Übertragung über einen potenziell unsicheren Kommunikationskanal wird von $B$ die Entschlüsselungsfunktion $D$ eingesetzt als $M = D(C, K^B_S)$. Neben diesem Verfahren sind ECC-basierte

Verschlüsselungsverfahren verbreitet, die häufig mit symmetrischen Verfahren wie AES kombiniert werden (Bos et al. 2013).

*3. Digitale Signaturen*

Die Erstellung und Überprüfung von Signaturen anhand von ECDSA ist Teil von Systemen wie Bitcoin und Ethereum (Bitcoin Core 2017; Wood 2014). Zur Übertragung einer signierten Nachricht $M$ zwischen den Teilnehmern $A$, $B$, von $A$ nach $B$, erzeugt $A$ eine Signatur $S$ anhand der Funktion $S = C(H(M), K_S^A)$. Die Funktion umfasst die Anwendung einer Hash-Funktion $H$ auf die Nachricht. An den Empfänger $B$ werden $M$ und $S$ übermittelt. Zur Überprüfung der Signatur prüft $B$, ob anhand von $S$ ein Hash-Wert erzeugt werden kann, der mit dem Hash-Wert der Nachricht übereinstimmt. Hierfür wird (1.) anhand der Funktion $H1 = V(S, K_P^A)$ ein Hash-Wert produziert, (2.) ein Hash-Wert aus $M$ errechnet als $H2 = H(M)$, (3.) die Übereinstimmung von $H1$ und $H2$ geprüft. Stimmen die Werte als $H1 = H2$ überein, wird eine gültige Signatur angenommen. Für $H1 \neq H2$ wird keine Gültigkeit der Signatur angenommen.

## 3.3 Architektur von Blockchain-Systemen

Das vorliegende Kapitel generalisiert den State of the Art der Architektur implementierter Blockchain-Systeme. Die hierfür entwickelte Architektur betrachtet Blockchain-Systeme als Zustandsraumsysteme, systematisiert deren Komponenten in einem Metamodell und diskutiert Beispiele konkreter Systeme und Verfahren. Das Kapitel basiert auf Literatur, etablierten Blockchain-Systemen und deren Quellcode (Antonopoulos 2017, 2018; Bitcoin Core 2019; Buterin et al. 2014; Ethereum 2019b; Nakamoto 2008a; Narayanan, Bonneau et al. 2016; Wood 2014; Xu, Weber und Staples 2019; Xu, Weber, Staples et al. 2017).

### Architektur von Blockchain-Systemen

Die Architektur eines Blockchain-Systems umfasst die Komponenten Netzwerk, Datenstruktur und Protokoll mit ihren Beziehungen. Die Komponenten bilden die in Abbildung 3.2 dargestellte Schichtenarchitektur, bestehend aus den Teilsystemen:

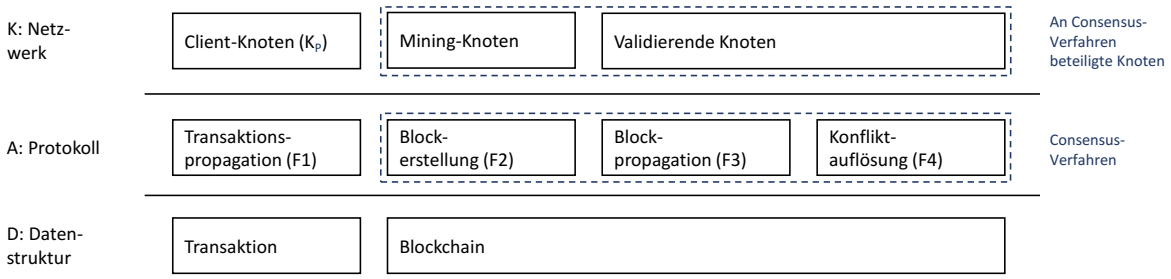



ABBILDUNG 3.2: Architektur von Blockchain-Systemen

- **Netzwerk** zur Kommunikation zwischen den abgebildeten Typen von Knoten und zur Ausführung der Funktionen des Protokolls.
- **Protokoll** zur Realisierung der Anwendungsfunktionen Transaktionspropagation (F1), Blockerstellung (F2), Blockpropagation (F3) und Konfliktauflösung (F4), die ein Consensus-Verfahren implementieren.
- **Datenstruktur** zur Datenverwaltung der Transaktionen und der Blockchain.

Die Schichten entsprechen einer funktionalen Systemabgrenzung (Ferstl und Sinz 2013, S. 321).

**Zustandsrepräsentation in Blockchain-Systemen**

Ein Modell zur Repräsentation von Zuständen eines verteilten Informationssystems ist ein Zustandsüberführungsgraph eines Zustandsraumsystems (Sinz 2014). Als Modell eines Blockchain-Systems repräsentiert ein Knoten hier genau einen Systemzustand, der sich durch den Block sowie ggf. dessen Vorgänger definiert. Ein Zustandsübergang wird durch das Protokoll anhand der Blockerstellung ausgelöst, die in einigen Systemen als Mining bezeichnet wird (Narayanan, Bonneau et al. 2016, 104 ff.). Transaktionen werden als signierte Nachrichten durch die Transaktionspropagation des Protokolls innerhalb des Netzwerks verteilt, im Rahmen der Blockerstellung als Teil eines Blocks persistent gespeichert und durch ein Verfahren zur Blockpropagation veröffentlicht (Narayanan, Bonneau et al. 2016, 51 ff.). Ein Block bildet damit einen Zustand des Systems ab. Die nebenläufige Ausführung des Protokolls in einzelnen Knoten erfordert eine Konfliktauflösung als Bestandteil des Consensus-Verfahrens, das eine Übereinkunft über eine konsistente Datenbasis erzielt. Die Menge derjenigen Blöcke, die nach Anwendung der Regeln des Consensus-Verfahrens einen konsistenten Systemzustand definieren, wird im Folgenden als Hauptkette (Main Chain) bezeichnet (Gervais et al. 2016). Die Kardinalität dieser Menge entspricht als *Block Height* der Anzahl gültiger Blöcke eines Systemzustandes.

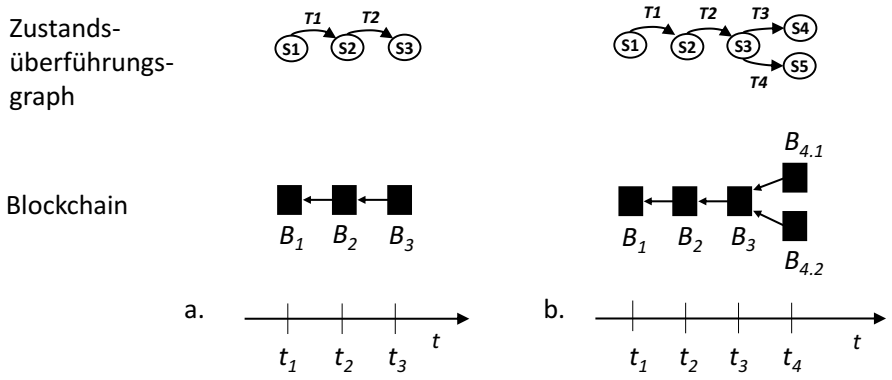



ABBILDUNG 3.3: Zustandsrepräsentation in Blockchain-Systemen

Abbildung 3.3 zeigt ein Beispiel der sich im Zeitverlauf ändernden Zustände eines Blockchain-Systems bis zu einem Zeitpunkt $t_3$ (a.) sowie $t_4$ (b.). Ausgehend von S1 wird T1 durch das Anfügen an $B_h$, $h = 1$ ausgelöst, das den Zustand in S2 und schließlich in S3 überführt. $B_h$ bezeichnet den h-ten Block der Sequenz als Block

Height. Die Datenbasis des Systems ist zu diesem Zeitpunkt konsistent, da serialisierte Operationsfolgen und Daten in einzelnen Blöcken vorliegen (siehe Abschnitt 3.3.5). Zum Zeitpunkt $t_4$ (b.) liegen zwei Blöcke vor, die innerhalb eines Zeitintervalls parallel von unterschiedlichen Knoten des Netzwerks angefügt wurden. Die Datenbasis ist zu diesem Zeitpunkt nicht konsistent, da jeder Knoten des Netzwerks ausgehend von S3 den Zustandsübergang T3 oder T4 wählen kann. Die zur Konfliktauflösung notwendige Veränderung der annehmbaren Zustände im Zeitverlauf charakterisiert einen Soft State. Im Zeitverlauf kann für den durch einen gegebenen Block definierten Zustand Konsistenz angenommen werden (siehe Abschnitt 3.3.5.2). Mit dem Anfügen eines weiteren Blocks kann die Auswahl einer der Teilketten erfolgen; die hierfür verwendete Regel zur Konfliktauflösung des Consensus-Verfahrens wählt beispielsweise die längste Kette aus, oder diejenige, in deren Erstellung die meisten Ressourcen eingegangen sind (siehe Proof-of-Work, 3.3.4.2).

## 3.3.1 Datenstruktur

Die Datenstruktur eines Blockchain-Systems ist eine verteilte Transaktionshistorie, oder Distributed Ledger. Auch ohne Betrachtung der Verteilung, d.h. ohne Berücksichtigung der Systemkomponenten Netzwerk und Protokoll, können eine Reihe der Merkmale eines Blockchain-Systems bereits erfüllt werden.

- Eine partielle oder totale *Ordnung* (1. Merkmal) über der Menge der Blöcke wird anhand der Verknüpfung eines jeden Blocks $B_h$ ($h > 0$) mit genau einem Vorgänger hergestellt.
- Die *Integrität* (2. Merkmal) der in Transaktionen hinterlegten Daten sowie der Verknüpfungen anhand von Blöcken ist überprüfbar.
- Die *Verbindlichkeit* (3. Merkmal) der Zuordnung der Identitäten von Teilnehmern zu Transaktionen ist anhand von Signaturen überprüfbar.

### 3.3.1.1 Konzeptuelles Metamodell

Das in Abbildung 3.4 dargestellte Metamodell beschreibt die Konzepte der Datenstruktur Blockchain. Das Modell bezieht sich auf dezentrale Blockchain-Systeme und unterscheidet die Systemtypen Blockchain-System und Smart-Contract-System. Die dargestellten Komponenten werden von State-of-the-Art Blockchain-Systemen wie Bitcoin und Smart-Contract-Systemen wie Ethereum instanziiert[1].

[1]Das Modell ist eine erweiterte und verallgemeinerte Fassung des Conceptual Blockchain Data Model (Härer und Fill 2019a).

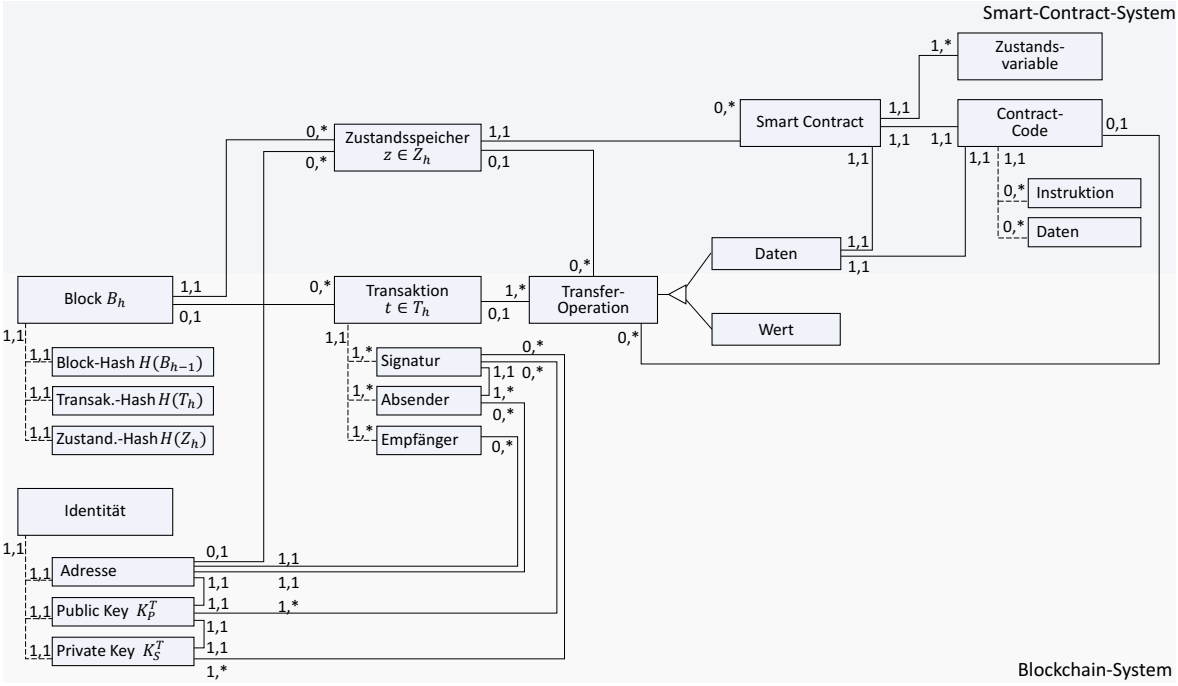



ABBILDUNG 3.4: Blockchain-Metamodell

*Block*

Ein Block ist stets mit genau einem vorhergehenden Block verknüpft, indem ein enthaltener Hash-Wert den Inhalt des Vorgänger-Blocks durch eine Abbildung auf diesen Wert zusammenfasst. Die Datenstruktur des Blocks wird nachfolgend separat erläutert.

*Adresse*

Die Teilnehmer des Systems besitzen Identitäten, die Adressen sowie private und öffentliche Schlüssel umfassen. Eine *Adresse* bestimmt sich aus einem Schlüsselpaar einer Identität, das aus einem öffentlichen Schlüssel $K_P^T$ und einem privaten Schlüssel $K_S^T$ besteht (Narayanan, Bonneau et al. 2016) (siehe z.B. $Adresse_{Bitcoin}^T$ in Abschnitt 3.2.2.3.). Eine Identität tritt in Erscheinung (1.) als *Absender* oder *Empfänger* einer *Transaktion*, (2.) für das Erstellen der *Signatur* einer zu versendenden Transaktion anhand von $K_S^T$, ggf. zusammen mit weiteren Absendern, und (3.) für die Überprüfung der Signatur anhand von $K_P^T$ durch beliebige Teilnehmer des Systems.

*Transaktion*

Eine *Transaktion* ist eine signierte und persistent zu speichernde Nachricht, die einen Transfer von Werten oder Daten durch die Ausführung von Operationen auf der Datenbasis bewirkt (Antonopoulos 2017; Bartoletti und Pompianu 2017). Sie

kann nach einer Übertragung gemäß der Nachrichtendefinition des Protokolls in einem neuen *Block* persistiert werden. Eine *Signatur* weist die Verbindlichkeit der Zuordnung zwischen Absender und Transaktion gegenüber beliebigen Systemteilnehmern nach. Der Nachweis der Verbindlichkeit besteht in der Signaturprüfung anhand von $K_P^T$.

Die Ausführung einer *Transfer-Operation* transferiert einen *Wert*, z.B. als Anzahl an Tokens, oder beliebige *Daten* zwischen 1 bis n *Absendern* und *Empfängern*. Ein Wert geht als Input in eine Transaktion ein und ist eine Summe von Werten aus Outputs von vorhergehenden Transaktionen (Narayanan, Bonneau et al. 2016). Sind *Daten* in Smart-Contract-Systemen Gegenstand einer *Transaktion*, können zwei Fälle unterschieden werden (Wood 2014).

1. Es handelt sich um eine Speicherung und Erstellung eines *Smart Contracts* anhand von *Contract-Code* als eine Folge von *Instruktion* und *Daten*.
2. Die Transaktion löst den Aufruf eines vorher erstellten *Smart Contracts* aus, der die Ausführung von *Contract-Code* nach sich zieht, z.B. durch Angabe einer Funktionssignatur zusammen mit Argumenten.

Die Ausführung in den Knoten des Systems greift auf eine virtualisierte Ausführungsumgebung zurück, z.B. die als Stack-Maschine ausgelegte Ethereum Virtual Machine (EVM). Parameter der Ausführung werden durch die Daten der Transfer-Operation spezifiziert. Instruktionen stellen Rechen- und Kontrollflussbefehle der Ausführungsumgebung dar, die auf den gegebenen Daten operieren. Damit werden Smart Contracts als Programme eines annähernd Turing-mächtigen Programmiersystems definiert (Bartoletti und Pompianu 2017). Instruktionen werden in einigen Systemen als „Opcodes“ bezeichnet (Antonopoulos 2017, S. 316).

*Zustandsspeicher*

Ein Block eines Smart-Contract-Systems enthält neben Transaktionen eine Zustandsrepräsentation des Systems, die in einen oder mehrere *Zustandsspeicher* untergliedert ist. Eine Erstellung eines Smart Contracts führt zur Bildung (Wood 2014) eines neuen Speichers mit einer neuen Adresse, unter der der Smart Contract aufrufbar ist. Ein somit persistierter Smart Contract ist anhand dieser Adresse wiederholt aufrufbar. Weiterhin können *Transfer-Operationen* durch *Contract-Code* ohne Transaktionen ausgelöst werden, wodurch Autonomie hinsichtlich der Auslösung von Smart Contracts vorliegt. Neben *Contract-Code* können *Zustandsvariablen* als Teil eines Smart Contracts innerhalb des Speichers hinterlegt werden, um Variablenwerte während der Ausführung zu lesen und zu schreiben. Ein *Zustandsspeicher* kann

neben Smart Contracts zudem für *Transfer-Operation* von *Werten* herangezogen werden.

Unabhängig von der Buchführung anhand von Transaktionen repräsentiert ein Speicher in diesem Fall den aggregierten Wert einer Adresse. Einige Systeme unterscheiden zudem zwischen extern zugreifbaren Accounts von Teilnehmern und von per Smart Contract zugreifbaren internen Accounts (Wood 2014).

#### 3.3.1.2 Blöcke zur Definition des Systemzustandes

Ein Systemzustand definiert such durch einen Block zusammen mit dessen Vorgängern und den darin enthaltenen Komponenten, die mindestens Transaktionen umfassen. Abbildung 3.5 zeigt Blöcke und Systemzustände für Transfer-Transaktionen. Einzelne Transfer-Transaktionen bilden einen Transaktionsgraphen. Eine als Kante dargestellte Transaktion führt einen Transfer zwischen den als Knoten dargestellten Adressen (A) durch. Die Transaktionen der Blöcke $B_0$, $B_1$, ..., $B_h$ definieren den Zustand von Block $B_h$ als $T_0 \cup T_1 \cup ... \cup T_h$.

Die Datenstruktur eines Blocks $B_h$ referenziert für jeden Block $h > 0$ genau einen Vorgänger $B_{h-1}$, indem der Hash-Wert einer kryptografischen Hash-Funktion $H(B_{h-1})$ als Bestandteil von $B_h$ aufgenommen wird (siehe Abschnitt 3.2.2.2). $B_h$ bezieht damit die Hash-Werte der vorhergehenden Blöcke sowie die im nachfolgenden Abschnitt besprochenen Komponenten ein und erlaubt eine Überprüfung der Integrität der Blockchain bis $B_{h-1}$. Erfolgt eine Veränderung der Daten eines beliebigen Blocks $B_i, i < h$, stimmt der Hash-Wert nicht mehr mit dem in $B_{i+1}$ hinterlegten Wert überein, d.h. die Kette ist unterbrochen.

Die beschriebene Datenstruktur wurde von D. Bayer et al. (1993) und Haber und Stornetta (1991) zur Absicherung von Zeitstempeln in digitalen Dokumenten vorgeschlagen. Mit einem Zeitstempel versehene Dokumente oder Daten werden anhand von Signaturen oder Hash-Werten verknüpft und erlauben Aussagen über die Integrität und die Abfolge und Ordnung. Zeitstempel in Blöcken realisieren die Anwendung heute (Gipp et al. 2015).

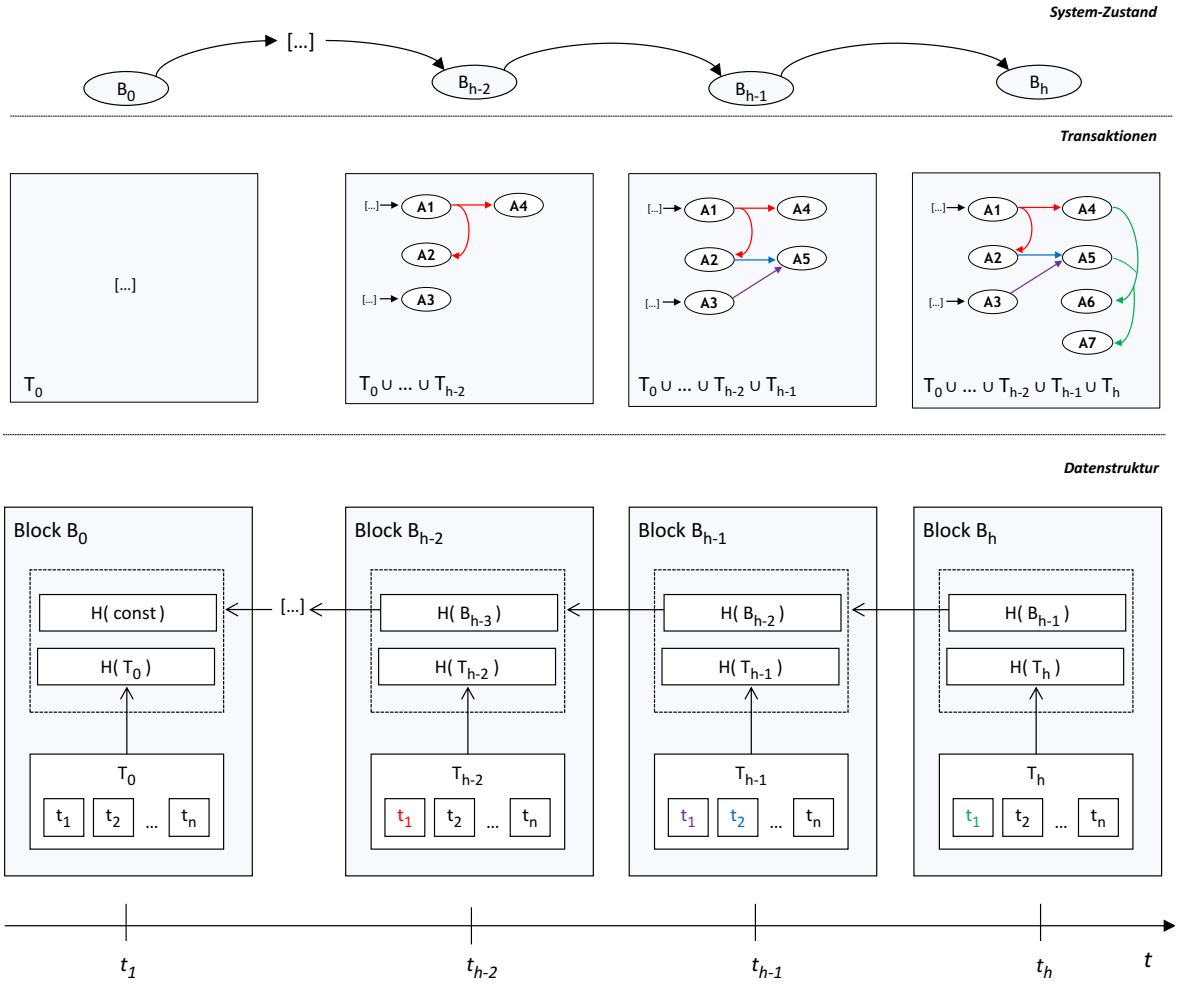



ABBILDUNG 3.5: Zusammenhang zwischen System-Zuständen, Transaktionen und Blöcken

#### 3.3.1.3 Komponenten eines Blocks in Blockchain-Systemen

Die Datenstruktur eines Blocks $B_h$ umfasst mindestens $B_h = (T_h, H(T_h), H(B_{h-1}))$:

- **Menge von Transaktionen** $T_h$: Die Menge enthält $n$ innerhalb des Blocks zu speichernde Transaktionen. Die Transaktionen gehen häufig in die Datenstruktur eines Merkle-Baumes ein (siehe folgender Abschnitt).
- **Transaktionen-Hash** $H(T_h)$: Die Funktion bildet die Menge der Transaktionen von Block h zur Sicherung von deren Integrität auf einen Wert ab und besitzt die Eigenschaften einer kryptografischen Hash-Funktion (siehe Abschnitt 3.2.2.2). Eine Festlegung erfolgt systemweit z.B. als H(x) = SHA-256(x). Die Berechnung der Funktion kann beispielsweise durch den Aufbau eines Merkle-Baumes erfolgen (siehe folgender Abschnitt), dessen Merkle-Root den Funktionswert darstellt.
- **Block-Hash des Vorgängers** $H(B_{h-1})$: Die Funktion bildet den vorhergehenden Block $B_{h-1}$ zur Sicherung von dessen Integrität auf einen Wert ab und besitzt die Eigenschaften einer kryptografischen Hash-Funktion. Der Funktionswert kann als Hash-Pointer oder Referenz auf den Vorgänger interpretiert werden (Narayanan, Bonneau et al. 2016). Die Berechnung der Funktion bezieht mindestens die zur Sicherung der Integrität erforderlichen Hash-Werte des vorhergehenden Blocks ein, d.h. $H(B_{h-1}) = H(H(T_{h-1})||H(B_{h-2}))$ unter Vereinigung der beiden Hash-Werte als Konkatenation. Die für diese Berechnung verwendeten Werte werden in einigen Systemen als Block-Header bezeichnet. Ein Block Header $BH_h$ bildet zusammen mit $T$ einen Block ($B_h = (BH_h, T)$) (Narayanan und Clark 2017).

Weitere Komponenten, die systemspezifisch weiterhin enthalten sein können, sind z.B. Hash-Bäume für Transaktionen und Signaturen (Lombrozo et al. 2018).

**Beispiel**

Für das Bitcoin-System ist $H(T_h) = Merkle_Root$ ein Wurzelknoten eines Merkle-Baumes (Nakamoto 2008b). Für das Ethereum-System ist $H(T_h) = transactionsRoot$ ein Wurzelknoten eines Merkle-Patricia-Baumes (Wood 2014). Beide Datenstrukturen sind Hash-Bäume, die im nachfolgenden Abschnitt erläutert werden. Die Datenstruktur wird beispielsweise für das Bitcoin-System anhand eines Block-Headers $BH_h$ für einen Block $h$ durch Konkatenation erzeugt (Antonopoulos 2017):

$$BH_h = Version||Previous_Block_Header_Hash||Merkle_Root||Time||nBits||Nonce$$

Mit den Werten:

- Version als Versionskennung,
- Previous_Block_Header_Hash als Block-Hash des Vorgängers, der sich durch $BH_{h-1}$ bestimmt als $H(B_{h-1}) = H(BH_{h-1})$ mit der Hash-Funktion SHA-256,
- $Merkle_Root = H(T_h)$ als Wurzel-Knoten eines Merkle-Baumes, siehe nachfolgender Abschnitt,
- Time als Zeitstempel, sowie,
- nBits und Nonce als Parameter des Consensus-Verfahrens.

Mit einer Betrachtung des Consensus-Verfahrens werden die Komponenten in Abschnitt 3.3.4.3 erneut aufgegriffen.

#### 3.3.1.4 Komponenten eines Blocks in Smart-Contract-Systemen

Die Datenstruktur eines Blocks $B_h$ umfasst hier mindestens die Bestandteile $B_h = (T_h, H(T_h), Z_h, H(Z_h), H(B_{h-1}))$:

- **Menge von Transaktionen** $T_h$: analog zu bisher betrachteten Systemen.
- **Transaktionen-Hash** $H(T_h)$: analog zu bisher betrachteten Systemen.
- **Menge von Zustandsspeichern** $Z_h$: Die Menge enthält $m$ innerhalb des Blocks zu speichernde Zustände. Ein Zustand wird häufig als Merkle-Baum oder Merkle-Patricia-Baum erfasst (siehe folgender Abschnitt). Ein Zustandsspeicher enthält Smart Contracts zusammen mit Contract Code (Quellcode oder Byte Code) sowie Zustandsvariablen.
- **Zustandsspeicher-Hash** $H(Z_h)$: Die Funktion bildet die Menge der Zustandsspeicher von Block h zur Sicherung von deren Integrität auf einen Wert ab und besitzt die Eigenschaften einer kryptografischen Hash-Funktion (siehe Abschnitt 3.2.2.2). Die Berechnung der Funktion kann beispielsweise durch den Aufbau eines Merkle-Baumes erfolgen (siehe folgender Abschnitt), dessen Merkle-Root den Funktionswert darstellt.
- **Block-Hash des Vorgängers** $H(B_{h-1})$: Die zur Sicherung der Integrität erforderlichen Hash-Werte umfassen im Unterschied zu bisher betrachteten Blockchain-Systemen zusätzlich den Zustandsspeicher-Hash. Die Berechnung des Vorgänger-Hash-Wertes erfolgt typischerweise als Konkatenation, deren Ergebnis auf eine Hash-Funktion angewendet wird, d.h. $H(B_{h-1}) = H(H(T_{h-1})||H(Z_{h-1})||H(B_{h-2}))$.

**Beispiel**

Abbildung 3.6 gibt den Aufbau von Blöcken zusammen mit einem Beispiel eines Smart Contracts *SalesContract* um Verkauf eines Produkts an. Der hier angegebene Smart Contract der Programmiersprache Solidity kommt in der EVM in den Knoten des Netzwerks zur Ausführung. Eine Transaktion zur Erstellung des Smart Contracts führt bei deren Aufnahme in einen Block $B_{h-1}$ zur Hinterlegung des Contract Code mit Zustandsvariablen in einem Zustandsspeicher einer Adresse A2. Dabei kommt ein Konstruktor zur Ausführung, der eine initial verfügbare Menge *stock* des Produkts setzt. Eine Transaktion zur Durchführung eines Kaufs ausgehend von A3 in Block $B_h$ enthält Daten eines Funktionsaufrufs als order(3) sowie einen Wert 3 Ether. Die Funktion führt den Kauf der Menge 3 durch, sofern die prozedural angegebenen Bedingungen des Vertrages eingehalten werden, die Menge verfügbar und der transferierte Betrag *msg.value* ausreichend sind. Der Verkauf führt zur Reduzierung der Menge verfügbarer Produkte *stock* und der Hinterlegung der Adresse des Käufers *msg.sender* mit der erworbenen Menge. Eine parallel mehrfache Ausführung der Funktion in einem Knoten wird nicht unterstützt (Antonopoulos 2018, S. 129). Die Ausführung der Funktion in allen Mining-Knoten und den validierenden Knoten des Netzwerks erlaubt eine Überprüfung der Korrektheit der Ausführung. Ein Knoten verwirft einen Block, wenn die lokal per Transaktion ausgelöste Ausführung zu Werten von Zustandsvariablen führt, die nicht mit den innerhalb des Blocks übermittelten Zustandsvariablenwerten übereinstimmen.

#### 3.3.1.5 Merkle-Bäume

Ein Merkle-Baum ist eine Datenstruktur für die Speicherung von Transaktionen oder Zuständen und deren Integritätssicherung anhand eines Hash-Wertes, der als Merkle-Root die Wurzel des Baumes darstellt.

**Merkle-Bäume**

Das ursprüngliche Konzept des Merkle-Baumes sieht eine Zusammenfassung einer Menge von beliebigen Daten in Dokumenten vor, deren Inhalte auf einen Wert abgebildet und signiert werden (Merkle 1988). Für den Aufbau des Baumes werden (1.) $n$ Dokumente, hier Transaktionen oder Zustände, jeweils auf eine Funktion H(x) angewendet. H(x) ist in Blockchain-Systemen eine kryptografische Hash-Funktion (Bashir 2017, S. 111), wobei Merkle ursprünglich eine Signaturfunktion vorsieht. (2.) Die Funktionswerte werden paarweise, unter Konkatenation von je zwei Werten, auf H(x) angewendet. Ein Wert kann um sich selbst oder eine Konstante konkateniert werden, sofern die Anzahl der Funktionswerte ungerade ist (Antonopoulos

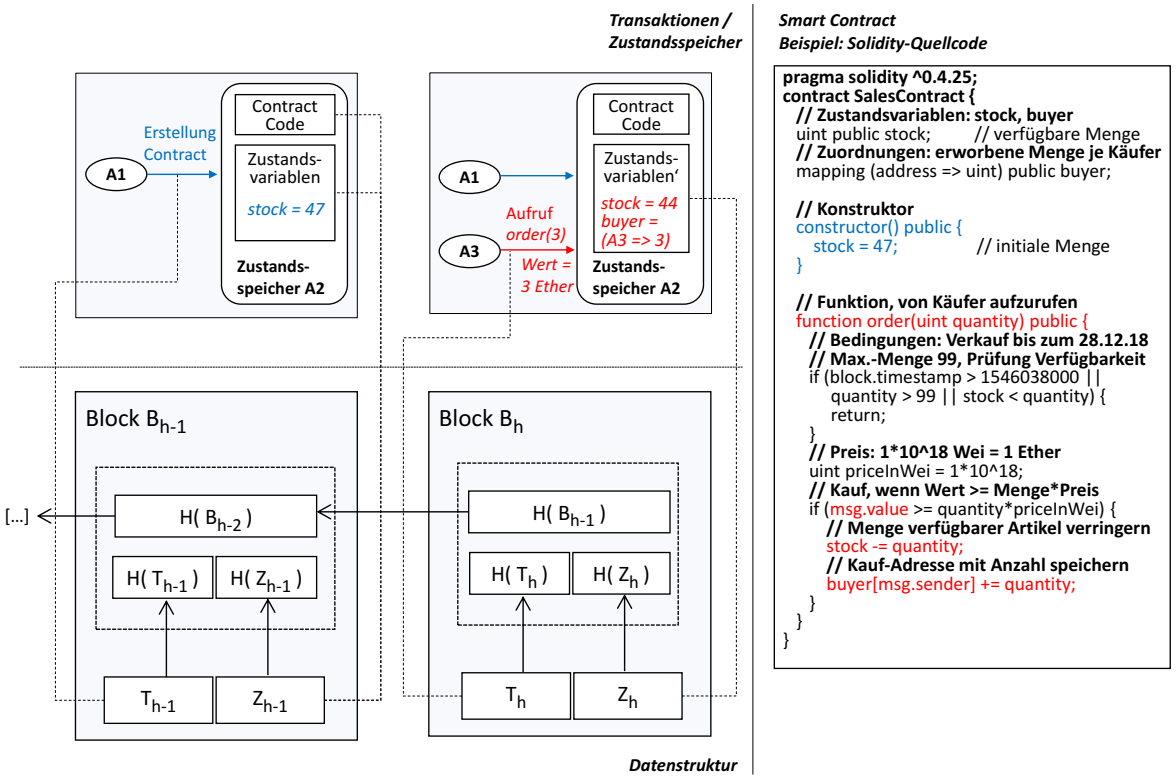



ABBILDUNG 3.6: Repräsentation von Smart Contracts in Blöcken

2017, S. 203). Schritt (2.) wird wiederholt, bis ein Wert als Wurzelknoten *Merkle-Root* vorliegt. Dieser Wert fasst den Inhalt des gesamten Baumes zusammen; eine Veränderung eines Dokuments verändert den Wert des Wurzelknotens. Die Anzahl der Funktionswerte halbiert sich je Ebene, d.h. für $n$ Blattknoten ergeben sich $\lceil \log_2(n) \rceil + 1$ Ebenen[2] von Hash-Werten mit einer Pfadlänge von $\lceil \log_2(n) \rceil$. Die Komplexität für eine Traversierung des Baumes liegt daher hinsichtlich der durchzuführenden Operationen und dem hierfür aufzuwendenden Speicher jeweils bei $O(\log_2(n))$ (Szydlo 2004).

### Überprüfung der Integrität durch Merkle-Proofs

Die Überprüfung der Integrität einer Transaktion oder eines Zustandes kann (1.) aufgrund von $O(\log_2(n))$ effizient und (2.) ohne Kenntnis aller Elemente der Mengen $T_h$ oder $Z_h$ eines Blocks $B_h$ erfolgen. Um die Integrität eines Elements $e_i$ nachzuweisen, wird $e_i$ durch die Konstruktion eines *Merkle-Proofs* als Bestandteil eines Merkle-Baumes nachgewiesen. Sofern die Wurzel des Baumes mit dem Hash-Wert $H(T_h)$ bzw. $H(Z_h)$ eines Blocks übereinstimmt, kann die Integrität von $e_i$ angenommen werden. Abbildung 3.7 zeigt ein Beispiel für den Nachweis von $e_5$, in dem

[2]Unter Einbeziehung der Ebene des Wurzelknotens.

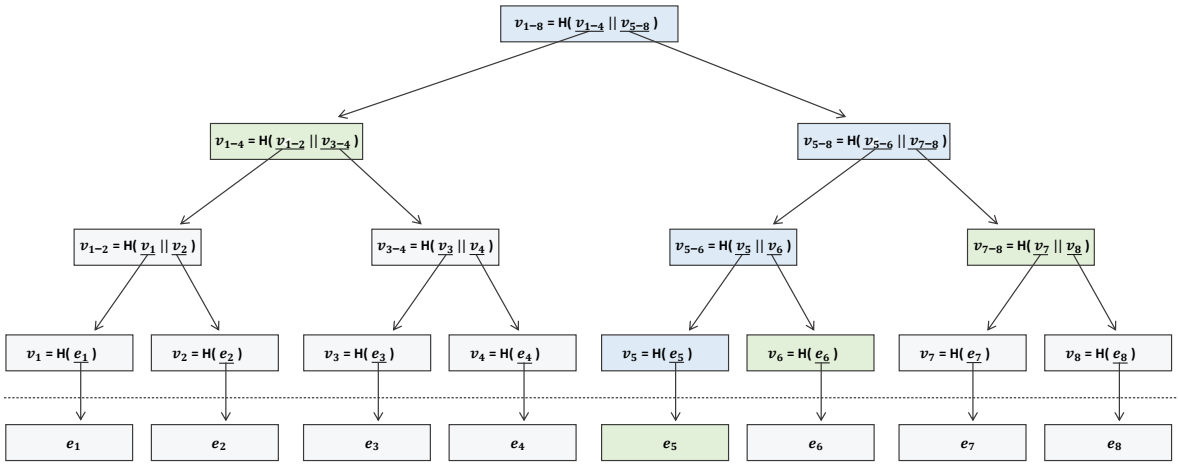



ABBILDUNG 3.7: Merkle-Baum (n = 8) mit Beispiel zur Integritätsprüfung von $e_5$

die während des Nachweises berechneten Knoten (blau) und die für den Nachweis erforderlichen Knoten (grün) hervorgehoben sind. Der Nachweis kann somit auf Basis von Hash-Werten geführt werden, wobei außer dem zugrunde liegenden $e_i$ keine Elemente bekannt sein müssen.

Zur Führung des Nachweises wird (1.) die zur Erstellung des Baumes verwendete Hash-Funktion $H(x)$ mit dem Element $e_i$ angewendet; hier: $H(e_5)$. (2.) Der Hash-Wert eines Geschwisterknotens von $e_i$ wird um den zuvor berechneten Funktionswert konkateniert und auf $H(x) = v$ angewendet; hier: $H(H(e_5)||H(e_6)) = v_{5-6}$. (3.) Der Nachweis wird fortgesetzt, sofern $v$ der Elternknoten (hier $v_{5-6}$) der beiden betrachteten Elemente ist und scheitert andernfalls. Die Schritte (2.) und (3.) werden wiederholt, bis der Nachweis scheitert oder ein Wert vorliegt, der mit der Merkle-Root übereinstimmt. In diesem Fall wird angenommen, dass Integrität für $e_i$ gegeben ist und dieses Element ein Bestandteil der Datenstruktur ist. Das Verfahren wird innerhalb des Bitcoin-Systems verwendet (Narayanan, Bonneau et al. 2016).

**Merkle-Patricia-Bäume**

Das Ethereum-System verwendet eine auf Patricia-Bäumen beruhende Variante (Bashir 2017, S. 111), die als Merkle-Patricia-Baum bezeichnet wird. Ein Patricia-Baum besteht aus einer Trie-Datenstruktur aus geordneten Knoten, die einen Suchbaum bilden. Ein Schlüssel je gespeichertem Wert definiert den Pfad vom Wurzelknoten zum Knoten des Wertes. In Ethereum werden drei Knoten unterschieden (Wood 2014). (1.) Branch-Knoten enthalten Verweise auf zwei oder mehr Kind-Knoten, deren Schlüssel bis zum Branch-Knoten übereinstimmen. Innerhalb dieser Knoten

werden keine Werte gespeichert. (2.) Extension-Knoten enthalten genau einen Wert für den Schlüssel des Knotens sowie einen Verweis auf einen Kind-Knoten. Dieser ist ein in der Ordnung der Schlüssel nachfolgender Knoten. (3.) Leaf-Knoten enthalten einen Wert für den Schlüssel des Knotens und besitzen keine Verweise.

Zur Bestimmung eines Schlüssels für einen Wert wird dieser auf die Keccak-Hash-Funktion angewendet. Der Funktionswert entspricht dem Schlüssel. Der gesamte Baum wird auf den Schlüssel des Wurzelknotens abgebildet, der zur Sicherung der Integrität in einen Block eingeht. Dabei werden mehrere Bäume verwendet; mit $H(T_h)$ als *transactionsRoot* und $H(Z_h)$ als *stateRoot*. Ein weiterer Merkle-Patricia-Baum fasst Transaktionsbelege als *receiptsRoot* zusammen.

## 3.3.2 Netzwerk

Ein Peer-to-Peer-Netzwerk realisiert die Verteilung der Datenbasis zwischen Knoten und die Ausführung des Protokolls in Knoten des Systems (siehe Abschnitt 3.2.1.2). Dabei wird die Kommunikation über ein offenes Netzwerk und daher ein asynchrones verteiltes System angenommen (siehe Abschnitt 3.2.1). Die Komponente Netzwerk realisiert das Merkmal *Verteilung* (4. Merkmal) und, durch Verteilung der Datenstruktur, das Merkmal Transparenz (5. Merkmal).

### 3.3.2.1 Architektur

Der Datenbestand eines Blockchain-Systems etabliert aus der Sicht des Peer-to-Peer-Netzwerks ein Overlay-Netzwerk aus Transaktionen, die direkt und ohne Intermediation zwischen einzelnen Teilnehmern verlaufen. Die als Identitäten herangezogenen Adressen der Teilnehmer bilden die Knoten des Overlay-Netzwerks, dessen Kanten Transaktionen darstellen. Ein Knoten der Software eines Blockchain-Systems (OSI-Schicht 7) ist mit einer begrenzten Anzahl von Knoten direkt verbunden.

Die Propagation von Blöcken und Transaktionen nutzt Gossip-Protokolle (siehe Abschnitt 3.2.1.2). Direkte Verbindungen zwischen Knoten greifen typischerweise auf TCP und auf IP zurück (OSI-Schicht 4 bzw. 3) (Antonopoulos 2017; Parity 2018a).

Im Kontext von Blockchain-Systemen bezeichnet „Knoten“ die Instanz einer Software zur Ausführung des Protokolls (OSI-Schicht 7).

#### 3.3.2.2 Typisierung beteiligter Knoten

In Abhängigkeit der folgenden Aufgaben eines Knotens lassen sich drei Typen von Knoten unterscheiden:

- **Client-Knoten** verwalten Identitäten, erstellen Transaktionen und führen als Teil des Protokolls die Transaktionspropagation durch,
- **Mining-Knoten** führen die Blockerstellung durch,
- **Validierende Knoten** führen die Blockpropagation und die Konfliktauflösung durch.

In dieser Klassifikation wird jeder Typ anhand der mindestens durchzuführenden Aufgaben charakterisiert. Prinzipiell ist eine Zuordnung von Knoten des Netzwerks zu den Aufgaben mehrerer Typen möglich. In der ursprünglichen Realisierungsform werden alle genannten Aufgaben von allen Knoten des Netzwerks ausgeführt (Nakamoto 2008a). Eine Zuordnung von Aufgaben verschiedener Typen zu einzelnen Knoten wird heute durch Software- und Hardwareanforderungen eingeschränkt.

**Client-Knoten**

Client-Knoten realisieren die Mensch-Computer-Schnittstelle (vgl. Antonopoulos 2017) anhand von drei Funktionen:

- **Verwalten von Identitäten**: Eine Identität definiert sich als Adresse, die anhand der Generierung privater und öffentlicher Schlüssel erstellt wird. Ein Client-Knoten verwaltet 1 bis n Identitäten.
- **Erstellen von Transaktionen:** Die Erstellung einer Transaktion besteht in der Konstruktion der Datenstruktur, der Signierung unter Nutzung des privaten Schlüssels einer Identität und der Absendung als Broadcast an die Knoten des Netzwerks.
- **Transaktionspropagation:** Die Verteilung von Transaktionen innerhalb des Netzwerks erfolgt durch Validierung und Weiterleitung eingehender Transaktionen an unmittelbar verbundene Knoten (Gossip-Protokoll).

An Client-Knoten werden im Vergleich zu anderen Knoten-Typen geringere Hardware-Anforderungen hinsichtlich des Hintergrundspeichers und der Rechenleistung gestellt. Die Datenbasis Blockchain liegt nicht notwendigerweise lokal vor. Eine Verarbeitung oder Validierung von Blöcken finden damit nicht notwendigerweise statt. Eine mögliche Realisierungsform ist eine Smartphone-App.

### Mining-Knoten

Mining-Knoten führen mindestens die Aufgabe Blockerstellung innerhalb des verteilten Systems durch (Narayanan, Bonneau et al. 2016). Das Consensus-Verfahren des Protokolls realisiert die Koordination der Blockerstellung durch drei untergeordnete Funktionen:

- **Erstellung der Block-Datenstruktur:** Die in Mining-Knoten eingehenden Transaktionen werden für den Aufbau der Datenstruktur von Blöcken herangezogen. Erstellte Blöcke sind zu diesem Zeitpunkt nicht finalisiert und in einzelnen Knoten lokal und transient gespeichert.
- **Knotenselektion:** Ein Selektionsverfahren wählt Knoten aus, deren Blöcke an die Datenstruktur Blockchain angefügt werden. Autonome Selektionsverfahren führen eine Selektion autonom in einzelnen Knoten durch. Nichtautonome Selektionsverfahren führen eine Selektion per Nachrichtenkommunikation durch.
- **Veröffentlichung:** Erstellte Blöcke selektierter Knoten werden als Broadcast an unmittelbar verbundene Knoten übertragen. Die finalisierte Block-Datenstruktur enthält einen Nachweis der Terminierung des Selektionsverfahrens.

Die Aufgaben werden als Funktion des Protokolls separat erläutert (siehe Abschnitt 3.3.3). An Mining-Knoten werden in Abhängigkeit des Consensus-Verfahrens erhöhte Hardware-Anforderungen gestellt. Verfahren wie Proof-of-Work erfordern eine systemspezifische Berechnung von Funktionswerten zur Lösung von Hash-Funktionen. Das Auffinden der Lösung des „kryptografischen Puzzles“ (Narayanan, Bonneau et al. 2016, S. 115) erfordert hier die Ressource Rechenleistung, die in Abhängigkeit des Verfahrens in Software, für GPU-Recheneinheiten oder als ASIC implementiert wird. Verfahren in privaten Blockchains stellen typischerweise geringere Anforderungen (siehe Abschnitt 3.3.4).

### Validierende Knoten

Validierende Knoten führen mindestens die Aufgaben Blockpropagation und Konfliktauflösung aus (Antonopoulos 2017; Xu, Weber, Staples et al. 2017). Das Consensus-Verfahren des Protokolls koordiniert die Durchführung der Aufgaben.

- **Blockpropagation:** Die Validierung unter Anwendung von Regeln des Consensus-Verfahrens überprüft eingehende Blöcke und verteilt valide Blöcke durch Weiterleitung an unmittelbar verbundene Knoten.

- **Konfliktauflösung:** Zur Auflösung von Konflikten zwischen mehreren Blöcken werden zwei Verfahrensklassen unterschieden. Sofern das Protokoll ein Consensus-Verfahren probabilistischer Finalität vorsieht, wählt ein Verfahren zur Auflösung von Konflikten anhand von definierten Kriterien zwischen parallel angefügten, konfliktären Blöcken. Verfahren nicht-probabilistischer Finalität vermeiden Konflikte unter Annahmen, welche die Verfahren für private Systeme prädestinieren.

An validierende Knoten bestehen die Anforderungen, die Datenbasis der Blockchain vollständig lokal zu indexieren, zu speichern und zu validieren. Hierdurch ergeben sich systemabhängige Hardware-Anforderungen hinsichtlich des Hintergrundspeichers und der Verarbeitung. Diese Anforderungen stellen einen limitierenden Faktor der Skalierung dezentraler Blockchain-Systeme dar (siehe Abschnitt 3.3.6).

### 3.3.3 Protokoll

Mit dem Protokoll eines Blockchain-Systems wird das Ziel verfolgt, die Merkmale *Unveränderlich* (6. Merkmal) und *Trustless* (7. Merkmal) aufbauend auf den durch die Datenstruktur gegebenen Merkmalen zu realisieren. Dabei wird ein dezentrales oder ein partiell dezentrales System unterstellt, dessen Knoten ein Protokoll zur Bestimmung konsistenter und übereinstimmender Systemzustände ausführen.

Das Protokoll regelt die Koordination beteiligter Knoten innerhalb des Netzwerks und definiert hierfür eine Reihe von Funktionen und Nachrichtendefinitionen. Jeder Knoten führt die Funktionen des Protokolls identisch aus, um das Eintreten determinierter Zustände hinsichtlich des Gesamtsystems sicherzustellen. Damit erfolgt die Verarbeitung von Blockchain-Transaktionen und deren persistente Speicherung lokal bei einzelnen Knoten in Übereinstimmung mit anderen Knoten des Netzwerks. Die Aufgabe des Protokolls ist damit das Herbeiführen eines übereinstimmenden und konsistenten Systemzustandes (siehe Abschnitt 3.2.1.7). Die Konsistenz des Gesamtsystems wird im Zeitverlauf durch das Erreichen identischer Zustände in einzelnen Knoten hergestellt. Eine periodische Ausführung des Verfahrens legt regelmäßig einen neuen Systemzustand fest. Dieser definiert eine totale Ordnung über der Menge der Blöcke, die ungeordnete Transaktionen beinhalten. Das Protokoll umfasst folgende Funktionen.

(F1) *Transaktionspropagation* zur Validierung und Verteilung von Transaktionen,

(F2) *Blockerstellung* zur Erstellung eines Blocks und der Selektion von Knoten zur Anfügung eines Blocks,

(F3) *Blockpropagation* zur Validierung und Verteilung von Blöcken sowie

(F4) *Konfliktauflösung* zur Bestimmung des Systemzustandes.

Die Funktion Transaktionspropagation (F1) betrifft die Verteilung valider Transaktionen ausgehend von Client-Knoten. Im engeren Sinne Teil des Consensus-Verfahrens sind die von Mining-Knoten durchgeführte Blockerstellung (F2) (Narayanan und Clark 2017) sowie die von validierenden Knoten durchgeführte Blockpropagation (F3) und die Konfliktauflösung (F4). Im Folgenden werden zunächst die vier Funktionen sowie anschließend Ausprägungen von Consensus-Verfahren in öffentlichen und privaten Blockchain-Systemen betrachtet.

#### 3.3.3.1 Transaktionspropagation (F1)

Blockchain-Transaktionen werden in Form von Nachrichten von beliebigen Nodes als Broadcast ausgesandt. Jede Transaktion enthält mindestens die Attribute Absender, Empfänger und Signatur, sowie mindestens eine Transfer-Operation (siehe Abbildung 3.4). Die Propagation von Transaktionen beschreibt die Verteilung syntaktisch valider Transaktionen an weitere Knoten des Systems.

**(F1.1) Transaktionsvalidierung**

Propagierte Transaktionen in Blockchain-Systemen erfordern eine Validierung zur Sicherstellung der anhand einer Transaktion übertragenen Daten und Informationen. Für jede Validierung gilt: die Transaktion wird verworfen, sofern ein für die Validierung herangezogenes Kriterium nicht erfüllt werden kann.

*(F1.1.1) Validierung der Transaktionssyntax*

Jede in einem Knoten eingehende Transaktion wird zur Vermeidung der Propagation von syntaktisch fehlerhaften Transaktionen validiert. Eine Transaktion wird verworfen, wenn ihre Syntax nicht der Nachrichtendefinition ihres Transaktionstyps entspricht.

*(F1.1.2) Informationelle Validierung*

Transfer-Operationen erfordern eine auf Adressen bezogene Validierung, welche das Vorhandensein zu transferierender Werte sicherstellt. Die Konsistenzprüfung aggregiert die Input-Werte aller eingehenden Transaktionen als Summe der Inputs

sowie die Output-Werte aller ausgehenden Transaktionen als Summe der Outputs (Nakamoto 2008a). Für jede syntaktisch valide Transaktion gilt: die Transaktion wird verworfen, sofern die Summe der Outputs die Summe der Inputs übersteigt. Nicht-verworfene Transaktion werden in Vorbereitung der Erstellung eines Blocks in einem transienten Speicher hinterlegt.

In Abhängigkeit des Systems können domänenspezifische Validierungskriterien zur Anwendung kommen. Ein Beispiel aus dem Bereich der Unternehmensmodellierung ist eine Überprüfung der Syntax von Modelloperationen (vgl. Abschnitt 3.4.5).

**(F1.2) Weiterleitung einer Transaktion**

Jeder Knoten des Systems ist mit einer Teilmenge aller Knoten direkt verbunden. Eine validierte Transaktion wird in Form eines Floodings durch Weiterleitung an alle erreichbaren Knoten propagiert, um die Dauer der Verteilung zu minimieren (Decker und Wattenhofer 2013). Eine nicht-verworfene Transaktion wird in einem Zwischenspeicher vorgehalten, bis deren Aufnahme in einen Block unter Anwendung des Consensus-Verfahrens abgeschlossen ist.

#### 3.3.3.2 Blockerstellung (F2)

Die Aufgaben der Blockerstellung umfassen (2.1) die Erstellung einer Datenstruktur zur Gewährleistung von Integrität und Verbindlichkeit, (2.2) die zufällige Selektion von 1 bis n verfügbaren Knoten und (2.3) die Erstellung eines Blocks durch die selektierten Knoten. Eine zufällige Selektion verhindert die Kontrolle des Systems durch einzelne manipulierte Knoten.

**(F2.1) Block-Datenstruktur**

Mining-Knoten erstellen die Datenstruktur eines Blocks unter Aufnahme validierter Transaktionen. Die Auswahl transient gespeicherter Transaktionen liegt in der Verantwortung einzelner Mining-Knoten und kann von weiteren Faktoren abhängen, z.B. Transaktionsgebühren (Antonopoulos 2017, S. 239). Der Aufbau der Block-Datenstruktur umfasst typischerweise den Aufbau eines Merkle-Baumes zur Berechnung eines Hash-Wertes der aufzunehmenden Transaktionen (siehe Abschnitt 3.3.1.1) sowie systemspezifische Werte, z.B. Versionskennungen (Antonopoulos 2017, S. 243). In Abhängigkeit des konkreten Verfahrens beginnt die Erstellung der Block-Datenstruktur in öffentlichen Systemen wie Bitcoin und Ethereum vor dem Beginn der Selektion (F2.2), um die Latenz zwischen Selektion (F2.2) und Broadcast (F2.3) zu minimieren (Antonopoulos 2017, S. 236–239).

**(F2.2) Knotenselektion**

Die Selektion von Knoten wählt aus der Menge aller Knoten 1 bis n Knoten aus, die eine erstellte Datenstruktur als neuen Block für andere Knoten des Netzwerks per Broadcast aussenden (F2.3). Mindestens ein Leader (vgl. Abschnitt 3.2.1.7) erhält damit eine Berechtigung zur Aussendung eines von allen Teilnehmern anzufügenden Blocks.

Eine notwendige Bedingung für das in diesem Schritt verwendete Selektionsverfahren ist die zufällige Selektion eines Knotens oder einer Gruppe von Knoten. Die Blockerstellung geht damit ohne zentrale Koordination von einzelnen, vorab unbestimmten Knoten aus, die keine Kontrolle über mehrere Iterationen des Verfahrens hinweg besitzen.

Selektionsverfahren unterscheiden sich hinsichtlich der Autonomie ihrer Ausführung. Verfahren, die eine Selektion von Leader-Knoten ohne Input durch das Netzwerk autonom und isoliert in einzelnen Knoten erwirken, werden im Folgenden als autonome Selektionsverfahren bezeichnet. Verfahren, die den Austausch von Nachrichten zwischen Knoten erfordern, werden als nicht-autonome Selektionsverfahren bezeichnet.

Autonome Selektionsverfahren gehen von einzelnen Knoten aus. Diese wenden einen nicht-deterministischen Algorithmus an, deren Output einen validierbaren Nachweis der Terminierung des Algorithmus darstellt. Ein Output repräsentiert einen Nachweis zur Berechtigung der Verteilung eines neuen Blocks. Der Output ist als Bestandteil des nachfolgenden Broadcasts von beliebigen Knoten validierbar. Ein Beispiel eines autonomen Selektionsverfahrens ist die Berechnung von Hash-Werten in Proof-of-Work. Gegenstand des Verfahrens ist die fortlaufende Neuberechnung einer Hash-Funktion $H(BH) = v$ unter Veränderung eines Block Headers $BH$ durch einen variablen Nonce-Wert, bis für $v$ eine vorab definierte Bedingung $v <= Target$ erfüllt ist (siehe Abschnitt 3.3.4.3). Das Auffinden eines gültigen $v$ in einem vorab nicht bestimmten Knoten entspricht einer (pseudo-)zufälligen Selektion des Knotens, die lokal und autonom erfolgt. Nach der Terminierung des Selektionsverfahrens weist $v$ die Terminierung und die damit einhergehende Berechtigung zur Anfügung eines Blocks gegenüber anderen Knoten nach. Der Nachweis erfolgt durch Aufnahme des $v$ als Bestandteil des Blocks (siehe Abschnitt 3.3.4.3).

Nicht-autonome Selektionsverfahren sehen den Austausch von Nachrichten zur Durchführung der Selektion von Knoten vor. Zur Terminierung des Verfahrens

werden mehrere Knoten benötigt, die sich unter Bezugnahme auf das Gesamtsystem koordiniert verhalten. Ein Beispiel ist das PBFT-Verfahren, das v.a. in privaten Blockchain-Systemen eingesetzt wird (siehe Abschnitt 3.3.4). Ein Beispiel ist der Austausch signierter Nachrichten, die jeweils eine Stimme einer Abstimmung über die Auswahl eines zu selektierenden Knotens darstellen.

Eine Kombination der Verfahren mit Delegationsverfahren erlaubt für eine Menge ausgewählter Knoten eine Delegation der Selektion an beliebig wählbare Knoten des Systems.

Die Selektion kann eine Incentivierung der ausgewählten Knoten umfassen, z.B. als Transaktionsgebühr (Antonopoulos 2017, 129 f.) in der Einheit der Transaktion oder in der Einheit eines Utility Tokens. Zu den Gründen einer Incentivierung zählen: (1.) die Sicherstellung des Systembetriebs durch Herstellung eines Angebots einer Menge auswählbarer Knoten aufgrund der geschaffenen Anreize, (2.) die Entschädigung der zur Verfahrensausführung aufzuwendenden Ressourcen und (3.) die Verringerung der Wahrscheinlichkeit der Auswahl gleicher Knoten über mehrere Iterationen des Verfahrens hinweg, d.h. die mit der Anzahl an Knoten zunehmende Absicherung des Netzwerks (siehe Abschnitt 3.3.5.3).

**(F2.3) Veröffentlichung**

Die erstellte Datenstruktur eines Blocks (Schritt 1.) wird als Broadcast-Nachricht an alle Knoten übertragen, die von den ausgewählten Knoten direkt erreichbar sind.

#### 3.3.3.3 Blockpropagation (F3)

Nach der Anfügung eines Blocks wird dieser in Form einer Nachricht an andere Knoten des Systems übertragen (Antonopoulos 2017, S. 254). Die Propagation von Blöcken beschreit die Verteilung von Broadcast-Nachrichten, die validierte Blöcke enthalten. Jeder in einem Knoten eingehende Block wird zur Einhaltung des Protokolls und zur Vermeidung der Propagation fehlerhafter Blöcke validiert.

Qualitative Ziele der Blockpropagation sind (1.) die Maximierung der Anzahl an Knoten, die den Broadcast empfangen sowie (2.) die Minimierung der Latenz zwischen Empfang und Weiterleitung. Mit abnehmender Propagationszeit sinkt die Wahrscheinlichkeit paralleler und konfliktärer Blöcke.

**(F3.1) Validierung eines Blocks**

Von Knoten des Netzwerks empfangene Blöcke durchlaufen eine Reihe von Validierungsschritten.

*(F3.1.1) Syntaktische Validierung*

Eine syntaktische Validierung überprüft die Syntax der Block-Datenstruktur. Ein Block wird verworfen, sofern dessen Syntax nicht der Spezifikation entspricht.

*(F3.1.2) Validierung der Selektion*

Eine Validierung der Selektion überprüft, ob die Erstellung eines Blocks durch einen per Selektion bestimmten Knoten erfolgt ist. In Abhängigkeit des betrachteten Blockchain-Systems können weitere informationelle Kriterien hinzukommen, welche die Validität eines Blocks determinieren. Ein Block wird verworfen, wenn dessen Erstellung nicht durch einen selektierten Knoten erfolgt ist.

Für autonome Selektionsverfahren ist ein per Broadcast verbreiteter Block gültig, wenn dieser einen Output enthält, der die Terminierung des Selektionsverfahrens nachweist (Narayanan, Bonneau et al. 2016, S. 105–107; Wood 2014). Ein Block wird verworfen, sofern der Output eine Terminierung nicht nachweist.

Die Korrektheit der Ausführung von nicht-autonomen Selektionsverfahren erfordert einen per Nachricht übertragenen, verfahrensabhängigen Nachweis der ausgewählten Knoten. Eine Möglichkeit ist die Verwendung von digitalen Signaturen. In diesem Fall wird ein Block verworfen, sofern die Signatur die Auswahl des Selektionsverfahrens nicht nachweist.

In Abhängigkeit des betrachteten Systems können weitere informationelle Kriterien hinzukommen, welche die Validität eines Blocks determinieren. Ein Beispiel ist die Überprüfung von Timestamps hinsichtlich einer maximal tolerierten Abweichung. In domänenspezifischen Systemen kann die Gültigkeit eines Blocks von weiteren Attributen abhängig sein, z.B. von der Gültigkeit von hinterlegten Modellen (siehe Abschnitt 3.4.5).

**(F3.2) Validierung der Blockchain**

Die Validierung der Integrität der Blockchain wird ausschließlich von validierenden Knoten ausgeführt, denen die Datenbasis Blockchain vorliegt.

*(F3.2.1) Synchronisation von Blöcken*

Eine Synchronisation von Blöcken ist erforderlich, sofern die Datenbasis eines Knotens nicht durch vorherige Iterationen unter Ausführung des Schrittes (3.5) vollständig besteht. Ein Knoten sendet als Teil des Protokolls definierte Request-Nachrichten zur Anforderung von lokal nicht vorhandenen Blöcken der Datenbasis

an unmittelbar erreichbare Knoten des Netzwerks, welche bei vorhandener Datenbasis Blöcke als Response-Nachrichten bereitstellen.

*(F3.2.2) Validierung der Integrität aller Blöcke*

Eine Validierung der Integrität der Blockchain besteht in der Berechnung der Abbildung des vorherigen Block Headers $BH_{h-1}$ auf einen Hash-Wert als $BH_{h-1} = v$ und einen Vergleich des in $BH_h$ enthaltenen Hash-Wertes $v'$. Integrität wird nur dann angenommen, wenn gilt: $v = v'$. Sofern eine Überprüfung der Integrität weiter zurückliegender Blöcke nicht in vorherigen Iterationen des Verfahrens erfolgt ist, wird diese anhand des beschriebenen Verfahrens bis zu einem bereits überprüften Block oder bis zum Beginn der Kette zurückgeführt. Ein Block wird verworfen, sofern die Integrität des Blocks nicht nachgewiesen werden kann. Wird die Integrität eines zurückliegenden Blocks nicht nachgewiesen, entspricht dies einer Unterbrechung der Kette, die das Verwerfen aller nach der Unterbrechung vorliegenden Blöcke nach sich zieht.

**(F3.3) Validierung und Ausführung von Transaktionen**

Im Rahmen der Validierung und Ausführung werden alle Transaktionen eines Blocks betrachtet.

*(F3.3.1) Validierung der Integrität der Transaktionen*

Die Validierung überprüft die Integrität der Transaktionen des Blocks, beispielsweise durch den Aufbau eines Merkle-Baumes ausgehend von Transaktionen und einem Vergleich der berechneten Merkle Root $r$ mit der innerhalb des Block Headers angegebenen Merkle Root $r'$. Integrität wird nur dann angenommen, wenn gilt: $r = r'$. Ein Block wird verworfen, sofern die Integrität der Transaktionen nicht angenommen werden kann.

*(F3.3.2) Validierung der Verbindlichkeit von Transaktionen*

Die Validierung der Verbindlichkeit betrifft die Überprüfung der Zuordnung einer Transaktion zu der Identität ihres Absenders. Hierfür wird die Signatur anhand des öffentlichen Schlüssels der Identität überprüft (siehe Abschnitt 3.2.2.3). In Abhängigkeit des Systems kann die Transaktion mehrere Absender sowie korrespondierende Signaturen besitzen. Eine Transaktion wird verworfen, sofern mindestens eine der Signaturen nicht korrekt ist. Weiterhin kann in Abhängigkeit des Systems die Anzahl der Absender beschränkt sein. Die Überprüfung der Signatur kann anhand von in der Transaktion hinterlegten Operationsfolgen oder durch Operationen

des Protokolls erfolgen. Die Signatur gewährleistet zudem Verbindlichkeit, d.h. die Durchführung der Transaktion wird damit nicht-abstreitbar mit den Identitäten der Absender verknüpft.

Ein privates Blockchain-System beschränkt den Zugriff gegenüber Identitäten abhängig von festzulegenden Rollen. Eine Transaktion wird verworfen, sofern rollenspezifische Berechnungen der durchzuführenden Aktion nicht gegeben sind. Ein Beispiel ist eine Rechtevergabe in Organisationen, mit der die Ausführbarkeit von Modell-Operationen in Unternehmensmodellen beschränkt wird (siehe Abschnitt 3.4.5).

*(F3.3.3) Informationelle Validierung und Ausführung von Transfer-Transaktionen*

Die informationelle Validierung von Transaktionen führt die zur Propagation herangezogenen Schritte für alle Transaktionen eines Blocks erneut aus (siehe F1.1.2). Eine Transaktion wird verworfen, sofern die zu transferierenden Werte bei den Absendern der Transaktion nicht vorliegen. Für Transfer-Operationen, die einen Smart Contract aufrufen, wird der hinterlegte Contract-Code als Operationsfolge in jedem Knoten ausgeführt. Der Output der Ausführung des Smart Contracts muss mit den innerhalb des Blocks hinterlegten Zustandsvariablen übereinstimmen. Die Transaktion wird verworfen, sofern keine Übereinstimmung vorliegt.

*(F3.4) Weiterleiten eines Blocks*

Ein Knoten ist mit einer Teilmenge der Knoten des Systems direkt verbunden. Ein in einem Knoten eingegangener und validierter Block wird durch Weiterleitung an alle erreichbaren Knoten propagiert. Die Propagation entspricht einem Gossip-Mechanismus, der in Form eines Floodings einen Block von Knoten zu Knoten über das gesamte System hinweg verteilt (Decker und Wattenhofer 2013).

*(F3.5) Hinzufügen eines Blocks zur Datenbasis*

Ein Knoten fügt validierte Blöcke zur lokalen Datenbasis des Knotens hinzu, um in nachfolgenden Iterationen daran anschließende Blöcke validieren zu können. Das Vorliegen aller bekannten Blöcke in den validierenden Knoten des Systems bestimmt einen konsistenten und übereinstimmenden Systemzustand.

#### 3.3.3.4 Konfliktauflösung (F4)

Die Bestimmung des Systemzustandes entspricht einem probabilistischen Verfahren zur Konfliktauflösung (Nakamoto 2008a; Wood 2014), das einen konsistenten Systemzustand im Zeitverlauf finalisiert. Konfliktäre Zustände treten in Fällen auf,

in denen Blöcke parallel erzeugt und an denselben zurückliegenden Block angefügt werden. Dies ist für einen zeitlichen Abstand zwischen zwei Anfügungen von Blöcken $\Delta t_B$ und die Zeit der Blockpropagation $t_P$ mindestens dann der Fall, wenn $\Delta t_B < t_P$ gilt. Weitere Gründe mit Bezug zur Konsistenz werden in Abschnitt 3.3.5.2 aufgegriffen.

Abbildung 3.8 zeigt ein Beispiel eines Blockchain-Systems, dessen Datenbasis zunächst konsistent (a.) und nach Anfügung der Blöcke $B_{3.1}$ sowie $B_{3.2}$ inkonsistent ist (b.). Die von den Knoten des Netzwerks ausgehend von S2 wahrnehmbaren Zustandsübergänge erlauben zum Zeitpunkt $t_3$ (b.) eine Überführung in die Zustände S3 und S4.

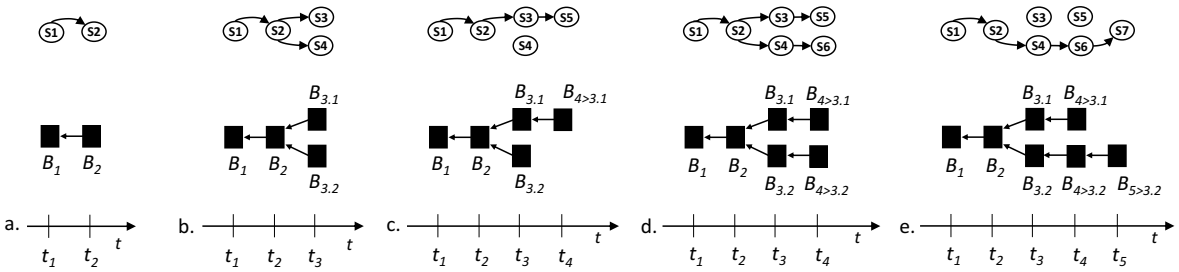



ABBILDUNG 3.8: Zustandskonsistenz

**Verfahren zur Konfliktauflösung**

Ein Verfahren zur Konfliktauflösung bestimmt eine konsistente Datenbasis durch die Auswahl von Blöcken einer Hauptkette (Main Chain) als Teilmenge der Menge aller Blöcke. In Abhängigkeit des Blockchain-Systems können beliebige Konfliktauflösungsverfahren herangezogen werden, die eine totale Ordnung über der Menge der Blöcke bilden.

Verbreitete und in bestehenden Systemen implementierte Konfliktauflösungsverfahren sind die (1.) Wahl der längsten sequenziell miteinander verknüpften Kette von Blöcken (Nakamoto 2008b), oder (2.) die Wahl derjenigen Kette, für welche die zur Erstellung benötigte Rechenleistung maximal ist (siehe Abschnitt 3.3.5.3) (L. M. Bach et al. 2018). Weitere Verfahren sind (3.) die Auswahl zusammenhängender Operationsfolgen als Teil eines BFT-Verfahrens wie PBFT (siehe Abschnitt 3.2.1.7), (4.) die Auswahl zusammenhängender Operationsfolgen anhand von gerichteten Graphen, z.B. als Teil von zyklenfreien Graphen (Directed Acyclic Graph, DAG) (Gramoli 2017; Lewenberg et al. 2015), sowie die (5.) die Ordnung anhand von anwendungsspezifischen Kriterien auf Basis eines BFT-Verfahrens (Sousa et al. 2018). Nicht alle Verfahren sind anhand von Implementierungen evaluierbar.

Bei nicht-probabilistischer Finalität besteht keine Möglichkeit zur Zustandsänderung im Zeitverlauf. Diese ist für PBFT sowie verwandte Verfahren (3., 5.) gegeben, die geringere Anforderungen hinsichtlich der Fehlertoleranz stellen und eine bekannte Menge von Knoten annehmen. Diese Protokolle sind v.a. in privaten Blockchain-Systemen vorzufinden (siehe Abschnitt 3.3.4).

**Konflikte bei probabilistischer Finalität**

Die Verfahren 1., 2. und 4. greifen auf probabilistische Finalität zurück. In diesem Fall kann sich für einen gegebenen Block eine annähernd unveränderliche und, in diesem Kontext, annähernd finale Speicherung ergeben, die im Zeitverlauf nach einer Anzahl von n nachfolgenden Blöcken angenommen werden kann (siehe Abschnitt 3.3.5.3). Bei probabilistischer Finalität können vor der Anfügung von n Nachfolgern die folgenden beiden Konflikte eintreten.

*Stale Block*

Falls für einen Block mehrere Nachfolger gefunden und an diesen angefügt werden, muss entschieden werden, welcher der Nachfolger den Zustand des Systems bestimmt. Andere Nachfolger werden anschließend als Stale Blocks bezeichnet. Für Abbildung 3.8 ergibt sich anhand der Konfliktauflösungsverfahren (1.) und (2.), unter der Annahme einer konstanten Rate der Rechenleistung des Netzwerks, die dargestellte Konfliktauflösung (c.) durch Anfügung von $B_{4>3.1}$ (an einen Block $B_3$)[3]. Ein Zustandsübergang ausgehend von S2 ist nun (c.) ausschließlich zu S3 und schließlich zu S4 möglich. Block $B_{3.2}$ ist als „Stale Block“ (Narayanan, Bonneau et al. 2016, S. 36) ein nicht-annehmbarer Zustand des Systems. Stale Blocks treten bei probabilistischer Finalität regelmäßig auf[4].

*Chain Split*

Mit der Annahme einer probabilistischen Finalität können zu diesem Zeitpunkt weiterhin Zustandsänderungen eintreten. Ein Beispiel ist die Anfügung eines Blocks $B_{4>3.2}$ zum Zeitpunkt $t_4$ (d.), der einen inkonsistenten Zustand hervorruft. Bei Auflösung des Konflikts in $t_5$ (e.) ergeben sich zwei Stale Blocks $B_{3.1}$ und $B_{4>3.1}$. Mehrere parallel hintereinander folgende Blöcke werden als Chain Split bezeichnet. Zur

[3]Notation $B_{h.i>k.j}$: Block $B_{h.i}$ ist direkter oder indirekter Nachfolger von Block $B_{k.j}$. D.h., $B_{h.i>k.j}$ identifiziert den $i$-ten Block der Block Height $h$, der Nachfolger eines Blocks $B_{k.j}$ ist, so dass gilt $h > k$ ($h, k, i, j \in \mathbb{Z}$). Die Angabe $.i$ und $.j$ entfällt, sofern sich ein Block $B_{h.i>k.j}$ als Nachfolger von $B_k$ eindeutig bestimmt.

[4]siehe z.B. https://www.blockchain.com/de/btc/orphaned-blocks .

definitorischen Abgrenzung von Stale Block wird ein Chain Split definiert als eine parallele Anfügung von mindestens zwei unterschiedlichen Blöcken, die jeweils mindestens einen Nachfolger besitzen, z.B. $B_{3.1}$ gefolgt von $B_{4>3.1}$ sowie $B_{3.2}$ gefolgt von $B_{4>3.2}$.

Ein Chain Split wird für den Normalfall der Ausführung des Verfahrens vermieden, da die Anfügung eines Blocks stets einen durch das Konfliktauflösungsverfahren bestimmten Block, hier $B_{4>3.1}$, heranzieht. Ein Chain Split tritt mindestens für den Ausnahmefall ein, in dem mehrmalig zufällig eintretende parallele Selektionen aufgrund von $\Delta t_B < t_P$ erfolgen[5].

Weitere Ursachen für Chain Splits sind als Hard Forks (Narayanan, Bonneau et al. 2016, 73 ff.) bezeichnete, absichtlich oder unabsichtlich herbeigeführte Protokolländerungen.

- Absichtlich herbeigeführte Protokolländerungen treten aufgrund von Protokoll-Updates ein, oder im Falle von Abspaltungen, die auf Grundlage eines Protokolls eines bestehenden Systems ein verändertes Protokoll als Bestandteil eines neuen Systems ableiten. Ein Beispiel sind die Blockchain-Systeme Ethereum und Ethereum Classic, die auf ein ursächliches Blockchain-System zurückzuführen sind (Berentsen und Schar 2018).
- Unabsichtlich herbeigeführte Protokolländerungen sind Software-Fehler in Node-Software (Andresen 2013), sowie Angriffe, die zu einer fehlerhaften Ausführung des Protokolls führen. Ein Beispiel sind verschiedene Formen von „Block Withholding Attacks“, bei denen mehrere Blöcke bewusst an einen Block vor dem selektierten Block angefügt werden (Courtois und Bahack 2014). In diesem Fall kann die Selektion von Blöcken über mehrere Iterationen des Verfahrens hinweg zwischen Blöcken beider Teilketten, hier $B_{x>3.1}$ und $B_{y>3.2}$, alternieren.

Von Hard Forks werden Soft Forks (Narayanan, Bonneau et al. 2016, 73 ff.) unterschieden, die eine Protokolländerung innerhalb der bestehenden Regeln des Protokolls bezeichnen. Als Soft Fork durchführbare Protokolländerung erlauben Protokoll-Updates ohne Auftreten eines Chain Splits (siehe z.B. Lombrozo et al. (2018)). Dabei kann anhand eines Soft Forks lediglich eine Konkretisierung bestehender Regeln erfolgen.

[5] siehe z.B. Nachfolger von $B_{363996}$: https://www.blockchain.com/de/btc/block-height/363996 .

**Transaktionsbestätigungen**

Eine Blockchain-Transaktion wird bei probabilistisch finalen Consensus-Verfahren nach einer Anzahl $c$ an Confirmations oder Bestätigungen als final abgeschlossen und unveränderlich bewertet. Der Wert des Parameters ist nicht Teil des Protokolls und wird von den an einer Transaktion beteiligten Absendern und Empfängern angenommen. Confirmations $c$ ist die Anzahl an Blöcken einer als gültig betrachteten Menge von Blöcken $MC$ einer Blockchain, die seit der Aufnahme der Transaktion in einen Block $B_h \in MC$ vorliegen. Die Teilmenge $MC$ (Main Chain) umfasst diejenigen Blöcke, die nach den Regeln des Protokolls den konsistenten Zustand der Datenbasis Blockchain definieren. Bei Aufnahme einer Transaktion in einen Block bestehen $c = 1$ Confirmations. Bei Erreichen eines Blocks $B_{h+n} \in MC$ ist $B_{h+n}$ ein direkter oder indirekter Nachfolger von $B_h$ und es bestehen $c = n + 1$ Confirmations.

## 3.3.4 Ausprägungen von Consensus-Verfahren

Unter Nutzung der im vorherigen Abschnitt besprochenen Funktionen des Protokolls erzielen die hier diskutierten Verfahren eine Übereinkunft hinsichtlich des Systemzustandes. Dabei treffen private und öffentliche Blockchain-Systeme unterschiedliche Annahmen hinsichtlich der zugrunde liegenden verteilten Systeme 3.2.1.7.

### 3.3.4.1 Consensus-Verfahren in privaten Blockchain-Systemen

Private Blockchain-Systeme erzielen ohne zentrale Koordination eine Übereinkunft zwischen einer Reihe von bekannten Knoten, z.B. für Unternehmen in Wertschöpfungsnetzen oder in Konsortien (siehe Abschnitt 3.1.1). Zur Abgrenzung von verteilten Datenbanksystemen werden in dieser Diskussion nur Verfahren berücksichtigt, die das Ziel verfolgen, das Problem der Übereinkunft zu lösen (siehe Abschnitt 3.2.1.7).

*Funktionen*

Die Verfahren realisieren die Funktion Blockerstellung (F2) mit den untergeordneten Funktionen zur Erstellung einer Datenstruktur der zu speichernden Daten (F2.1), der Selektion von Knoten (F2.2) und der Übertragung anzufügender Daten innerhalb des Netzwerks (F2.3). Die Übertragung anzufügender Daten wird in diesem

Kontext oft durch den Commit-Begriff beschrieben. Die Verfahren realisieren Konfliktauflösungsverfahren (F4) bei gegebener Finalität und vermeiden Stale Blocks und Chain Splits.

*Verfahrensklassen*

In privaten Blockchain-Systemen können drei Verfahrensklassen identifiziert werden.

- **Verfahren von öffentlichen Blockchain-Systemen**, die für den Einsatz in privaten Systemen prinzipiell übernommen werden können, da öffentliche Systeme höhere Anforderungen an Fehlertoleranz und Resilienz stellen. Ein Beispiel ist Proof-of-Work (Parity 2018b) (siehe nachfolgender Abschnitt).
- **BFT-Verfahren** in Implementierungen wie PBFT aus dem Bereich der Consensus-Algorithmen verteilter Systeme (siehe Abschnitt 3.2.1.7). Teilweise werden gegenüber der ersten Verfahrensklasse höhere Durchsatzraten sowie eine höhere Anzahl von Transaktionen pro Zeiteinheit erzielt. Neben PBFT wird teilweise der Einsatz von Paxos-Implementierungen verfolgt. (Androulaki et al. 2018; Vukoli 2016).
- **Verfahren aus dem Cloud-Computing** für das Daten-Management von Cloud-Computing-Systemen, die Daten abstrahiert von konkreten Server-Infrastrukturen verteilt verwalten. Beispiele hierfür sind der Zab-Algorithmus von Apache ZooKeeper (Apache 2018) sowie der Raft-Algorithmus als eine erweiterte Implementierung des Paxos-Algorithmus (Dinh et al. 2018).

Private Blockchain-Systeme treffen gegenüber öffentlichen Systemen eine Reihe von Annahmen (siehe Abschnitt 3.2.1.7). Eine Annahme ist die Bekanntheit der Knotenmenge, die in einem privaten System einem ausgewählten Nutzerkreis entspricht. Hinsichtlich der Nachrichtenkommunikation treffen BFT- und Cloud-Computing-Verfahren Annahmen, indem Timeouts als Obergrenzen für Nachrichtenzustellungen festgelegt werden. Zudem wird unter der Annahme einer beliebig oft wiederholbaren Absendung von Nachrichten eine stets erfolgende Zustellung angenommen. Ein Ausbleiben von Nachrichten beeinträchtigt die Durchführung von nicht-autonomen Selektionsverfahren (F2). Eine zwischen mehreren Knoten erfolgende Abstimmung anhand von Nachrichten verzögert das Verfahren, sofern Knoten aus der Menge der bekannten Knoten nicht erreichbar sind. Die Fehlertoleranz ist aufgrund der geringeren Resilienz gegenüber Protokollabweichungen daher im Vergleich zu den prototypischen Verfahren öffentlicher Blockchain-Systeme geringer.

Gleichzeitig erlauben BFT- und Cloud-Computing-Verfahren eine höhere Skalierbarkeit der Performance, bezogen auf die Rate der Transaktionen pro Zeit.

Insbesondere BFT-Verfahren wie PBFT treten zunehmend in den Vordergrund. Darauf aufbauende Verfahren unter Verwendung von SMR werden von einer Reihe von Blockchain-Systemen herangezogen. Hierzu gehören beispielsweise Hyperledger Indy mit dem RBFT-Verfahren (Linux Foundation 2017), Hyperledger Fabric in Version 0.6, dessen Consensus-Verfahren in neueren Versionen durch Apache Kafka realisiert wird (Linux Foundation 2017), Chain (Core 2018), Corda (Corda 2018) und Tendermint (Tendermint 2018). Aktuelle Entwicklungen umfassen spezialisierte Erweiterungen von BFT, welche die von partiell asynchronen Systemen getroffenen Annahmen verringern. Das HoneyBadgerBFT-Verfahren erlaubt eine Fortführung des Verfahrens (Liveness) u.a. ohne Annahmen über Laufzeiten (A. Miller et al. 2016). Das Verfahren wird mit der Erweiterung BEAT (Duan et al. 2018) für unterschiedliche Anwendungsszenarien in Durchsatz und Latenz verbessert.

*Merkmale von Consensus-Verfahren in privaten Blockchain-Systemen*

Die folgenden Merkmale ergeben sich typischerweise für Consensus-Verfahren in privaten Blockchain-Systemen:

- Annahme einer bekannten Knotenmenge des Netzwerks:
  - Einsatz nicht-autonomer Selektionsverfahren.
  - Die Aufwendung von Ressourcen ist kein inhärenter Bestandteil der Selektion von Knoten.
  - Hohe Skalierbarkeit, d.h. eine Steigerung des Durchsatzes und eine Verringerung der Latenz, sowie eine Senkung der Dauer der periodischen Verfahrensausführung gegenüber Consensus-Verfahren mit autonomen Selektionsverfahren.
- Fehlertoleranz unter Annahmen zur Nachrichtenkommunikation:
  - Einsatz von Konfliktauflösungsverfahren probabilistischer Finalität.
  - Absenz von Stale Blocks.
  - Absenz von Chain Splits.
- Validierung der Ausführung der Verfahren des Protokolls unter Zugriffsbeschränkungen.

Dabei besteht ein Zusammenhang zwischen der Annahme einer bekannten Knotenmenge des Netzwerks und der Möglichkeit des Einsatzes von nicht-autonomen Selektionsverfahren (F2). Autonome Selektionsverfahren erfordern eine Selektion, die von Knoten ausgeht. Nicht-autonome Selektionsverfahren erlauben eine Auswahl durch Nachrichtenkommunikation zwischen den Elementen der Knotenmenge.

Weiterhin besteht ein Zusammenhang zwischen den Annahmen zur Nachrichtenkommunikation in asynchronen verteilten Systemen und der Finalität im Zuge der Konfliktauflösung (F4). Die Annahmen betreffen die Festlegung von Timeouts als Obergrenzen für Nachrichtenzustellungen und die Annahme stets durchgeführter Zustellungen (siehe Abschnitt 3.2.1.7).

#### 3.3.4.2 Consensus-Verfahren in öffentlichen Blockchain-Systemen

Öffentliche Blockchain-Systeme treffen aufgrund des unbeschränkten Zugriffs durch beliebige Teilnehmer keine Annahmen hinsichtlich der Nachrichtenkommunikation sowie des Verhaltens der Systemteilnehmer (siehe Abschnitt 3.2.1.7). Die zentrale Anforderung ist das Erzielen einer Übereinkunft zwischen einander nicht vertrauenden Knoten, unter Gewährleistung der Fehlertoleranz des Systems und der Resilienz gegenüber Protokollabweichungen. Zur nachfolgenden Diskussion werden die folgenden Verfahrensklassen herangezogen:

- Proof-of-Resource-Verfahren sehen eine Blockerstellung (F2) unter dem Einsatz von nicht-autonomen Selektionsverfahren durch die Aufwendung von Ressourcen wie Rechenleistung, Speicherplatz oder Zeit vor. Die Verfahren werden mit Konfliktauflösungsverfahren (F4) kombiniert und sind in implementierten Systemen verbreitet (L. M. Bach et al. 2018).
- Graph-basierte Verfahren bilden Operationsfolgen von Transaktionen als Knoten eines gerichteten und meist azyklischen Graphen ab, dessen Kanten eine partielle Ordnung über der Menge der Transaktionen bilden (Gramoli 2017; Lewenberg et al. 2015). Das Selektionsverfahren zur Anfügung von Graphen-Knoten ist autonom (F2). Das Konfliktauflösungsverfahren (F4) realisiert eine probabilistische Finalität.
- Modifizierte BFT-Verfahren bauen auf in privaten Systemen verbreiteten Verfahren auf, ohne den Systemzugang einzuschränken. Dabei werden Annahmen zur Nachrichtenkommunikation getroffen. Die Verfahren sind gegenüber den anderen beiden Klassen weniger verbreitet (L. M. Bach et al.

2018; Bano et al. 2017). Die Charakteristika folgen den BFT-Verfahren privater Systeme. Eingesetzte Selektionsverfahren sind nicht-autonom bei nicht-probabilistischer Finalität.

*Proof-of-Resource-Verfahren*

Proof-of-Resource-Verfahren sehen eine Blockerstellung (F2) unter Aufwendung von Ressourcen wie Rechenleistung, Speicherplatz oder Zeit vor. Das Teilverfahren zur Selektion von Knoten (F2) wählt Knoten anhand eines Nachweises aus, welcher die Aufwendung einer Ressource nachweist. Der Nachweis entspricht einem Wert, der die Terminierung des Selektionsverfahrens belegt. Dabei determiniert die Quantität der Ressourcen je Knoten die Wahrscheinlichkeit der Auswahl eines Knotens, so dass eine als Sybil-Angriff bezeichnete Manipulation der Selektion in Form einer Erhöhung der Anzahl der eigens betriebenen Knoten nicht möglich ist.

Die Verfahren werden mit einem Konfliktauflösungsverfahren (F4) kombiniert, das zwischen konfliktären Blöcken in Abhängigkeit der Menge der aufgewendeten Ressource auswählt (L. M. Bach et al. 2018; Garay, Kiayias und Leonardos 2015; Narayanan, Bonneau et al. 2016).

**Proof-of-Work oder Nakamoto Consensus** bildet historisch das erste Verfahren dieser Klasse (Nakamoto 2008a). Zur Blockerstellung (F2) muss die Ressource Rechenleistung aufgewendet werden, die aufgrund der dafür erforderlichen Energie als Absicherung des Systems interpretiert wird (siehe Abschnitt 3.3.5.3). Dies stellt zugleich einen Nachteil dar. Der Nachweis wird als Lösungswert eines kryptografischen Puzzles geführt, der die Terminierung des Selektionsverfahrens nachweist. Das Verfahren ist heute in implementierten Systemen verbreitet (L. M. Bach et al. 2018). Aktuelle Realisationen des Verfahrens bestehen u.a. in den Systemen Bitcoin und Ethereum. Zudem werden Erweiterungen diskutiert, u.a. (a.) die Verwendung nutzbringender Berechnungen (Ball et al. 2017), (b.) die Verwendung von Selektionsverfahren basierend auf anderen kryptografischen und nicht-kryptografischen Puzzeln (Bitansky et al. 2016) sowie (c.) Erweiterungen hin zu einer determinierten Selektion eines Knotens durch eine oder mehrere vorhergehende Verfahrensiterationen (Andrychowicz und Dziembowski 2015; Garay, Texas et al. 2017).

**Proof-of-Stake** sieht zur Blockerstellung (F2) eine Selektion von Knoten in Abhängigkeit eines Anteils je Knoten vor. Ein Anteil entspricht je Knoten einer Anzahl aufgewendeter Tokens an der Anzahl der Tokens des Gesamtsystems (Antonopoulos 2018, S. 320 f.). Die Auswahl entspricht einer anhand von Anteilen gewichteten Abstimmung unter den Knoten. Im Gegensatz zu Proof-of-Work wird dabei nicht

der Prozess der Aufwendung der Ressource, sondern deren Existenz nachgewiesen. Die Konfliktauflösung (F4) wählt Blöcke oder Teil-Ketten, im Falle von Chain-Splits, in Abhängigkeit der mehrheitlich zur Erstellung aufgewendeten Anteile aus. Das Nothing-At-Stake-Problem bezeichnet diesbezüglich die Möglichkeit der Beeinflussung der Auswahl einer Teil-Kette durch diejenigen Knoten, die eine Verschiebung des mehrheitlichen Anteils zwischen den Teilketten verursachen können.

Neben der Grundform des Verfahrens existieren weitere Varianten. Hierzu gehören (a.) der z.T. als Proof-of-Service bezeichnete Betrieb von Master-Knoten unter gleichzeitiger Hinterlegung von Tokens innerhalb der Knoten sowie (b.) Delegated Proof-of-Stake als delegatives Verfahren, das eine Übertragung von Anteilen, oder den sich daraus ergebenden Abstimmungsgewichten, an andere Knoten oder Gruppen von Knoten vorsieht. Anhand der Delegation soll die Auswahl nicht-verfügbarer Knoten vermieden werden. Die Anwendung des Verfahrens für das System Ethereum wird vor dem Hintergrund der Skalierung diskutiert (Zamfir 2015) (siehe Abschnitt 3.3.6) und in Systemen wie Cardano und EOS (Delegated Proof-of-Stake) erprobt (L. M. Bach et al. 2018).

**Proof-of-Elapsed-Time** ist ein von Intel und Hyperledger (Bano et al. 2017) vorgeschlagenes Verfahren, das die Aufwendung von Zeit vorsieht. Der Nachweis der Aufwendung der Ressource zur Selektion von Knoten (F2) wird anhand von CPU-Instruktionen realisiert, die in einer Secure-Enclave einer CPU zur Ausführung kommen. Das Verfahren wird in Hyperledger Sawtooth anhand von Intel-SGX-Instruktionen implementiert. Die Selektion eines Knotens erfolgt in Abhängigkeit einer von CPUs abgewarteten Zeitdauer. Unter Annahmen zur Nachrichtenkommunikation werden Konflikte vermieden. Die Evaluierbarkeit des Verfahrens wird durch die geringe Verbreitung und Auslastung eingeschränkt.

**Proof-of-Space** sieht die Verwendung der Ressource Speicher vor. Zur Vermeidung von Sybil-Angriffen ist ein Gestaltungsziel der Verfahren die Verhinderung des Einsatzes spezialisierter Hardware wie ASIC. Die Verfahren sehen einen Nachweis zur Selektion von Knoten (F2) in Abhängigkeit des verfügbaren Speichers vor (Ateniese et al. 2013). Die Anwendung ist mit der Bereitstellung von Speicherplatz im Sinne von IaaS kombinierbar (Filecoin 2014).

*Graph-basierte Verfahren*

Verfahren dieser Art bilden gerichtete und zyklenfreie Graphen als Directed Acyclic Graph (DAG). Knoten repräsentieren Transaktionen, über denen gerichtete Kanten eine partielle Ordnung bilden. Das Anfügen neuer Transaktionen entspricht dem

Anfügen von Knoten, wobei ausgehende Kanten zu bestehenden Knoten des Ausgangsgrades 0 hinzukommen, die hin zu den anzufügenden Knoten verlaufen. Eine Zusammenfassung von Transaktionen in Blöcken ist hierfür nicht notwendig. Implementierungen sind u.a. in Hashgraph und dem Tangle-Verfahren von IOTA vorhanden (El Ioini und Pahl 2018). Die Evaluierbarkeit der Verfahren ist aufgrund der geringeren Verbreitung und Auslastung der Systeme eingeschränkt.

*Modifizierte BFT-Verfahren*

Modifizierte BFT-Verfahren basieren auf den Verfahren privater Systeme. Aufgrund des öffentlichen Zugriffs unterstellen einige Verfahren eine dynamische, sich verändernde Knotenmenge. Diese besteht in der Verwaltung einer Datenstruktur bekannter Knoten, deren operationaler Zustand durch Annahmen über Timeouts determiniert wird. Die Anwendung der Verfahren tritt insbesondere in Systemen auf, die den Zugang im Sinne eines öffentlichen Systems nicht beschränken und gleichzeitig die im vorhergehenden Abschnitt diskutierten Annahmen zur Knotenmenge und Nachrichtenkommunikation treffen. Ein Beispiel ist das nicht-zugangsbeschränkte Ripple-System, dessen Infrastruktur nicht-öffentlich betrieben wird (Armknecht et al. 2015).

*Merkmale von Consensus-Verfahren in öffentlichen Blockchain-Systemen*

Die folgenden Merkmale ergeben sich typischerweise für Consensus-Verfahren in öffentlichen Blockchain-Systemen:

- Annahme einer unbekannten Knotenmenge des Netzwerks:
  - Einsatz autonomer Selektionsverfahren.
  - Die Aufwendung von Ressourcen als Nachweis der Terminierung des Selektionsverfahrens (Sybil-Resistenz).
  - Eine geringe Skalierbarkeit bei höherer Dauer der periodischen Verfahrensausführung gegenüber Consensus-Verfahren mit nicht-autonomen Selektionsverfahren.
- Fehlertoleranz ohne Annahmen zur Nachrichtenkommunikation:
  - Einsatz probabilistisch finaler Konfliktauflösungsverfahren.
  - Entstehung von Stale Blocks als Normalfall.
  - Entstehung von Chain Splits in Ausnahmefällen.
- Eine öffentliche Validierung der Ausführung der Verfahren des Protokolls.

Dabei besteht ein Zusammenhang zwischen der Annahme einer unbekannten Knotenmenge des Netzwerks und der Möglichkeit des Einsatzes von autonomen Selektionsverfahren (F2). Autonome Selektionsverfahren erfordern eine Selektion, die von den Knoten selbst ausgeht und Ressourcen als Nachweis der Terminierung erfordert. Das Anlegen mehrerer Instanzen von Knoten durch einzelne Teilnehmer beeinflusst das Verfahren daher nicht (Sybil-Resistenz).

Im Zeitverlauf erfolgende Zustandsänderungen sind prinzipiell beliebig lange möglich. Verfahren zur Gewährleistung einer probabilistischen Finalität bilden dieses Systemverhalten auf eine im Zeitverlauf stattfindende Konfliktauflösung unter abnehmender Wahrscheinlichkeit nicht-eintreffender Nachrichten ab.

#### 3.3.4.3 Proof-of-Work oder Nakamoto Consensus

Der diesem prototypischen Proof-of-Resource-Verfahren zugrunde liegende Proof-of-Work-Ansatz definiert ursprünglich eine CPU-abhängige Kostenfunktion, die zur Vermeidung von E-Mail-Spam (Dwork und Naor 1992) und zur Generierung von transferierbaren Tokens (Back 2002; Finney 2004) definiert wurde. Zur Erstellung von Blöcken in einem dezentralen System wird das Verfahren erweitert um (a.) eine autonome Selektion als Teil der Blockerstellung (F2) sowie (b.) ein Konfliktauflösungsverfahren, das ursprünglich auf Basis der längsten zusammenhängenden Kette eine probabilistische Finalität (F4) gewährleistet (Nakamoto 2008a).

Proof-of-Work beschreibt damit ein Consensus-Verfahren, das auf Grundlage eines probabilistischen Verfahrens (a.) periodisch eine zufällige Auswahl eines vorab nicht bekannten Knotens zur Anfügung von Operationsfolgen oder Transaktionen durchführt und (b.) konfliktäre Operationen durch definierte Regeln auflöst. Die periodische Terminierung des Verfahrens unter Aufwendung der Rechenleistung des gesamten Netzwerks tritt nach einer festgelegten mittleren Zeitdauer $t_i$ ein; in Abhängigkeit des Systems (Gervais et al. 2016) etwa $t_i = 14$ s oder $t_i = 600$ s (Narayanan, Bonneau et al. 2016).

a. Die Selektion des Knoten (F2) erfolgt autonom, indem von allen Knoten zeitaufwendige und nicht-deterministische Berechnungen zur Lösung eines kryptografischen Puzzles über eine mittlere Dauer von z.B. $t_i = 10$ min (Narayanan und Clark 2017) durchgeführt werden. Das probabilistische Auffinden einer Lösung erlaubt keine Vorhersage der genauen Zeitdauer der Periode oder des nächsten Knotens. Eine Periode endet mit dem Auffinden einer Lösung eines Knotens, der per Algorithmus berechtigt ist, den Systemzustand durch Anfügung eines vorab erstellten Blocks zu verändern.

b. Konfliktäre Zustände (F4) können auftreten, sofern kurz nacheinander, innerhalb eines Zeitintervalls $\Delta t_i$, mehrere Lösungen von verschiedenen Knoten gefunden werden. Der Systemzustand kann damit ausgehend von einem Vorzustand parallel durch mehrere Knoten verändert werden (3.3.5). Dieser Fall kann eintreten, wenn für die Zeit der Propagation einer Zustandsänderung $t_p$ gilt: $t_p < \Delta t_i$. Zur Auflösung wird in aktuellen Realisationen des Verfahrens diejenige Teil-Kette gewählt, für welche die zur Erstellung benötigte Rechenleistung maximal ist (Narayanan und Clark 2017).

Ein kryptografisches Puzzle ist definiert durch eine Funktion $H(BH) = v$ sowie eine oder mehrere an $v$ geknüpfte Nebenbedingungen. Zur Bestimmung einer Lösung wird ein Wert $BH$ gesucht, für den der Funktionswert $v$ hinsichtlich der Nebenbedingungen gültig ist. Die Nebenbedingung schränkt den Lösungsraum ein. Häufig erfolgt die Einschränkung durch Angabe eines Maximalwertes (Narayanan und Clark 2017) *Target*, d.h.:

$$H(BH) = v \,\text{mit}\, v <= Target$$

Die Funktion besitzt die Merkmale einer kryptografischen Hash-Funktion (siehe Abschnitt 3.2.2.2), d.h. das Auffinden eines BH unter Erfüllung von $b <= Target$ ist nicht effizient, während das Führen eines Nachweises für ein ermitteltes $BH$ anhand der Berechnung effizient ist.

Das Verfahren zur Blockerstellung (F2) sieht zunächst (2.1) die Erstellung eines Blocks vor, dessen Block Header $BH$ der Parameter der Hash-Funktion ist. Die Selektion eines Knotens (2.2) erfolgt durch fortlaufende Berechnungen von H unter Veränderung des Wertes *Nonce* in $BH$, bis zur Ermittlung eines gültigen $v$. Die fortlaufenden Neuberechnungen von H finden in Knoten des Netzwerks statt und erfordern unter Aufwendung der Ressourcen aller Knoten des Netzwerks im Mittel eine bestimmte Zeitdauer $t_i$. Ein gültiges $BH$ wird zusammen mit dem erstellten Block als Broadcast an andere Knoten des Netzwerks übertragen (2.3). $BH$ schließt neben Nonce u.a. den Hash-Wert des vorhergehenden Blocks sowie die Merkle Root zur Sicherung der Integrität aller Transaktionen ein (siehe 3.3.1.3). Während der Blockpropagation (F3) erfolgt eine Validierung des Nachweises durch die einmalige Berechnung von $H(BH)$ sowie die Überprüfung der Nebenbedingung. Die Propagation des Blocks durch Knoten des Netzwerks erfolgt genau dann, wenn der enthaltene Block Header $BH$ bei Berechnung von $H(BH)$ die Nebenbedingung erfüllt und somit die Terminierung des Selektionsverfahrens nachweist.

Die Ermittlung einer Lösung erfordert, unter Aufwendung aller Ressourcen des Netzwerks, im Mittel eine Zeitdauer von $t_i$. Zur Einhaltung einer mittleren Dauer $t_i$ wird die Schwierigkeit zur Lösung des kryptografischen Puzzles regelmäßig per Protokoll an die innerhalb des Systems verfügbare Rechenleistung $r_{network}$ angepasst (Narayanan und Clark 2017). Die Anpassung erfolgt durch eine Veränderung der Parameter der Nebenbedingung, etwa durch eine Verringerung oder Erhöhung von *Target*. Der folgende Abschnitt erläutert ein kryptografisches Puzzle sowie die Anpassung von *Target* für das Blockchain-System Bitcoin.

*Beispiel zum Nachweis der Terminierung des Selektionsverfahrens*

Ein häufig gewähltes kryptografisches Puzzle ist die Suche eines Funktionswertes unterhalb des Zielwertes *Target*. Für das Bitcoin-System ist das Puzzle definiert durch die rekursive Anwendung zweier Hash-Funktionen SHA-256(SHA-256$(BH)$) (Bitcoin 2018), deren Parameter BH genau dann ein Nachweis der Terminierung ist, wenn der Funktionswert maximal den Wert der Target-Variable nBits annimmt.

$$\text{SHA-256}(\text{SHA-256}(BH)) = v <= nBits$$

$BH$ (siehe Abschnitt 3.3.1.3) ist eine Konkatenation der Werte des Block-Headers des zuvor erstellten Blocks (F2) (Bitcoin 2018). Für Nonce wird zunächst ein beliebiger Wert angenommen, der beispielsweise fortlaufend inkrementiert wird.

Aufgrund der Eigenschaften kryptografischer Hash-Funktionen (siehe Abschnitt 3.2.2.2) erfolgt die Ermittlung eines gültigen $v$ durch wiederholte Neuberechnungen unter Veränderung von Nonce, bis gilt $v <= nBits$.

Eine Anpassung von Target (Bitcoin Core 2018) erfolgt nach dem Abschluss von 2016 Blöcken, deren Berechnung sich mit $t_i = nPowTargetSpacing * 1s = 600s$ auf die Dauer $nTargetTimespan = t_i * 2016 = 14 * 24 * 60 * 60s$ oder ca. 14 Tage beläuft. Ein neuer Target-Wert $nBits_{neu}$ berechnet sich aus einem bestehenden Target-Wert $nBits$ als $nBits_{neu} = nBits * nActualTimespan/nTargetTimespan$ mit $nActualTimespan$ als Zeitdifferenz des aktuellen Blocks und eines 2016 - 1 Blöcke zurückliegenden Blocks. Die Zeitdifferenz wird auf den Minimalwert 3,5 d und den Maximalwert 56 d begrenzt (Bitcoin Core 2018).

Weitergehende Betrachtungen der innerhalb des Netzwerks aufzuwendenden Rate $r_{network}$ als Anzahl von Funktionsberechnungen pro Zeit werden vor dem Hintergrund der Konsistenz in den folgenden Abschnitten aufgegriffen.

### 3.3.5 Konsistenz und Unveränderlichkeit

Blockchain-Systeme wählen hinsichtlich des CAP-Theorems (siehe Abschnitt 3.2.1.3) die Merkmale Verfügbarkeit und Partitionstoleranz. Jeder Knoten, d.h. jede Instanz einer Nodes-Software des Peer-to-Peer-Netzwerks, wählt anhand des Protokolls und des damit implementierten Consensus-Verfahrens einen konsistenten und zusammenhängenden Teil einer Blockchain aus. Die Konsistenz ist dabei nicht zu jedem Zeitpunkt sichergestellt, sondern wird als Eventual Consistency im Zeitverlauf erreicht. Die folgenden beiden Abschnitte erläutern zunächst die Herstellung der Konsistenz nach einer Partitionsbildung sowie für Fälle, in denen der Zustand des Systems im Sinne eines Soft State verändert wird. Verfügbarkeit, Soft State und Eventual Consistency sind die hier vor dem Hintergrund von BASE betrachteten Merkmale.

#### 3.3.5.1 Partitionsbildung und Eventual Consistency

Tritt nach Block $B_h$ eine Partitionierung auf, wird die Blockchain innerhalb eines jeden Teilsystems durch Anfügung neuer Blöcke an $B_h$ unterschiedlich fortgesetzt, so dass Konsistenz nicht mehr gegeben ist. Die Nachfolger von $B_h$ sind innerhalb der Teilsysteme Teil je einer Hauptkette (Main Chain) separater Blockchains, d.h. die Nachfolger sind Teil eines als gültig betrachteten, konsistenten und zusammenhängenden Teils einer Blockchain. Die Verfügbarkeit wird somit nicht beeinträchtigt. Nach einer Partitionierung bestehen beispielsweise zwei Nachfolger als $B_{h+1.1}$ und $B_{h+1.2}$ die m bzw. n Nachfolger besitzen, so dass zwei Blöcke $B_{h+m>h+1.1}$ bzw. $B_{h+n>h+1.2}$ existieren (zur Notation siehe Abschnitt 3.3.3.4). Die Blockchain entspricht damit einer Baumstruktur, in der ein Pfad zwischen dem Knoten $B_{h+m>h+1.1}$ und $B_0$, sowie ein Pfad zwischen $B_{h+n>h+1.2}$ und $B_0$ existiert. Das Protokoll wählt schließlich im Falle einer Zusammenführung der Partitionen einen Pfad nach definierten Kriterien aus, z.B. in Abhängigkeit von m und n denjenigen, für den die Pfadlänge maximal ist oder, wie etwa bei Bitcoin, denjenigen, für den die kumulierte Rechenleistung zur Erzeugung des Pfades maximal ist (Antonopoulos 2017, S. 240). Damit ist das Merkmal Eventual Consistency für Blockchain-Systeme gegeben.

#### 3.3.5.2 Soft State und Eventual Consistency

In Blockchain-Systemen, die eine Finalität von Blöcken nicht garantieren, ist der Zustand im Sinne eines Soft State prinzipiell veränderbar. Die beiden folgenden Szenarien erläutern Zustandsänderungen sowie deren Auswirkungen auf die Konsistenz,

die auch ohne die Bildung von Partitionen in beiden Fällen vorübergehend nicht garantiert werden kann.

*1. Bildung von Stale Blocks*

Ein Stale Block tritt infolge einer parallelen Anfügung von zwei Blöcken auf (siehe Abschnitt 3.3.3.4). Werden an einen Block $B_h$ der Hauptkette (Main Chain) parallel zwei unterschiedliche Blöcke $B_{h+1.1}$ und $B_{h+1.2}$ angefügt, so besteht bis zur Anfügung von $B_{h+2}$ an $B_{h+1.1}$ oder $B_{h+1.2}$ keine Konsistenz. Das Protokoll stellt Konsistenz nach der Anfügung eines Blocks her, z.B. nach der Anfügung von $B_{h+2>h+1.2}$, der Teil der Hauptkette wird. $B_{h+1.1}$ ist damit ein Stale Block. Der Zustand des Blockchain-Systems definiert sich als Zustand der Main Chin zu diesem Zeitpunkt durch $B_{h+2>h+1.2}$ sowie dessen Vorgänger. Das Protokoll sieht nun die Fortsetzung nachfolgender Blöcke ausgehend von $B_{h+2>h+1.2}$ vor.

Prinzipiell ist eine Anfügung eines Blocks an $B_{h+1.1}$ weiterhin möglich. Wird dort $B_{h+2>h+1.1}$ angefügt, ist Konsistenz erneut nicht gegeben. Eine Bildung von Stale Blocks dieser Art wird definitorisch als Chain Split behandelt.

*2. Chain Split*

Ein Chain Split tritt im Falle von mehreren parallel angefügten Blöcken einer Hauptkette mit jeweils mindestens einem Nachfolger auf (siehe Abschnitt 3.3.3.4). Werden an $B_h$ die Blöcke $B_{h+1.1}$ und $B_{h+1.2}$ sowie $B_{h+2>h+1.2}$ und $B_{h+3>h+1.2}$ angefügt, bestehen ausgehend von $B_h$ n = 3 Nachfolger der Hauptkette. In diesem Fall ist z.B. weiterhin eine Anfügung von $B_{h+2>h+1.1}$ möglich, die in $B_{h+3>h+1.1}$ und $B_{h+4>h+1.1}$ fortgesetzt wird. Der Zustand definiert sich in diesem Fall durch $B_{h+4>h+1.1}$, sowie zudem durch dessen Vorgänger. Die in $B_{h+1.2}$, $B_{h+2>h+1.2}$ und $B_{h+3>h+1.2}$ enthaltenen Daten beeinflussen den Zustand nicht mehr. Zustandsänderungen dieser Art sind somit ohne eine garantierte Finalität von Blöcken prinzipiell möglich. In Abhängigkeit des Protokolls ist für Proof-of-Work-Consensus-Verfahren eine Durchführung der Zustandsänderung ausgehend von Blöcken vor dem letzten Block der Hauptkette allerdings mit der Aufwendung unverhältnismäßig hoher Rechenleistung verbunden, die nun betrachtet wird.

#### 3.3.5.3 Unveränderlichkeit

Im Falle des verbreiteten Proof-of-Work-Consensus-Verfahrens (siehe Abschnitt 3.3.4) kann eine Änderung des Zustandes entsprechend der vorhergehenden Diskussion nur unter der Aufwendung von Rechenleistung zur Berechnung von Hash-Operationen erfolgen. Dies wirkt sich auf das Merkmal der Unveränderlichkeit sowie auf das davon abhängige Merkmal Trustless aus.

Zur Anfügung eines Blocks ist die Berechnung von Hash-Operationen mit der Rate des Netzwerks $r_{network}$ [H/s] erforderlich. Die erforderliche Rate $r_z$ [H/s], die für die Herbeiführung einer Zustandsänderung ausgehend von einem Block $B_h$ einer Hauptkette (Main Chain) mit dem letzten Block $B_{h+n}$ notwendig ist, kann in Abhängigkeit von $r_{network}$ angegeben werden. Zur Veränderung des Zustandes werden n+1 Blöcke benötigt, so dass $r_z = (n+1) * r_{network}$ gilt.

Für eine annähernd unveränderliche Speicherung von Daten muss $r_z$ eine zur Herbeiführung einer Zustandsänderung global bereitstellbare Rate $r_g$ [H/s] übersteigen:

$$r_g < (n+1) * r_{network} \text{ mit } n \geq 0, n \in \mathbb{N}$$

Dieser Zusammenhang gilt, solange die durch den Target-Wert determinierte Rate für das Anfügen eines Blocks konstant ist. Dies ist beispielsweise für Bitcoin über je 2016 Blöcke der Fall (siehe Abschnitt 3.3.4). In Relation stehen zwei Raten, d.h. effektiv die Rechenleistung als Anzahl an Hash-Operationen. Für die Herbeiführung einer Zustandsänderung unter Verletzung der Ungleichung muss für $n = 0$ mindestens die Rechenleistung des gesamten Netzwerks aufgewendet werden, sowie für $n > 0$ entsprechend des linearen Zusammenhangs $r_g$, $n$ ein Vielfaches der Rechenleistung. Bei Betrachtung der unteren Schranke $n = 0$ wird die Ungleichung verletzt, wenn nach einer Bereitstellung von $r_g$ der Anteil von $r_g$ an der Rate des Netzwerks $r_{network_post}$ über 0.5 liegt. Eine Änderung dieser Art wird als 51%-Angriff bezeichnet (Barber et al. 2012).

In Abhängigkeit von $r_{network}$ des betrachteten Blockchain-Systems kann eine unveränderliche Speicherung von Daten in einem Block $B_h$ mit zunehmendem n im Zeitverlauf angenähert werden. Die Variable n ist daher für die Absender und Empfänger der in $B_h$ enthaltenen Transaktionen relevant (siehe Confirmations 3.3.3.4). Im Mittel erhöht sich n je $t_i$ um 1. Bei $n = 4$ und konstanter $r = r_{network}$ ergeben sich für Bitcoin die Parameter $t_i = 600s$ und $r = 47,6 * 10^{18}$ H/s (18.11.2018), so dass $r = 4 * 47,6 * 10^{18}$ H/s $= 190,4 * 10^{18}$ H/s und $r * t_i = 190,4 * 10^{18}$ H/s $* 600s \approx 114,2 * 10^{21}$ H $= 114,2$ ZH.

**Energieaufwand zur Erstellung von Blöcken**

Die für die Anfügung von Blöcken aufzuwendende Leistung $P$ ist abhängig von $r_{network}$ und der durchschnittlichen Energieeffizienz der Berechnung von Hash-Operationen $e_{network}$ [J/H] mit $P = r_{network} * e_{network}$. Zur Etablierung einer unteren Grenze kann für Bitcoin die Effizienz aktueller ASIC-basierter Rechner-Hardware mit 94,0 J/TH angenommen werden (Bitmain 2018), so dass $P = 47,6 * 10^{18}$ H/s $*$ $94,0 J/H/10^{12} \approx 4,47 * 10^9$ W $= 4,47$ GW . Innerhalb des Beispiels muss zur Herbeiführung einer Zustandsänderung von Daten in $B_h$ zu einem Zeitpunkt mit $n = 4$ mindestens die Energie $(114 * 10^{21} * 94) * 10^{-12} * 3600^{-1}$ Wh $\approx 3,0$ GWh aufgewendet werden.

Eine mit zunehmendem n annähernd unveränderliche Speicherung wird durch $r_{network}$ determiniert. Die Rate wird dabei von externen Faktoren beeinflusst. Hierzu gehören mindestens die Kosten der Energie, der Hardware, Produktionsfaktoren von ASIC-Chips und der Fertigung von „Mining-Hardware“, Faktoren der Marktentwicklung und rechtliche Rahmenbedingungen.

#### 3.3.5.4 ACID- und BASE-Eigenschaften

Die ACID-Eigenschaften (siehe Abschnitt 3.2.1.3) beziehen sich auf Transaktionen sowie eine von diesen veränderte Datenbasis. Blockchain-Transaktionen stellen hingegen Nachrichten dar, die unter bestimmten Bedingungen in die Datenstruktur der Blockchain einbezogen werden. Nach dem Absenden einer Blockchain-Transaktion kann die Aufnahme der Transaktion in einen Block anhand des Verfahrens zur Blockerstellung nicht garantiert werden, z.B. sofern keine freien Kapazitäten bestehen. Dabei besteht keine zeitliche Obergrenze, die eine Nichtausführung definiert. Die Atomarität (1.) einer solchen Transaktion ist dennoch gegeben, da Teilausführungen nicht möglich sind. Zur Erfüllung der weiteren drei Eigenschaften müssen Annahmen getroffen werden, d.h., diese können nicht garantiert werden. Sofern für ein betrachtetes Blockchain-System eine annähernd unveränderliche Speicherung nach n Blöcken in Abhängigkeit von $r_{network}$ angenommen wird, treffen für die in Blöcken aufgenommenen Transaktionen die Eigenschaften Isolation gegenüber anderen Transaktionen (3.) sowie Dauerhaftigkeit der Daten (4.) zu. Die Eigenschaft (2.), der Überführung der Datenbasis in einen konsistenten Zustand, kann bei einer Bildung von Stale Blocks und für Chain Splits nicht gewährleistet werden.

Hinsichtlich des Konsistenzbegriffs des CAP-Theorems bestehen die in den vorhergehenden Abschnitten diskutierten Einschränkungen während der Bildung von Partition, die durch die Priorisierung der Verfügbarkeit gegenüber der Konsistenz

bedingt sind. Ein Blockchain-System erfüllt in Bezug auf die BASE-Eigenschaften damit Basically Available (1.). Weiterhin bedingen die diskutierten Zustandsänderungen die Eigenschaft Soft State (2.) und die Herstellung eines konsistenten Zustandes im Zeitverlauf als Eventual Consistent (3.). Ein Blockchain-System ist anhand von BASE charakterisierbar.

### 3.3.6 Limitationen zur Skalierbarkeit

Die Ablage von Daten in Blockchain-Systemen unterliegt Limitationen, die sich durch die Ablage aller Daten in einer Blockchain als „Single Point of Truth“ ergeben. Die Verteilung der Datenstruktur als Replikation führt zu Einschränkungen hinsichtlich der Skalierbarkeit von Blockchain-Systemen.

Der Durchsatz eines Blockchain-Systems ist systemspezifisch von unterschiedlichen Parametern abhängig. Faktoren für die Bestimmung des maximalen Durchsatzes sind (1.) das Block-Intervall als mittleres Zeitintervall $t_i$ zwischen zwei Blöcken, das direkt oder indirekt durch das Protokoll vorgegeben ist, (2.) die Block-Größe $b_B$, die durch das Protokoll auf einen Maximalwert festgelegt werden kann sowie (3.) die Datenstruktur eines Blocks, die für das Anfügen von Blöcken prozedural durch das Protokoll spezifiziert ist. In Abhängigkeit von (3.) ist (2.) nicht notwendigerweise konstant.

*Block-Intervall und Block-Größe*

Für das Blockchain-System Bitcoin besteht bis September 2017 eine Limitierung von $b_B$ = 1 MB je Block. Nach der Segregated-Witness-Erweiterung der Datenstruktur (3.) ist $b_B$ durch das Protokoll limitiert, allerdings nicht mehr als konstanter Wert $b_B$, sondern in Abhängigkeit einer Gewichtung, die zu Blöcken bis knapp unter 4 MB führen kann (Lombrozo et al. 2018). Vom 01.01.2017 bis zum 25.11.2018 liegt die maximal erreichte Blockgröße bei 1,22 MB (22.11.2018) (Blockchain.com 2018). Hinsichtlich des Block-Intervalls wird der Mittelwert $t_i = 600$ s angestrebt. Rechnerisch ergibt sich unter Annahme des Block-Intervalls $t_i$ für den Zeitpunkt des Maximums ein Durchsatz von 1220 KB/600 s $\approx 2,0$ KB/s.

Für das Blockchain-System Ethereum besteht keine Limitierung von $b_B$. Die mögliche Größe wird aufgrund von (3.) durch Transaktionskosten der Einheit Gas beschränkt, die mit jeder Transaktion abgeführt werden (Wood 2014). Vom 01.01.2017 bis zum 25.11.2018 liegt die maximal erreichte Blockgröße bei 33,684 KB (04.01.2018)

(Etherchain.org 2018). Dabei gilt im Mittel $t_i = 14$ s (Etherchain.org 2018). Rechnerisch ergibt sich unter Annahme des Block-Intervalls $t_i$ für den Zeitpunkt des Maximums ein Durchsatz von $33,684$ KB$/14$ s $\approx 2,4$ KB/s.

*Transaktionsdurchsatz*

Eine Kennzahl zur Abbildung der zuvor diskutierten Faktoren ist die Anzahl möglicher Transaktionen pro Zeiteinheit. Abbildung 3.9 zeigt eine Anzahl von Transaktionen je Tag für das Blockchain-System Bitcoin (Blockchain.com 2018). Werte der Kennzahl Transaktionen pro Sekunde (TPS) bewegen sich, bezogen auf Intervalle von 24 h, zwischen $490644/24/3600 \approx 5,7$ TPS (14.12.2017) und $131875/24/3600 \approx 1,5$ (01.08.2017) TPS. Eine Bitcoin-Transaktion kann dabei an mehrere Empfänger adressiert sein und mehrere Transfers durchführen. Die seit der Segregated-Witness-Erweiterung erhöhte Block-Größe wird derzeit nicht ausgeschöpft, so dass höhere TPS-Werte erreichbar sind.

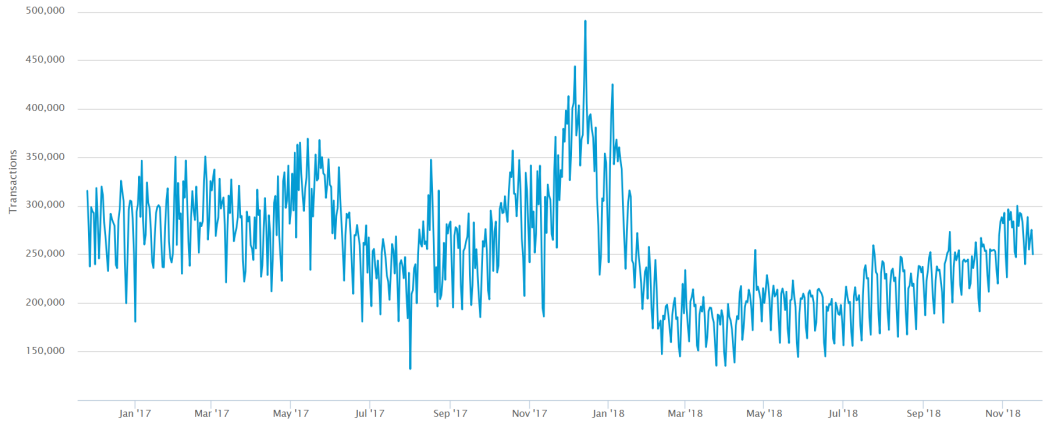



ABBILDUNG 3.9: Transaktionen pro Tag für die Bitcoin-Blockchain (Blockchain.com 2018)

Für das Blockchain-System Ethereum zeigt Abbildung 3.10 die Anzahl von Transaktionen je Tag (Etherchain.org 2018). Die Kennzahlwerte der TPS bewegen sich, bezogen auf Intervalle von 24 h, zwischen $1351307/24/3600 \approx 15,6$ TPS (04.01.2018) und $38730/24/3600 \approx 0,4$ (01.01.2017) TPS. Eine Ethereum-Transaktion ist an genau einen Empfänger adressiert. In den TPS-Wert fließen Währungstransfers sowie Erstellungen und Aufrufe von Smart-Contracts ein. Aufgrund der nicht-konstanten maximalen Block-Größe kann im Falle einer höheren Auslastung eine Steigerung der TPS erwartet werden (siehe vorheriger Abschnitt).

Weitere externe Faktoren existieren, wie z.B. die Netzwerk-Latenz und der Netzwerk-Durchsatz. Mit einer zunehmenden Latenz und mit abnehmendem

ABBILDUNG 3.10: Transaktionen pro Tag für die Ethereum-Blockchain (Etherchain.org 2018)

Durchsatz steigt die Wahrscheinlichkeit der Bildung von Stale Blocks (siehe Abschnitt 3.3.5). Zudem nimmt mit zunehmender Größe der Blockchain bei unveränderter Incentivierung die Anzahl der Nodes im Netzwerk ab, die diese vollständig speichern. Die Größe der Bitcoin-Blockchain beträgt am 24.11.2018 192,0 GB (Blockchain.com 2018), die der Ethereum-Blockchain umfasst in Abhängigkeit der Archivierung der Komponenten ein bis zwei TB. Sofern lokal keine vollständige Historie der Zustandsspeicher erforderlich ist, kann die Größe unter Beibehaltung aller Transaktionen auf 100-200 GB reduziert werden (ETH Statistics 2018; Härer und Fill 2019b; Parity 2018a). Ansätze zur Skalierung von Blockchain-Systemen bespricht der nachfolgende Abschnitt.

Neuere und weniger verbreitete Blockchain-Systeme basieren auf effizienteren Consensus-Algorithmen und neueren Architekturen, die prinzipiell höhere TPS zulassen (siehe z.B. Androulaki et al. 2018). Ein grundlegendes Problem der TPS-Metrik sind mitunter fehlende Evaluierungsmöglichkeiten, etwa aufgrund von geringer Netzwerkauslastung oder externen Faktoren (siehe z.B. L. M. Bach et al. 2018). Wenig verbreitete Systeme können zudem geringere Konsistenzgarantien hinsichtlich $r_{network}$ aufweisen (siehe Abschnitt 3.3.5).

#### 3.3.6.1 Ansätze zur Skalierung von Blockchain-Systemen

Die Blockchain-Systeme Bitcoin und Ethereum verfolgen eine Reihe von Projekten zur Untersuchung der Skalierbarkeit. Die vorgeschlagenen Ansätze gliedern sich in On-Chain- und Off-Chain-Ansätze, die eine Skalierung innerhalb bzw. außerhalb der jeweiligen Blockchain vorschlagen.

**On-Chain-Ansätze**

Für Ethereum werden On-Chain-Sharding-Ansätze diskutiert, die eine Aufteilung der Blockchain in *Shards* über mehrere Gruppen von Teilnehmern hinweg vorsehen. Die Blockchain liegt damit nicht mehr repliziert bei einzelnen Teilnehmern vor. Ein Beispiel ist Teil der sich entwickelnden Spezifikation des Systems Ethereum 2.0 (Ethereum 2019a), die nach einer initialen Betriebsphase (Beacon Chain) eine Erweiterung hin zu verteilten Shard Chains plant. Dabei besteht die Beacon Chain als zentrale Infrastruktur, welche die Zustandskonsistenz der Shard Chain in regelmäßigen Abständen anhand von auf Merkle-Bäumen basierenden Nachweisen absichert.

**Off-Chain-Ansätze**

Ein Off-Chain-Ansatz für Ethereum ist das Plasma Framework (Poon und Buterin 2017). Zur Abwicklung von Off-Chain-Transaktionen erfolgt ein Transfer von Ether-Einheiten zu einem Smart Contract. Der Betrag steht ab diesem Zeitpunkt für die Durchführung von Off-Chain-Transaktionen zur Verfügung. Die Transaktionshistorie der Off-Chain-Transaktionen wird als sog. Sidechain separat verwaltet, die von beliebigen Anbietern initiiert und betrieben werden, z.B. von sozialen Netzwerken. Innerhalb einer Sidechain ist die reguläre Durchführung von Transaktionen möglich. Das Verlassen der Sidechain ist in allen Fällen möglich, da der ursächlich genutzte Smart Contract Ausbuchungen zu jedem Zeitpunkt anhand von Bonded-Fraud-Proofs (Poon und Buterin 2017, S. 4–6) ermöglicht. Die Bildung weiterer Sidechains ist durch Rekursion möglich, die zu einer Hierarchie von Sidechains führt. Der Zustand von untergeordneten Sidechains wird in regelmäßigen Abständen zu nächsthöheren Chain zurückgeführt. Weitere Optimierungen wie ein Proof-of-Stake-basiertes Consensus-Protokoll und Merged Mining zur Nutzung der vorhandenen Infrastruktur sind Bestandteil des Vorschlags.

*Second-Layer-Ansätze*

Als Off-Chain-Ansatz wird die Einführung einer weiteren Abstraktionsschicht als „Second Layer“ diskutiert, die für Bitcoin anhand des Lightning-Protokolls (Poon und Dryja 2016) implementiert ist und erprobt wird. Ethereum verfolgt diesen Ansatz anhand des Raiden-Projekts (Raiden 2018).

Ein „Second Layer“ bildet nicht jede Transaktion zwischen Teilnehmern eines Netzwerks als Blockchain-Transaktion ab, sondern führt mehrere Transaktionen zwischen mehreren Teilnehmern zunächst „Off-Chain“ durch. Nach einer Reihe gegenläufiger Transaktionen ist im Falle von Währungstransfers ausschließlich der Ausgleich ausstehender Beträge notwendig. Dabei werden durch das Routing von

Lightning-Transaktionen innerhalb des Netzwerks mehr als je zwei Teilnehmer in eine Transaktion einbezogen. Eine Transaktion etabliert einen Payment-Channel über mehrere Teilnehmer hinweg, der in Analogie zu einem Kommunikationskanal einer Ende-zu-Ende-Verbindung das Routing innerhalb des Netzwerks abstrahiert. Ein Kanal kann zu einem späteren Zeitpunkt unter Ausgleich ausstehender Beträge geschlossen werden. Im Falle eines vorzeitigen Schließens oder einer ausbleibenden Reaktion eines Teilnehmers werden nach einem Timeout, angegeben durch nLockTime als Anzahl von Blöcken (Antonopoulos 2017), automatisierte Rückzahlungen und Disincentivierungsmechanismen ausgelöst. Durch die Off-Chain-Zahlungsabwicklung werden hohe Transaktionsraten möglich, wobei die Komplexität hinsichtlich der Anforderungen der Disincentivierungsmechanismen mögliche Anwendungsfälle gegenüber On-Chain-Zahlungen einschränkt.

## 3.4 Anwendungsklassen von Blockchain-Systemen

### 3.4.1 Anwendungen von Blockchain- und Smart-Contract-Systemen

Blockchain-Systeme sind aufgrund des Merkmals Trustless insbesondere für Anwendungen geeignet, die auf direkten vertrauenswürdigen Beziehungen zwischen mehreren Beteiligten beruhen (siehe Abschnitt 3.1.1), ohne eine Intermediation durch eine Trusted-Third-Party (Kaulartz und Heckmann 2016; Wiefling et al. 2017). Für Blockchain-Systeme ergeben sich ausgehend von virtuellen Währungen für den Transfer von Geld- oder Werteinheiten eine Reihe von aufbauenden Anwendungen zur Durchführung von beliebigen Transaktionen auf der Basis von Daten. Smart-Contract-Systeme erlauben die Umsetzung weitergehender Anwendungen, die neben Daten ausführbaren Smart-Contract-Code umfassen. Die Anwendungen werden in Tabelle 3.3 klassifiziert.

**Kriterien**

Die Klassifikation betrachtet die Kriterien *Integrität*, *Systemimmanente Vertrauensbeziehung*, *Beziehungstyp* und *Öffentlichkeit*. Das Kriterium *Integrität* fordert eine nichtmodifizierbare Speicherung von Daten. Dabei wird prinzipiell der Einsatz von dezentralen oder partiell dezentralen Blockchain-Systemen angenommen (siehe Abschnitt 3.1.2.4). Eine *Systemimmanente Vertrauensbeziehung* wird unterstellt, sofern die auf Vertrauen beruhenden Beziehungen zwischen Teilnehmern vollständig innerhalb des Systems abgebildet werden. Bei Interaktionen mit physischen Objekten trifft das Kriterium nicht zu. Bei Interaktionen mit nicht-physischen Objekten, z.B. Daten oder Software, trifft das Kriterium zu, sofern die beteiligten nichtphysischen Objekte Teil des Systems sind. Der *Beziehungstyp* unterscheidet zwischen 1:n-Beziehungen, die zwischen einem Bezugspunkt und mehreren Teilnehmern bestehen, sowie n:n-Beziehungen, die zwischen mehreren Teilnehmern erbracht werden. Eine 1:n-Beziehung besteht z.B. durch das Angebot eines Dienstes gegenüber mehreren Teilnehmern. Eine n:n-Beziehung trifft etwa für Transfers zwischen beliebig vielen Teilnehmern zu. *Öffentlichkeit* beschreibt die Anforderung, die Daten einer Anwendung öffentlich abrufbar bereitzustellen.

| | | Anwendungsklasse | | Merkmal | | | |
|---|---|---|---|---|---|---|---|
| | | Blockchain | Smart Contract | Integrität | Systemimmanente Vertrauensbeziehung | Beziehungstyp | Öffentlichkeit |
| **Werttransfer** | **Transfer quantifizierbarer Währungseinheiten** | D-1 | S-1 | ● | ● | n:n | ○ |
| | **Transfer quantifizierbarer Tokens** | D-2 | S-2 | ● | ● | n:n | ○ |
| **Identität** | **Registrierung und Tracking von Datenobjekten** | D-3 | S-3 | ● | ● | 1:n | ◐ (ausgewählter Personenkreis) |
| | **Beglaubigung von Datenobjekten** | ○ | S-4 | ● | ● | 1:n | ◐ (ausgewählter Personenkreis) |
| **Bereitstellung von Gütern** | **Materielle Objekte als Smart Property** | ○ | S-5 | ● | ○ | 1:n | ○ |
| | **Immaterielle Objekte** | ○ | S-6 | ● | ● | 1:n | ○ |
| **Bereitstellung von Dienstleistungen** | **Cloud-Computing-Ressourcen** | ○ | S-7 | ● | ● | 1:n (nicht-austauschbar) / n:n (austauschbar) | ○ |
| **Märkte** | **Handelsplattformen** | ○ | S-8 | ● | ◐ (nicht zutreffend für Oracle, sonst zutreffend) | n:n | ○ |
| | **Prognosemärkte** | ○ | S-9 | ● | ◐ (nicht zutreffend für Oracle, sonst zutreffend) | n:n | ○ |
| **Organisationen** | **DAO** | ○ | S-10 | ● | ◐ (zutreffend bei vollständiger Autonomie, sonst nicht zutreffend) | 1:n | ○ |

● zutreffend ◐ teilweise zutreffend (siehe Erläuterung) ○ nicht zutreffend

TABELLE 3.3: Klassifikation von Anwendungen

#### 3.4.1.1 Blockchain-Systeme

Eine erste Klasse bilden Anwendungen für Blockchain-Systeme, die allein die Funktion einer verteilten Transaktionshistorie realisieren, d.h. die Speicherung von Transaktionen in sequenziell geordneten Blöcken, die eine Transaktionshistorie bilden. Anwendungen dieser Klasse umfassen:

(D-1) **Transfer quantifizierbarer Währungseinheiten** anhand von virtuellen Währungen (Chohan 2018; CoinMarketCap 2018; Nakamoto 2008a),

(D-2) **Transfer quantifizierbarer Tokens**, die (a.) Wertrepräsentationen von Gütern, Wertpapieren oder anderen Objekten als Einheiten von Asset-Tokens bzw. Wertpapier-Tokens (auch Security-Tokens) abbilden, oder (b.) quantifizierbare Einheiten ohne Wertrepräsentation als „Utility Tokens“ abbilden (Notheisen et al. 2017; Oliveira et al. 2018). Der Transfer von Währungseinheiten ist als Spezialisierung dieser Klasse abbildbar, stellt historisch allerdings eine eigene Anwendung dar (Nakamoto 2008a).

(D-3) **Registrierung und Tracking von Datenobjekten** zur Repräsentation identifizierbarer Dokumente, digitaler Identitäten oder beliebiger anderer Datenobjekte (Lemieux 2016), z.B. Güter in einer Supply Chain (Baruffaldi und Sternberg 2018), Grundbucheinträge (Barbieri 2017), Einträge des Domain Name Systems (Benshoof et al. 2016; Namecoin 2018) oder verschlüsselt abgelegte, persönliche Daten (Zyskind et al. 2015). Datenobjekte werden einmalig (Registrierung) oder periodisch (Tracking) in Transaktionen hinterlegt, die zu einem späteren Zeitpunkt abrufbar sind und die Existenz des Datenobjekts zu einem Zeitpunkt bestätigen. Beispielsweise anhand eines Hash-Wertes in einer Transaktion, der die Existenz eines Dokuments zum Zeitpunkt der Transaktion, d.h. zum Zeitpunkt des Blocks, beglaubigt (Härer und Fill 2019b; OpenTimestamps 2018).

#### 3.4.1.2 Smart-Contract-Systeme

Eine zweite Klasse bilden Anwendungen für Smart-Contract-(Blockchain-)Systeme, die neben einer verteilten Transaktionshistorie die Funktion einer Plattform für die Ausführung von Smart-Contracts darstellen. Transaktionen können zusätzlich ausführbaren Contract-Code beinhalten, der durch weitere Transaktionen innerhalb des Blockchain-Systems verteilt zur Ausführung kommt. Die auf Smart Contracts beruhende Software einzelner Anwendungen wird auch als DApp (Decentralized Application) (Swan 2015) bezeichnet.

(S-1) **Transfer quantifizierbarer Währungseinheiten** analog zu Blockchain-Systemen.

(S-2) **Transfer quantifizierbarer Token-Objekte** analog zu Blockchain-Systemen.

(S-3) **Registrierung von Datenobjekten** analog zu Blockchain-Systemen.

(S-4) **Beglaubigung von Datenobjekten** (Lemieux 2016), die identifizierbare Dokumente, digitale Identitäten oder beliebige andere Datenobjekte einmalig oder periodisch registrieren, z.B. durch einen Hash-Wert, um damit deren Existenz und ggf. Metadaten zu einem späteren Zeitpunkt nachweisen zu können. Gegenüber der Anwendung „Registrierung und Tracking von Datenobjekten“ in Blockchain-Systemen können die Registrierung und die Validierung explizit erfasst, über mehrere Perioden gespeichert und mit der Ausführung von Code verknüpft werden. Ein Smart Contract erlaubt neben Registrierung und Tracking eine interaktive Überprüfung von Identitäten anhand von öffentlichen Schlüsseln.

(S-5) **Materielle Objekte als Smart Property**, das die Funktion und das Verhalten materieller Objekte von Smart Contracts abhängig macht, um z.B. den Zugang zu einem Fahrzeug nur für den Eigentümer zu erlauben oder, um den Zugang von Transaktionen abhängig zu machen (Szabo 1994). Prinzipiell ist eine Verknüpfung mit beliebigen IoT-Devices möglich.

(S-6) **Immaterielle Objekte**, d.h. der Verkauf oder die Vermietung nicht-physischer Objekte, wie z.B. Lizenzrechte, Dokumente, Musik- und Video-Streams (Fujimura et al. 2015; Sitonio und Nucciarelli 2018), Spiele und nicht-fungible Objekte mit Seltenheitswert (Digital Collectibles) (CryptoKitties 2018).

(S-7) **Cloud-Computing-Ressourcen** wie Infrastruktur, Plattformen oder Software, die anhand von nutzungsbasierten Abrechnungsmodellen vertrieben werden. Ressourcen dieser Art können z.B. als Grundlage für dezentrale Webseiten und Webdienste herangezogen werden. Beispiele für Infrastruktur sind Rechenzeit, Speicherplatz oder virtuelle Maschinen. Darauf aufbauend ist eine Einbindung von Cloud-Plattformen zur Entwicklung und Ausführung von Software sowie von Software-as-a-Service möglich. Austauschbare Ressourcen besitzen den Beziehungstyp n:n, da gegenseitige Angebots- und Abnahmebeziehungen zwischen beliebig vielen Teilnehmern bestehen können (Filecoin 2014). Nicht-austauschbare Ressourcen, z.B. Bereitstellungen einer Software-Instanz, gehen stets von einer Instanz aus und besitzen den Beziehungstyp 1:n.

(S-8) **Handelsplattformen**, die in Smart Contracts implementiert sind, handeln quantifizierbare Währungseinheiten, quantifizierbare Token-Einheiten (EtherDelta 2018) oder beliebige andere Datenobjekte. Ein Datenobjekt stellt beispielsweise Assets oder Leistungen von Zulieferer-Abnehmer-Beziehungen in Business-to-Business-Geschäftsbeziehungen dar. Datenobjekte sind entweder systemintern hinterlegt oder werden systemextern von sog. Oracles bereitgestellt (Kaulartz und Heckmann 2016).

(S-9) **Prognosemärkte** verbinden Handelsplattformen mit Prognosen. Die Preisbildung des Handels erfolgt basierend auf Prognosen der systeminternen Teilnehmer oder basierend auf Angaben von systemexternen Oracles (Augur 2018; Clark et al. 2014; Eskandari et al. 2017).

(S-10) **Dezentrale autonome Organisationen (DAO)** (Aragon 2018; Swan 2015). Ein DAO ist eine Organisation oder Unternehmung, die in Smart Contracts implementiert ist. Die Organisation kann im Sinne eines Investmentfonds agieren,

dessen Anteilseigner Währungseinheiten zur Verwaltung durch den Smart Contract einzahlen und über dessen Verwendung abstimmen. Zudem sind über Investments hinausgehende Entscheidungen durch Abstimmungsverfahren des Smart Contracts bestimmbar. Menschliche Akteure können als Anteilseigner und an Abstimmungsverfahren beteiligt sein. Eine als Investmentfonds ausgelegte Implementierung scheiterte aufgrund von Bugs innerhalb des Smart-Contract-Codes (Dupont 2017). Prinzipiell ist eine DAO unter vollständiger Autonomie realisierbar, d.h. ohne die Beteiligung von menschlichen Akteuren. Nach einmaliger Initialisierung können beispielsweise Anwendungen wie (S-7) betrieben werden, um Infrastruktur über öffentlich auffindbare APIs anzumieten und zu vertreiben.

### 3.4.2 Integration anhand von Blockchain-Systemen

Die Integration von Blockchain-Systemen als Bestandteil von Informationssystemen greift auf etablierte Integrationsarchitekturen zurück. Ein Bedarf nach Integration entsteht stets dann, wenn die Komponenten mehrerer betrieblicher Aufgaben überlappen und gleichzeitig verschiedenen Anwendungssystemen oder personellen Aufgabenträgern zugeordnet sind (Ferstl und Sinz 2013, 237 ff.). Mögliche Überlappungen beziehen sich auf die Komponente Aufgabenobjekt oder die Komponenten des Lösungsverfahrens, d.h. Aktion, Aktionensteuerung sowie deren Beziehungen. Die hiermit auf der Aufgabenebene identifizierten Integrationsbedarfe ziehen eine Integration der Aufgabenträgerebene nach sich. Gegenstand der Integration sind damit die Typen und Instanzen der Aufgaben sowie die Typen und Instanzen der ihnen zugeordneten Aufgabenträger. Die Kopplung mehrerer Systemkomponenten wird durch eine Reihe von Integrationskonzepten der Funktionsintegration, der Datenintegration sowie der Objektintegration beschrieben (Ferstl und Sinz 2013, S. 251).

- **Funktionsintegration** sieht Datenflüsse zwischen den Komponenten des Systems vor. Jede Komponente umfasst einen Speicher sowie eine Reihe von Funktionen, deren Ein- und Ausgänge die zu übertragenden Daten determinieren. Eine Durchführung von Aufgaben basiert je Komponente auf lokal vorliegenden Daten und Funktionen. Hinsichtlich des Gesamtsystems wird daher Daten- und Funktionsredundanz angenommen. Reihenfolgebeziehungen zwischen Aufgaben sind unmittelbar auf Datenflüsse abbildbar.

- **Datenintegration** sieht einen gemeinsamen Speicher für alle Komponenten des Systems vor. Jede Komponente greift anhand einer Sicht auf einen Ausschnitt der zentralisierten Datenbasis zu. Eine Durchführung von Aufgaben erfordert Zugriff auf den gemeinsamen Speicher sowie auf lokal je Komponenten vorliegende Funktionen, die hinsichtlich des Gesamtsystems zu einer Funktionsredundanz führen. Neben der Abbildung von Reihenfolgebeziehungen sind partiell gleiche Typen und Instanzen von Aufgabenobjekten innerhalb des gemeinsamen Speichers abbildbar.
- **Objektintegration** betrachtet die beteiligten Systemkomponenten als Objekte im Sinne der Objektorientierung, die Informationen über ein globales Kommunikationssystem austauschen. Objekte kapseln Zustand und Verhalten, sind lose gekoppelt und kommunizieren anhand von Nachrichten. Reihenfolgebeziehungen definieren sich durch Nachrichten zwischen Objekten des Typs Vorgangsobjekttyp (siehe 2.2.4). Partiell gleiche Typen und Instanzen von Aufgabenobjekten sind auf den Zustand gemeinsam zugreifbarer Objekte des Typs konzeptueller Objekttyp (siehe 2.2.4) abbildbar, deren Zustand durch die Anpassung von Attributen verändert wird. Partiell gleiche Lösungsverfahren sind auf das Verhalten von gemeinsam zugreifbaren konzeptuellen Objekttypen oder Vorgangsobjekttypen abbildbar, d.h. auf Operatoren oder Methoden einer Klasse. Damit besteht weder Daten- noch Funktionsredundanz.

#### 3.4.2.1 Integrationskonzepte für Blockchain-Systeme

Für Blockchain-Systeme und darauf aufbauende Smart-Contract-Systeme lassen sich zwei Integrationskonzepte unterscheiden. Ein erstes und grundlegendes Integrationskonzept für beide Systemtypen folgt der Datenintegration. Smart-Contract-Systeme erlauben zudem ein auf Smart Contracts beruhendes Integrationskonzept, das eine Kombination aus Datenintegration und Objektintegration darstellt. Beide Formen sind in den folgenden Abbildungen 3.11 und 3.12 dargestellt.

#### 3.4.2.2 Integration durch Transaktionen in Blockchain-Systemen

Der Architektur von Blockchain-Systemen folgend, sieht das System einen zentralisierten Speicher Blockchain als gemeinsame Datenbasis vor. Die Datenbasis stellt einen Single-Point-of-Truth dar. Jeder Teilnehmer erhält eine Sicht auf das System, welche die Adressen des Teilnehmers mit allen ein- und ausgehenden Transaktionen aggregiert. Transaktionen transferieren Werte oder beliebige Daten, die schließlich unter der Adresse einer Identität persistent in einem Block gespeichert werden. Beispielsweise bildet sich eine Sicht zur Anzeige des Wertes oder Guthabens einer Adresse durch eine Aggregation aller ein- und ausgehenden Transaktionen. D.h. eine Aggregationsfunktion einer Adresse $a$ bestimmt die Differenz $Wert_a = Inputs_a - Outputs_a$ mit $Inputs$ als Summe aller Outputs eingehender Transaktionen für Empfänger $a$ sowie mit $Outputs$ als Summe aller Inputs ausgehender Transaktionen von Absender $a$. Die Sicht entspricht im Falle von Währungstransfers der Anzeige eines Kontostandes in einer Wallet-Anwendung.

Das Integrationskonzept entspricht der Datenintegration. Reihenfolgebeziehungen sind in Form von einzelnen Transaktionen abbildbar, die für alle Teilnehmer lesend zugreifbar sind. Typen und Instanzen von partiell gleichen Aufgabenobjekten lassen sich auf Daten von Transaktionen abbilden, die Teil des gemeinsamen

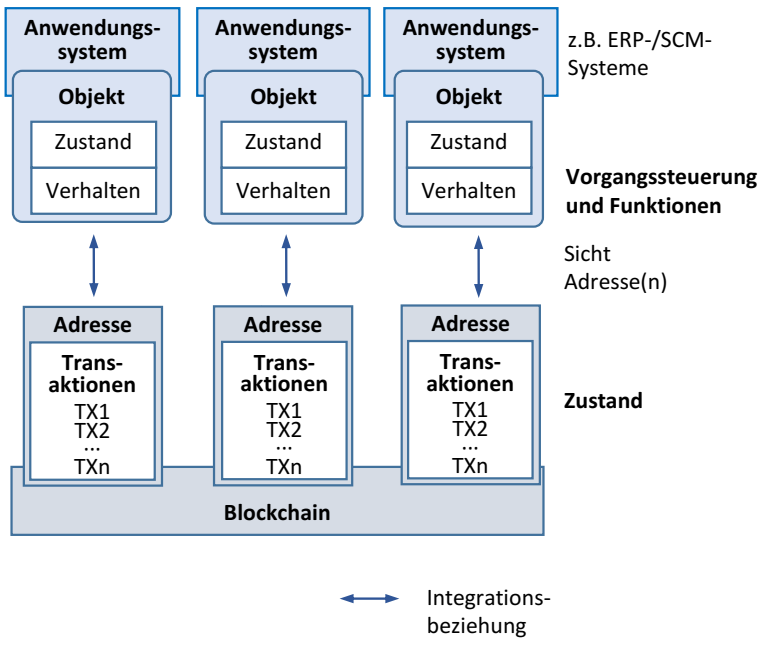



ABBILDUNG 3.11: Integration anhand von Transaktionen

Speichers werden. Eine Vorgangssteuerung oder gemeinsam zugreifbare Funktionen werden nicht unterstützt. Jeder Teilnehmer unterhält eigene Anwendungssysteme und stellt dort Funktionen lokal zur Verfügung. Die lokale Verarbeitung führt zu einer Funktionsredundanz. Datenredundanz besteht aufgrund des gemeinsamen Speichers nicht. Die Abbildung von Reihenfolgebeziehungen ist Bestandteil des in Kapitel 4 beschriebenen Ansatzes.

#### 3.4.2.3 Integration durch Smart Contracts in Smart-Contract-Systemen

Smart Contracts ermöglichen eine Integrationsform, die Datenintegration und Objektintegration kombiniert. Es besteht weiterhin eine Datenbasis als Single-Point-of-Truth, die sämtliche Transaktionen und Smart Contracts umfasst. Ein Smart Contract entspricht einem Programm, das als Contract Code mit Instruktionen und Zustandsvariablen anhand von Transaktionen zugreifbar ist. Jeder Teilnehmer besitzt eine Sicht auf das System, welche definierte Zustandsvariablen von Smart Contracts als Zustand bereitstellt. Der Zustand ist anhand von Transaktionen durch definierte Operationen des Contract Codes veränderbar. Mit diesem Integrationskonzept liegt hinsichtlich der Daten der Transaktionen und der in der Blockchain abgelegten Smart Contracts eine Datenintegration vor, die Daten redundanzfrei speichert.

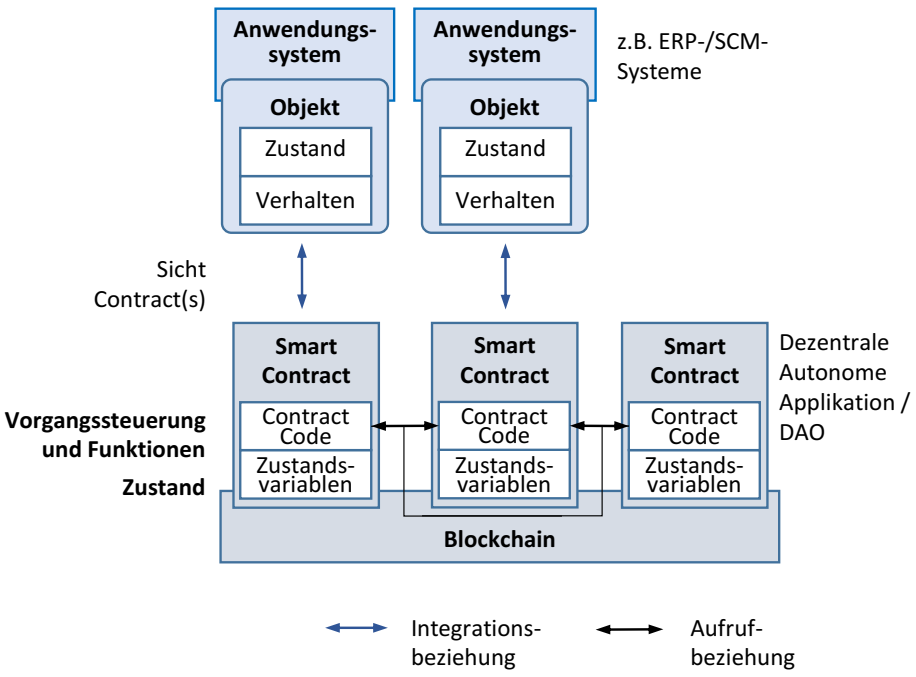



ABBILDUNG 3.12: Integration anhand von Smart Contracts

Zudem besteht eine typorientierte Kapselung von Verhalten und Zustand je Smart Contract, die Funktionen in Form von aufrufbarem Contract Code zur Definition des Verhaltens, sowie Zustandsvariablen zur Definition des Zustandes umfasst. Im Unterschied zur Objektintegration wird der Zustand des Smart Contract global und öffentlich verwaltet. Jeder Aufruf bezieht sich auf globale Zustandsvariablen und verändert diese, ohne Zustände individueller Instanzen zu unterscheiden. Einige Systeme ermöglichen Vererbungsbeziehungen zwischen Smart Contracts (Antonopoulos 2017), ohne aber Instanzen zu unterscheiden. In allen Fällen führt ein globales Kommunikationssystem Aufrufe nachrichtenbasiert durch und nimmt Zustandsänderungen stets global vor. Reihenfolgebeziehungen sind durch Nachrichten der aufrufenden Transaktionen unmittelbar abbildbar. Partiell gleiche Typen und Instanzen von Aufgabenobjekten werden anhand der Zustandsvariablen von Smart Contracts erfasst. Partiell gleiche Lösungsverfahren sind durch die in den gemeinsam zugreifbaren Smart Contracts hinterlegten Funktionen des Contract Codes realisiert. Dabei besteht keine Funktionsredundanz und keine Datenredundanz. Eine Vorgangssteuerung ist durch Aufrufbeziehungen zwischen Smart Contracts und deren Funktionen abbildbar. Spezialisierte Objekttypen zur Erfassung von konzeptuellen Objekten und Vorgangsobjekten bestehen nicht.

Ein einmal hinterlegter Smart Contract ist aufgrund der autonomen Ausführung nicht notwendigerweise auf ein bestimmtes Objekt eines betrieblichen Informationssystems zurückführbar. Eine Integration kann somit prinzipiell ausschließlich auf Smart Contracts beruhen, die eine autonome Anwendung realisieren. Sofern eine solche Anwendung betriebliche Funktionen implementiert, kann von einer dezentralen autonomen Organisation (DAO) gesprochen werden (siehe Abschnitt 3.4.1.2).

### 3.4.3 Kriterien zur Beurteilung des Einsatzes von öffentlichen und privaten Blockchain-Systemen

Basierend auf dem Merkmal Trustless schlagen Wüst und Gervais (2017) Kriterien vor, die im Folgenden als Anforderungen für den Einsatz von öffentlichen und privaten Blockchain-Systemen formuliert werden. Aufgrund des Merkmals Trustless wird der Einsatz von dezentralen oder partiell dezentralen Blockchain-Systemen angenommen. Die Erfüllung der folgenden Anforderungen kann zu einem Einsatz eines Blockchain-Systems führen.

1. **Ordnung**: Die zu speichernden Daten sind auf unterscheidbare Zustände abbildbar. Das System definiert eine Ordnung über der Menge der Zustände.
2. **Verteilung**: Schreib-Operationen werden durch mehrere Teilnehmer ausgeführt, die verteilt sind.
3. **Verbindlichkeit, Trustless**: Auf eine stets online verfügbare Trusted-Third-Party kann nicht zurückgegriffen werden.
4. **Alternative 1. Unbekannte Knotenmenge**: Die Menge der Teilnehmer ist unbekannt.
   → Einsatz einer öffentlichen Blockchain.
5. **Alternative 2. Bekannte Knotenmenge mit Erfordernis von Transparenz**: Die Menge der Teilnehmer ist bekannt und eine öffentliche Validierung ist erforderlich.
   → Einsatz einer Public Permissioned Blockchain.
6. **Alternative 3. Bekannte Knotenmenge ohne Erfordernis von Transparenz**: Die Menge der Teilnehmer ist bekannt und eine öffentliche Validierung ist nicht erforderlich.
   → Einsatz einer privaten Blockchain.

Die verteilte Speicherung von Daten unter Sicherung der Integrität und der Verbindlichkeit wird dabei noch nicht berücksichtigt. Eine weitere Anforderung ist: hohe Fehlertoleranz bei verteilter Datenhaltung unter Absicherung der Integrität. In dezentralen Blockchain-Systemen lässt die Verteilung der Daten auf alle Knoten in Abhängigkeit des Systems eine annähernd unveränderliche Speicherung zu (siehe Abschnitt 3.3.5.3).

### 3.4.4 Blockchain-Systeme in Verbindung mit Software-Systemen

Software-Anwendungen, die nicht innerhalb eines Smart Contracts implementiert werden, können Schnittstellen zu Blockchain-Systemen besitzen (Xu, Pautasso et al. 2016). Die Daten des Software-Systems werden entweder (a.) nicht innerhalb einer Blockchain gespeichert, (b.) lesbar innerhalb der Blockchain gespeichert oder (c.) lesbar und schreibbar innerhalb der Blockchain gespeichert.

(a.) Entsprechend der diskutierten Kriterien lässt sich die Integrität von Daten des Software-Systems anhand von innerhalb des Blockchain-Systems abgelegten Hash-Werten der Daten sicherstellen. Zudem kann die Existenz von Daten

durch den Integritätsnachweis öffentlich oder nicht-öffentlich nachgewiesen werden.

(b.) Die Integrität und die Informationen gespeicherter Daten sind von Teilnehmern des Blockchain-Systems lesbar und validierbar.

(c.) Die Integrität und die Informationen gespeicherter Daten sind von Teilnehmern des Blockchain-Systems lesbar und validierbar sowie durch Append-Operationen schreibbar.

Die Kopplung von Software-Systemen und Blockchain-Systemen kann insbesondere für Anwendungen erfolgen, die Beziehungen zwischen mehreren verteilten Teilnehmern herstellen. Beispiele hierfür sind Systeme für das Supply Chain Management (SCM) (Baruffaldi und Sternberg 2018), mit denen mehrere Unternehmen ein Blockchain-System für Business-to-Business-Transaktionen als Teil einer Supply Chain, z.B. zum Abschließen von Verträgen, für Procure-to-Pay-Prozesse (Linke und Strahringer 2018) oder für die Absicherung von Sensordaten einzelner Lieferbeziehungen nutzen. Die Architekturen sehen eine Blockchain als Single-Point-of-Truth vor und nutzen Software-Konnektoren, die Schnittstellen zu existierenden Systemen bilden. Mit den Ansätzen wird die Integrität der Daten syntaktisch gewährleistet, die nicht notwendigerweise Aussagen über die semantische Integrität und die Realwelt zulassen. Neben der Integrität ist daher insbesondere das Merkmal der Transparenz für Anwendungen wie SCM interessant, mit dem eine Validierung der Daten und auch der Informationen von mehreren Beteiligten möglich wird.

**Blockchain-Systeme zur Absicherung und Speicherung von Daten**

Blockchain-Systeme können neben Kommunikations- und Anwendungsfunktionen für eine verteilte Datenhaltung herangezogen werden (Lemieux 2016), um eine annähernd unveränderliche Speicherung und eine Sicherung der Integrität zu gewährleisten. Eine Möglichkeit zur Sicherung der Integrität besteht hier in der Bildung von Hash-Werten zu einzelnen Relationen, die aggregiert oder einzeln in Blockchain-Transaktionen aufgenommen werden. Die Relationen sind hinsichtlich ihrer Integrität und Verbindlichkeit, bzgl. der Identität des Transaktionsabsenders, überprüfbar. Eine Anwendung ist die Absicherung von Daten in Data-Warehouse-Systemen (Helland 2015), bei der neben der Integrität insbesondere die Unveränderlichkeit von Bedeutung für historisierte Daten ist. Für diese Anwendung bestehen in öffentlichen Blockchain-Systemen Limitation hinsichtlich der Datenmenge je Transaktion sowie der Zeitdauer zur Durchführung einer Transaktion (siehe Abschnitt 3.3.6).

### 3.4.5 Blockchain-Systeme zur Speicherung und Verarbeitung von Modellen

Die Entwicklung von Informationssystemen greift auf Modelle zurück, die das zu entwickelnde System, oder einzelne Teilsysteme, in Struktur und Verhalten abbilden. Gegenstand der Modellierung sind beispielsweise Prozesse und Workflows sowie Anwendungssysteme, d.h. Aufgaben und Aufgabenträger. Weiterhin wird der Betrieb einzelner Systeme zur Laufzeit durch Modelle unterstützt, die beispielsweise ein Monitoring einzelner Prozessinstanzen erlauben.

Die Merkmale Integrität und Verbindlichkeit von permissioned Blockchain-Systemen, sowie die Merkmale Unveränderlichkeit und Trustless von dezentralen Blockchain-Systemen, lassen sich auf die Speicherung und Verarbeitung von Modellen übertragen.

**Integrität und Verbindlichkeit** erlaubt in einer permissioned Blockchain die Erkennung und Dokumentation von autorisierten oder unautorisierten Veränderungen von Modellen unter Angabe der Identitäten der Modellierer. Anwendungen dieser Art erfordern eine Datenstruktur zur Absicherung der Integrität und der Verbindlichkeit, die z.B. auf Hash-Funktionen und Signaturen zurückgreift. Diese Anwendung bedingt nicht notwendigerweise den Einsatz eines dezentralen Blockchain-Systems, sondern kann auf private Blockchain-Systeme oder auch Datenbanksysteme zurückgreifen. Die konzeptuelle Modellierung kann damit auf Domänen angewendet werden, die eine Absicherung der Integrität der zu speichernden Informationen als Dokumentation aller Änderungen oder auch eine Verbindlichkeit der Zuordnung von Modell-Operationen zu Identitäten erfordern. Ein nachfolgend erläutertes Beispiel ist die abgesicherte Speicherung von Modellen in einer permissioned Blockchain, die als Knowledge Blockchain organisationales Wissen erfasst. Hiermit können Identitäten von Teilnehmern und Modelle erfasst werden, um Integrität und Verbindlichkeit beispielsweise für Compliance-Prüfungen gegenüber externen Prüfern nachzuweisen.

**Unveränderlichkeit und Trustless** eines dezentralen Blockchain-Systems erlauben, neben den beschriebenen Anwendungen, die Durchführung von Transaktionen zwischen verteilten Teilnehmern unter dezentraler Koordination. Das System ist permissionless und trifft keine Annahmen über die Bekanntheit der Teilnehmer.

Transaktionen von Prozessmodellen lassen sich auf Blockchain-Transaktionen abbilden, die keine vorab etablierten Beziehungen zwischen Teilnehmern voraussetzen. Mit der Anwendung auf Prozessmodelle wird etwa die Erfassung von Zulieferbeziehungen und Netzwerken von Supply Chains möglich. Eine Validierung von Transaktionen ist durch alle Teilnehmer des verteilten Systems möglich.

#### 3.4.5.1 Modelle in Permissioned Blockchains am Beispiel Knowledge Blockchain

Knowledge Blockchain (Fill und Härer 2018) ist ein Ansatz zur Speicherung von Modellen in der Datenstruktur einer permissioned Blockchain. Die Datenstruktur codiert beliebige Modelle bestehend aus Extensionen der Elemente Class und Relation-Class des ADOxx-Metamodells, z.B. BPMN-Aktivitäten bzw. Sequenzflüsse zusammen mit Attributen.

- Eine Transaktion besteht aus signierten Modellen einer Domäne (M1) und Permission-Modellen (M2), die mit der Anfügung eines Blocks in zwei separaten Merkle-Bäumen codiert werden. Ein Teilnehmer *i* des Systems erstellt oder verändert Elemente der Modelle und sendet beide Modelle als Transaktion an eine zentrale Identität *j* zur Aufnahme in einen Block. Permission-Modelle (M2) definieren Identitäten und deren Rechte zur Durchführung von Modell-Operationen in beiden Modellen.
- Ein Block umfasst einen *Block-Header*, einen *Block-Hash* zur Abbildung des Headers auf einen Hash-Wert sowie eine *Signatur*, so dass eine Verbindlichkeit der Zuordnung eines Blocks zu *j* besteht. Der Block Header enthält zur Gewährleistung der Integrität die Merkle Root beider Bäume sowie eine Signatur der sendenden Identität *i* zur Gewährleistung der Verbindlichkeit der Zuordnung zum Absender der Transaktion. Die Modelle werden vollständig innerhalb des Blocks gespeichert.

Zur Anfügung eines Blocks unter Aufnahme einer Transaktion wendet *j* definierte Protokoll-Regeln eines Mining-Verfahrens an und erstellt im Falle von validen Modell-Veränderungen einen Block. Das Verfahren überprüft die von einer sendenden Identität *i* auf beide Modelle angewendeten Operationen.

- Die Rechte zur Durchführung von Operationen innerhalb des Permission-Modells (M2) sind je Identität durch das Permission-Modell des vorhergehenden Blocks definiert. Dabei werden Create-, Delete- und Transfer-Permissions

unterschieden. Eine Transaktion wird von *j* verworfen, wenn eine von *i* durchgeführte Operation zur Erstellung oder Löschung von Elementen nicht durch die Rechte einer Create- oder Delete-Permission für *i* in M2 des vorherigen Blocks erlaubt sind. Zudem wird die Transaktion verworfen, wenn eine von *i* durchgeführte Delegation von Rechten an eine weitere Identität *k* nicht durch das Recht einer Transfer-Permission für *i* in M2 des vorherigen Blocks erlaubt ist.

- Die Rechte zur Durchführung von Operationen innerhalb des Modells der Domäne (M1) sind je Identität durch das Permission-Modell der Transaktion definiert. Sofern eine Transaktion nicht aufgrund von Veränderungen des Permission-Modells verworfen wurde, werden die von einem Teilnehmer *i* auf M1 angewendeten Operationen hinsichtlich der Rechte des mit der Transaktion übertragenen M2 überprüft. Dabei werden Create- und Delete-Permissions unterschieden. Eine Transaktion wird von *j* verworfen, wenn eine von *i* durchgeführte Operation zur Erstellung oder Löschung von Elementen nicht durch die Rechte einer Create- oder Delete-Permission für *i* in M2 der Transaktion erlaubt sind.

Die Anwendung der Protokoll-Regeln ist von allen Teilnehmern des Systems oder externen Prüfern überprüfbar.

**Implementierung**

Der Ansatz ist prototypisch in der ADOxx-Metamodellierungsplattform implementiert. In ADOxx sind Modelltypen zur Erstellung von Modellen und Permission-Modellen implementiert. Eine C++-DLL erstellt private und öffentliche Schlüssel je Identität und erlaubt das Signieren von Modellen und Blöcken anhand von ECDSA unter Verwendung der Kurve Secp256k1 mittels OpenSSL. Bei der Erstellung eines Blocks werden Modelle anhand einer AdoScript-Implementierung in die beiden Merkle-Bäume überführt. Jedes Element erhält eine UUID. Nach einer erfolgreichen Validierung der Signaturen gegenüber den Identitäten des Permission Models durch die DLL folgt die Aufnahme der Merkle-Bäume in die beschriebene Datenstruktur der permissioned Blockchain. Die Hash-Werte des Merkle-Baumes, des Block-Headers und des Blocks werden durch die Hash-Funktion SHA-256 anhand der DLL erzeugt. Die Hash-Werte werden als Teil des Modells aufgenommen und zur Sicherstellung der Integrität überprüft.

**Anwendungen**

Anwendungen umfassen damit (a.) die Verfolgung der Evolution von Modellschemata über die Zeit, (b.) die Rückverfolgung von Modell-Änderungen auf einen Zeitpunkt und eine Identität, (c.) die Vergabe und Delegation von Rechten und (d.) der Nachweis von Mustern als Teil von Modellen gegenüber externen Prüfern.

Die Anwendungen (a.) und (b.) beziehen sich auf Blöcke mit den darin enthaltenen Modellen der Domäne sowie Permissions mit Identitäten. Anwendung (c.) nimmt auf Transfer-Permissions Bezug. Anwendung (d.) besteht in einem Existenz-Nachweis von einzelnen oder zusammengesetzten Elementen eines Modells als nicht-interaktiver Zero-Knowledge-Proof.

**Zero-Knowledge-Proofs**

Ein nicht-interaktiver Zero-Knowledge-Proof erlaubt den Nachweis der Existenz eines Elements $m$ als Bestandteil eines Modells gegenüber einem externen Beteiligten $e$ ohne Kenntnis oder Offenlegung des Modells. Dabei wird die Überprüfung der Existenz von $m$ durch $e$ angenommen, z.B. um das Bestehen von Aufgaben als Teil eines Prozessmodells nachzuweisen oder um Bedingungen und die Ausgestaltung von Verträgen nachzuweisen.

Die Struktur der Codierung von Modellen in Merkle-Bäumen erlaubt den Nachweis von $m$ durch einen Hash-Wert $H(m)$. Dieser weist die Existenz von einzelnen oder zusammengesetzten Elementen als Bestandteil des Merkle-Baumes eines Blocks nach. Das Verfahren prüft für jeden Block und darin für jeden Knoten eines Merkle-Baumes, ob $H(m)$ mit dem Wert eines betrachteten Knotens übereinstimmt. Ist dies für einen Knoten $k$ eines Blocks $B_h$ der Fall, so ist die Existenz von $e$ zum Zeitpunkt des Zeitstempels von $B_h$ nachgewiesen. Dabei wird eine Überprüfung konkreter und exakter Attributwerte durch $e$ unterstellt.

Ein Existenznachweis kann sich beispielsweise auf das Vorhandensein von Aufgaben eines Geschäftsprozesses beziehen, die aus Compliance-Gründen gegenüber einer dritten Partei nachzuweisen sind; etwa eine Identitätsprüfung aufgrund von Know-Your-Customer-Richtlinien. Weitere Attribute und darauf basierte Nachweise können die Überprüfung über das Modell hinaus erweitern. Zudem können die Elemente als Bestandteil des Kontrollflusses nachgewiesen werden, da Beziehungen anhand des gleichen Verfahrens als Elemente des Merkle-Baumes nachweisbar sind.

#### 3.4.5.2 Modelle in dezentralen Blockchain-Systemen

Dezentrale Blockchain-Systeme bieten über die Gewährleistung von Integrität und Verbindlichkeit hinausgehende Anwendungen, die sich auf die Merkmale Unveränderlichkeit und Trustless beziehen. Zwischen den Teilnehmern des Systems müssen keine vorab etablierten Beziehungen bestehen. Analog zur Durchführung von Transaktionen für den Transfer von Wert zwischen Teilnehmern können beliebige Transaktionen zwischen Organisationen auf die gleichen Merkmale zurückgreifen.

Eine Entwicklung von Informationssystemen kann über Organisationsgrenzen hinaus beliebige Teilnehmer umfassen, die sich dem System zur Laufzeit anschließen. Die Voraussetzung hierfür ist die Nutzung eines dezentralen Blockchain-Systems, das nicht die Annahme einer vorab festgelegten Menge an Teilnehmern trifft. Die mit dem Consensus-Verfahren des Protokolls einhergehende Validierung der Korrektheit der Protokollausführung incentiviert eine verteilte Ausführung unter Beteiligung aller Teilnehmer.

Das nachfolgende Kapitel 4 entwickelt vor diesem Hintergrund einen Ansatz, der die Merkmale von Blockchain-Systemen auf die Gestaltung und Ausführung eines anhand von Prozessen beschriebenen Systems zwischen global verteilten Teilnehmern bezieht.

## 3.5 Ergebnisdiskussion

Blockchain-Systeme umfassen die Komponenten Datenstruktur, Protokoll und Netzwerk. Die Verteilung und Validierung der aus Transaktionen bestehenden Datenstruktur innerhalb eines Peer-to-Peer-Netzwerks regelt ein Protokoll durch Anwendung eines Consensus-Verfahrens. Das Herstellen einer definierten Ordnung innerhalb der verteilten Datenstruktur führt zu global eindeutigen Systemzuständen, deren Integrität und Verbindlichkeit validierbar ist. Der Systemzustand wird daher von allen Systemteilnehmern als vertrauenswürdig angesehen und kann für die verbindliche Speicherung von Transaktionen zwischen a priori unbekannten Peers herangezogen werden. Smart Contracts erweitern dieses Konzept um die Ausführung von Programmen, die damit als verbindliche und validierbare Verträge herangezogen werden können.

Dezentrale Systeme sind verteilte Systeme, die eine verteilte Koordination durch die validierbare Ausführung eines Protokolls ausgehend von den Systemkomponenten etablieren. Dezentrale Blockchain-Systeme sind öffentlich und unbeschränkt (permissionless) zugreifbare dezentrale Systeme. Eine dezentrale Koordination innerhalb dieser Systeme wird durch das Protokoll definiert und von Smart Contracts zur Laufzeit erweitert. Ein dezentrales Blockchain-System ist eine offene Plattform, welche die Eigenschaften von offenen Kommunikationssystemen um den verbindlichen Austausch von Daten sowie deren Speicherung und Ausführung erweitern.

Die Annahme a priori unbekannter Teilnehmer in dezentralen Systemen erfordert derzeit den Einsatz von Consensus-Verfahren probabilistischer Finalität wie Proof-of-Work, die Konsistenz, Verfügbarkeit und Partitionstoleranz bezogen auf abgrenzbare Systemzustände im Zeitverlauf sicherstellen. Die Limitationen des BASE-Paradigmas treten mit zunehmender Knotenanzahl und Rechenleistung des Netzwerks in den Hintergrund. Weiterhin bestehen Limitationen hinsichtlich der Skalierung, bezogen auf die Kapazität und die Rate der Transaktionen. In privaten Blockchain-Systemen eines festgelegten Teilnehmerkreises stehen geringere Limitationen einer eingeschränkten Anwendbarkeit als dezentrale Plattform aufgrund der getroffenen Annahmen gegenüber.

Als technische Basis zur Realisierung von dezentralen Organisationen ermöglichen Blockchain-Systeme eine globale und verbindliche Integration von Anwendungssystemen ohne zentrale Koordination. Dezentrale Blockchain-Systeme etablieren offene Plattformen zur Ausführung von betrieblichen Transaktionen und zur Abwicklung von Verträgen zwischen personellen Aufgabenträgern und Software.

# Kapitel 4

# Ein Ansatz zur integrierten Entwicklung und Ausführung von Prozessen in dezentral organisierten Systemen

## 4.1 Dezentrale Systementwicklung

Ein dezentrales System[1] entwickelt sich als verteiltes und dezentral koordiniertes System ausgehend von den Systemkomponenten. Diese sind autonom und selbstorganisiert, und interagieren ohne Intermediäre oder Koordinatoren direkt untereinander. Dezentrale Blockchain-Systeme implementieren ein solches System bereits für Transaktionen von Werten und Smart Contracts, die zwischen beliebigen Teilnehmern ohne vorab etablierte Beziehungen zustande kommen. Das mit diesem Ansatz angestrebte System ist ein Peer-to-Peer-Wertschöpfungsnetz, dessen Teilnehmer anhand von Transaktionen unmittelbar Vereinbarungen treffen oder Verträge schließen. Ziel dieses Ansatzes ist es, die Entwicklung und Ausführung von Prozessen zwischen Organisationen, Einzelpersonen oder Software ohne Intermediation dezentral auf der Basis von Transaktionen zu realisieren.

**Ziele des Ansatzes**

Z-1. Dezentrale Systementwicklung

- Z-1.1. Prozessorientierte Gestaltung des Systems
- Z-1.2. Entwicklung durch kooperative Modellierung

[1] siehe Abschnitt 3.1.2

Z-1.3. Dezentrale Koordination der Entwicklung

Z-2. Dezentrales Zielsystem

Z-2.1. Offenes Peer-to-Peer-Wertschöpfungsnetz

Z-2.2. Unterstützung von kooperativen Prozessen

Z-2.3. Dezentrale Koordination der Ausführung

Der Ansatz basiert auf Methoden der Systementwicklung anhand von Modellen[2] und ist verwandt zu Ansätzen der kooperativen Modellierung (Collaborative Modeling)[3] sowie zur Modellierung von interorganisationalen Prozessen und Kooperationen[4]. Damit werden eine Reihe von bestehenden Ansätzen zur Modellierung kombiniert und zusammen mit Methoden der verteilten Versionsverwaltung und Blockchain-Technologien implementiert.

### 4.1.1 Zielarchitektur

Die Zielarchitektur ist ein dezentrales System, dessen Komponenten als autonome Knoten eines globalen Netzwerks anhand von direkten Interaktionen eine dezentral koordinierte Wertschöpfung realisieren. Dabei wird eine durch das Peer-to-Peer-Netzwerk koordinierte Wertschöpfung angenommen, deren Transaktionen anhand von Modellen geplant, vereinbart und unveränderlich gespeichert werden. Ein transparent zugreifbares und unveränderlich gespeichertes Modell besitzt den Charakter eines Vertrages, der die Ausführung der Transaktionen regelt.

Zudem wird eine dezentrale und interaktive Wertschöpfung unterstellt, an der Organisationen, Einzelpersonen und Software als Peers beteiligt sind. Das Paradigma der Wertschöpfung basiert damit auf der Netzwerkorganisation sowie den hierauf aufbauenden Formen der interaktiven, vernetzten und dezentralen Wertschöpfung[5].

Das in Abbildung 4.1 dargestellte System zeigt einen Ausschnitt des Gesamtsystems sowie Teilsysteme aus der Sicht einzelner Komponenten. Ein Zusammenwirken mehrerer Komponenten erfordert eine innerhalb des Systems stattfindende Koordination, an der die als Wertschöpfungsstufen gekennzeichneten Diskursweltobjekte (Ferstl und Sinz 2013, S. 219) in Kooperation beteiligt sind. Umweltobjekte (Ferstl

[2] siehe Abschnitt 2.2.4
[3] siehe Abschnitt 2.4
[4] siehe Abschnitt 2.3
[5] siehe Abschnitt 2.3.2

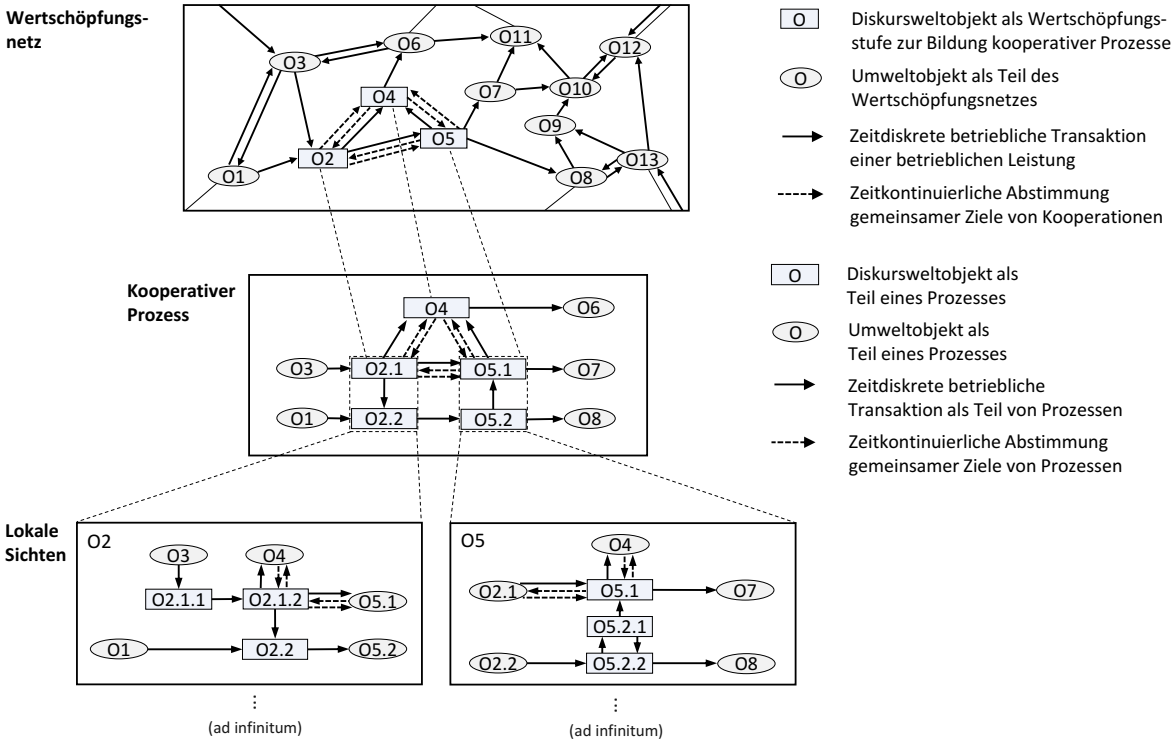



ABBILDUNG 4.1: Zielarchitektur des dezentralen Systems

und Sinz 2013, S. 219) sind kooperationsexterne Teilnehmer des Wertschöpfungsnetzes, die mit Diskurswelttobjekten z.B. in Zuliefer-Abnehmer-Beziehungen interagieren. Die Knoten des Netzes und ihre Interaktionsbeziehungen bilden innerhalb der Teilsysteme kooperative Wertschöpfungsprozesse. Zeitkontinuierliche Beziehungen legen dort die Koordination zur Vereinbarung von Zielen (Ferstl und Sinz 2013, S. 8, 215–218) anhand eines Protokolls fest. Zeitdiskrete Beziehungen modellieren fachliche Transaktionen (Ferstl und Sinz 2013, S. 8, 215–218) zwischen den Beteiligten, die Aufgaben zur Erstellung von Leistungen durchführen. Objekte umfassen einen objektinternen Speicher sowie eine Aktionensteuerung und Aktionen zur Veränderung ihres Zustandes[6]. Eine zeitkontinuierliche Beziehung zwischen zwei autonomen Objekten ist zum gegenseitigen Zugriff auf Parameter der objektinternen Speicher erforderlich, sofern Kooperationen eine Abstimmung von Zielen zur Gestaltungszeit erfordern. Eine zeitdiskrete Beziehung zwischen zwei autonomen Objekten beschreibt die Übertragung von Lenkungsnachrichten und Leistungspaketen (Ferstl und Sinz 2013, 203 f.) zur Ausführungszeit. Die Objekte weiterer Teilsysteme bilden sich durch eine strukturelle Zerlegung (Ferstl und Sinz 2013, 207 f.). Die lokale Sicht eines autonomen Objekts umfasst mindestens die aus der Zerlegung

[6]siehe Abschnitt 2.1.3.3

hervorgehenden Diskurswelttobjekte sowie die unmittelbar in Beziehung stehenden Objekte der Umwelt. Die Zerlegung eines Teilsystems kann über beliebig viele Ebenen fortgesetzt werden.

**Aufbau des Kapitels**

Die Architektur des Ansatzes wird im nachfolgenden Abschnitt 4.2 konstruiert. Die daraus hervorgehenden Teilsysteme sind Gegenstand der Abschnitte 4.3 zur Modellierung, 4.4 zur Kooperation und 4.5 zur Ausführung.

## 4.2 Architektur des Ansatzes

### 4.2.1 Teilsysteme und Schichten des Ansatzes

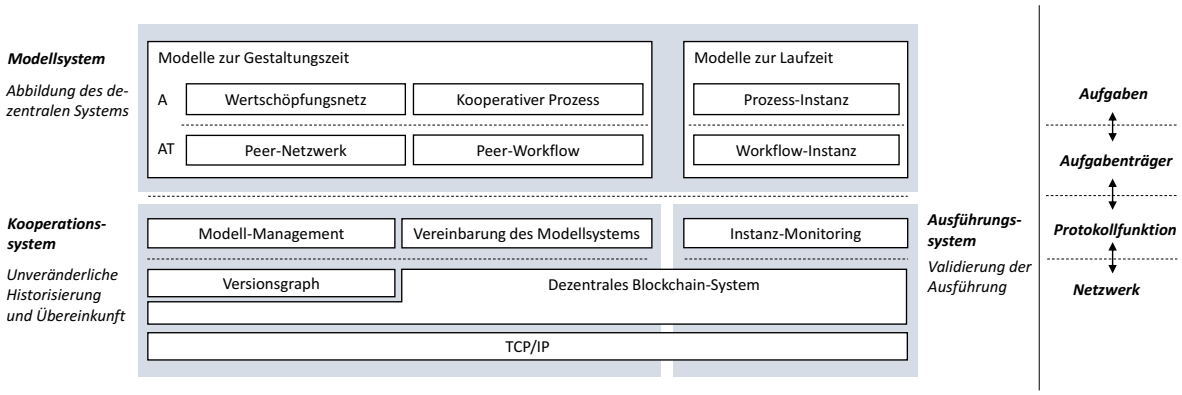



ABBILDUNG 4.2: Architektur des Ansatzes

Die Architektur des Ansatzes wird anhand von funktionalen Abgrenzungsmerkmalen gebildet, die eine Strukturierung in Subsysteme und Schichten ergeben. Abbildung 4.2 zeigt die Systeme als Bestandteile der aus vier Schichten bestehenden Architektur[7]. Das Modellsystem definiert Beschreibungsmittel, anhand derer sich ein dezentrales System auf Modelle zur Gestaltungszeit und zur Laufzeit abbilden lässt. Die oberste Ebene erlaubt eine Beschreibung des Wertschöpfungsnetzes, kooperativer Prozesse und deren Instanzen auf der Ebene der Aufgaben (A). Das Wertschöpfungsnetz setzt sich aus den Knoten des Peer-Netzwerks zusammen. Diese bilden die Ebene der Aufgabenträger (AT) zusammen mit Peer-Modellen, die sich aus den kooperativen Prozessen ableiten. Diese Modelle sind schließlich in Instanz-Modelle überführbar. Zur Koordination der Planung und Ausführung interagieren die selbstorganisierten Aufgabenträger als Peers anhand eines Protokolls, das Funktionen des Kooperationssystems und des Ausführungssystems umfasst. Das Kooperationssystem realisiert die unveränderliche Historisierung und Übereinkunft der innerhalb des dezentralen Systems geplanten Prozesse anhand der Modelle der Gestaltungszeit. Das Ausführungssystem erlaubt die Validierung der Ausführung anhand eines Instanz-Monitorings, das auf die Modelle der Laufzeit zurückgreift. Eine Validierung eines kooperativen Prozesses kann durch jedes Peer erfolgen, das an der Kooperation beteiligt ist. Die Funktionen des Protokolls interagieren mit der

[7] Hiermit wird die in Härer (2018) beschriebene Architektur erweitert.

Schicht des Netzwerks. Das Modell-Management[8] sichert Modelle in einem Distributed Version Control System (DVCS), das als Replikation jedem Peer lokal vorliegt. Der Zustand des DVCS und dessen Integrität werden anhand eines dezentralen Blockchain-Systems abgesichert. Dieses System implementiert zudem die Vereinbarung des Modellsystems über Abstimmungsmechanismen anhand von Smart Contracts sowie das Instanz-Monitoring zur Protokollierung und Validierung ausgeführter Instanz-Zustände. Die Datenstruktur der Blockchain enthält die globale Sicht des Wertschöpfungsnetzes, des kooperativen Prozesses und der Instanzen sowie, zur Sicherung der Konsistenz, Hash-Bäume zur Abbildung der hierarchischen Zerlegung lokaler Teilsysteme. Die globale Vernetzung über ein dezentrales Blockchain-System, basierend auf einer TCP/IP-Infrastruktur, ist eine notwendige Voraussetzung zur Anwendung des Ansatzes.

### 4.2.2 Konstruktion der Architektur

Die Architektur des Ansatzes definiert einen Rahmen zur Beschreibung der zugrunde liegenden Komponenten mit ihren Beziehungen sowie deren Anwendungen, Funktionen und Technologien.

Zur Realisierung der Zielarchitektur leitet sich die Architektur des Ansatzes aus den Grundlagen der geschäftsprozessorientierten Entwicklung betrieblicher Systeme (Kapitel 2) und dem Aufbau von dezentralen Systemen und Blockchain-Systemen ab (Kapitel 3). Ausgehend von betrieblichen Aufgaben und ihren Phasen werden im Hinblick auf die Ziele zunächst die zeitlichen Bezüge zur Gestaltungs- und Laufzeit diskutiert. Anschließend bildet sich aus den Grundlagen der Modellierung (Kapitel 2.2) und aus den Methoden kooperativer Geschäftsprozesse (Kapitel 2.3) die Komponente Modellsystem. Weiterhin basierend auf den Konzepten kooperativer Geschäftsprozesse, führen diese zusammen mit der Architektur von Blockchain-Systemen zur Bildung des Kooperationssystems, mit dem das Zusammenwirken verteilter Objekte u.a. anhand von Smart Contracts koordiniert wird. Unter Hinzunahme der Instanziierung bildet sich, ebenfalls auf Basis der zuletzt genannten Grundlagen, das Ausführungssystem. Dieses bildet einzelne Ausführungszustände ab und verteilt sie. Die Grundlage zur Modellierung der Ausführung ist die Petri-Netz-Semantik der verhaltensorientierten Modelle des Modellsystems.

[8] siehe Abschnitt 2.3.5.2

### 4.2.3 Zeitbezogene Abgrenzung

Die Aufgabenphasen Analyse, Synthese und Aufgabenträger-Zuordnung betreffen die Gestaltungszeit, wobei die Konstitution von Aufgaben aus Außensicht durch Sach- und Formalziele gegeben ist. Einzelne Aktionen und ihre Steuerung bilden das Lösungsverfahren, das eine Aufgabe in ihrer Innensicht beschreibt.

Die zeitliche Abgrenzung der Gestaltung und Ausführung von Aufgaben als Gestaltungszeit oder Build Time bzw. Laufzeit oder Run Time ist den Aufgabenphasen zuordenbar. Bezogen auf eine Aufgabe betreffen Analyse, Synthese und Aufgabenträger-Zuordnung die Gestaltungszeit, wobei sich die Durchführung und die Kontrolle während der Durchführung auf die Laufzeit beziehen. Die Aufgabenphase der Kontrolle nach der Durchführung fällt nicht der Laufzeit zu. In Prozessen treffen die genannten Zeitbezüge auf die dem Prozess zugehörigen Aufgaben zu (Kapitel 2.1.3.3), die hinsichtlich der Gestaltungs- und Laufzeit überlappen können. Bezüglich des Prozesses sind damit die Gestaltungszeit und die Laufzeit des Prozesses als Prozess-Gestaltungszeit bzw. Prozess-Laufzeit abgrenzbar, die sich auf die Planung bzw. die Durchführung von Vorgängen bezieht.

| | **Gestaltungszeit / Build Time** | **Laufzeit / Run Time** |
|---|---|---|
| **Aufgabe** | Aufgabengestaltung durch Analyse, Synthese und Aufgabenträger-Zuordnung | Durchführung als Vorgang, Kontrolle während Durchführung |
| **Prozess** | Zusammenfassung von Aufgaben und Beziehungen mit gemeinsamen, übergeordneten Zielen | Leistungserstellung durch Ausführung zusammengehöriger Aufgaben als Prozess-Instanz, Überwachung während der Ausführung |
| **System** | Verknüpfung von Prozessen als Wertschöpfungsstufen mit Output-Input-Beziehungen eines Wertschöpfungsnetzes | Wertschöpfung durch Ausführung von Prozessen des Wertschöpfungsnetzes |

TABELLE 4.1: Zeitbezogene Abgrenzung

Aufgrund der Betrachtung von interorganisationalen Netzwerken sind zudem zeitliche Bezüge zwischen einzelnen Prozessen unterscheidbar, die sich auf die Abfolge mehrerer Prozesse in einem System aus vernetzten Prozessen beziehen. Zur zeitbezogenen Abgrenzung ergeben sich somit die in Tabelle 4.1 dargestellten Zeitbezüge

als Aufgaben-Gestaltungszeit und Aufgaben-Laufzeit, als Prozess-Gestaltungszeit und Prozess-Laufzeit sowie als System-Gestaltungszeit und System-Laufzeit.

Eine Überlappung der Gestaltungszeit- und der Laufzeit kann insbesondere in interorganisationalen Systemen mit mehreren Prozessen auftreten, etwa in Supply Chains, in denen Teilsysteme Prozesse planen, während andere Teilsysteme Prozesse zeitgleich ausführen. Im Falle einer Überlappung von Planung und Ausführung sind die Merkmale hochflexibler Prozesse erfüllt (Kapitel 2.3.4.2). Dezentrale Systeme erfordern somit die Berücksichtigung einer potenziell überlappenden Prozess-Gestaltungszeit und Prozess-Laufzeit sowie einer überlappenden System-Gestaltungszeit und System-Laufzeit.

# 4.3 Modellsystem

## 4.3.1 Generische Beschreibung des Modellsystems

### 4.3.1.1 Rahmen zur Beschreibung von Modellen und Sichten der Zielarchitektur

Anhand des in Abbildung 4.3 dargestellten Rahmens legt das Modellsystem eine gemeinsame Sprache fest, die als Basis der Kommunikation und Vereinbarung von Prozessen des dezentralen Systems von allen Teilnehmern herangezogen wird. Modelle der Aufgabenebene enthalten globale Struktur- und Verhaltenssichten, die im Sinne eines Public Process öffentlich zugreifbar sind. Die Aufgabenträgerebene enthält lokale Struktur- und Verhaltenssichten, die im Sinne eines Private Process auf ein Objekt und dessen Zerlegungsprodukte beschränkt sind.

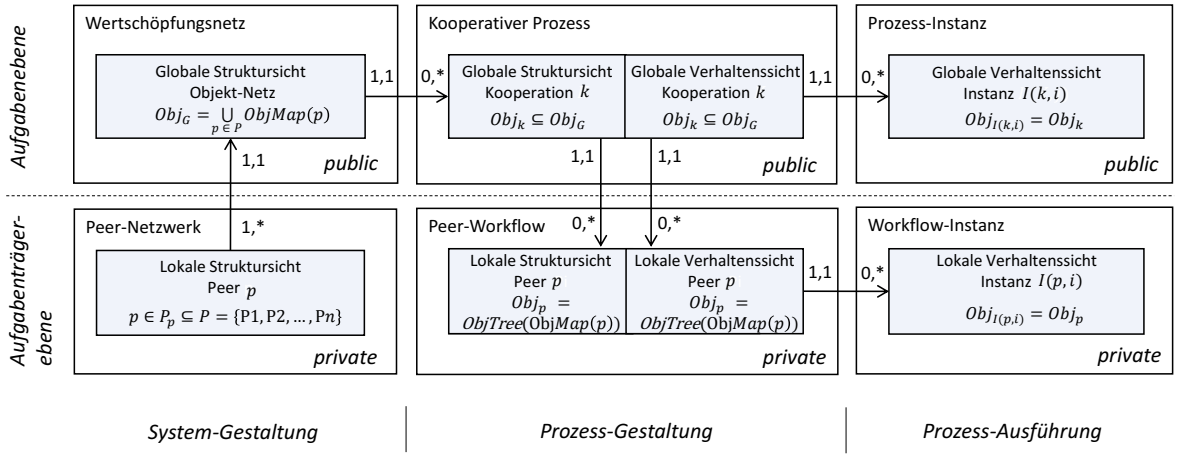



ABBILDUNG 4.3: Beschreibungsrahmen der Architektur des Modellsystems

### 4.3.1.2 System-Gestaltung

Der Gestaltung des Systems liegen Systemkomponenten zugrunde, die autonom und selbstorganisiert sind. Die Kommunikation lose gekoppelter Objekte ist die Basis zur Bildung von Netzwerken, die eine kooperative Planung von Aufgaben ermöglichen und in ein globales Netz von Objekten eingehen. Im Hinblick auf die definierte Zielarchitektur betreffen die Modelle der System-Gestaltung die Vernetzung von Aufgabenträgern und die Bildung eines Wertschöpfungsnetzes.

**Peer-Netzwerk**

*Ziel:* Identifikation und Abbildung bekannter Aufgabenträger.

Die von den Komponenten des Systems ausgehende Planung beginnt mit dem Modell des Peer-Netzwerks, das eine lokale Sicht eines betrachteten Peers $p$ darstellt. Das Netzwerk enthält die lokal bekannten Knoten $p \in P_p$, die Teil einer globalen Knotenmenge $P_p \subseteq P$ sind.

*Ansatz:* Das Modell kann auf Basis eines Graphen gebildet und um domänenspezifische Attribute ergänzt werden. Das Peer-Netzwerk geht in das globale Wertschöpfungsnetz ein. Hierfür ist die Definition der in Abbildung 4.4 dargestellten Modelltransformation erforderlich.

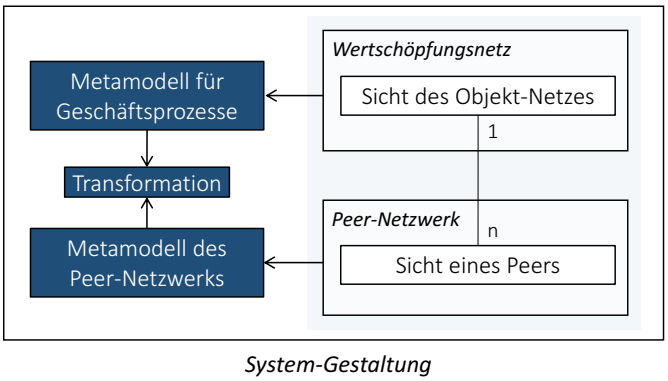



ABBILDUNG 4.4: Transformation zwischen Sichten der System-Gestaltung

**Wertschöpfungsnetz**

*Ziel:* Vernetzung und Festlegung von gemeinsamen Zielen kooperierender Objekte.

Das Modell zur Abbildung des dezentralen Systems umfasst die globale Struktur des Wertschöpfungsnetzes, die sich durch die Vernetzung der Peer-Netzwerke anhand einer Zuordnung von Peers zu Objekten ergibt. Das Modell bildet kooperierende und nicht-kooperierende Objekte zusammen mit Ziel- und Transaktionsbeziehungen ab. Kooperierende Objekte besitzen gemeinsame Ziele und interagieren im Rahmen einer gemeinsamen Wertschöpfung. Die Menge der globalen Objekte $Obj_G$ des Modells ist die Vereinigungsmenge derjenigen Objekte, die einzelnen $p$ zugeordnet sind. $ObjMap(p) = O$ ist eine Funktion, welche die Menge $O$ der $p$ zugeordneten Objekte zurückgibt[9].

[9] siehe Smart Contract GC, Abschnitt 4.4.1.4

*Ansatz:* Die Beschreibung der Struktur der Objekte mit ihren Interaktionen kann auf Grundlage einzelner Elemente des Metamodells der Aufgabenebene des SOM[10] erfolgen. Das Wertschöpfungsnetz ist der Ausgangspunkt zur Beschreibung der Struktur von kooperativen Prozessen.

#### 4.3.1.3 Prozess-Gestaltung

Die Gestaltung von Prozessen umfasst die Phase der Planung einzelner Aufgaben. Die bislang aggregiert als Teil von Objekten erfassten Aufgaben werden durch Analyse und Synthese bestimmt.

**Kooperativer Prozess**

*Ziel:* Bildung eines globalen Prozesses der kooperierenden Objekte.

Das Modell beschreibt die Struktur und das Verhalten eines kooperativen Prozesses anhand der daran beteiligten Objekte $Obj_k$ für eine Kooperation $k$. Die aus dem Wertschöpfungsnetz hervorgehende Struktur wird anhand des Kooperationssystems festgelegt und um Angaben des Verhaltens ergänzt. Die Interaktionsbeziehungen zwischen den Elementen von $Obj_k$ werden auf Grundlage der zuvor definierten Zielbeziehungen der Kooperationen um transaktionale Leistungsbeziehungen erweitert, um einen kooperativen Prozess einer initialen Zerlegungsstufe zu etablieren. Eine weitergehende Transaktions- und Objektzerlegung definiert den kooperativen Prozess. Die Festlegung von Transaktionen führt dabei zur Angabe des Verhaltens, das als Abfolge einzelner Aufgaben anhand von Ordnungsrelationen oder Ereignissen beschrieben wird.

*Ansatz:* Die Struktur und das Verhalten können anhand von Interaktions- und Vorgangs-Ereignis-Schemata des SOM abgebildet werden. Ein kooperativer Prozess ist der Ausgangspunkt zur Beschreibung von Peer-Workflows und von Instanzen des Prozesses. Eine Transformation zwischen den in Abbildung 4.5 dargestellten Sichten muss hierfür definiert werden.

**Peer-Workflow**

*Ziel:* lokale Detaillierung und Implementierung einzelner Objekte.

Ein Peer-Workflow detailliert und implementiert die Struktur und das Verhalten eines kooperativen Prozesses für ein Peer $p$ als lokale Sichten. Beide Sichten gehen initial aus einem kooperativen Prozess hervor und sind diesem untergeordnet. Der Workflow beschreibt die Innensicht der $p$ zugeordneten Objekte $ObjMap(p)$ durch

[10] siehe Abschnitt 2.2.4

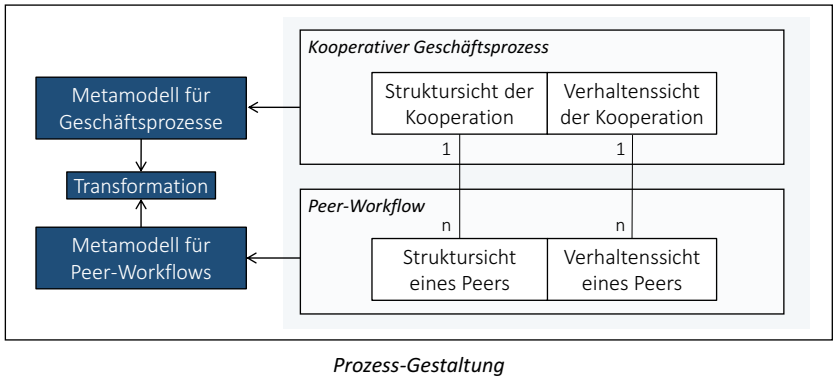



ABBILDUNG 4.5: Transformation zwischen Sichten der Prozess-Gestaltung

eine hierarchische Zerlegung in detaillierte Objekte. Die Sichten zeigen die Menge der Objekte $Obj_p$, die aus den Objekten der Zerlegungsbäume von $ObjMap(p)$ besteht. $ObjTree(P) = Q$ ist eine Funktion, welche die Menge der Objekte Q zurückgibt, die in den Zerlegungsbäumen der Objekte der Menge P enthalten sind[11]. Das Modell beschreibt die Implementierung der Objekte $Q$ anhand von Blockchain-Transaktionen und Smart Contracts.

*Ansatz:* Die Abbildung der Struktur und des Verhaltens des Workflows können auf Grundlage des Interaktions- und des Vorgangs-Ereignis-Schemas zusammen mit Aufgabenträgerzuordnungen sowie Pre- und Post-Conditions erfolgen. Die Darstellung des Verhaltens kann alternativ anhand von BPMN erfolgen (Härer 2018).

#### 4.3.1.4 Prozess-Ausführung

Modelle zur Laufzeit beschreiben Instanzen für die Funktion des Instanz-Monitorings. Damit wird die Phase der Kontrolle während und nach der Durchführung einzelner Aufgaben realisiert.

**Prozess-Instanz**

*Ziel:* globale Nachvollziehbarkeit der Ausführung des kooperativen Prozesses.

Eine Prozess-Instanz beschreibt das Verhalten eines kooperativen Prozesses zur Laufzeit. Ein Modell erfasst den Ausführungszustand einer Instanz $I(k,i)$ für die i-te Instanz ($i \in \mathbb{Z}$) eines Prozesses der Kooperation $k$. Die Menge der Objekte $Obj_{I(k,i)}$ entspricht der Menge der Objekte des zugrunde liegenden kooperativen Prozesses. Die Bildung einzelner Instanzen greift somit auf die globale Verhaltenssicht des

[11] siehe Objektbaum, Abschnitt 4.4.1.3

kooperativen Prozesses und die lokalen Verhaltenssichten einzelner Peers zurück. Abbildung 4.6 zeigt den Zusammenhang.

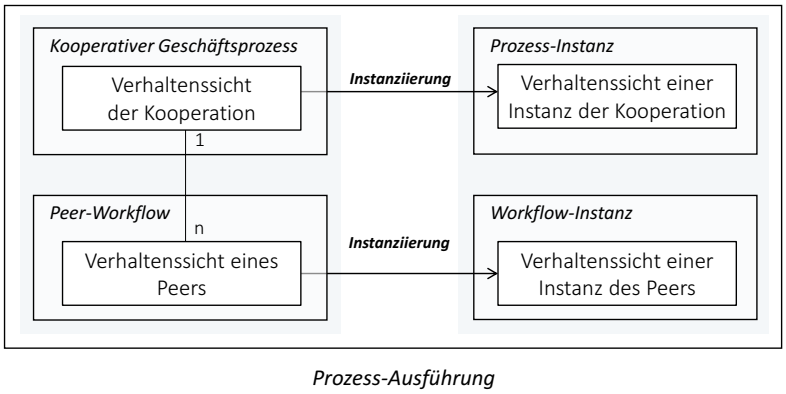



ABBILDUNG 4.6: Bildung von Instanz-Sichten zur Prozess-Ausführung

*Ansatz:* Eine Möglichkeit zur Modellierung von Ausführungszuständen des Verhaltens ist das Vorgangs-Ereignis-Schema, dessen Petri-Netz-Semantik um die Semantik gefärbter Petri-Netze erweiterbar ist. Eine Abbildung anhand von BPMN erfordert eine analoge Erweiterung.

**Workflow-Instanz**

*Ziel:* Lokale Nachvollziehbarkeit der Ausführung anhand von Peer-Workflows.

Eine Workflow-Instanz beschreibt das Verhalten eines Peer-Workflows zur Laufzeit. Ein Modell erfasst den Ausführungszustand einer Instanz $I(p,i), i \in \mathbb{Z}$ für einen Workflow eines Peers $p$. Die Menge der Objekte $Obj_{I(p,i)}$ entspricht der Menge der Objekte des zugrunde liegenden Peer-Workflows.

*Ansatz:* Die Modellierung von Ausführungszuständen des Verhaltens kann auf Basis der Verhaltenssicht des Peer-Workflows erfolgen. Analog zur Modellierung von Prozess-Instanzen kann hierfür die Petri-Netz-Semantik des Vorgangs-Ereignis-Schemas um die Semantik gefärbter Petri-Netze erweitert werden. Eine Abbildung anhand von BPMN erfordert eine analoge Erweiterung.

### 4.3.2 Abstraktes Beispiel zur Instanziierung des Modellsystems

Abbildung 4.7 zeigt ein abstraktes Beispiel zur Instanziierung des Modellsystems. Das Beispiel veranschaulicht die Schemata des Modellsystems in einer zusammenhängenden Darstellung. Eine vollständige Definition der Notation wird in den nachfolgenden Abschnitten zusammen mit Metamodellen entwickelt.

Ausgehend von der Vernetzung und Objekt-Zuordnung der Knoten der Peer-Netzwerke von P2, P4, P5 bildet sich ein Netz aus Objekten; hier anhand einer ein-eindeutigen Zuordnung von Objekten ($ObjMap(\mathrm{P}i) = \{\mathrm{O}i\}$ für $i \in \{1, 2, ..., 8\}$). Das Netzwerk zeigt drei Objekte, die durch die Festlegung gemeinsamer Ziele anhand von drei Zielbeziehungen kooperieren. Interaktionsbeziehungen zwischen Objekten definieren aggregierte transaktionale Beziehungen. Die Struktur des kooperativen Prozesses definiert sich auf dieser Basis durch Transaktionen und Zielbeziehungen zwischen Objekten. Je Transaktion ergeben sich zwei in der Verhaltenssicht dargestellte Aufgaben. Die Sicht stellt den Ablauf des Prozesses je Objekt dar.

Die Peer-Workflows der Peers P2, P4 und P5 sind dem kooperativen Prozess hierarchisch untergeordnet. Die Objektmengen bilden sich durch Zerlegung der einem Peer zugeordneten Objekte, z.B. $Obj_{P4} = ObjTree(ObjMap(P4)) = \{O4, O4.1, O4.2\}$. Diese Objekte definieren die Sicht des privaten Modells von P4. Das Modell implementiert Struktur und Verhalten für ein betrachtetes Peer, durch Detaillierung bis hin zu einem Workflow. Die in diesem Modell angedeutete Detaillierung wird durch die lokale Zerlegung von O4, O2.1 und O5.2 sichtbar.

Die Modelle der Workflow-Instanzen leiten sich aus der Verhaltenssicht der Peer-Workflows ab. Gleichzeitig sind Workflow-Instanzen aufgrund der hierarchischen Objektzerlegung einer Prozess-Instanz untergeordnet. Der Ausführungszustand der Instanziierungen von Peer-Workflows wird als Prozess-Instanz des kooperativen Prozesses sichtbar.

Der Zustand der hier dargestellten Instanzen wird durch die Marken eines gefärbten Petri-Netzes beschrieben. Eine Transition repräsentiert die Durchführung einer Aufgabe. Parallele Verzweigungen des Kontrollflusses führen in dem hier dargestellten Fall zu vier Marken. Parallele Verzweigungen in Peer-Workflows können zur Produktion von lokal sichtbaren Marken führen, die innerhalb der Prozess-Instanz nicht sichtbar sind. Die Konsistenz wird anhand der hierarchischen Zerlegung sichergestellt, die hier zu hierarchischen Petri-Netzen führt.

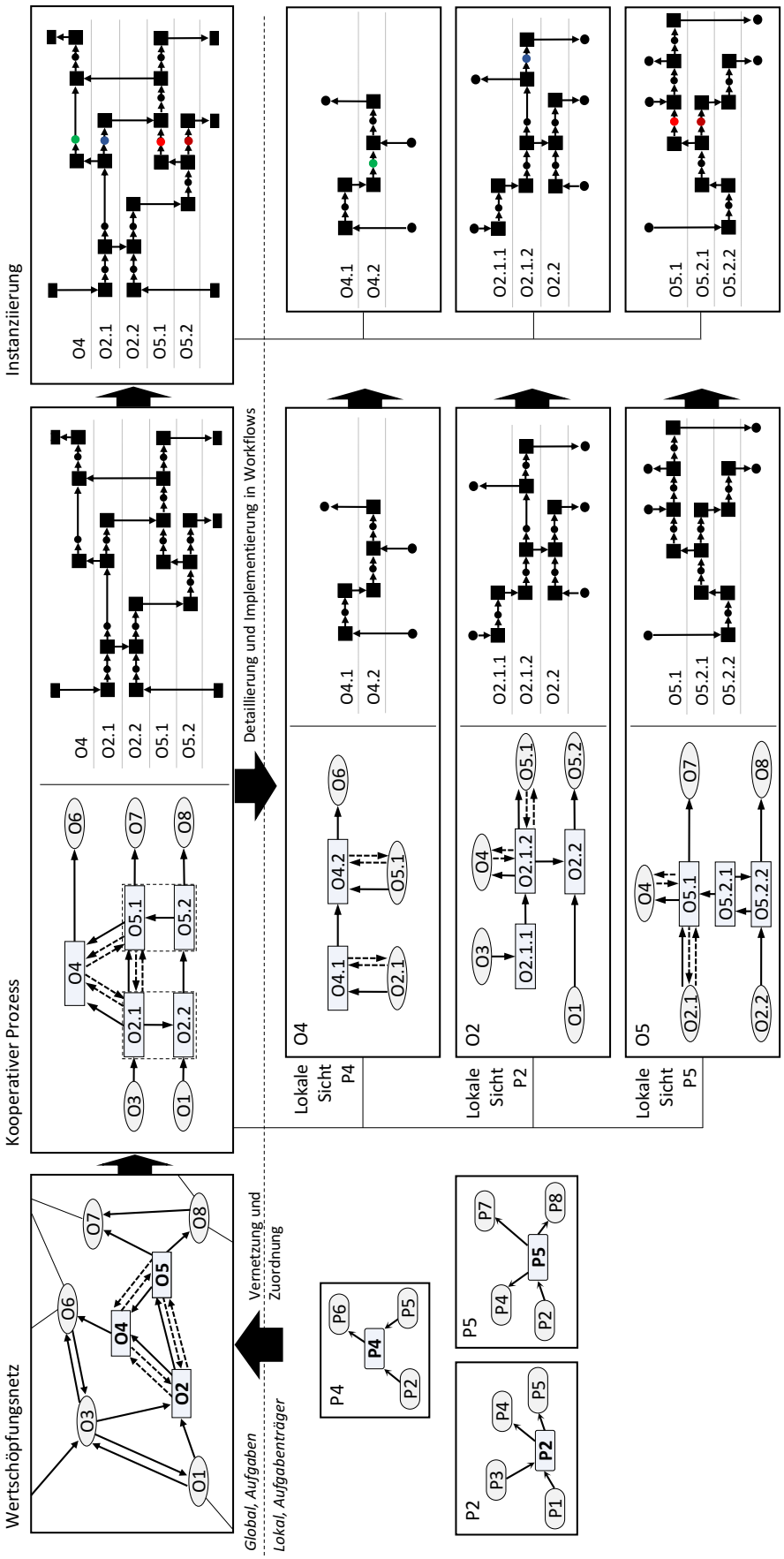



ABBILDUNG 4.7: Abstraktes Beispiel zur Modellsystem-Instanziierung

### 4.3.3 Instanziierung des Modellsystems

Der Rahmen des Absatzes (4.3) wird in den nachfolgenden Abschnitten auf Grundlage des SOM[12] mit Erweiterungen zu den jeweiligen Domänenkonzepten instanziiert. Die hierfür verwendeten Modelle und Schemata stellt Abbildung 4.8 dar.

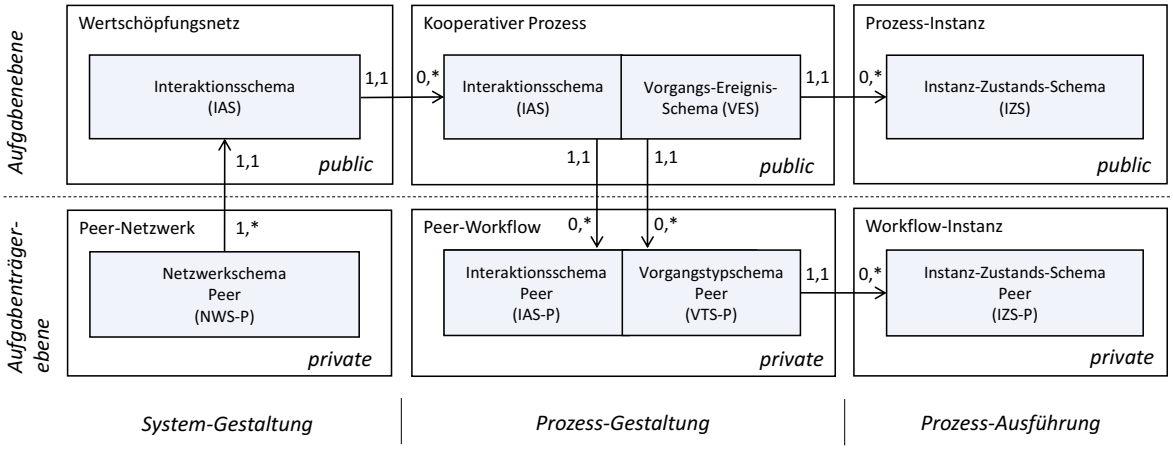



ABBILDUNG 4.8: Instanziierung des Modellsystems anhand von SOM

Die Definition der Modellsyntax greift auf die im vorhergehenden Abschnitt beispielhaft besprochene Notation zurück und beschreibt diese. Syntax und Notation werden anhand von Metamodellen der System-Gestaltung, der Prozess-Gestaltung und der Prozess-Ausführung definiert.

#### 4.3.3.1 Metamodelle der System-Gestaltung

Die Gestaltung des Systems umfasst die Beschreibung des Peer-Netzwerks und des Wertschöpfungsnetzes anhand der in Abbildung 4.9 dargestellten Metamodelle.

**Peer-Netzwerk: Netzwerkschema**

*Ziel: Identifikation und Abbildung bekannter Aufgabenträger.*

Ein Peer-Netzwerk bildet die Struktur bekannter Knoten einzelner Peers mit ihren Beziehungen aus der lokalen Sicht eines Peers ab. Anhand des Modells erfolgt eine Erfassung eines potenziell bereits bestehenden Unternehmensnetzwerks (Österle, Fleisch et al. 2002). Bestehende Teilnehmer des Netzwerks, z.B. Zulieferer und Abnehmer, sind beispielsweise aus bereits implementierten Prozessen bekannt. Diese

[12] siehe Abschnitt 2.2.4

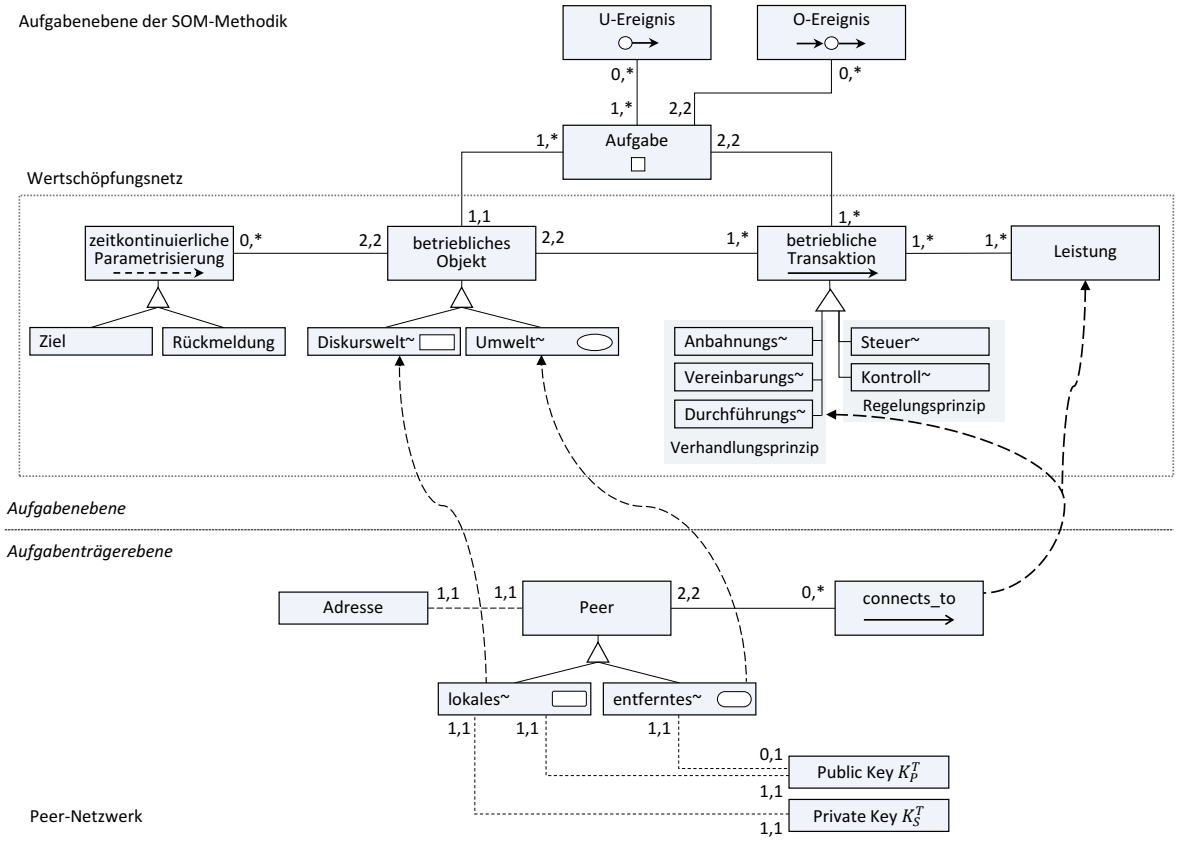



ABBILDUNG 4.9: Metamodelle und Transformation der System-Gestaltung

stellen Aufgabenträger dar, die im weiteren Verlauf nach dem Prinzip der Selbstorganisation an der Aufgabenphase der Planung mitwirken. Das in Abbildung 4.9 dargestellte Metamodell des Peer-Netzwerks erlaubt die Erfassung von miteinander verbundenen Peers zusammen mit den auf Aufgabenträgerebene benötigten Attributen. Die für die Durchführung von Blockchain-Transaktionen benötigten Attribute betreffen die Adressierung sowie ein Schlüsselpaar zur Sicherstellung der Verbindlichkeit anhand von Signaturen[13]. Die privaten Schlüssel entfernter Peers sind nicht bekannt.

### Wertschöpfungsnetz: Interaktionsschema

*Ziel: Vernetzung und Festlegung von gemeinsamen Zielen kooperierender Objekte.*

Ein Modell des Wertschöpfungsnetzes beschreibt die Struktur global miteinander vernetzter betrieblicher Objekte anhand der Aufgabenebene der SOM-Methodik.

[13] siehe Abschnitt 3.2.2.3

Zwischen betrieblichen Objekten verlaufen zeitkontinuierliche Ziel- und Rückmeldungsbeziehungen (Z/R-Beziehungen) sowie zeitdiskrete Transaktionsbeziehungen zur Erstellung von Leistungen.

Zwei Unterziele dieser Modellierung geben das Vorgehen zur Transformation entsprechend der Metamodell-Abbildung (Abbildung 4.9) und den weiteren Verlauf der Modellierung vor:

1. **Modellierung vernetzter Objekte:** Die Transformation von Netzwerkschemata in ein Interaktionsschema des globalen Wertschöpfungsnetzes sieht die Überführung von Peer-Elementen in betriebliche Objekte vor. Die lokal im Einzelnen durchgeführten Peer-Objekt-Zuordnungen führen mit der Zusammenführung aller Objekte in einem Modell zu einem globalen Netz aus Objekten. Aus der Sicht eines Peers führt die Transformation von lokalen Peers zu Diskursweltobjekten, die mit den aus entfernten Peers hervorgehenden Umweltobjekten in Beziehung stehen. Eine Beziehung zwischen Peers wird in eine Durchführungstransaktion (D-Transaktion) überführt, welche die Erstellung einer Leistung repräsentiert.
2. **Modellierung der gemeinsamen Ziele von kooperierenden Objekten:** Eine Beteiligung an kooperativen Prozessen zur Wertschöpfung setzt eine innerhalb des Systems von den Komponenten ausgehende Koordination voraus. Die Modellierung von zeitkontinuierlichen Z/R-Beziehungen bildet die Abstimmung der Ziele kooperierender Objekte ab. Ein Paar von Objekten O1, O2 kooperiert, sofern eine von O1 ausgehende Angabe einer Z/R-Beziehung zu O2 sowie eine von O2 ausgehende Z/R-Beziehung zu O1 als Teil des globalen Schemas modelliert ist. Anhand der Beziehungen werden die Ziele der Kooperation sowie nachfolgend das Modell des kooperativen Prozesses vereinbart.

Durch die Modellierungen von Zielbeziehungen zwischen Objekten bilden sich Teilgraphen, die jeweils einen kooperativen Prozess bedingen. Jeder kooperative Prozess wird anhand der Schemata zur Prozess-Gestaltung festgelegt.

#### 4.3.3.2 Metamodelle der Prozess-Gestaltung

Die Gestaltung eines Prozesses umfasst die globale Modellierung des kooperativen Prozesses auf Aufgabenebene sowie die Modellierung der lokalen Peer-Workflows als Implementierung und Detaillierung eines Prozesses. Abbildung 4.9 stellt die hierfür vorgesehenen Metamodelle zusammen mit einer Metamodell-Abbildung dar.

### Kooperativer Geschäftsprozess: Interaktions- und Vorgangs-Ereignis-Schema

*Ziel: Bildung eines globalen Prozesses der kooperierenden Objekte.*

Die Schemata eines kooperativen Geschäftsprozesses beschreiben die Struktur und das Verhalten des Prozesses als Interaktionsschema bzw. Vorgangs-Ereignis-Schema entsprechend der SOM-Methodik. Das initial aus einem Teilgraph des Wertschöpfungsnetzes übernommene Interaktionsschema bildet den Ausgangspunkt zur Spezifikation von betrieblichen Transaktionen und Objekten. Das Modell der Aufgabenebene erfasst anschließend den innerhalb der Kooperation globalen und öffentlichen Prozess durch hierarchische Zerlegungen der initialen Struktur. Hierfür werden ein Interaktionsschema (IAS) sowie ein Vorgangs-Ereignis-Schema herangezogen (VES).

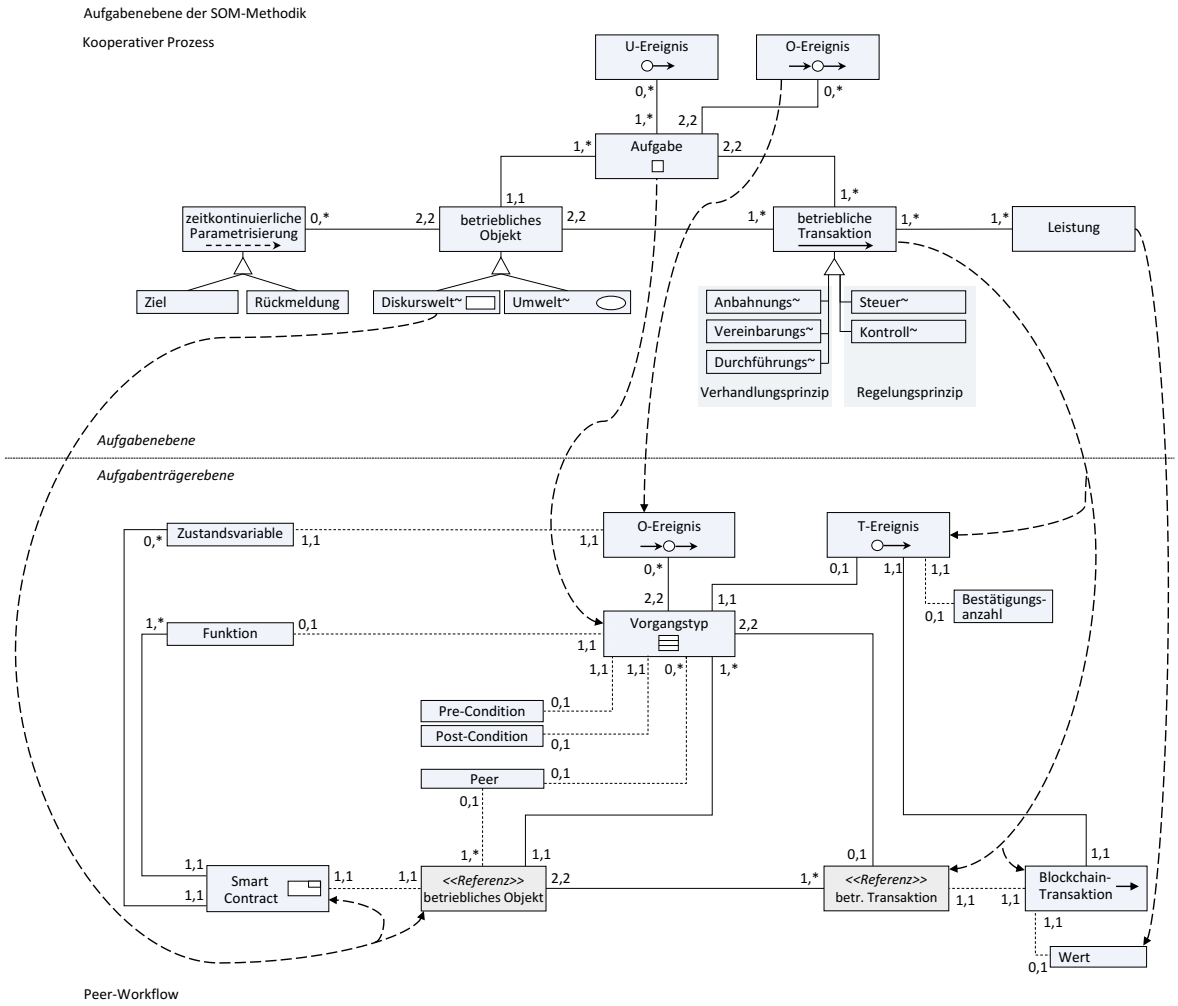



ABBILDUNG 4.10: Metamodelle und Transformation der Prozess-Gestaltung

Das Vorgehen zur Modellierung kooperativer Prozesse entsprechend des Metamodells (Abbildung 4.9) umfasst die Angabe der Struktur und des Verhaltens der Kooperation ausgehend von den Zielbeziehungen der Objekte:

1. **Modellierung des initialen Objektsystems:** Der Teilgraph der anhand von Z/R-Beziehungen verbundenen Objekte des Wertschöpfungsnetzes grenzt die Diskurswelt der Kooperation ab. Die adjazenten Objekte aller Objekte des Teilgraphen bilden die Umweltobjekte der Kooperation. Dieses initiale Objektsystem stellt zusammen mit den definierten Zielen eine globale Planung des Systems der Kooperation dar.
2. **Modellierung kooperierender Objekte in IAS und VES:** Die Realisierung der Planung erfolgt durch die Modellierung der Leistungserstellung als Angabe einer Zerlegung der bestehenden Durchführungstransaktionen und der daran beteiligten Objekte. Die Abbildung der Objekte und Transaktionen erfolgt in einer oder in mehreren Zerlegungsstufen, die iterativ zunächst die Struktur als IAS sowie anschließend das Verhalten als VES erfassen. Das IAS beschreibt Objekte und Transaktionen in einer Struktur, über die Nachrichten zur Koordination und zur Leistungserstellung austauschbar sind. Das VES definiert den ereignisgesteuerten Ablauf des Prozesses. Die Zerlegungen führen nach dem Verhandlungsprinzip unter Anwendung der Zerlegungsregeln der SOM-Methodik zu einem System, dessen autonome Objekte unter nichthierarchischer Koordination zusammenwirken. Eine Zerlegung nach dem Regelungsprinzip ist auf der Ebene des globalen kooperativen Prozesses aufgrund der Autonomie der beteiligten Objekte nicht möglich. Die Zerlegung wird so lange durchgeführt, bis jedes Diskursweltobjekt ein für die Kooperation elementares Objekt beschreibt. Die Innensicht eines solchen Objekts ist nicht-öffentlich und für die Kooperation nicht einsehbar.

Die erreichte Zerlegungsstufe des IAS definiert je Diskursweltobjekt eine Schnittstelle zwischen den globalen und öffentlichen Prozessmodellen der Aufgabenebene sowie den lokalen und privaten Workflow-Modellen einzelner Peers der Aufgabenträgerebene.

#### Peer-Workflow

*Ziel: Lokale Detaillierung und Implementierung einzelner Objekte.*

Ein Peer-Workflow beschreibt die lokal durch einzelne Peers stattfindende Detaillierung und Implementierung des kooperativen Prozesses. Das einem Peer zugeordnete Objekt mit dessen Zerlegungsprodukten wird innerhalb der privaten Struktur-

und Verhaltenssichten bis hin zum Workflow durch weitere hierarchische Zerlegungsschritte detailliert. Die Ebene spezifiziert die Innensicht des einem Peer zugeordneten Objekts anhand von Objekten und Transaktionen als Interaktionsschema eines Peers (IAS-P). Für diese Objekte spezifiziert ein Vorgangstypschema eines Peers (VTS-P) einzelne Vorgänge auf Typebene. Die Auslösung von Vorgängen wird anhand von Blockchain-Transaktionen und Smart Contracts festgelegt.

Das Vorgehen zur Transformation entsprechend der Metamodell-Abbildung (Abbildung 4.10) nimmt zunächst auf die Detaillierung der beiden Schemata der Ebene Bezug (1.) und geht anschließend auf die Implementierung (2.) ein:

1a. **Detaillierung der Struktur anhand des IAS-P:** Das mit dem kooperativen Prozess erfasste Diskursweltobjekt des Peers sowie die dort durch Zerlegung entstandenen Objekte und Transaktionen grenzen die Diskurswelt aus der Sicht eines Peers ab. Bei der Transformation von IAS zu IAS-P werden nach der Übernahme dieser abgegrenzten Diskursweltobjekte alle noch nicht transformierten Diskursweltobjekte des IAS in Umweltobjekte überführt. Andere Objekte der Kooperation sind aus Sicht des Peers prinzipiell ein Teil der Umwelt. Die Modellierung von weiteren Umweltobjekten als Teil des privaten Peer-Prozesses steht dem nicht entgegen. Die Detaillierung wendet nach den Zerlegungsregeln der SOM-Methodik das Verhandlungsprinzip oder das Regelungsprinzip an. Hierdurch entsteht ein System, welches das betriebliche Objekt, z.B. ein Unternehmen, in seiner Innensicht abbildet.

1b. **Detaillierung des Verhaltens anhand des VTS-P:** Für die innerhalb des IAS-P definierten Diskursweltobjekte beschreibt das VTS-P die Auslösung und Implementierung von Vorgangstypen, die sich gemäß der Metamodell-Transformation aus einzelnen Aufgaben ergeben. Die mit der Detaillierung der Struktur gleichermaßen detaillierten betrieblichen Transaktionen bedingen je zwei Vorgangstypen in den anhand der Transaktion verbundenen Objekten. Die Auslösung von Vorgängen beschreiben O-Ereignisse, mit denen Reihenfolgebeziehungen zwischen Vorgangstypen festgelegt werden, sowie T-Ereignisse, welche die Auslösung eines Vorgangstyps durch je eine eingehende Blockchain-Transaktion implementieren. Die operationale Semantik entspricht der eines Petri-Netzes, wobei jeder Vorgangstyp eine Transition und jedes Ereignis eine Stelle sowie damit verbundene Kanten darstellt. Zudem spezifizieren Pre- und Post-Conditions je Vorgangstyp Bedingungen, die zur Auslösung eines Vorgangs erfüllt sein müssen. Conditions werden durch eine logische Funktion definiert, die mit der Implementierung des Smart Contracts

angegeben wird. Hiermit wird die Implementierung der Automatisierung der Vorgangsauslösung und der Aktionensteuerung vorbereitet.

Das weitere Vorgehen betrifft die Automatisierung der Vorgangsauslösung durch die Nachricht einer Blockchain-Transaktion (BC-Transaktion) sowie die Automatisierung der Aktionensteuerung durch Smart-Contract-Funktionen.

2a. **Implementierung von betrieblichen Transaktionen in BC-Transaktionen:** Jeder betrieblichen Transaktion ist auf Typebene eine BC-Transaktion (Abbildung 4.10) zugeordnet, mit der die Kommunikation anhand von Nachrichten zwischen lose gekoppelten Objekten dezentral implementiert wird. Die erfassten Transaktionen spezifizieren die während der Ausführung durchzuführenden BC-Transaktionen. Sofern die Leistung der ursächlichen betrieblichen Transaktion einen innerhalb des Blockchain-Systems erfassbaren Wert transferiert, wird dieser als Attribut der Blockchain-Transaktion modelliert. Die Transaktion löst den damit zusammenhängenden Vorgangstyp anhand des T-Ereignisses aus. Ein T-Ereignis tritt ein, nachdem eine Transaktion zur Laufzeit in den Datenbestand der Blockchain aufgenommen wurde. Das Attribut Bestätigungsanzahl $n$ gibt die Anzahl der einer Transaktion nachfolgenden Blöcke ausgehend von einem Block $B_i$ an, die zur Auslösung des Ereignisses erforderlich sind. Das Ereignis löst den verknüpften Vorgangstyp bei Block $B_{n+i}$ aus. Die Auslösung eines Vorgangstyps kann darüber hinaus von Pre- und Post-Conditions abhängen, die als Funktion eines Smart Contracts implementiert sind.

2b. **Implementierung von betrieblichen Objekten in objektspezifischen Smart Contracts:** Jedem betrieblichen Objekt ist auf Typebene ein Smart Contract zugeordnet (Abbildung 4.10), der den objektinternen Speicher eines Objekts in Zustandsvariablen sowie die Steuerung von Vorgangstypen in Funktionen abbildet. Funktionen können zur Implementierung der Vorgangsauslösung und Aktionensteuerung angegeben werden. Die Auslösung eines Vorgangstyps durch ein Ereignis führt in diesem Fall zum Aufruf einer Funktion, mit der die elementaren Aktionen des ausgelösten Vorgangstyps gesteuert werden. Im Falle von automatisierbaren Aktionen können weitere Funktionsaufrufe erfolgen, die Aktionen innerhalb des Smart Contracts ausführen. Dem steht eine nicht-automatisierte Durchführung von Aktionen nicht entgegen. Nach der Durchführung eines Vorgangs kann eine ereignisgesteuerte Auslösung weiterer Vorgänge erfolgen.

Der Automatisierungsgrad bei zusätzlicher Automatisierung der Aktionen wird

durch das Dreitupel (m,m,m) beschrieben[14]. Dieses beschreibt im Falle von autonom ausgeführten Smart Contracts das Vorliegen einer dezentralen autonomen Organisation (DAO) innerhalb des betrachteten betrieblichen Objekts. Bei vollständig autonomer Ausführung kann die Zuordnung zu einem Peer entfallen. Eine DAO ist typischerweise auf digitale Leistungen beschränkt[15].

#### 4.3.3.3 Metamodelle der Prozess-Ausführung

Die Instanziierung eines kooperativen Prozesses betrifft den Prozess global sowie einzelne Instanzen von Workflows lokal. Die hierfür verwendbaren Modelle ergeben sich durch eine Erweiterung der Metamodelle der Aufgaben- und der Aufgabenträgerebene.

**Prozess-Instanz**

*Ziel: Globale Nachvollziehbarkeit der Ausführung des kooperativen Prozesses.*

Das Verhalten der Instanzen von globalen kooperativen Prozessen wird anhand von einzelnen Instanz-Zuständen zur Laufzeit erfasst. Der hiermit nachvollziehbare Ablauf der Prozess-Ausführung ist von den Beteiligten der Kooperation einsehbar und validiert die korrekte Ausführung des Prozesses innerhalb des dezentralen Systems. Ein Instanz-Zustands-Schema (IZS) beschreibt als Erweiterung des VES Zustände anhand von Ereignissen und Tokens. Die operationale Semantik des IZS entspricht der eines Petri-Netzes. Abbildung 4.11 zeigt das hierfür verwendete Metamodell.

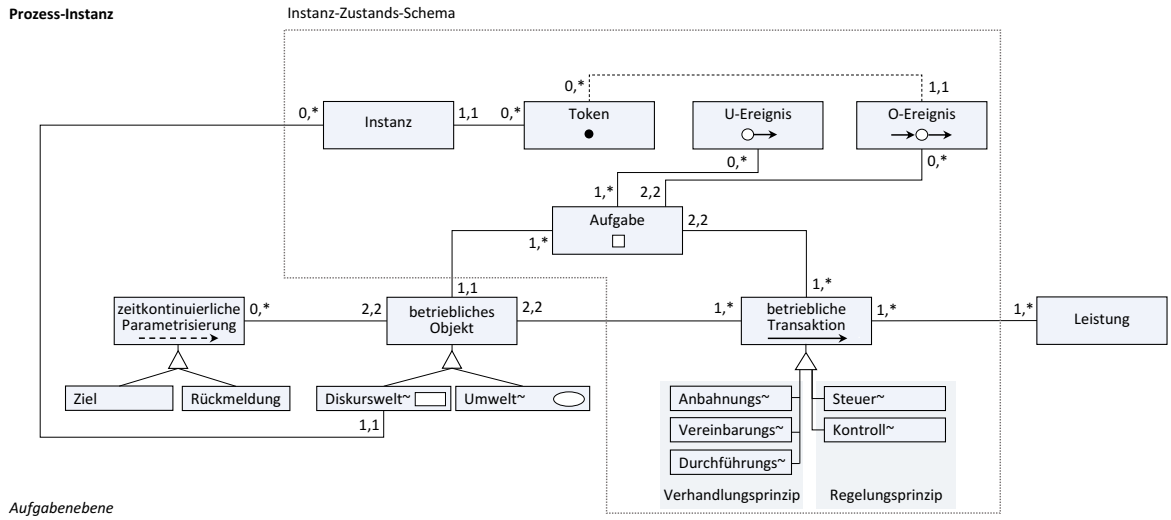



ABBILDUNG 4.11: Metamodell der Prozess-Instanz

[14] siehe Abschnitt 2.1.4.2
[15] siehe Abschnitt 3.4

Das Vorgehen zur Modellierung eines IZS entsprechend des Metamodells umfasst die Angabe von Tokens, die einer Instanz zugeordnet sind:

1. **Angabe einer Instanz:** Jeder kooperative Prozess ist beliebig oft instanziierbar. Eine Instanz eines kooperativen Prozesses wird dabei dem betrieblichen Objekt zugeordnet, das die Kooperation umfasst. Das VES dieser Kooperation wird zur Angabe des Ausführungszustandes übernommen.
2. **Angabe des Ausführungszustandes einer Instanz:** Ein Ausführungszustand bildet eine Prozess-Instanz zu einem Zeitpunkt ab. Jede abgeschlossene Durchführung einer betrieblichen Transaktion führt zu einem Instanz-Zustand, der durch die Durchführung der zugeordneten Blockchain-Transaktionen innerhalb von Peer-Workflows global bekannt ist. Analog zur Semantik eines Petri-Netzes kann der Zustand des Systems anhand von Tokens beschrieben werden. Die Angabe eines Ausführungszustandes erfolgt durch die Zuordnung von Tokens zu Ereignissen, die Stellen repräsentieren.

Die Angabe von Instanzen und Tokens realisiert die Semantik eines gefärbten Petri-Netzes unterscheidbarer Tokens, die jeweils eine Instanz-Zuordnung besitzen.

**Workflow-Instanz**

*Ziel: Lokale Nachvollziehbarkeit der Ausführung anhand von Peer-Workflows.*

Das Verhalten der Instanzen einzelner Peer-Workflows wird anhand von Instanz-Zuständen zur Laufzeit erfasst. Der Ablauf der Prozess-Ausführung ist ausschließlich lokal für das dem Workflow zugeordnete Peer nachvollziehbar. Ein Instanz-Zustands-Schema eines Peers (IZS-P) beschreibt als Erweiterung des VTS-P Zustände anhand von Ereignissen und Tokens. Die operationale Semantik des IZS-P entspricht der eines Petri-Netzes. Abbildung 4.12 zeigt das hierfür verwendete Metamodell.

Das Vorgehen zur Modellierung eines IZS-P entsprechend des Metamodells umfasst die Angabe von Tokens, die einer Instanz zugeordnet sind:

1. **Angabe einer Instanz:** Jeder Peer-Workflow ist beliebig oft instanziierbar. Eine Instanz eines Workflows bezieht sich stets auf das einem Peer zugeordnete betriebliche Objekt sowie dessen Zerlegungsprodukte. Das VTS-P des Peer-Workflows wird zur Angabe des Ausführungszustandes übernommen.

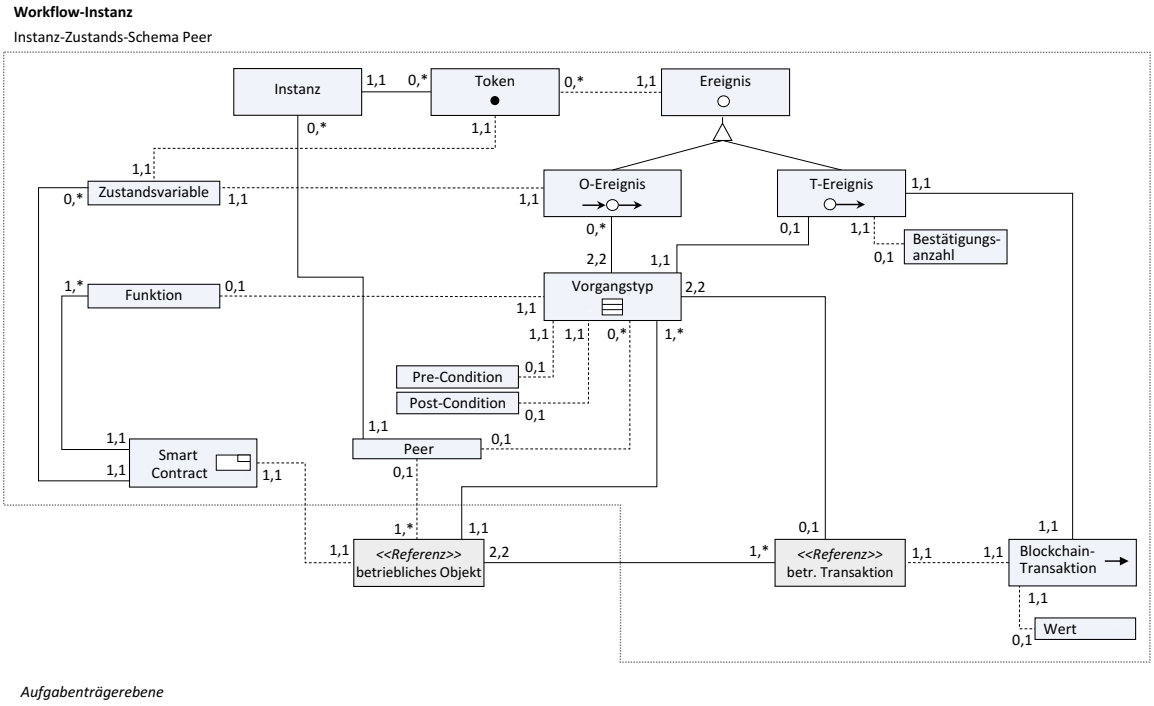



ABBILDUNG 4.12: Metamodell der Workflow-Instanz

2. **Angabe des Ausführungszustandes einer Instanz:** Ein Ausführungszustand bildet eine Workflow-Instanz zu einem Zeitpunkt ab. Jede abgeschlossene Durchführung einer betrieblichen Transaktion führt zu einem Instanz-Zustand, der lokal bekannt und durch die Durchführung der zugeordneten Blockchain-Transaktionen global veröffentlicht und unveränderlich gespeichert wird. Zustände, die sich auf nur lokal sichtbare Objekte beziehen, sind global nicht interpretierbar. Die Angabe eines Ausführungszustandes erfolgt durch die Zuordnung von Tokens zu Ereignissen, die in der Semantik des Petri-Netzes Stellen repräsentieren.

Aufgrund der hierarchischen Zerlegung des kooperativen Prozesses ist das Petri-Netz einer Workflow-Instanz dem Petri-Netz des kooperativen Prozesses hierarchisch untergeordnet. Die hierarchische Zerlegung des Prozesses bedingt hierarchische Petri-Netze.

## 4.4 Kooperationssystem

Für den Entwurf von Prozessen in dezentral organisierten Systemen schlägt dieser Abschnitt einen Ansatz vor, der die Modelle der Gestaltungszeit als verbindliche Grundlage zur kooperativen Gestaltung von kooperativen Prozessen heranzieht. Modelle, die unter Gewährleistung von Integrität und Verbindlichkeit entwickelt und vereinbart werden, besitzen den Charakter eines Vertrages. Die hierfür verwendete Architektur zeigt Abbildung 4.13.

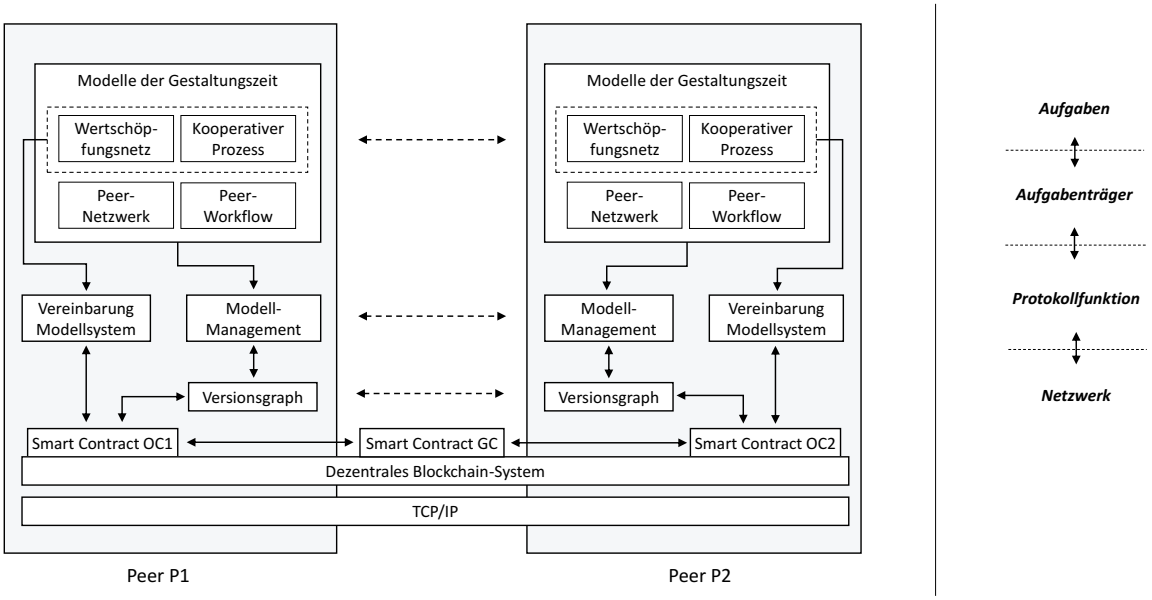



ABBILDUNG 4.13: Architektur des Kooperationssystems

### Modell-Management

Die Entwicklung der globalen Modelle der Aufgabenebene realisiert die Anforderung der gemeinsamen und verteilten Verwaltung als Modell-Management. Die Protokollfunktion greift auf einen Versionsgraphen zurück, der anhand einer dezentralen Blockchain die Integrität und Verbindlichkeit von Modellen gewährleistet. Der Begriff der Integrität bezieht sich damit auf verteilte Modelle, die syntaktisch und semantisch in intendierter Form, ohne unerwünschte Modifikation, vorliegen. Der Begriff der Verbindlichkeit betrifft die verbindliche Zuordnung von Modellen zu Identitäten von Peers, die eine Voraussetzung für die dezentrale Vereinbarung des Modellsystems ist. Ein Versionsgraph hält die Modelle kooperierender Objekte öffentlich oder nicht-öffentlich im Zeitverlauf fest, indem während der Modellierung aufeinanderfolgende Zustände von Modellen zeitdiskret in einem Distributed Version Control System (DVCS) erfasst werden. Teil der Blockchain sind Daten

zur Absicherung der Integrität und Verbindlichkeit der Zustände, die für Kooperationen in einem globalen Smart Contract (GC) sowie für betriebliche Objekte in je einem objektspezifischen Smart Contract (OC) hinterlegt werden.

**Vereinbarung des Modellsystems**

Die verteilte Entwicklung bedingt das Herstellen einer Übereinkunft zwischen den Beteiligten als Vereinbarung des Modellsystems. Die Protokollfunktion adressiert die Anforderung, die innerhalb einer Kooperation konsensual entwickelten Modelle für anschließende Instanziierungen verbindlich festzulegen. Eine notwendige Voraussetzung hierfür ist ein gemeinsamer und vertrauenswürdiger Informationsstand, auf dessen Grundlage eine verteilte Vereinbarung erfolgen kann. Die mit dem Modell-Management verteilten und in ihrer Integrität und Verbindlichkeit abgesicherten Modelle kooperierender Objekte werden anhand eines dezentral koordinierten Abstimmungsverfahrens vereinbart. Das Verfahren entspricht einem Zwei-Phasen-Commit (Steen und Tanenbaum 2017, S. 485), dessen korrekte Ausführung durch den globalen Smart Contract (GC) sichergestellt wird.

Die Architektur beschreibt ein dezentrales System[16], das aufgrund der TCP/IP-basierten Kommunikation ein global verteiltes System, sowie aufgrund der Blockchain-basierten Kommunikation anhand der Protokollfunktionen ein dezentral koordiniertes System ist. Der Aufbau von dezentralen Systemen auf Grundlage einer Blockchain setzt die Verwendung eines dezentralen Blockchain-Systems als globale und offene Infrastruktur zur Kommunikation voraus. Die Verwendung der Blockchain als Kommunikationsschicht erlaubt den Aufbau von skalierenden dezentralen Systemen, deren Datenhaltung in DVCS oder Datenbanken ausgelagert werden kann. Zur Integration können beispielsweise Hash-basierte Verfahren der Integritätssicherung herangezogen werden[17].

### 4.4.1 Modell-Management

Das Modell-Management zur gemeinsamen und verteilten Entwicklung von Prozessen basiert auf einer transaktionalen Erfassung von Modell-Operationen. Kooperierende Objekte der Aufgabenebene, die einen zusammenhängenden Teilgraphen

[16] Abschnitt 3.1.2
[17] Abschnitt 3.4.2.1

durch Ziel- und Rückmeldungsbeziehungen bilden, verwalten eine Transaktionshistorie zur Erstellung eines kooperativen Prozesses in Versionsgraphen. Der gegenseitige Abgleich der Graphen basiert auf einem DVCS. In dieser verteilten Umgebung werden Integrität und Verbindlichkeit der Transaktionen durch Blockchain-Transaktionen zu Smart Contracts abgesichert.

#### 4.4.1.1 Transaktionale Modellierung

Die Erfassung von Modell-Operationen als Commit folgt dem Konzept der transaktionalen Modellierung (Bork und Sinz 2013, S. 31). Eine Entwurfstransaktion überführt ein Modell von einem konsistenten Zustand in einen neuen konsistenten Zustand, unter Anwendung der ACID-Prinzipien. Ein Modell $M$ liegt nach der Anwendung von Modelloperationen durch einen Modellierer als $M'$ vor. Ein Commit zur Definition eines zeitlich abgegrenzten Zustands als Version des Modells kann ausgelöst werden, wenn das editierte Modell gegenüber dem Metamodell syntaktisch valide sowie hinsichtlich der Domäne semantisch valide ist. Die syntaktische Validität ist gegeben, sofern ein betrachtetes Modell eine Extension des hierzu definierten Metamodells ist. Die semantische Validierung überprüft die Konsistenz des Modells gegenüber der Domäne und wird nicht-automatisiert durch den Modellierer als Domänen-Experten bewertet.

Zur globalen Vergleichbarkeit einzelner Zustände sowie zur Vorbereitung der Integritätssicherung wird jeder durchgeführte Commit anhand einer inhaltsbasierten ID identifiziert, die sich als Funktionswert einer Hash-Funktion berechnet (Chacon und Straub 2014). Zur Berechnung des Hash-Wertes wird ein Modell eines Objektsystems ausgehend von einem Objekt $O$ zusammen mit dessen Zerlegungsprodukten betrachtet. Die Berechnung des Hash-Wertes als $HM(M'_O) = V'$ greift auf die nachfolgend in Abschnitt 4.4.1.3 beschriebene Objektbaum-Datenstruktur zurück. Der Hash-Wert repräsentiert den durch $V'$ global identifizierbaren Zustand des Modells $M'$ als Objektsystem, das ein Objekt $O$ und dessen Zerlegungsprodukte umfasst.

Die Durchführung einer Blockchain-Transaktion (BC-T) sichert $V'$ in einem Smart Contract, dessen Funktion $commit(V', O)$ den Wert in einer Zustandsvariable für das betrachtete Objekt ablegt[18]. Für das Absenden der BC-T wird der innerhalb des lokalen Peer-Netzwerk-Modells hinterlegte private Schlüssel $K_S^p$ von Peer $p$ herangezogen, der dem Modellierer zugeordnet ist.

[18] Die Besprechung der Funktion folgt in Abschnitt 4.4.1.4.

Die Durchführung der BC-T ist nach deren Aufnahme in einen Block und nach einer in Abhängigkeit des Blockchain-Systems festzulegenden Bestätigungsanzahl $c$ ($c > 0$) nach $c - 1$ nachfolgenden validen Blöcken[19] abgeschlossen. Mit dem Abschluss der BC-T ist zudem die Entwurfstransaktion abgeschlossen. Abbildung 4.14 fasst die Durchführung einer Entwurfstransaktion zusammen.

### Durchführung einer Entwurfstransaktion

Die Durchführung einer Entwurfstransaktion erfordert die erfolgreiche Ausführung der nachfolgenden Schritte durch den Modellierer und die systemseitige Software-Implementierung.

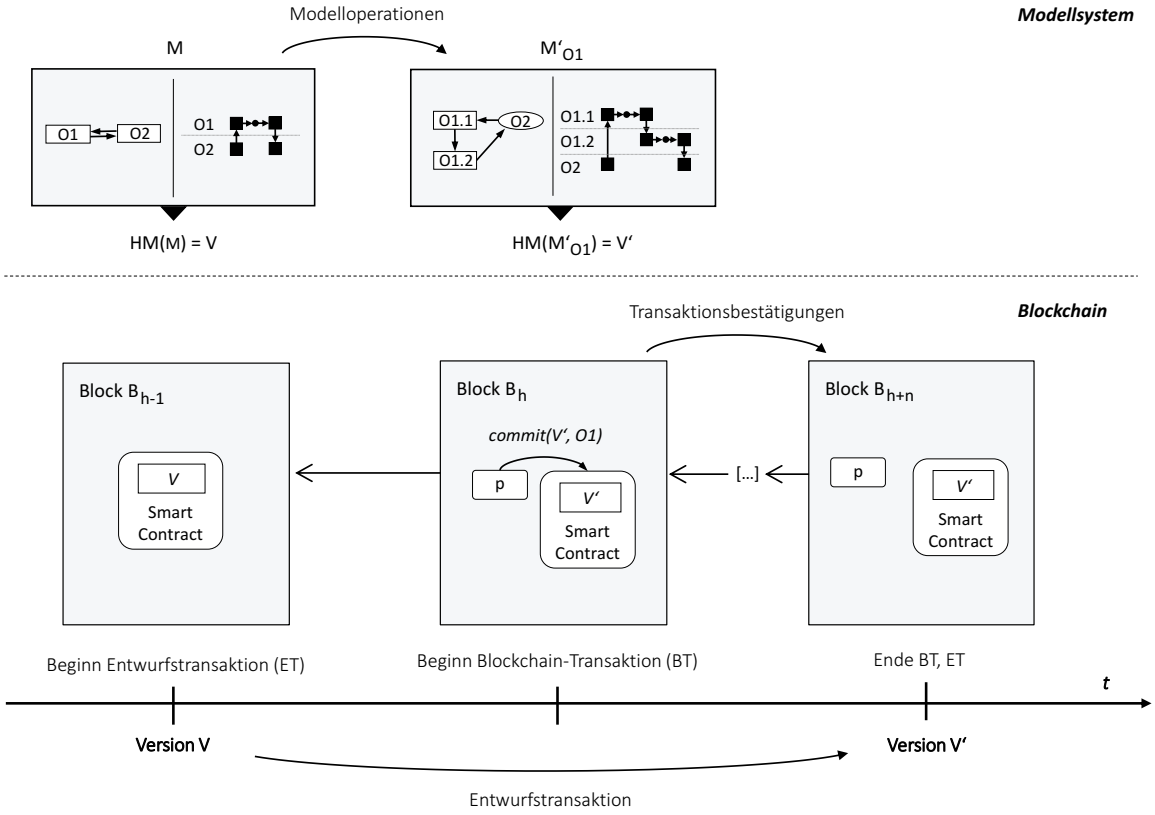



ABBILDUNG 4.14: Transaktionale Modellierung

1. **Beginn der Entwurfstransaktion**.
2. **Modellierer**: Anwenden von Modelloperationen zur Überführung des Modells $M$ in $M'_O$ unter Veränderung des Objekts $O$.
3. **System**: syntaktische Validierung von $M'_O$ gegenüber dem Metamodell von $M'$.

[19] siehe Abschnitt 3.3.3.4

4. **Modellierer**: semantische Validierung der Konsistenz von $M'_O$ gegenüber der Domäne.
5. **Modellierer**: Auslösen einer Commit-Operation.
6. **System**: Berechnung des Hash-Wertes $HM(M'_O) = V'$ eines Modells $M'$ von Objekt $O$.
7. **Beginn der Blockchain-Transaktion**.
   (a) **System**: Broadcast einer Blockchain-Transaktion (BC-T) als Aufruf einer Smart-Contract-Funktion $commit(V', O)$.
   (b) **System**: Abwarten der Aufnahme von BC-T in einen Block $B_h$.
   (c) **System**: Abwarten von $n = c - 1$ nachfolgenden validen Blöcken bis Block $B_{h+n}$.
8. **Ende der Blockchain-Transaktion**.
9. **Ende der Entwurfstransaktion**.

**Verbindlichkeit und Integrität**

Die von der Adresse des Peers ausgehende Blockchain-Transaktion sichert zum einen die Verbindlichkeit, durch die Zuordenbarkeit des Modells in Version $HM(M'_O)$ zur Adresse des Peers $p$, zum anderen die Integrität, durch die Validierbarkeit des Modell-Hash-Wertes $HM(M'_O)$. Das Modell wird hierbei nicht veröffentlicht. Eine Validierung der Verbindlichkeit entspricht einer Überprüfung der Signatur der Transaktion, deren Gültigkeit anhand des öffentlichen Schlüssels $K_P^p$ durch beliebige Teilnehmer des Netzwerks zu einem späteren Zeitpunkt nachgewiesen werden kann. Die Validierung der Integrität durch beliebige Teilnehmer ist nur dann möglich, wenn das Modell verteilt vorliegt. Die Validierung besteht in einer Neuberechnung und einem Vergleich des Modell-Hash-Wertes mit dem in Block $B_h$ hinterlegten Hash-Wert als $HM(M'_O) \stackrel{?}{=} V'$ . Die hierfür notwendige Verteilung von Modellen anhand des Versionsgraphen ist Gegenstand des nachfolgenden Abschnitts.

#### 4.4.1.2 Versionsgraphen zur verteilten Verwaltung von Modellen

Ein Versionsgraph erfasst die Entwicklung eines Prozesses von kooperierenden Objekten über die Zeit. Die Historisierung der Entwurfstransaktionen durch Commit-Operationen bildet zeitdiskrete Zustände, die als Versionen eines Versionskontrollsystems herangezogen werden können.

Der Versionsgraph eines kooperativen Prozesses ist auf die beteiligten Peers verteilt (siehe Abbildung 4.13). Die Modelle liegen als lokale Replikation unter gegebener Integrität und Verbindlichkeit bei denjenigen Peers vor, die den betrieblichen Objekten des Prozesses zugeordnet sind.

Ein Versionsgraph umfasst je Peer global-öffentliche und lokal-private Zweige (Branches). Abbildung 4.15 zeigt den Aufbau anhand eines abstrakten Beispiels. Die kooperative Entwicklung wird durch den Abgleich der global-öffentlich abgelegten Versionen des kooperativen Prozessmodells realisiert.

**Operationen des Versionsgraphen**

Die in Abbildung 4.15 dargestellten Operatoren des Datenobjekts Versionsgraph werden zur Durchführung eines Commits unterstützt. Diese entsprechen Operationen einer Nutzermaschine (Ferstl und Sinz 2013, 218 ff.), die durch die Software-Implementierung auf die Basismaschine eines DVCS abgebildet werden.

Folgende Operatoren werden zur Durchführung von Entwurfstransaktionen benötigt:

- **Commit**: Commit eines Modells $M$ innerhalb eines selektierten Branches.
- **Branch-Commit**: Commit eines Modells $M$ innerhalb eines neu erstellten Branches.
- **Merge-Commit**: Commit einer Zusammenführung der Modelle $M_1$, $M_2$ zweier Branches.

Eine Zusammenführung von Modellen für ein Peer $p1$ ist erforderlich, wenn das Modell des kooperativen Prozesses durch ein Peer $p2$ $(p1 \neq p2)$, verändert wird. Dabei handelt es sich um eine semantische Zusammenführung, die zu einem gegenüber der Domäne konsistenten Modell führen muss und daher nicht-automatisiert durchgeführt wird. Ein Beispiel ist das Hinzukommen einer neuen Zulieferbeziehung innerhalb des globalen kooperativen Prozesses, die mit einem zuvor lokal detaillierten Peer-Workflow zusammengeführt werden muss. Ein Peer ist für die lokale Implementierung eines veränderten kooperativen Prozesses und die Zusammenführung mit dem lokalen Workflow verantwortlich.

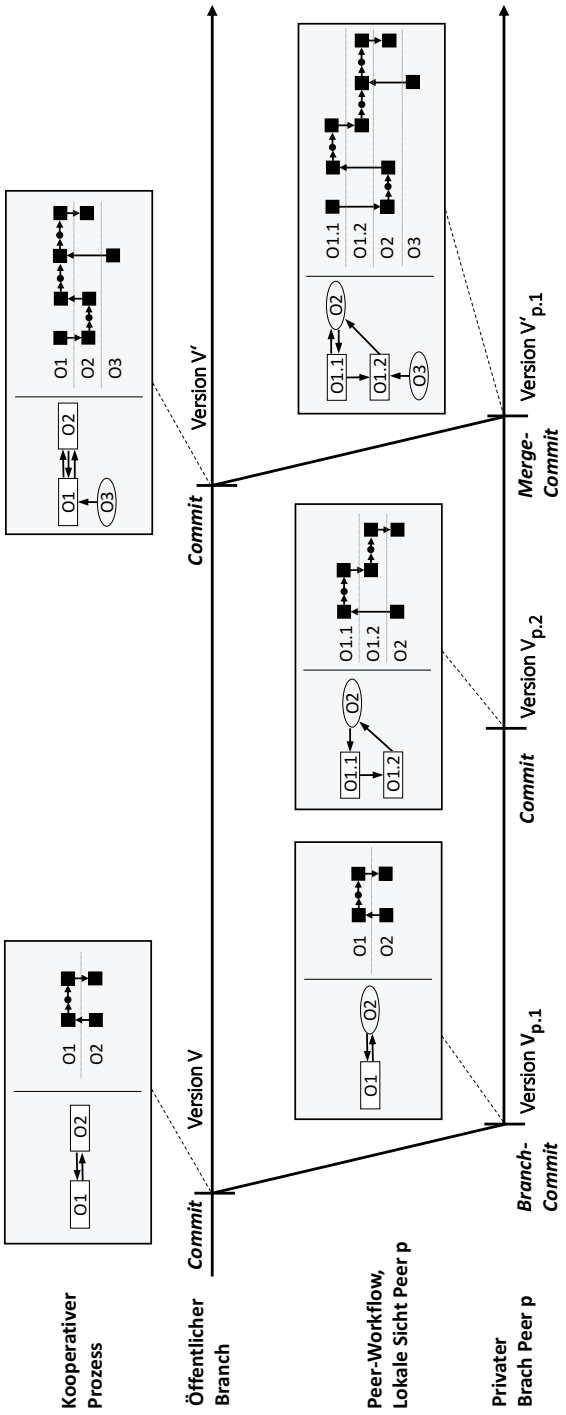



ABBILDUNG 4.15: Commit-Operationen und Branches des Versionsgraphen

Die Auslösung eines Abgleichs der Versionsgraphen erfolgt durch die mit einer Entwurfstransaktion durchgeführte Blockchain-Transaktion. Deren Abschluss in einem Block $B_{h+n}$ ist durch die im Rahmen der Entwurfstransaktion festgelegte Anzahl an Bestätigungen ($c = n + 1$) definiert. Durch $B_{h+n}$ wird der Abgleich zwischen dem sendenden Peer der Blockchain-Transaktion sowie allen weiteren Peers ausgelöst.

Diese führen jeweils anhand des DVCS Checkout- und Merge-Operationen aus, um die Version des Commits lokal zu replizieren und zusammenzuführen.

#### 4.4.1.3 Konsistenz und Integrität verteilter öffentlicher und privater Modelle

Die syntaktische Integrität einzelner Modelle und die Konsistenz der globalen und lokalen Modelle wird durch eine Datenstruktur sichergestellt, welche die globalen und öffentlichen Modelle der Aufgabenebene mit den lokalen und privaten Modellen der Aufgabenträgerebene anhand von Hash-Werten verknüpft. Die zusammen mit einem öffentlichen Modell hinterlegten Hash-Werte entsprechen einer Zuordnung der Objekte des Modells zu den Objekten privater Modelle der Aufgabenträgerebene. Die Datenstruktur stellt die Kompatibilität der Modelle anhand von Hash-Werten der darin enthaltenen Objekte sicher.

**Objektbaum-Datenstruktur**

Die aus der Objektzerlegung entstandene Hierarchie von betrieblichen Objekten definiert die Ebenen des Baumes. Ein Knoten ist ein Tupel $K_O = (O, H(O), Z_O)$. $K_O$ definiert ein Objektsystem, bestehend aus einem Objekt $O$ und dessen Zerlegungsprodukten. In einen Knoten $K_O$ gehen ein:

1. ein betriebliches Objekt $O$,
2. dessen Hash-Wert als Funktionswert einer Hash-Funktion $H(O)$,
3. eine Menge an referenzierten Zerlegungsprodukten $Z_O$, die je Zerlegungsprodukt $OZ$ eine Objekt-Referenz als Referenz-Hash-Wert $RH(OZ) \in Z_O$ enthält. Dieser bestimmt sich als $RH(OZ) = H(H(OZ)||Z_{OZ})$ durch den Nachfolgeknoten.

Ein Referenz-Hash-Wert $RH(OZ)$ eines referenzierten Objekts $OZ$ bestimmt sich somit aus den Hash-Werten des Knotens $K_{OZ}$. Die Referenz-Hash-Werte bilden einen Hash-Baum.

Ein Modell $M$ in Form eines Objektsystems wird durch einen Knoten des Baumes zusammen mit dessen Zerlegungsprodukten repräsentiert. Der Hash-Wert eines Modells $HM(M_O)$ berechnet sich durch die Objekte $O$ und deren Zerlegungsprodukte $Z_O$ in der Form $HM(M_O) = RH(O) = H(H(O)||Z_O)$ als Referenz-Hash-Wert.

*Verknüpfung und Konsistenz von öffentlichen und privaten Modellen*

Abbildung 4.16 zeigt die Datenstruktur anhand eines Beispiels. Der Wurzelknoten des Baumes enthält die betrieblichen Objekte der initialen Zerlegungsstufe und repräsentiert das Modell anhand des Hash-Wertes $HM(M_O) = RH(O)$. Dieses global übergeordnete Modell stellt einen kooperativen Prozess dar, dem lokale Peer-Workflow-Modelle für die innerhalb des kooperativen Prozesses modellierten Objekte untergeordnet sind (siehe Abbildung 4.7).

Ein Peer-Workflow-Modell repräsentiert ein Objektsystem eines Objekts $OZ$, hier $O1, O2$, das innerhalb des Prozessmodells einem Objekt $O$ untergeordnet ist. Die Datenstruktur verknüpft beide Modelle, indem in den Knoten $K_O$ ein Referenz-Hash-Wert $RH(OZ)$ für $OZ$ aufgenommen wird. In Abbildung 4.16 werden die Modelle der Objekte $O1$, $O2$ durch $HM(M_{O1}) = RH(O1)$ bzw. $HM(M_{O2}) = RH(O2)$ repräsentiert. Global ist damit eine Prüfsumme der lokalen Modelle bekannt, die zur Referenzierung der Objekte verwendet werden kann. Lokal liegt einem Peer zudem $K_{OZ}$ mit den darunterliegenden Objekten vor, d.h. die Datenstruktur repräsentiert lokal beide Modelle in einer Hierarchie.

Durch Anwendung von Modelloperationen auf die lokale Datenstruktur innerhalb des globalen oder lokalen Teils des Baumes wird die Konsistenz und Kompatibilität der Modelle gewahrt. Darüber hinaus kann die nachfolgend beschriebene Validierung der Datenstruktur unter Einbeziehung globaler und lokaler Modelle ohne Kenntnis der lokalen Modelle erfolgen, indem die Objekte lokaler Modelle aus der Datenstruktur entfernt werden.

*Validierung der Datenstruktur*

Die Datenstruktur stellt eine Variante der Merkle-Baum- und Patricia-Trie-Datenstrukturen (Nakamoto 2008a; Wood 2014) dar, die Objekthierarchien repräsentieren. Die Integrität und Konsistenz eines Objekts $O$ kann hier mit jedem untergeordneten Zerlegungsprodukt $OZ$ nachgewiesen werden. Die Führung des Nachweises besteht in der Neuberechnung des Hash-Wertes $H(OZ) = V$ und einer Validierung der Übereinstimmung von $V$ mit dem in $O$ hinterlegten Wert. Darüber hinaus ist eine Validierung der Datenstruktur über mehrere Ebenen auf Basis aller $H(O)$ und $H(OZ)$ ohne Kenntnis der enthaltenen Objekte möglich. Die Validierung eines Baumes, aus dessen Knoten die Objekte $O$ entfernt wurden, ist ausgehend von den Hash-Werten $H(O)$ der Blattknoten nach dem Verfahren eines Merkle-Proofs[20] durchführbar.

[20] siehe Abschnitt 3.3.1.5

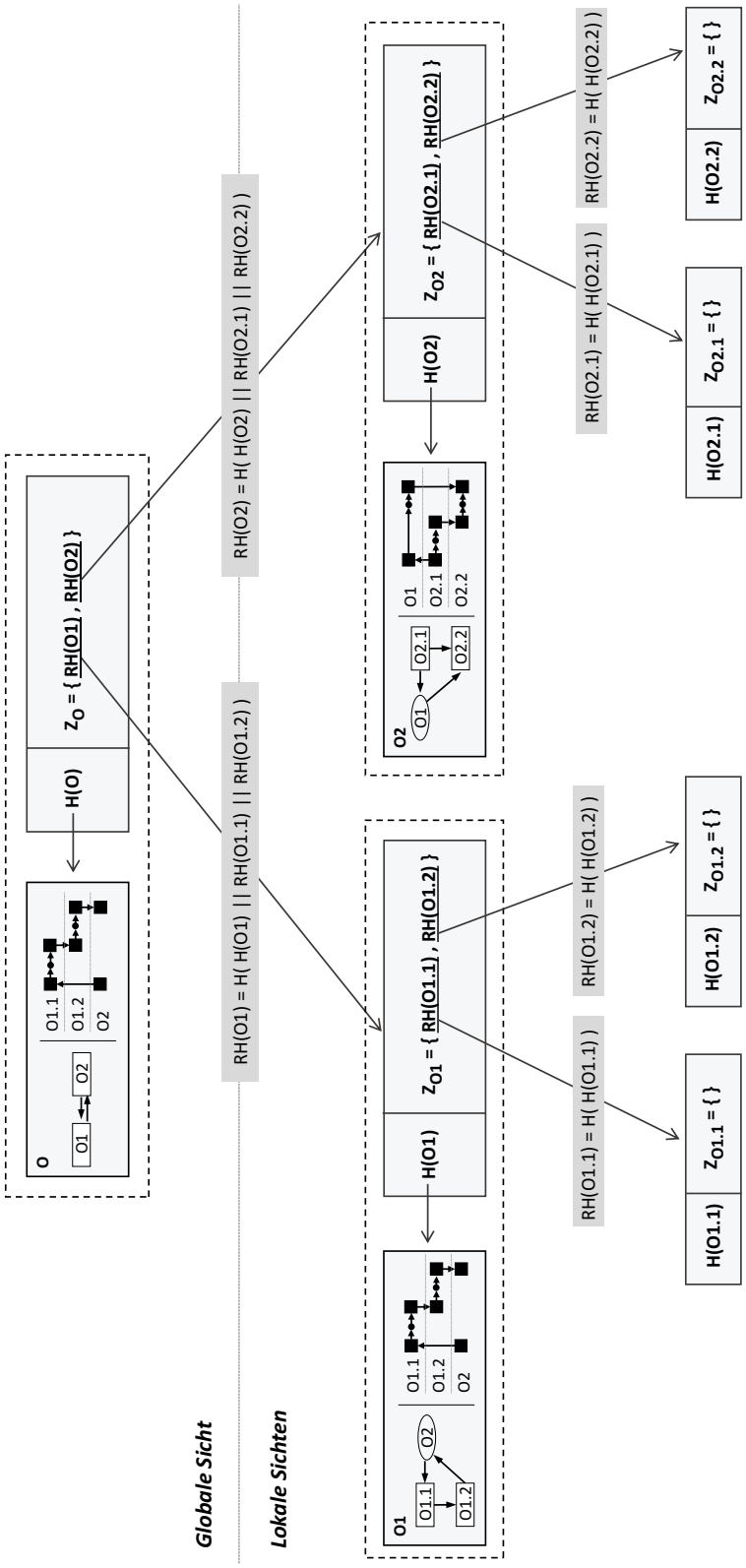



ABBILDUNG 4.16: Objektbaum-Datenstruktur

Die Validierung beginnt mit den Blattknoten. Für einen Blattknoten $L$ entspricht die Menge $Z_L$ der leeren Menge. Das Verfahren berechnet iterativ je Baum-Ebene und Knoten die Referenz-Hash-Werte der dort referenzierten Knoten neu und validiert deren Übereinstimmung mit den innerhalb des Knotens hinterlegten Werten. Die Validität eines Knotens $K_O$ wird nachgewiesen, indem je referenziertem Objekt $OZ$ der Wert $RH(OZ)* = H(H(OZ)||Z_{OZ})$ neu berechnet und gegenüber dem hinterlegten Wert $RH(OZ)$ validiert wird als $RH(OZ)* \stackrel{?}{=} RH(OZ)$. Bei Übereinstimmung der Werte aller referenzierten Knoten ist $K_O$ validiert. Das Verfahren wird für alle weiteren Knoten der Baum-Ebene wiederholt. Nach Validierung einer Ebene wird das Verfahren für die nächsthöhere Ebene wiederholt. Die erfolgreiche Anwendung bis zum Wurzelknoten weist die Validität des Baumes nach.

#### 4.4.1.4 Absicherung der Integrität und Verbindlichkeit in Smart Contracts

Die anhand der Objektbaum-Datenstruktur berechneten Hash-Werte eines Modells $HM(M_O)$ gehen als Bestandteil einer Entwurfstransaktion in einen Smart Contract ein. In Abhängigkeit davon, ob die Entwurfstransaktion ein globales oder lokales Modell betrifft, wird der globale Smart Contract (CG) oder ein objektspezifischer Smart Contract (OC) herangezogen[21].

**Objektspezifischer Smart Contract zur Absicherung lokaler Modelle**

Ein objektspezifischer Smart Contract (OC) repräsentiert ein betriebliches Objekt, das einem Peer zugeordnet ist. Quellcode 4.1 zeigt den Beginn des Smart Contracts[22]. Die Zustandsvariablen `object` und `peer` legen das Objekt und die Adresse des zugeordneten Peers für die Durchführung nachfolgender Commits fest. Ein Aufruf der commit-Funktion sichert einen übergebenen Hash-Wert unter Vergabe einer ID in der Zustandsvariable `versionHashValues`. Die öffentlich zugreifbare Variable erlaubt Abfragen von Hash-Werten je Objekt und Version. Neben der Sicherung der Integrität und Verbindlichkeit wird ein OC zudem für das Instanz-Monitoring des Ausführungssystems herangezogen[23].

```
1 pragma solidity ^0.5.3;
2 /// @title Objektspezifischer Smart Contract
3 contract ObjectSpecificContract {
4     // Objekt
5     bytes16 public object;
```

[21] siehe Abbildung 4.13
[22] Der vollständige Quellcode ist in Anhang A abgebildet.
[23] Die Besprechung des Ausführungssystems folgt in Abschnitt 4.5.1.3.

```
6     // Peer-Zuordnung
7     address public peer;
8     // Zuordnung: Objekt-Zerlegungsprodukt -> Versions-Hash-Werte
9     mapping(bytes16 => VersionHashValues) public versionHashValues;
10    // Datentyp VersionHashValues: Speicherung von Hash-Werten je
          Version
11    struct VersionHashValues {
12       uint16 nVersions;                        // Anzahl Versionen
13       mapping(uint16 => bytes32) hashValues;  // Hash-Werte
14       mapping(uint16 => uint16) commitProcedureID;  // Commit-IDs
15    }
16    /// Einleiten eines lokalen Commits
17    /// @param hashValue Modell-Hash-Wert des uebergebenen Objekts
18    /// @param object Objekt, von dem der Commit ausgeht
19    function commit(bytes32 hashValue, bytes16 object) public {
20       // Transaktionsabsender ueberpruefen
21       if (msg.sender != peer) {
22          return;
23       }
24       // Sequenz-ID der Version (0 < versionSequenceID <= nVersions)
25       uint16 versionSequenceID = ++versionHashValues[object].nVersions
             ;
26       // ID des letzten globalen Commits ermitteln
27       uint16 commitProcedureID =
28           GlobalContract(globalContract).getCommitProcedureID(object);
29       // Zuweisung Versions-Hash-Wert
30       VersionHashValues storage vhv = versionHashValues[object];
31       vhv.hashValues[versionSequenceID] = hashValue;
32       vhv.commitProcedureID[versionSequenceID] = commitProcedureID;
33    }
34    // [...]
35 }
```

QUELLCODE 4.1: Objektspezifischer Smart Contract

### Globaler Smart Contract zur Absicherung kooperativer Modelle

Der globale Smart Contract (GC) repräsentiert die anhand des globalen Wertschöpfungsnetzes gebildeten Kooperationen und sichert deren Zustand. GC realisiert Funktionen zur Verwaltung von Objekt-Peer-Zuordnungen sowie die Durchführung und Koordination des Commit-Verfahrens für globale Modelle.

*Verwaltung von Objekt-Peer-Zuordnungen*

Eine Objekt-Peer-Zuordnung erfolgt durch den Aufruf einer Funktion `assignObject(bytes16 object, bytes32 name)`, die für die Zeichenfolge des übergebenen semantischen Objekts eine Zuordnung zum Absender der aufrufenden Transaktion in der Zustandsvariable `objectMap` hinterlegt[24]. Peer und Objekt stehen zueinander in einer 1:n-Relation. Eine hierzu gehörende Funktion `objMap(address peer)` gibt die Objektzuordnung für die angegebene Adresse eines Peers als Array zurück. Die Abfrage einzelner Zuordnungen ist anhand der Methode `knownObjects(bytes16 object)` möglich.

*Auslösen des verteilten Commit-Verfahrens*

Den Quellcode-Ausschnitt zur Auslösung des Commits zeigt Quellcode 4.2[25]. Die Funktion ermittelt das dem Objekt zugeordnete Peer und löst einen verteilten Commit aus, sofern die beteiligten kooperierenden Objekte keinen Commit durchführen. Ist dies der Fall, erfolgt die Einleitung eines verteilten Commits durch die Vergabe einer `commitProcedureID`, das Setzen des Zustandes *Wait* und die Hinterlegung des übergebenen Hash-Wertes des Modells. Der Hash-Wert wird im Falle der Ausführung des verteilten Commits persistent gespeichert. Die Ausführung zur Realisierung der Protokollfunktion „Vereinbarung des Modellsystems“ wird nachfolgend betrachtet.

```
1  pragma solidity ^0.5.3;
2  /// @title Globaler Smart Contract
3  contract GlobalContract {
4      /// Einleiten eines globalen Commits
5      /// @param hashValue Modell-Hash-Wert des uebergebenen Objekts
6      /// @param object Objekt, von dem der Commit ausgeht
7      function commit(bytes32 hashValue, bytes16 object) public {
8          // Transaktionsabsender ueberpruefen
9          uint16 peerID = knownObjects[object].peerID;
10         if (peers[peerID].peer != msg.sender) {
11             return;
12         }
13         uint16 coID = knownObjects[object].collaborationID;
14         uint16 cpID = knownCollaborations[coID].commitProcedureID;
15         // Commit bei Vorliegen des Zustands Init beginnen
16         if (commitProcedures[cpID].state == CommitProcedureState.Init)
               {
```

[24]Der vollständige Quellcode ist in Anhang A abgebildet.
[25]Der vollständige Quellcode ist in Anhang A abgebildet.

```
17              cpID++;
18              commitProcedures[cpID].state = CommitProcedureState.Wait;
19              commitProcedures[cpID].modelHashValue = hashValue;
20              knownCollaborations[coID].commitProcedureID = cpID;
21              emit VoteRequest(coID, cpID, object);
22          }
23      }
24      // Event zur UEbermittlung der Vote-Request-Nachricht
25      event VoteRequest(uint16 indexed collaborationID, uint16
            commitProcedureID,
26           bytes16 object);
27      // [...]
```

QUELLCODE 4.2: Globaler Smart Contract

### 4.4.2 Vereinbarung des Modellsystems

Die Vereinbarung eines für die weitere Entwicklung und die Instanziierung heranzuziehenden globalen Modells erfolgt unter Beteiligung der innerhalb des Prozesses kooperierenden Objekte. Der Commit einer Entwurfstransaktion basiert für globale Modelle daher auf einem verteilten Commit-Verfahren, dessen erfolgreiche Durchführung einer fachlichen Vereinbarung eines Modells entspricht.

#### 4.4.2.1 Koordination des verteilten Commits

Der verteilte Commit wird im Sinne eines Zwei-Phasen-Commit realisiert, dessen Koordinator der globale Smart Contract (CG) ist. Die Kommunikation zwischen den beteiligten Objekten und dem Koordinator erfolgt durch Funktionsaufrufe in Form von Blockchain-Transaktionen sowie das Aussenden von global empfangbaren Events durch GC[26].

Aus Sicht des Koordinators umfasst das Commit-Verfahren einer Entwurfstransaktion die Zustände eines verteilten Commit-Verfahrens (Steen und Tanenbaum 2017, S. 485).

[26] Der Quellcode des Commit-Verfahrens ist als Teil des GC in Anhang A abgebildet.

**Durchführung des verteilten Commits:**

1. **Init**: initialer Zustand zur Auslösung des Commits. Die Auslösung durch den Funktionsaufruf `commit(bytes32 hashValue, bytes16 object)` initiiert das Senden eines `VoteRequest`-Events und den Übergang in den Zustand Wait.
2. **Wait**: Warten auf das Eingehen von `Vote-Abort`- und `Vote-Commit`-Nachrichten. Die beteiligten Objekte senden eine Nachricht `Vote-Abort` zur Signalisierung des Abbruchs der Entwurfstransaktion oder `Vote-Commit` zur Signalisierung zur Bereitschaft der Durchführung der Entwurfstransaktion. Das Eingehen einer `Vote-Abort`-Nachricht führt zur Auslösung eines `VoteAbort`-Events und führt in den Zustand *Abort* über. Das Eingehen von `Vote-Commit`-Nachrichten von allen Beteiligten löst ein `VoteCommit`-Event aus und führt zum Übergang in den Zustand *Commit*.
3. (a) **Alternative 1 - Abort:** Ein Abbruch des Commits verwirft den beim Koordinator vorliegenden Hash-Wert des Modells. Das Auslösen eines `GlobalAbort`-Events führt zum Abbruch des Commits der Entwurfstransaktion bei den Beteiligten.

   (b) **Alternative 2 - Commit:** Die Durchführung des Commits legt den Hash-Wert in einer Zustandsvariable dauerhaft ab. Das Auslösen eines `GlobalCommit`-Events löst eine Validierung des globalen Modells in den Versionsgraphen aller Beteiligten gegenüber dem Hash-Wert aus und markiert den Commit im Erfolgsfall als valide Version.

Der Zustand des Verfahrens ist zu jedem Zeitpunkt in den Zustandsvariablen des GC global nachvollziehbar.

#### 4.4.2.2 Vereinbarung anhand des Commit-Verfahrens

Das Commit-Verfahren einer Entwurfstransaktion umfasst aus Systemsicht die beschriebenen Zustände eines Zwei-Phasen-Commit-Verfahrens. Zur validen Durchführung des Verfahrens aus der fachlichen Sicht eines beteiligten Objekts kommen die folgenden syntaktischen und semantischen Evaluierungen hinzu:

1. **Init**: Der Empfang eines `VoteRequest`-Events löst die Beurteilung und Abstimmung des Commits aus.

   (a) **Evaluierung:** Die Evaluierung umfasst die Validierung der Integrität und Verbindlichkeit sowie die semantische Validierung des über den Versionsgraphen verteilten Modells.

i. **Überprüfung der Integrität und Verbindlichkeit**: Das Modell wird anhand der Objektbaum-Datenstruktur gegenüber dem hinterlegten Hash-Wert validiert. Das Ergebnis der Überprüfung ist bei übereinstimmenden Hash-Werten positiv. Die Verbindlichkeit ist in diesem Fall anhand des in GC hinterlegten Objekts gegeben.

ii. **Semantische Evaluierung**: Die semantische Evaluierung beurteilt das Modell aus fachlicher Sicht im Kontext der Domäne. Die Beurteilung liegt nach dem Autonomieprinzip in der Verantwortung der Beteiligten, die den Prozess als Domänen-Experten entwerfen und implementieren. Eine Zustimmung zur Weiterentwicklung des evaluierten Prozesses führt zu einem positiven Ergebnis der Beurteilung.

(b) **Abstimmung:** Ein positives Ergebnis beider Beurteilungen führt zum Absenden von `Vote-Commit` durch den Aufruf einer Smart-Contract-Funktion `voteGlobalCommit(bytes16 object, bool voteCommit)` unter Angabe des Objekts und `voteCommit = TRUE`. Andernfalls führt `Vote-Abort` mit `voteCommit = FALSE` zur Signalisierung eines Abbruchs.

2. **Ready**: Zustand in Vorbereitung des Commits oder Abbruchs der Entwurfstransaktion. Der Empfang eines `GlobalAbort`-Events löst den Übergang in den Zustand *Abort* aus. Ein eingehendes `GlobalCommit`-Event führt in den Zustand *Commit* über.

3. (a) **Alternative 1 - Abort:** Der Abbruch des Commits verwirft das innerhalb des Versionsgraphen hinterlegte Modell und bricht die Entwurfstransaktion ab.

   (b) **Alternative 2 - Commit:** Die Durchführung des Commits validiert das innerhalb des Versionsgraphen hinterlegte Modell und führt zum Abschluss der Entwurfstransaktion.

Die Durchführung von Entwurfstransaktionen anhand der Protokollfunktion zur Vereinbarung von Modellen führt zur konsensualen Entwicklung von syntaktisch und semantisch validen Modellen. Die Instanzen dieser Modelle sind anhand des Ausführungssystems abbildbar.

## 4.5 Ausführungssystem

Dieser Abschnitt betrifft die Prozess-Ausführung zur Abbildung von global nachverfolgbaren Instanzen auf Basis der zuvor vereinbarten Prozessmodelle. Das Ziel des Ausführungssystems ist eine von allen Beteiligten verteilt nachvollziehbare Ausführung von kooperativen Prozessen. Hiermit wird die Kontrolle während und nach der Ausführung einzelner Aufgaben in den Phasen Durchführung bzw. Kontrolle adressiert. Darüber hinaus ist ein nachgeordnetes Ziel die Unterstützung der Koordination während der Phase der Durchführung.

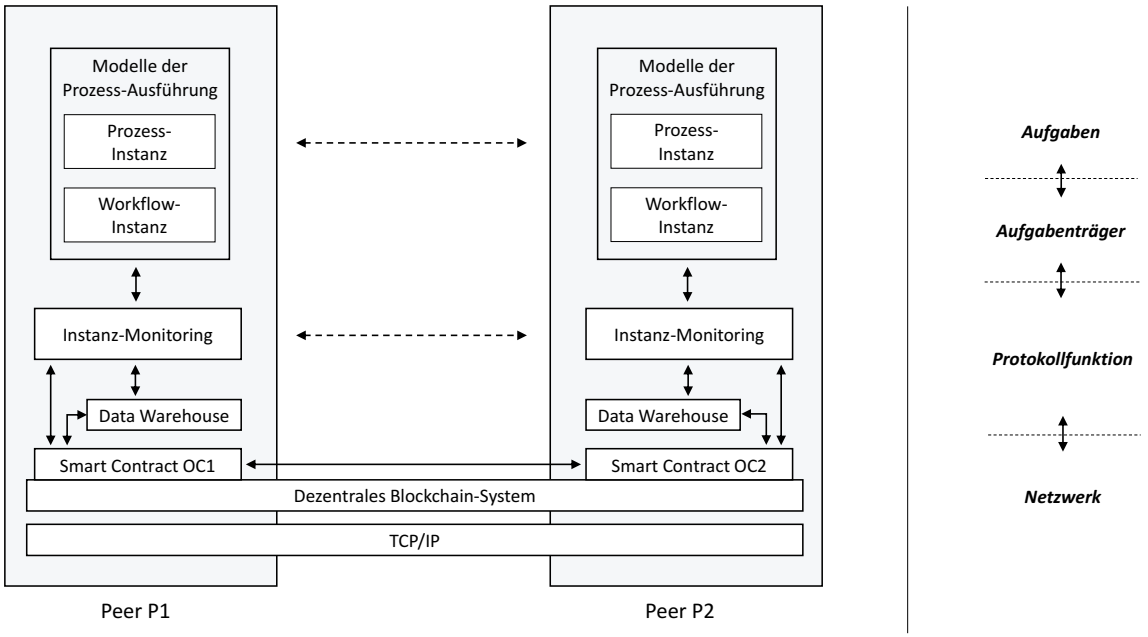



ABBILDUNG 4.17: Architektur des Ausführungssystems

Abbildung 4.17 zeigt die Architektur des Ausführungssystems. Die Modelle der Prozess-Ausführung umfassen auf Aufgabenebene das aus dem kooperativen Prozess abgeleitete globale Modell einer Prozess-Instanz sowie auf Aufgabenträgerebene eine Reihe von Modellen, die Workflow-Instanzen einzelner Peer-Workflows abbilden[27]. Die Protokollfunktion „Instanz-Monitoring“ realisiert die Kontrolle während und nach der Ausführung, indem die Petri-Netz-Semantik der beiden Modelle zur Angabe des Ausführungszustandes und zur darauf basierenden Erstellung von Protokollen herangezogen wird. Die Integration der Funktion „Instanz-Monitoring“ über die Objekte mehrerer Peers hinweg, basiert auf objektspezifischen

[27] siehe Abschnitt 4.3.3.3

Smart Contracts (OC), die Zustände einzelner Instanzen auf Basis von Ereignissen je Objekt verwalten und global zur Verfügung stellen. Eine Historisierung von Instanz-Zuständen im Zeitverlauf auf der Basis eines Data-Warehouse-Systems protokolliert die Prozessausführung zur Überprüfung der fachlichen Validität.

### 4.5.1 Instanz-Monitoring

Jeder Teilnehmer eines dezentral organisierten Systems ist während der Ausführung von kooperativen Prozessen auf Informationen hinsichtlich der verteilt eintretenden Ausführungszustände angewiesen. Die Informationsstände einzelner Instanzen von kooperativen Prozessen müssen bei allen Teilnehmern in einheitlicher Form vorliegen, um den Zielerreichungsgrad von Aufgaben zu bewerten und darauf basierend Entscheidungen hinsichtlich des weiteren Prozessablaufs abzuleiten.

#### 4.5.1.1 Modellierung von Ausführungszuständen

Der Ausführungszustand von Prozess- und Workflow-Instanzen wird als Instanz-Zustands-Schema eines Prozesses (IZS) bzw. eines Peer-Workflows (IZS-P) erfasst. Eine Instanz definiert sich durch die nachfolgend erläuterten Elemente der zugehörigen Metamodelle (Abbildungen 4.11 bzw. 4.12). Die dort definierte Syntax wird zur Beschreibung der Ausführung um die operationale Semantik von Petri-Netzen ergänzt und greift dabei auf die Petri-Netz-Semantik des Vorgangs-Ereignis-Schemas (VES) zurück (Ferstl und Sinz 2013, 206 f.).

**Prozess-Instanz der Aufgabenebene**

Zur Modellierung des Ausführungszustandes einer Prozess-Instanz als IZS werden die folgenden Meta-Objekte herangezogen:

1. **Instanz:** Ein Diskursweltobjekt der kooperierenden Objekte eines Prozesses kann die Instanziierung des Prozesses auslösen. Einem Diskursweltobjekt sind damit 0,* Instanzen zugeordnet, während sich eine Instanz auf genau ein Objekt bezieht.
2. **Token:** Der Zustand einer Instanz wird anhand von 0,* Token beschrieben. Der Semantik eines gefärbten Petri-Netzes folgend, bildet eine Menge von 0,* Token den Zustand einer Instanz zu einem Zeitpunkt statisch in einem Modell ab. Ein Token kann stets auf eine Instanz bezogen werden.
3. **O-Ereignis:** Ein objektinternes Ereignis tritt nach der Durchführung einer Aufgabe als Nachereignis ein und löst eine nachfolgende Aufgabe aus, sofern

deren Pre-Condition die Auslösung nicht verhindert. In der Semantik eines Petri-Netzes entspricht die Durchführung der mit einer betrieblichen Transaktion verknüpften Aufgabe dem Schalten eines Übergangs. Das Vorliegen eines O-Ereignisses definiert damit einen Ausführungszustand, der den Abschluss einer Transaktion und den Beginn einer weiteren Transaktion beschreibt. Das Metamodell sieht daher die Zuordnung eines Tokens zu einem O-Ereignis vor, das eine Stelle eines Petri-Netzes repräsentiert. Dem folgend, bezieht sich ein O-Ereignis zur Zustandsbeschreibung auf 0,* Token.

Das IZS einer Prozess-Instanz auf Aufgabenebene bildet damit den Ausführungszustand von betrieblichen Transaktionen ab.

**Workflow-Instanz der Aufgabenträgerebene**

Die Modellierung des Ausführungszustandes der Workflow-Instanz eines Peers als IZS-P basiert auf den Meta-Objekten des IZS und erweitert diese in Verbindung mit Blockchain-Transaktionen um ein T-Ereignis[28]. Die folgenden Meta-Objekte werden unter Berücksichtigung der Semantik der Aufgabenträgerebene herangezogen:

1. **Instanz:** Ein Peer kann die Instanziierung des Workflows auslösen. Einem Peer sind 0,* Instanzen zugeordnet, während sich eine Instanz auf genau ein Peer bezieht.
2. **Token:** Analog zu IZS wird der Zustand einer Instanz durch 0,* Token beschrieben, während ein Token stets auf eine Instanz beziehbar ist.
3. **Ereignis:** Ein Ereignis liegt spezialisiert als O-Ereignis oder T-Ereignis vor. Einem IZS entsprechend ist ein Token stets einem Ereignis zugeordnet, um eine Zustandsrepräsentation analog zu den Stellen eines Petri-Netzes abzubilden. Ein Ereignis umfasst 0,* Token.
   (a) **O-Ereignis:** Ein O-Ereignis definiert in Übereinstimmung zur Aufgabenebene den Abschluss einer betrieblichen Transaktion sowie den Beginn einer weiteren betrieblichen Transaktion.
   (b) **T-Ereignis:** Mit der Abbildung von betrieblichen Transaktionen auf Blockchain-Transaktionen (BC-T) wird die Unterscheidung zwischen objektinternen Ereignissen (O-Ereignis) und Ereignissen der Blockchain-Transaktion (T-Ereignis) notwendig. Ein T-Ereignis tritt ein, wenn die

[28] siehe Peer-Workflow (Abschnitt 4.3.3.2), Metamodell (Abbildung 4.12).

Durchführung einer Blockchain-Transaktion zur Übertragung der Informationen einer betrieblichen Transaktion abgeschlossen ist. Das Ereignis entspricht einem Nachereignis einer Aktion „BC-T senden“ und einem Vorereignis einer Aktion „BC-T empfangen“. Das Ereignis wird zur Automatisierung der Vorgangsauslösung sowie ggf. auch der Aktionensteuerung eingesetzt.

Das IZS-P einer Workflow-Instanz bildet die anhand von Blockchain-Transaktionen nachverfolgbaren Ausführungszustände eines Peers ab. Ein IZS-P ist einem IZS aufgrund der hierarchischen Zerlegung untergeordnet[29]. Damit entsteht ein Petri-Netz, das Ausführungszustände des globalen Prozesses auf lokale Sichten abbildet.

#### 4.5.1.2 Globale Identifikation von Modellelementen

Die Objekte eines kooperativen Prozesses werden während der Prozess-Gestaltung als Teil des Prozessmodells der Aufgabenebene definiert und führen zur Ableitung und hierarchischen Zerlegung je eines Peer-Workflows. Jede Instanz eines IZS und IZS-P ist damit auf ein Objekt der Aufgabenebene zurückführbar. Zur Realisierung von verteilt vorliegenden lokalen Sichten, die Ausführungszustände konsistent zu einem globalen Modell angeben, ist eine global eindeutige Identifikation von Objekten, Aufgaben, Instanzen, Tokens und Ereignissen erforderlich. Diese stellt die Zuordenbarkeit zwischen Elementen über Modellebenen hinweg sicher.

**Vergabe von Identifikatoren**

Die Vergabe von global eindeutigen Identifikatoren für die folgenden Elemente basiert auf randomisiert generierten UUIDs (Version 4)[30] (Fill und Härer 2018). Eine Vergabe kann damit ohne Nachrichtenkommunikation lokal erfolgen. Das Verfahren entspricht der lokalen Generierung eindeutiger Peer-Adressen[31].

- **Objekt:** Vergabe während der Zerlegung von Objekten der Aufgabenebene.
- **Ereignis:**
  - **O-Ereignis:** Vergabe während der Zerlegung von Objekten der Aufgabenebene.
  - **T-Ereignis:** Vergabe bei Ableitung eines Peer-Workflow-Modells der Aufgabenträgerebene.

[29] vgl. Abschnitt 4.7
[30] siehe RFC 4122: https://tools.ietf.org/html/rfc4122#section-4.4
[31] siehe Abschnitt3.2.2.3

- **Instanz:** Vergabe bei Instanziierung.
- **Token:** Vergabe bei Vorgangsauslösung.

Die UUIDs von Objekten und O-Ereignissen sind als Teil des kooperativen Prozesses global bekannt. Die UUIDs von T-Ereignissen, Instanzen und Tokens werden je Objekt anhand des zugehörigen objektspezifischen Smart Contracts global verteilt.

#### 4.5.1.3 Objektspezifischer Smart Contract zur Nachverfolgung von Instanzen

Zur Beschreibung einzelner Instanz-Zustände des IZS und IZS-P werden deren Elemente unter Hinzunahme der globalen Identifikatoren anhand der objektspezifischen Smart Contracts (OC) verwaltet.

Die Interaktion zwischen $OC_1$ und $OC_2$ zweier Objekte erfolgt gemäß der besprochenen Architektur (Abbildung 4.17). Ein Übergang eines Tokens zu einem O-Ereignis repräsentiert die Auslösung eines Vorgangs $V>$ innerhalb des Objekts $O1$. Hierfür wird die in Quellcode 4.3 als Signatur angegebene Funktion `triggerTransactionEvent(...)` herangezogen.

Diese Auslösung beginnt die mit $V>$ verbundene betriebliche Transaktion, die zusammen mit einem Vorgang $>V$ eines weiteren Objekts durchgeführt wird. Die Auslösung von $>V$ wird durch einen Funktionsaufruf `triggerTransactionEvent(...)` signalisiert, der das mit der Transaktion verbundene T-Ereignis in $OC2$ auslöst.

Der Funktionsaufruf ist effektiv eine als Blockchain-Transaktion übertragene Nachricht zwischen zwei Objekten, die den Beginn einer betrieblichen Transaktion übermittelt. Ebenso wird der Abschluss durch die Nachereignisse von $V>$ und $>V$ mit dem Aufruf von `triggerObjectEvent(...)` signalisiert.

Die Auslösung eines Ereignisses führt gemäß Petri-Netz-Semantik zur Veränderung der Zuordnung von Tokens zu den als Ereignissen repräsentierten Stellen des Netzes. Die Funktion `triggerTransactionEvent(...)` löst die Überführung des übergebenen Tokens hin zu dem übergebenen T-Ereignis aus. Hierfür wird die bestehende Zuordnung anhand der Zustandsvariable `taskEventAssignments` ermittelt und durch Anpassung der Variable `taskEventID` des Datentyps `TaskEventAssignment` gespeichert. Das Smart-Contract-Event `TransactionTaskEvent` gibt die Ereignisauslösung anschließend global bekannt.

```
1 pragma solidity ^0.5.3;
2 /// @title Objektspezifischer Smart Contract
3 contract ObjectSpecificContract {
```

```
4     // [...]
5
6     // Zuordnung: instanceID => Instanz
7     mapping (bytes16 => Token) public knownInstances;
8     // Zuordnung: taskEventAssignmentID => Event-Token-Zuordnung
9     mapping (uint16 => TaskEventAssignment) public taskEventAssignments
          ;
10    // Datentyp Instanz
11    struct Instance {
12        bytes16 objectID;
13        bytes16 instanceID;
14    }
15    // Datentyp Event-Zuordnung
16    struct TaskEventAssignment {
17        bytes16 taskEventID;
18        bytes16 instanceID;
19        bytes16 objectID;
20        bytes32 versionID;
21    }
22    /// T-Ereignis-Ausloesung: UEbergang tokenID zu tEventID
23    /// @param tEventID UUID des auszuloesenden Transaktionsereignisses
24    /// @param instanceID UUID der Instanz
25    /// @param objectID UUID des instanziierenden Objekts
26    /// @param versionID Hash-Wert des Prozessmodells
27    function triggerTransactionEvent(bytes16 tTaskEventID, bytes16
          instanceID, bytes16 objectID, bytes32 versionID) public {
28        // [...]
29        // Nachricht zur Ausloesung des Ereignisses per Event
30        emit TransactionTaskEvent(tTaskEventID, instanceID, objectID,
              versionID);
31    }
32
33    // Events zur Ausloesung von Ereignissen
34    event TransactionTaskEvent(bytes16 indexed taskEventID, bytes16
          instanceID, bytes16 objectID, bytes32 versionID);
35    event ObjectTaskEvent(bytes16 indexed taskEventID, bytes16
          instanceID, bytes16 objectID, bytes32 versionID);
36    // [...]
37 }
```

QUELLCODE 4.3: Objektspezifischer Smart Contract

Die objektspezifischen Smart Contracts aller Objekte bilden in ihren Zustandsvariablen den Ausführungszustand des kooperativen Prozesses zu einem Zeitpunkt ab.

**Korrektheit der Ausführung**

Aus Sicht des dezentralen Systems verläuft die Ausführung korrekt, wenn die erfassten Zustände das Protokoll des Systems nicht verletzen. Dies ist der Fall, wenn die folgenden Bedingungen erfüllt sind:

1. Ein Zustand $Z_i$ ist hinsichtlich des IZS-Metamodells valide. D.h., der Zustand muss die Instanzen eines zuvor fachlich vereinbarten kooperativen Prozesses abbilden.
2. Ein Paar aufeinanderfolgender Zustände $(Z_i, Z_{i+1})$ bildet einen Zustandsübergang $Z_i \Rightarrow Z_{i+1}$ ab, der unter Anwendung der operationalen Semantik eines Petri-Netzes zulässig ist. D.h., jede Veränderung der Zuordnung von Tokens zu Ereignissen muss auf ein Schalten von 1 bis n Transitionen des repräsentierten Petri-Netzes abbildbar sein.

Ein Ausführungszustand $Z_i$ wird verworfen, sofern Bedingung 1. nicht zutrifft. Ein Zustand $Z_{i+1}$ eines Paares aufeinanderfolgender Zustände $(Z_i, Z_{i+1})$ wird verworfen, sofern Bedingung 2. nicht zutrifft.

#### 4.5.1.4 Generierung von Instanz-Protokollen

Eine Historisierung von Zuständen im Zeitverlauf führt zur Nachverfolgbarkeit durchgeführter Vorgänge ex post. Instanz-Protokolle erfassen einzelne Zustände als Ergebnis einer multidimensionalen Auswertung, um (1.) die Korrektheit der Ausführung eines Prozesses im Nachhinein zu belegen und (2.) die Nachvollziehbarkeit verteilt ausgeführter Aufgaben für alle Prozessbeteiligten transparent zu ermöglichen. Hiermit wird die Aufgabenphase *Kontrolle* realisiert.

Der Generierung von Instanz-Protokollen liegt ein ETL-Prozess zum Aufbau eines Data Warehouse (DWH) zugrunde. Anfragen an das DWH erzeugen nach dem OLAP-Prinzip Instanz-Protokolle (Sinz 2010a). Die Implementierung des DWH basiert auf einem in Abbildung 4.18 gezeigten Star-Schema. Für die Auswertung von Instanz-Zuständen sind die hier gezeigten Dimensionen von Relevanz. Zudem sind Analysen anhand von abgeleiteten Kennzahlen durchführbar, z.B. zur Anzahl von Instanzen, zur Ausführungsdauer oder zur Rate wiederholter Ausführungen.

**ETL-Prozess**

Die in objektspezifischen Smart Contracts (OC) erfassten Ausführungszustände sind die Quelldaten des DWH. Gegenstand der Extraktion sind die in Blockchain-Transaktionen festgehaltenen Aufrufe von Funktionen der OC.

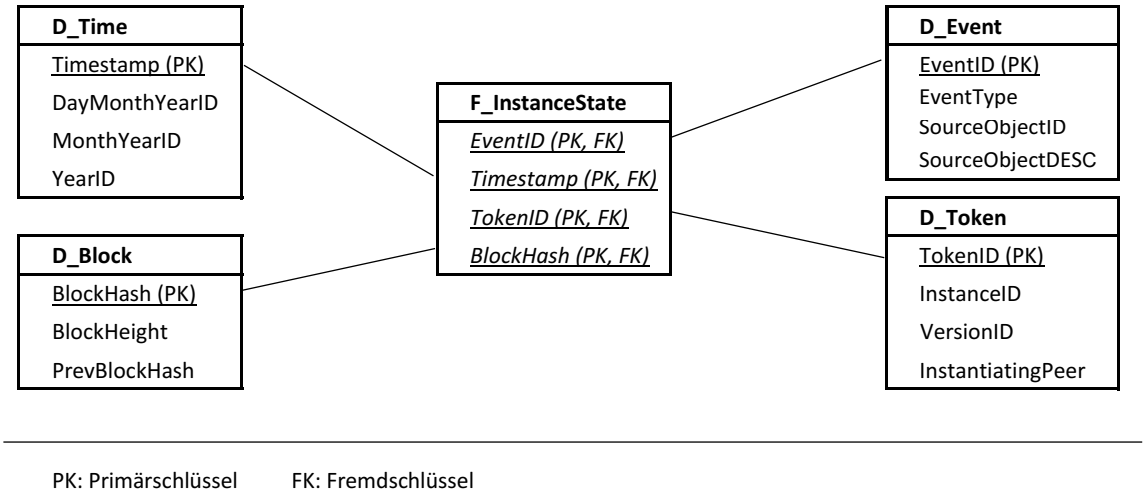



ABBILDUNG 4.18: Star-Schema zur Protokollierung von Instanz-Zuständen

Diese lösen einzelne Events aus (siehe Abschnitt 4.3). Die Extraktion wird mit jeder Benachrichtigung über neue Events durch eine Anwendungsfunktion der Node-Software des Blockchain-Systems ausgelöst. Jedes Event wird in eine Relation der in Abbildung 4.19 angegebenen Extraktionstabelle überführt.

Die anschließende Transformation überführt extrahierte Daten einzelner Ereignisauslösungen anhand einer Staging Area in Dimensionsdaten und Fakten. Dabei werden Instanzen (T_Token), Ereignisse (T_Event), Zeitpunkte und Zeiträume (T_Timestamp) sowie Blöcke (T_Block) berücksichtigt, die in der einer Faktentabelle

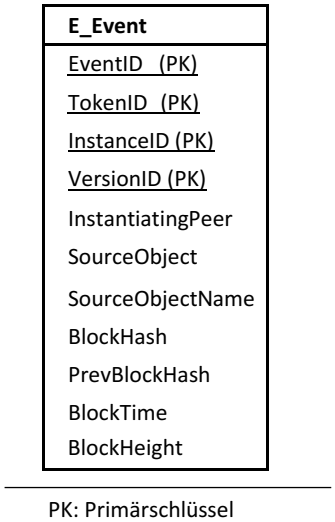



ABBILDUNG 4.19: Extraktionstabelle

(T_InstanceState) zusammengeführt werden. Das Laden der Daten in die gleichermaßen benannten Dimensionstabellen (Präfix D) und die Faktentabelle (Präfix F) des Star-Schemas schließt den ETL-Prozess ab.

### Anfragen zur Generierung des Protokolls

Die beschriebene Architektur erlaubt die Ausführung von ad-hoc-Anfragen an das Data-Warehouse-System. Die in Quellcode 4.4 dargestellte Anfrage bildet die Ausführungszustände aller betrieblichen Objekte für einen Prozess einer Version *selectedVersion* als globale Sicht über alle Instanzen ab. Weitere Dimensionen können hinzugezogen werden, z.B. *selectedInstance* und *selectedObject* zur Abbildung einzelner Instanzen bzw. Objekte. Die Analyse der Ausführungszustände der lokalen Sicht eines Peers erfordert diese beiden Angaben.

```
1  SELECT InstanceID, F.EventID, SourceObjectID, SourceObjectDESC,
       InstantiatingPeer, BlockHeight, F.Timestamp
2    FROM
3      F_InstanceState F
4      INNER JOIN D_Event EV ON F.EventID = EV.EventID
5      INNER JOIN D_Token TK ON F.TokenID = TK.TokenID
6      INNER JOIN D_Time TI ON F.Timestamp = TI.Timestamp
7      INNER JOIN D_Block BL ON F.BlockHash = BL.BlockHash
8    WHERE
9      InstanceID = $selectedInstance
10   ORDER BY
11     BlockHeight ASC
```

QUELLCODE 4.4: Anfrage an das DWH-System

| | InstanceID | EventID | SourceObjectID | SourceObjectDESC | InstantiatingPeer | lockHeigh | Timestamp |
|---|---|---|---|---|---|---|---|
| 1 | 319f8ae9-... | 7742d278-ca2... | 665c4e18-23ff-4... | Auftragsabwicklung | 0x6C249D8C7A3A7... | 7773563 | 2019-05-16T20:52:12 |
| 2 | 319f8ae9-... | 96ebabc9-ecc... | 665c4e18-23ff-4... | Auftragsabwicklung | 0x6C249D8C7A3A7... | 7773563 | 2019-05-16T20:52:12 |
| 3 | 319f8ae9-... | 4f22913c-9a2... | 7df392dd-93fd-3... | Zulieferer B | 0x6C249D8C7A3A7... | 7773653 | 2019-05-16T21:11:37 |
| 4 | 319f8ae9-... | a201a41d-8e... | 571ed6d0-8155-... | Makler | 0x6C249D8C7A3A7... | 7773664 | 2019-05-16T21:14:30 |
| 5 | 319f8ae9-... | 7f670d9a-dba... | e799ab03-1615-... | Zulieferer A | 0x13AE9557122D1... | 7773691 | 2019-05-16T21:21:26 |

ABBILDUNG 4.20: Beispiel eines Instanz-Protokolls

Abbildung 4.20 zeigt ein Beispiel eines Instanz-Protokolls unter Nutzung der angegebenen Anfrage. Das Protokoll kann anhand der folgenden Dimensionsattribute eingeschränkt werden.

- **Ereignisse der Dimension D_Event:** Vorgangsauslösung Aufgabe (EventID), Typ O oder T-Ereignis (EventType), Objekt (SourceObjectID, SourceObjectDESC)

- **Instanzen der Dimension D_Token:** Token (TokenID), Instanz (InstanceID), Prozess-Version (VersionID), instanziierendes Peer (InstantiatingPeer)
- **Zeitpunkte und Zeiträume der Dimension D_Time:** Zeitstempel (Timestamp), Datum (DayMonthYearID), Monat und Jahr (MonthYearID), Jahr (YearID)
- **Blöcke der Dimension D_Block:** Block-Hash-Wert (BlockHash), Block-Nummer (BlockHeight), vorhergehender Block-Hash (PrevBlockHash)

Das Instanz-Monitoring realisiert anhand des Instanz-Protokolls in Kombination mit dem IZS eines kooperativen Prozesses die Abbildung einer globalen Sicht auf Instanzen sowie in Kombination mit den zugehörigen IZS-P von Peer-Workflows die Abbildung von lokalen Sichten je Peer.

## 4.6 Ergebnisdiskussion

Der Ansatz umfasst die Modellierung, die Vereinbarung sowie das Instanz-Monitoring von auf Kooperation beruhenden Prozessen in einem dezentralen System.

Das Teilsystem *Modellsystem* legt einen Rahmen fest, der (1.) von der Modellierung der System-Gestaltung des dezentralen Systems in Peer-Modellen und einem globalen Wertschöpfungsnetz ausgeht, diese in (2.) Modelle der Prozess-Gestaltung in Form von global-öffentlichen Prozessmodellen und lokal-privaten Workflow-Modellen überführt und (3.) Modelle der Prozess-Ausführung für Prozess-Instanzen der Aufgabenebene sowie Workflow-Instanzen der Aufgabenträgerebene heranzieht. Der Rahmen wird durch die Definition von Metamodellen instanziierbar.

1. **System-Gestaltung:** Das Metamodell eines Peer-Schemas initiiert die Modellierung des dezentralen Systems ausgehend von dessen Komponenten. Das Metamodell der Aufgabenebene der SOM-Methodik bildet die Objekte des Wertschöpfungsnetzes in ihrer Struktur als Interaktionsschema ab.

2. **Prozess-Gestaltung:** Die Entwicklung von kooperativen Prozessen bezieht entsprechend des Metamodells der Aufgabenebene der SOM-Methodik Struktur- und Verhaltenssichten als Interaktionsschema (IAS) und Vorgangs-Ereignis-Schema (VES) ein. Eine Ableitung von Peer-Workflows wird durch ein Metamodell der Aufgabenträgerebene definiert, das die Bildung der Struktur- und Verhaltenssichten IAS-P und VTS-P unterstützt. Diese beziehen sich auf Interaktionsschemata (IAS-P) einzelner Peers und deren Implementierung als Workflows oder Vorgangstypen in Vorgangstypschemata (VTS-P).

3. **Prozess-Ausführung:** Die Ausführung von Prozessen erweitert die Verhaltenssichten der Aufgaben- und Aufgabenträgerebene um Meta-Objekte zur Darstellung von Instanz-Zuständen. Das Instanz-Zustands-Schema (IZS) erweitert die Prozessmodelle der Aufgabenebene um Instanz-Zustände, die unter Hinzunahme von Tokens die operationale Semantik eines Petri-Netzes unterstützen. Dies trifft für das Instanz-Zustands-Schema eines Peers (IZS-P) und das Vorgangstypschema eines Peers (VTS-P) gleichermaßen zu.

Das Teilsystem *Kooperationssystem* implementiert eine innerhalb des dezentralen Systems durchführbare kooperative Modellierung des Modellsystems. Der Ansatz

verbindet die Entwicklung von interorganisationalen und auf Kooperation beruhenden Prozessen mit Versionierungsverfahren zur kooperativen Modellierung. Zwei Protokollfunktionen des Systems werden von einzelnen Peers zugegriffen. Die Protokollfunktionen basieren auf einem Versionsgraphen und Smart Contracts, die über die Infrastruktur eines dezentralen Blockchain-Systems eine Netzwerk- und Kommunikationsschicht bilden.

1. **Modellmanagement:** Die Verwaltung von privaten Modellen greift auf einen privaten Branch des Versionsgraphen zurück und sichert die Integrität und Verbindlichkeit von dort hinterlegten Modellen anhand des Blockchain-Systems. Die Verwaltung von öffentlichen Modellen basiert auf einem verteilten Commit-Protokoll, das die fachliche Vereinbarung von Modellen anhand des Blockchain-Systems umfasst.
2. **Vereinbarung des Modellsystems:** Die Durchführung des verteilten Commits vereinbart das Modellsystem eines kooperativen Prozesses. Ein Commit legt eine Version eines Prozesses innerhalb des Versionsgraphen ab und sichert die Integrität und Verbindlichkeit des betrachteten Modells anhand des Blockchain-Systems. Ein vereinbarter Prozess ist der Ausgangspunkt der Instanziierung innerhalb des Ausführungssystems.

Das Teilsystem *Ausführungssystem* sichert die Korrektheit verteilt ausgeführter Prozesse, die anhand der Protokollfunktion Instanz-Monitoring von Prozessbeteiligten nachverfolgt werden können.

Das **Instanz-Monitoring** verteilt die in IZS und IZS-P erfassten Instanz-Zustände anhand von Smart Contracts und erlaubt damit die Nachvollziehbarkeit der Ausführung. Die Ausführung kann ex post über mehrere Dimensionen unter Nutzung eines Data-Warehouse-Systems verfolgt und fachlich bewertet werden. Die Protokollfunktion realisiert in der Architektur des dezentralen Systems die verteilte Validierung der Ausführung von Prozessen.

# Kapitel 5

# Fallstudie zur Entwicklung und Ausführung von Geschäftsprozessen

## 5.1 Betriebliche Prozesse in dezentralen Systemen

Das vorliegende Kapitel demonstriert (1.) die Anwendbarkeit des vorgeschlagenen Ansatzes anhand einer Fallstudie sowie (2.) die Implementierbarkeit anhand eines als Proof-of-Concept implementierten Software-Tools. Weiterhin soll die Anwendung des Ansatzes (3.) limitierende Faktoren aufzeigen. Das Kapitel behandelt zunächst die Anwendung des Ansatzes auf interorganisationale Prozesse, deren Diskussion zu einer Fallstudie eines Supply-Chain-Szenarios führt. Gegenstand der Fallstudie ist ein kooperativer Geschäftsprozess zur Produktion, der während der Anwendung des Ansatzes entwickelt wird.

### 5.1.1 Interorganisationale Prozesse

Untereinander vernetzte Objekte eines dezentralen Systems bilden ein Netzwerk, das die Herausbildung von Prozessen und deren Ausführung basierend auf einer verteilten Speicherung und Verarbeitung von Modellen erlaubt. Die Entwicklung von Prozessen innerhalb eines solchen Systems basiert auf den Merkmalen dezentraler Blockchain-Systeme, anhand derer die Verbindlichkeit und Integrität vereinbarter Prozesse gewährleistet werden können. Die Ausführung von Prozessen in einem solchen System gewährleistet durch die Nachvollziehbarkeit von Instanz-Zuständen eine verteilte Validierung durch beliebige Teilnehmer des Systems. Die Zusammenführung mehrerer Prozesse führt zu Wertschöpfungsnetzen.

Der Ansatz richtet sich daher an Prozesse, in denen die genannten Merkmale nutzbringend und notwendig sind. Hierzu zählen in erster Linie interorganisationale

Prozesse, die auf dem Austausch von Geschäftstransaktionen zwischen Unternehmen beruhen. Dies betrifft beispielsweise Zuliefer-Abnehmer-Prozesse in Supply-Netzwerken sowie prinzipiell beliebige Business-to-Business-Prozesse unter Beteiligung verschiedener autonomer Organisationen.

An jedem Punkt in den Abläufen interorganisationaler Prozesse, an dem der Austausch von Informationen über Unternehmensgrenzen hinweg erforderlich ist, erlaubt die Abbildung betrieblicher Transaktionen auf Blockchain-Transaktionen eine unabhängig von den beteiligten Unternehmen gewährleistete Absicherung der Integrität und Verbindlichkeit. Über mehrere Transaktionen hinweg kann damit die Konsistenz der Informationen eines Prozesses transparent nachverfolgt werden. Die Absicherung von Informationen des Wertschöpfungsprozesses über mehrere Stufen hinweg erlaubt beispielsweise die Rückverfolgbarkeit von Produkten hin zum Ursprung und die Auflösung von Konflikten anhand der abgesicherten Transaktionsdaten. Neben der Reduktion von Transaktionskosten durch die Vermeidung von rechtlichen Auseinandersetzungen werden hierdurch potenziell Intermediäre eliminiert.

Ein in diesem Zusammenhang diskutiertes Anwendungsszenario von Blockchain- und Distributed-Ledger-Systemen ist die überbetriebliche Planung von Ressourcen in ERP-Systemen sowie das Supply-Chain-Management (Abeyratne und Monfared 2016; Korpela et al. 2017; Linke und Strahringer 2018). Die in einem Supply-Chain-Szenario bestehenden Transaktionen des Informationsaustausches zwischen Zulieferern und Abnehmern beschreiben ein Informationssystem, das eine verteilte Koordination der Leistungserstellung ohne zentralen Koordinator erfordert.

### 5.1.2 Supply-Chain-Szenario

Das im Folgenden betrachtete Szenario beschreibt das Netzwerk verteilter Unternehmen einer Supply Chain, die einen verteilten Produktionsprozess implementieren. Der Prozess des Systems erfordert als kooperativer Geschäftsprozess ein Zusammenwirken der beteiligten Unternehmen.

Das Supply-Chain-Szenario (Fdhila et al. 2015) umfasst die folgenden Organisationen:

1. **Produzent:** Ein produzierendes Unternehmen führt Produktionsaufträge eines Großabnehmers aus. Die Fertigung erfordert Zwischenprodukte (ZP-A, ZP-B), die von zwei Zulieferern gefertigt werden.

2. **Makler:** Ein Makler erbringt Leistungen zur Beschaffungsdisposition für Produzent. Die Beschaffung von ZP-A wird zusammen mit einem Zulieferer A koordiniert.
3. **Zulieferer A:** Zulieferer A erhält Produktionsaufträge des ZP-A, führt diese aus und stellt ZP-A für den Transport zu Produzent bereit.
4. **Zulieferer B:** Zulieferer B erhält Produktionsaufträge des ZP-B, führt diese aus und liefert ZP-B an Produzent aus.
5. **Logistikdienstleister:** Ein durch den Makler beauftragter Logistikdienstleister plant die Distribution von ZP-A und führt diese aus.
6. **Großabnehmer:** Ein Großabnehmer vereinbart und erteilt Produktionsaufträge für die durch die Supply Chain erzeugten Leistungen.

Der kooperative Geschäftsprozess enthält als öffentlicher Prozess die Teilprozesse einzelner betrieblicher Objekte insoweit, als sie für die gemeinsame Ausführung und Koordination erforderlich sind. Die Innensicht der beteiligten Objekte implementiert einzelne Teilprozesse in privaten Prozessen und Workflows.

### 5.1.3 Kriterien zur Auswahl des Szenarios

Der Auswahl der Fallstudie (Fdhila et al. 2015) mit den dort beschriebenen Prozessen liegen folgende Merkmale zugrunde:

1. Die Konzeption des Gesamtprozesses erfolgt durch mehrere, verteilte Teilnehmer.
2. Eine Kooperation der Teilnehmer ist für die Durchführung der Prozesse erforderlich.
3. Die Merkmale Verteilung, Autonomie und Selbstorganisation sind für die Teilnehmer gegeben.
4. Es erfolgen Veränderungen der Geschäftsprozesse zur Gestaltungs- und Ausführungszeit.

Zur Anwendung des Ansatzes wird das Szenario im Folgenden um eine dezentrale Koordination erweitert. D.h., die Entwicklung und Ausführung der Prozesse des Systems findet innerhalb eines dezentralen Systems statt, das sich durch die Verteilung der Komponenten und die verteilte Koordination basierend auf den Protokollfunktionen des Ansatzes definiert.

## 5.2 Anwendung des Ansatzes zur System-Gestaltung

Die Gestaltung des dezentralen Wertschöpfungsnetzes geht von einzelnen Systemkomponenten aus, die, in Abgrenzung zu ihrer jeweiligen lokalen Umwelt, bestehende Beziehungen zu Zulieferern und Abnehmern besitzen. Die Zusammenführung dieser Beziehungen, die Festlegung von gemeinsamen Zielen und die Angabe von Leistungsbeziehungen führen zu miteinander vernetzten betrieblichen Objekten eines Wertschöpfungsnetzes.

### 5.2.1 Peer-Netzwerk: Netzwerkschema

Die Identifikation und Abgrenzung einzelner gleichrangiger Systemkomponenten ist Teil der Aufgabenträgerebene. Diese initiieren nach dem Prinzip der Selbstorganisation die Herausbildung von Aufgaben als Bestandteil von betrieblichen Objekten des Wertschöpfungsnetzes. Ein Netzwerkschema eines Peers (NWS-P) erfasst die Struktur initial bestehender Beziehungen einzelner Organisationen als Teil von Unternehmensnetzwerken (Österle, Fleisch et al. 2002). Abbildung 5.1 zeigt das in der Fallstudie initial bestehenden Netzwerk für „Produzent“ innerhalb des Software-Tools.

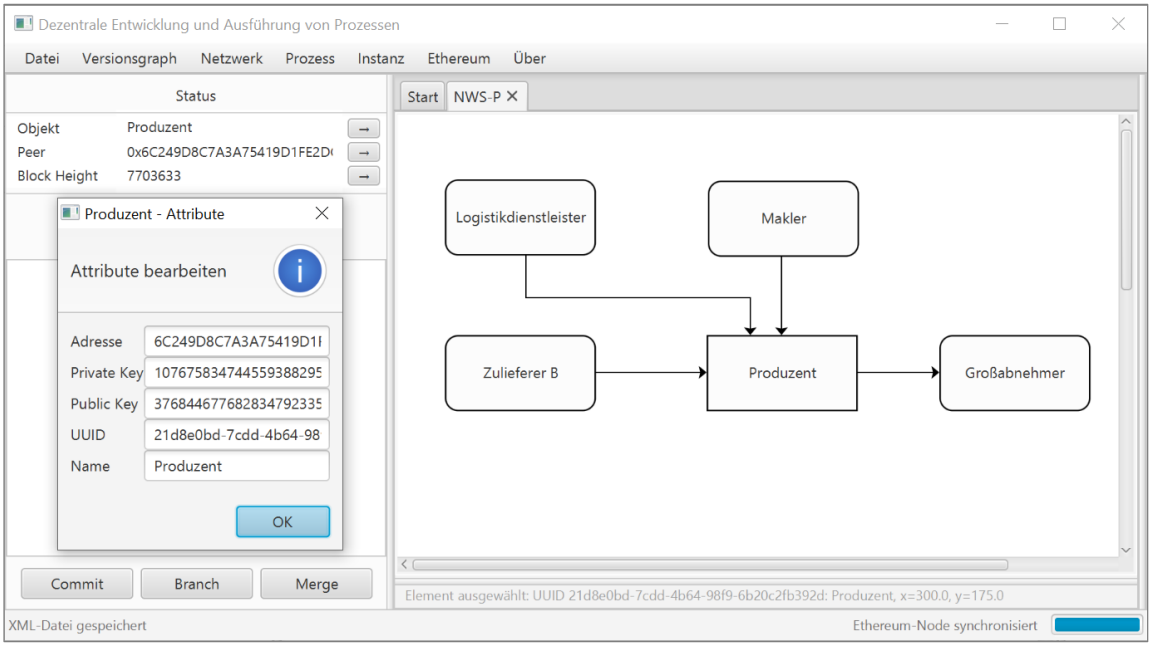



ABBILDUNG 5.1: Netzwerkschema von Produzent

Lokale und entfernte Peer-Elemente werden mit den zuvor definierten Attributen modelliert und innerhalb des „Global Contract“, hier in der Ethereum-Blockchain, registriert. In Kombination mit den in Abbildung 5.2 dargestellten Netzwerkschemata weiterer Peers sind nun einzelne Netzwerke, Adressen und öffentliche Schlüssel global bekannt[1].

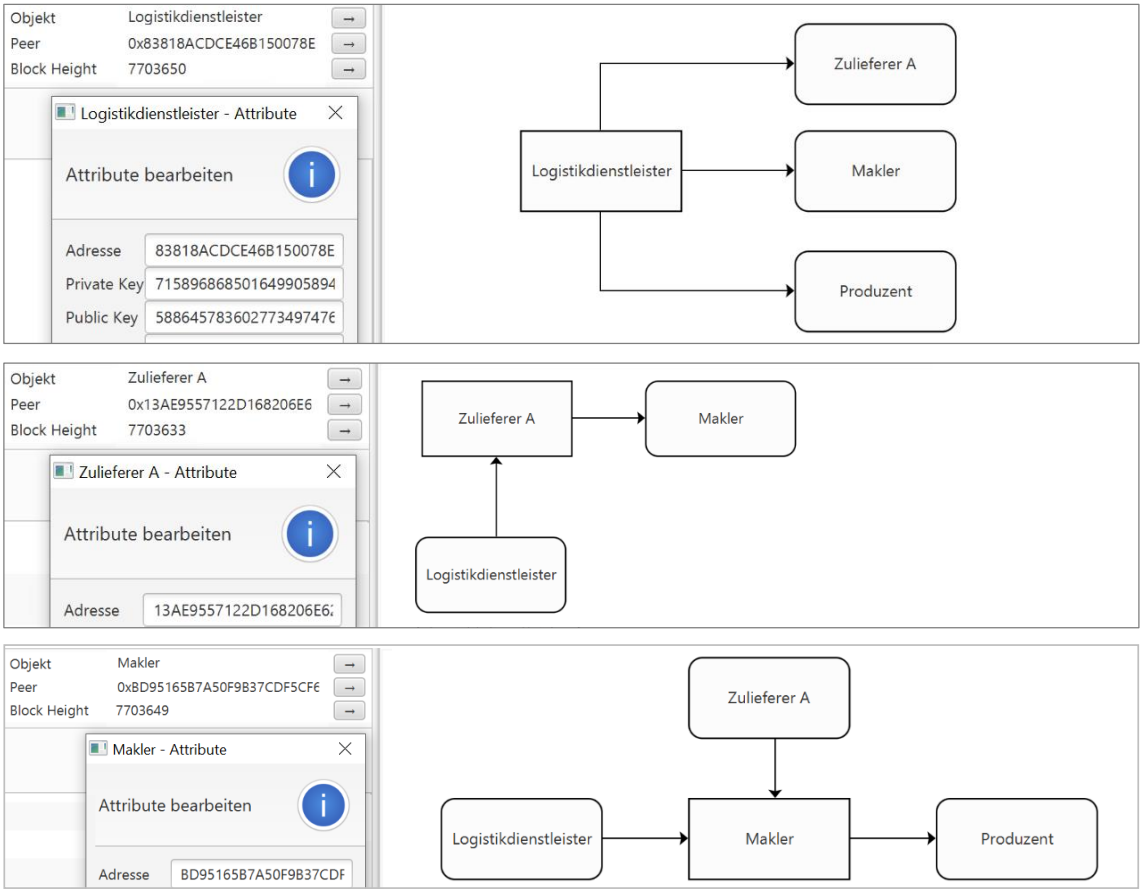



ABBILDUNG 5.2: Netzwerkschemata von Logistikdienstleister, Zulieferer A und Makler

### 5.2.2 Wertschöpfungsnetz: Interaktionsschema

Die Vernetzung betrieblicher Objekte innerhalb des Wertschöpfungsnetzes legt gemeinsame Ziele sowie Leistungsbeziehungen zwischen kooperierenden Objekten fest. Die Modellierung umfasst zwei Schritte.

[1]Siehe z.B. die Transaktion zur Registrierung von Produzent (Block 7721013): https://etherscan.io/address/0x6c249d8c7a3a75419d1fe2dcfb0644fb5ea9a60a .

#### 5.2.2.1 1. Modellierung vernetzter Objekte

Die Transformation einzelner Netzwerkschemata legt die Struktur betrieblicher Objekte der Aufgabenebene in Form eines Interaktionsschemas an. Das hiermit entstehende Objekt-Netzwerk spezifiziert betriebliche Objekte und Transaktionen zur Abbildung von Leistungen. Abbildung 5.3 zeigt das initiale Interaktionsschema des Wertschöpfungsnetzes sowie die im nachfolgenden Schritt vereinbarten Zielbeziehungen.

#### 5.2.2.2 2. Modellierung der gemeinsamen Ziele von kooperierenden Objekten

Die Herausbildung von kooperativen Prozessen unter Einbeziehung mehrerer Objekte beruht auf Ziel- und Rückmeldungsbeziehungen. Die Modellierung dieser Beziehungen innerhalb des dezentralen Systems realisiert eine verteilt erfolgende Koordination. Abbildung 5.3 zeigt die zeitkontinuierlichen Beziehungen. Der zusammenhängende Teilgraph von Objekten (blau hinterlegt), die gemeinsame Ziele verfolgen, führt zur Registrierung von kooperativen Prozessen zwischen den verbundenen Objekten innerhalb des „Global Contract“.

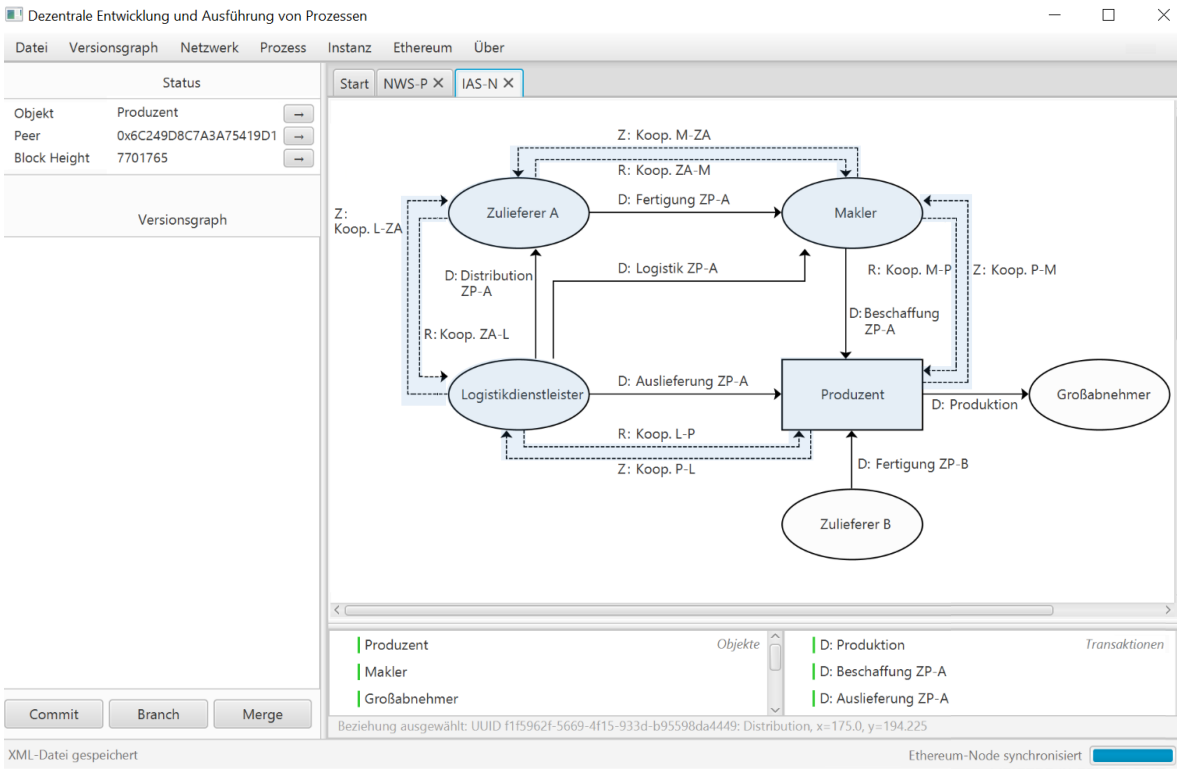



ABBILDUNG 5.3: Interaktionsschema des globalen Wertschöpfungsnetzes aus der Sicht von „Produzent“

## 5.3 Anwendung des Ansatzes zur Prozess-Gestaltung

Die Planung eines kooperativen Geschäftsprozesses übernimmt die Struktur eines zusammenhängenden Teilgraphen des Wertschöpfungsnetzes. Die Strukturansicht entspricht dem initialen Interaktionsschema des kooperativen Prozesses, die zur Bildung der Aufgabenebene anhand des Vorgangs-Ereignis-Schemas um eine Verhaltenssicht ergänzt wird. Dieses Modell ist global-öffentlich für alle Beteiligten zugreifbar. Die Aufgabenträgerebene umfasst die Spezifikation von lokalen und privaten Peer-Workflows, mit denen die Innensichten der beteiligten Objekte beschrieben werden.

Die Entwicklung des kooperativen Prozesses basiert auf Entwurfstransaktionen, die Modelle per Commit persistieren und anhand der Protokollfunktion Modell-Management global-öffentlich oder lokal-privat in separaten Branches von Versionsgraphen zugänglich machen und anhand von Smart Contracts absichern. Der Commit eines global-öffentlichen Modells löst anhand der Protokollfunktion zur Vereinbarung von Modellen einen verteilten Zwei-Phasen-Commit aus, der die Vereinbarung des Modellsystems per Smart Contract koordiniert.

### 5.3.1 Kooperativer Prozess: Interaktions- und Vorgangs-Ereignis-Schema

Die Modellierung des kooperativen Prozesses umfasst zwei Schritte:

#### 5.3.1.1 1. Modellierung des initialen Objektsystems

Der Prozess der kooperierenden Objekte Produzent, Makler, Logistikdienstleister und Zulieferer A liegt in einer initialen Struktur als Interaktionsschema vor, die Abbildung 5.3 entspricht. Die genannten Objekte bilden die Diskurswelt des von den beteiligten Objekten gemeinsam durchgeführten Prozesses. Die Entwicklung der Struktur- und Verhaltenssicht erfolgt sukzessive durch Entwurfstransaktionen.

#### 5.3.1.2 2. Modellierung kooperierender Objekte in IAS und VES

Ausgangspunkt der Entwicklung ist das Interaktionsschema des Netzwerks, welches in das initiale Interaktionsschema des kooperativen Prozesses in Abbildung 5.4 übergeht.

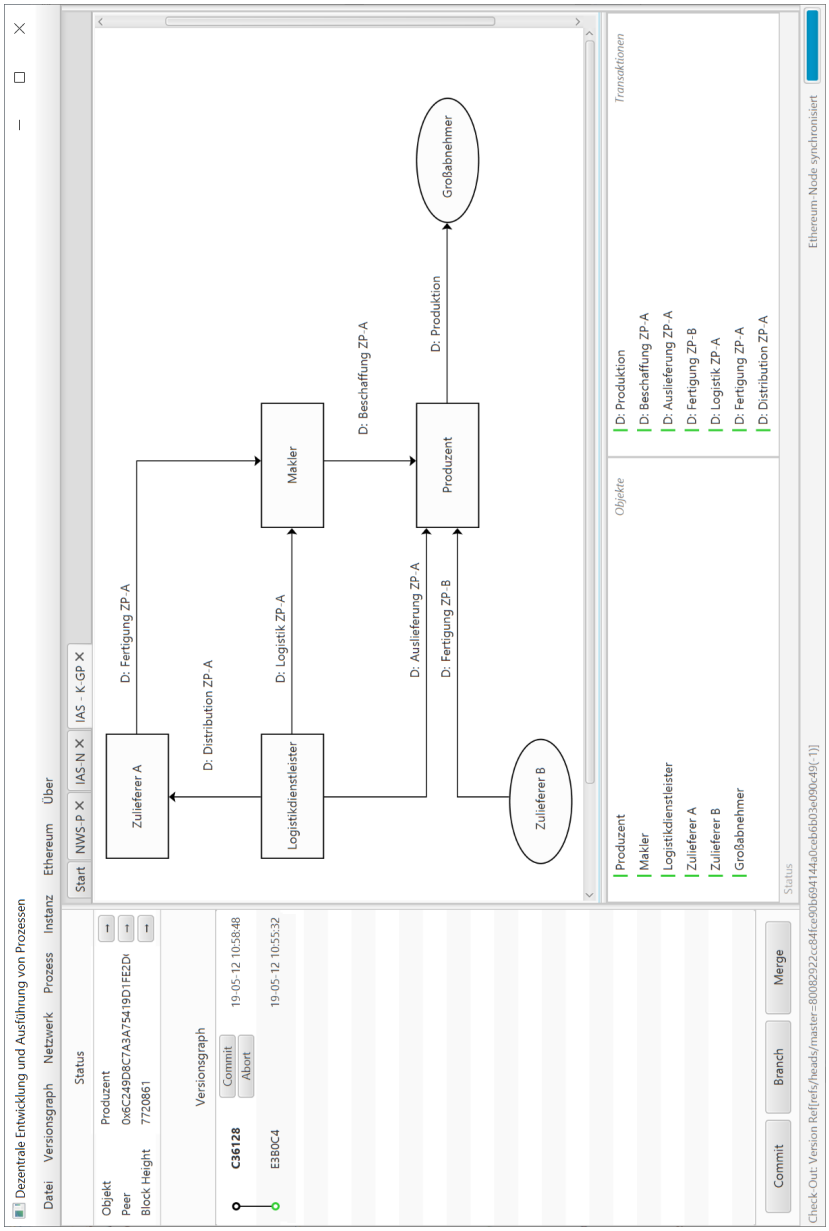



ABBILDUNG 5.4: Initiales Interaktionsschema des kooperativen Prozesses

Der öffentliche und globale Prozess ist nach der Durchführung einer ersten Entwurfstransaktion innerhalb des Versionsgraphens sichtbar. Die von „Produzent“ initiierte Transaktion löst einen globalen Commit aus, der übereinstimmende Vote-Commit-Nachrichten je Peer erfordert. Die Abbildung zeigt die Versionskennung C36128 nach der Initiierung des Commits mit der Möglichkeit zum Absenden von Vote-Commit- oder Vote-Abort-Nachrichten an den globalen Smart Contract. Die Versionskennung bezieht sich auf den SHA256-Hash-Wert des Modellsystems. Als Teil von Anhang B zeigt Abbildung B.8 einen Ausschnitt der zur Berechnung des

Wertes herangezogenen Objektbaum-Datenstruktur in der innerhalb des Versionsgraphen hinterlegten XML-Serialisierung. Abbildung 5.5 zeigt die Statusausgabe während des Initiierens der Commit-Transaktion sowie nach dem Absenden einer Vote-Commit-Nachricht durch „Zulieferer A“.

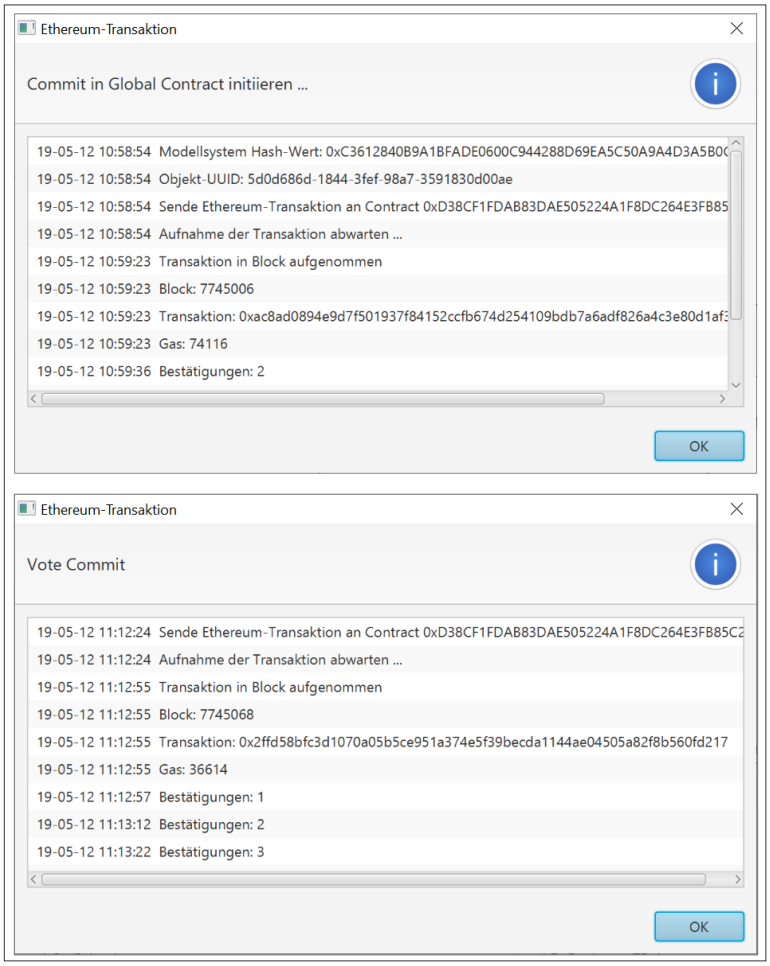



ABBILDUNG 5.5: Initiieren eines globalen Commits und Absenden einer Vote-Commit-Nachricht

Die Vereinbarung des kooperativen Prozesses führt nach weiteren Entwurfstransaktionen durch Makler (BA6833, 2F793D) und Logistikdienstleister (4FCBD) im Zuge einer Transaktionszerlegung sowie einer von Produzent (40F00A) durchgeführten Objektzerlegung zu dem in Abbildung 5.6 dargestellten Interaktionsschema eines kooperativen Prozesses.

Die Zerlegungsschritte sind in einzelnen Commits des Versionsgraphen hinterlegt. Der Commit 2F793D wurde aufgrund einer Vote-Abort-Nachricht nicht durchgeführt. Die vollständigen Zerlegungsbäume sind in Abbildung B.1 des Anhangs B

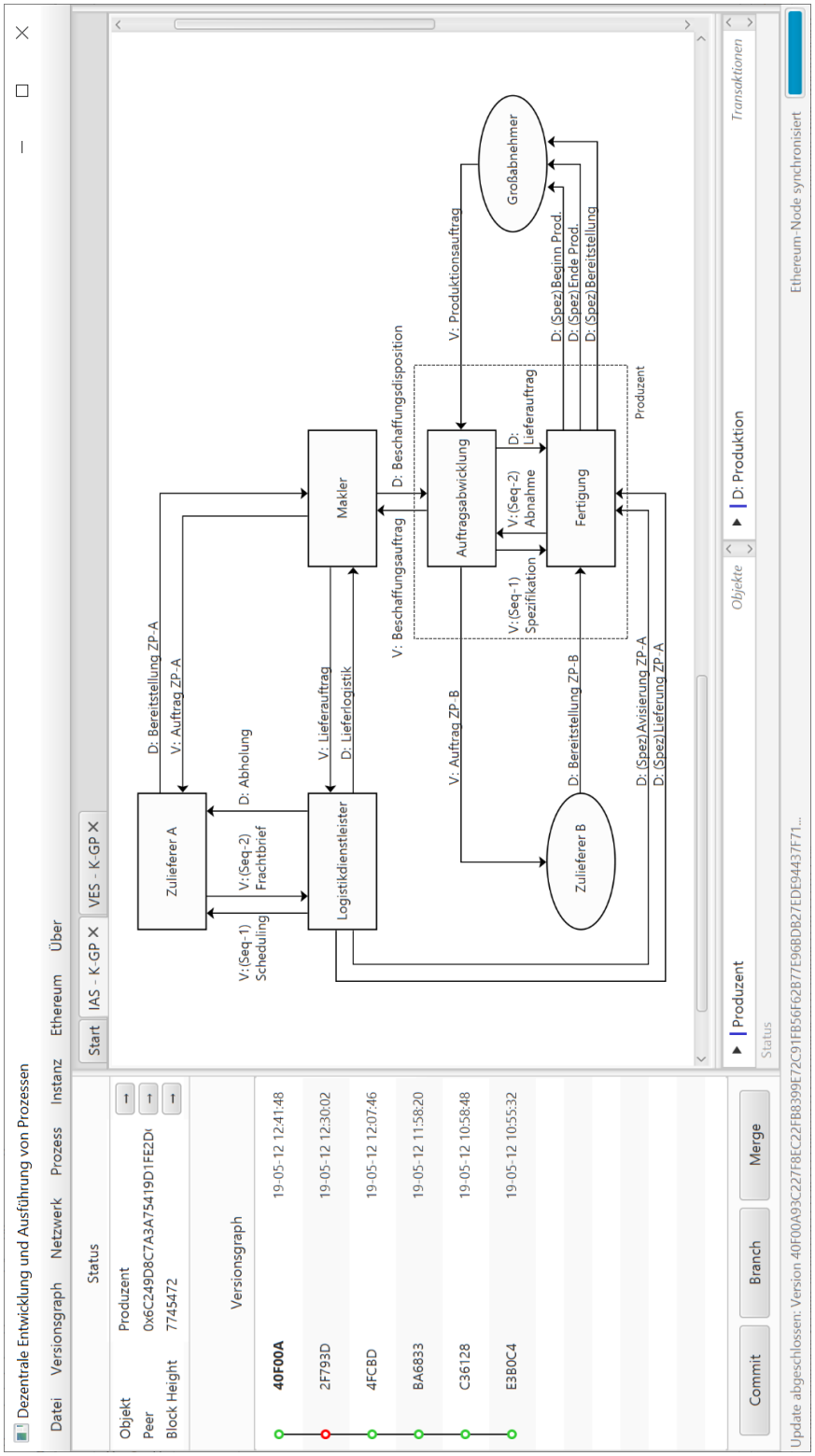



ABBILDUNG 5.6: Interaktionsschema des kooperativen Prozesses nach Transaktions- und Objektzerlegung

zu sehen. Die anhand der Struktur erstellte Ablaufsicht ist als Vorgangs-Ereignis-Schema ebenso Teil des Anhangs (Abbildung B.2). Die Objekte und Transaktionen

sind in den angegebenen Versionen des öffentlichen Branches enthalten und in der Ethereum Blockchain unter der Adresse des globalen Smart Contracts einsehbar[2].

### 5.3.2 Peer-Workflow: Interaktionsschema und Vorgangstypschema

Die Detaillierung und Implementierung lokaler Peer-Workflows findet je Diskursweltobjekt innerhalb des zugeordneten Peers lokal und nicht-öffentlich in einem separaten Branch des Versionsgraphen statt. Die Objekte Produzent, Makler, Logistikdienstleister und Zulieferer A vollziehen die Schritte (1.) Detaillierung der Struktur anhand des IAS-P, (2.) Detaillierung des Verhaltens anhand des VTS-P sowie (3.) Implementierung der Vorgangsauslösung in objektspezifischen Smart Contracts.

#### 5.3.2.1 1. Detaillierung der Struktur anhand des IAS-P

Aus der Innensicht eines betrachteten Objekts spezifiziert das IAS-P die Struktur zur Durchführung der öffentlichen Teile des Prozesses. Die Transformation des IAS bildet ausschließlich das jeweils betrachtete Objekt als Diskursweltobjekt ab. Abbildung 5.7 zeigt das Interaktionsschema des Peers Produzent als Teil einer Version des privaten Branches (blaue Markierung).

Die je Objekt hinterlegten Adressen sind den in Schritt 3. angelegten objektspezifischen Smart Contracts zugehörig. Sämtliche Zerlegungsprodukte des Objekts „Produzent“ liegen nur innerhalb der Objektbaum-Datenstruktur des privaten Branches vor. Das Schema ist damit eine lokale Sicht des globalen Prozesses. Ein weiteres Beispiel ist zusammen mit der lokalen Zerlegung von Objekten und Transaktionen in Abbildung 5.8 für „Zulieferer A“ zu sehen.

#### 5.3.2.2 2. Detaillierung des Verhaltens anhand des VTS-P

Die Modellierung der Verhaltenssicht leitet sich per Transformation aus dem VES der Aufgabenebene ab. Dabei leiten sich Vorgangstypen zur Angabe von Workflows aus Aufgaben ab. Ein VTS-P erfasst die Verhaltenssicht für die innerhalb des IAS-P vorgenommene Detaillierung und ergänzt Pre- und Post-Conditions zur ereignisgesteuerten Vorgangsauslösung. Die VTS-P-Schemata in Abbildung 5.9 stellt exemplarisch die Vorgangstypen von Zulieferer A dar. Die durch Blockchain-Transaktionen ausgelösten T-Ereignisse initiieren zu Beginn des lokalen Workflows den Vorgangstyp „>Auftrag ZP-A“. Der Vorgang kann nach Eingang der Blockchain-Transaktion

[2] Adresse: 0xD38CF1FDAB83DAE505224A1F8DC264E3FB85C24E (Siehe z.B. Etherscan Block-Explorer: https://etherscan.io/address/0xD38CF1FDAB83DAE505224A1F8DC264E3FB85C24E)

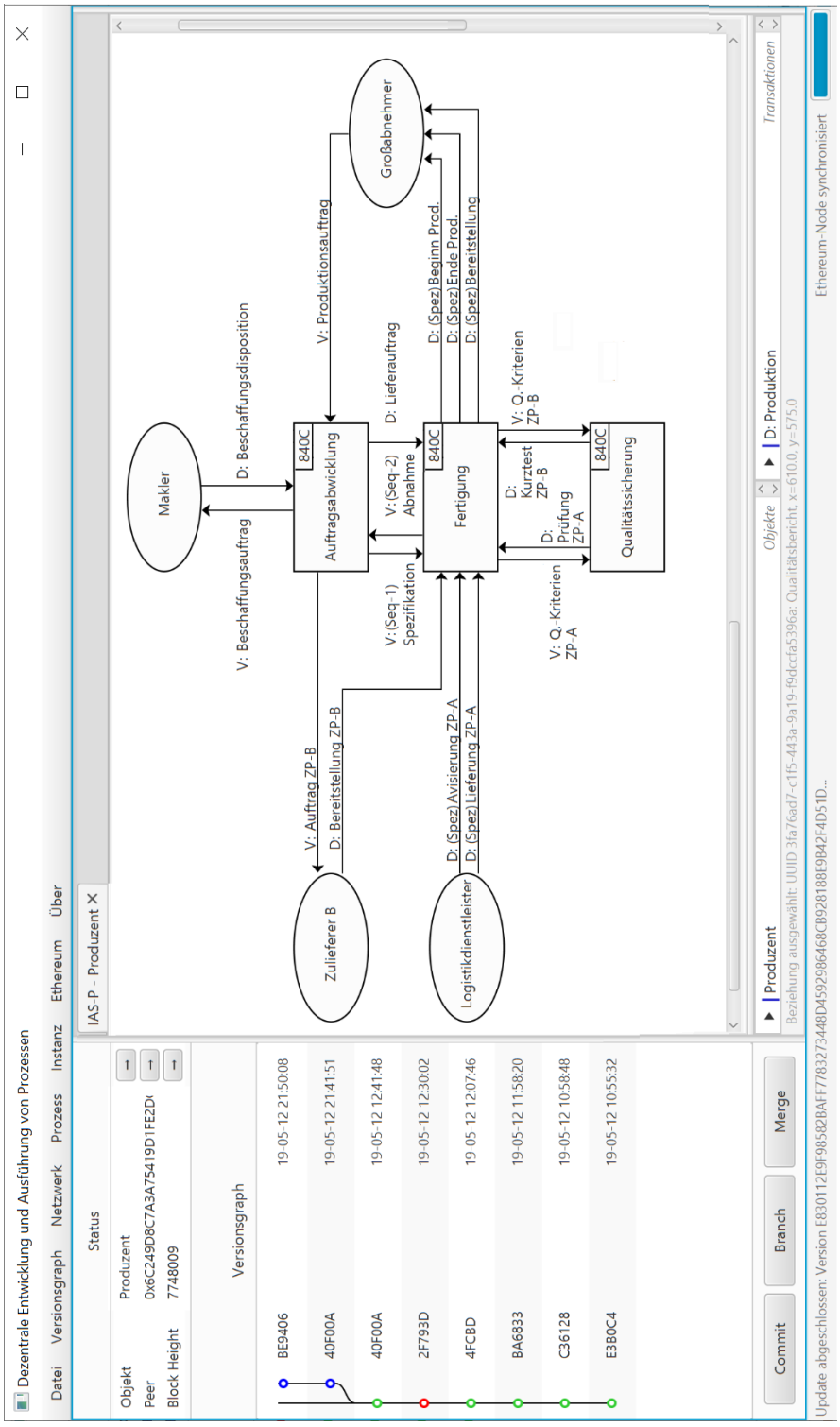



ABBILDUNG 5.7: IAS-P von Produzent

durch den objektspezifischen Smart Contract des Objekts manuell oder automatisiert ausgelöst werden. Die Angabe einer Pre-Condition für einen Vorgangstyp ist

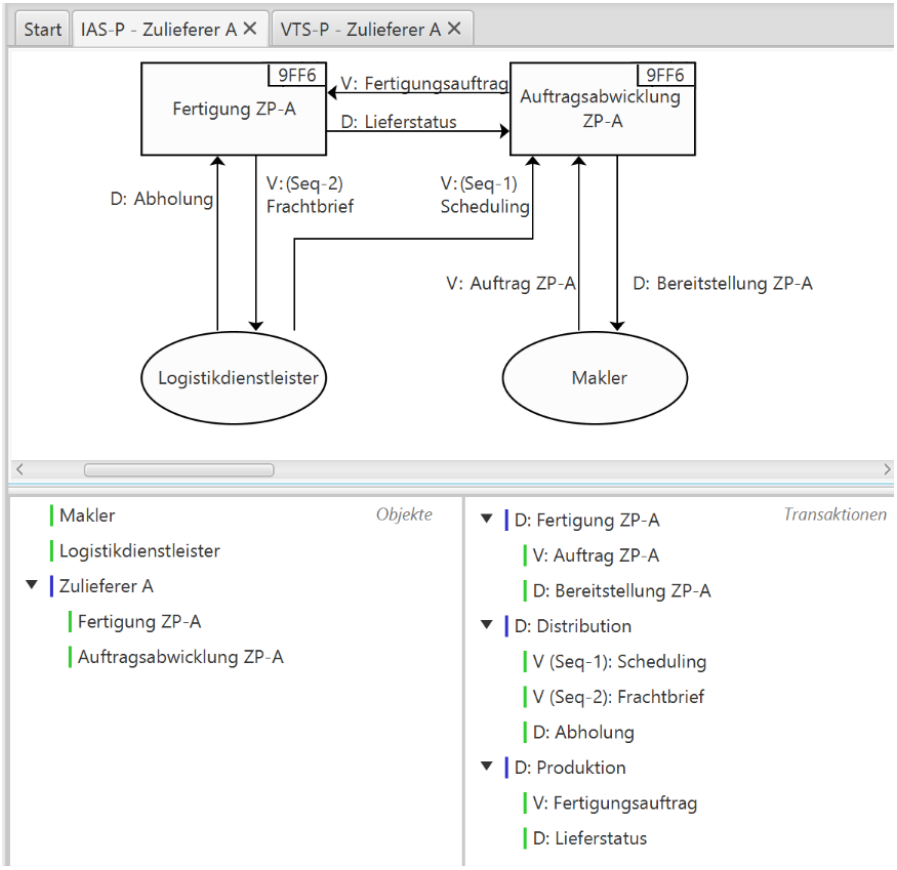



ABBILDUNG 5.8: IAS-P von Zulieferer A

hier für „>Scheduling“ erforderlich. Die hier angegebene Bedingung entspricht einer parallelen Verzweigung. Die Vorgangstypen der Transaktion „V: Fertigungsauftrag“ zwischen den nicht-öffentlich sichtbaren Zerlegungsprodukten von „Zulieferer A“ (siehe Abbildung 5.8) sind ebenso nicht-öffentlich.

#### 5.3.2.3 3. Implementierung der Vorgangsauslösung in objektspezifischen Smart Contracts

Objektspezifische Smart Contracts implementieren die objektinternen Speicher von Objekten in ihren Zustandsvariablen. Eine Vorgangsauslösung wird anhand von Funktionen ausgelöst. Weitere Funktionen zur Automatisierung der Aktionensteuerung und Aktionen können hinzukommen. Für die zuvor angeführten IAS-P für „Produzent“ und „Zulieferer A“ beinhalten die in den Schemata je Objekt angegebenen Adressen (Abbildung 5.7 bzw. 5.8) jeweils einen objektspezifischen Smart Contract. Der Quellcode ist in Anhang A angegebenen.

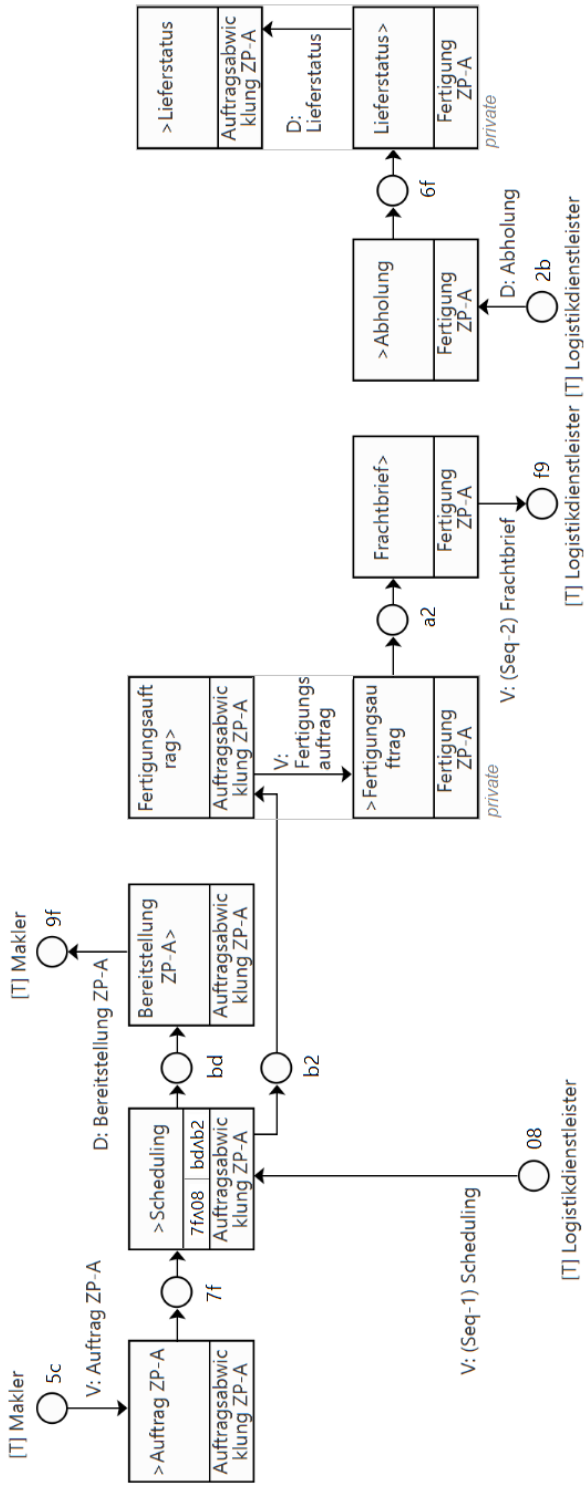



ABBILDUNG 5.9: VTS-P Zulieferer A

## 5.4 Anwendung des Ansatzes zur Prozess-Ausführung

Die Ausführung des Prozesses innerhalb des dezentralen Systems betrifft die Protokollfunktion Instanz-Monitoring. Die Auslösung des Vorereignisses der Aufgabe „V: Produktionsauftrag“ initiiert die Bildung der Instanz anhand des OC von Produzent.

### 5.4.1 Prozess-Instanz

Die Petri-Netz-Semantik des VES erlaubt eine Abbildung von Ausführungszuständen in global-öffentlichen Instanz-Zustands-Schemata (IZS) des kooperativen Prozesses. Mit dem Beginn einer Instanziierung wird das aus dem VES hervorgehende IZS um die in den OC erfassten Instanz-Zustände ergänzt. Abbildung 5.10 stellt eine Prozess-Instanz (IZS 1) nach Eingang eines Produktionsauftrages als Teil eines IZS dar.

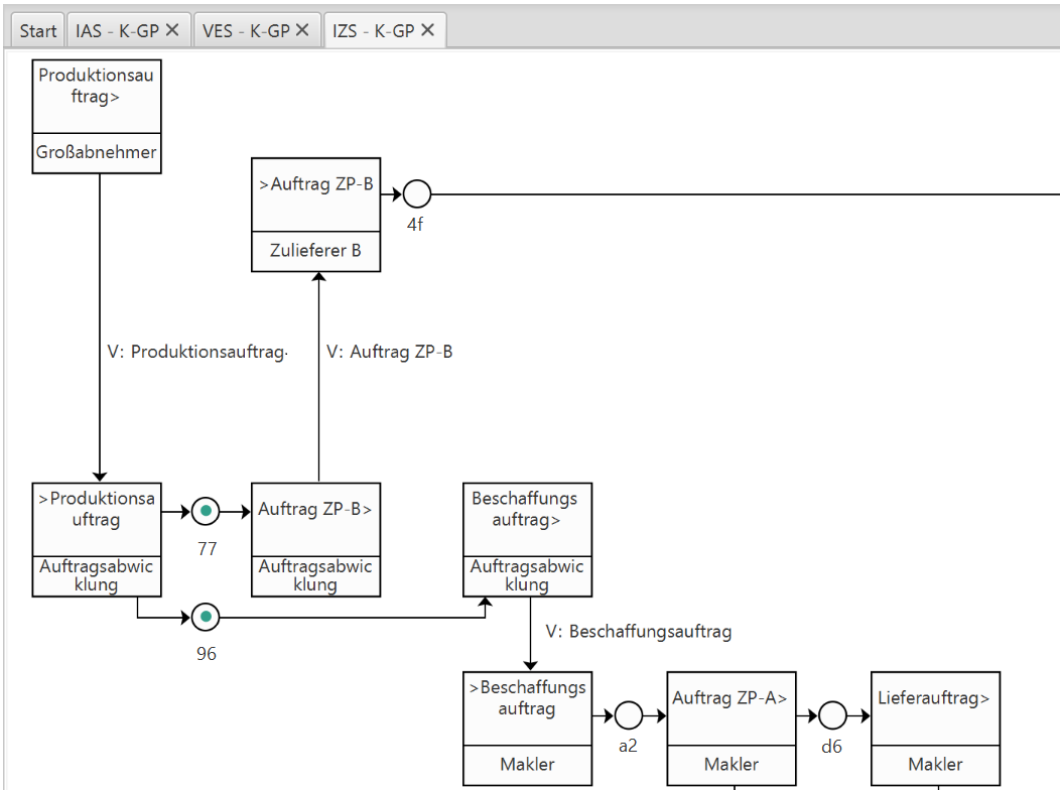



ABBILDUNG 5.10: IZS 1: Beginn des kooperativen Prozesses

Die von einzelnen Peers ausgehenden Vorgangsauslösungen führen anschließend zur Vermittlung eines Beschaffungsauftrages für ZP-A durch „Makler“. Der hierbei vor der Erfüllung der Leistung „Beschaffungsdisposition“ auftretende Zustand ist Gegenstand des IZS 2 in Abbildung 5.11.

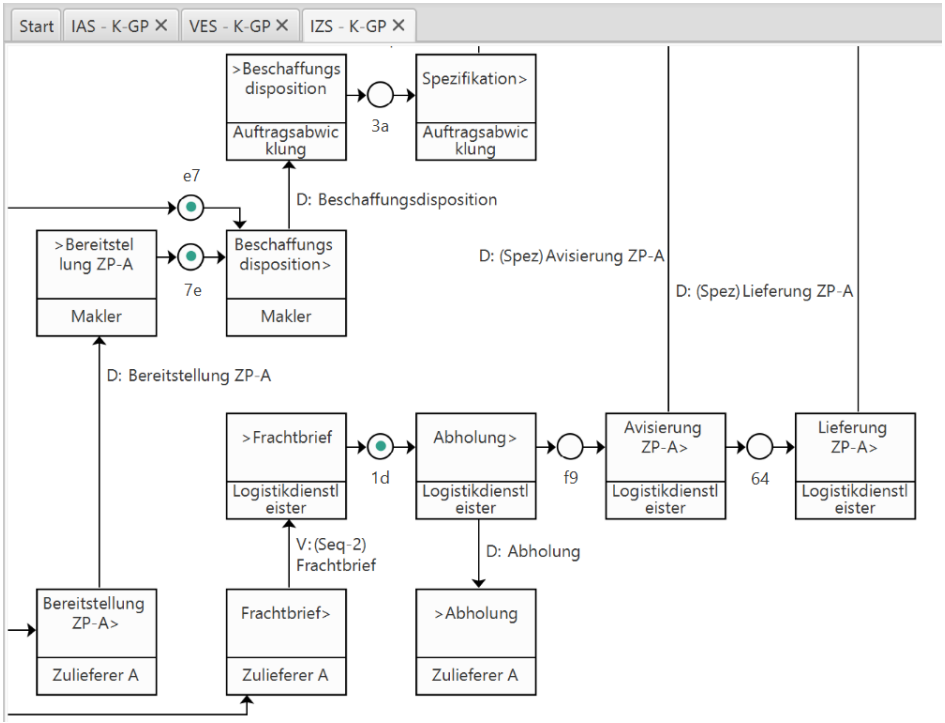



ABBILDUNG 5.11: IZS 2: Vermittlung des Fertigungsauftrages von ZP-A

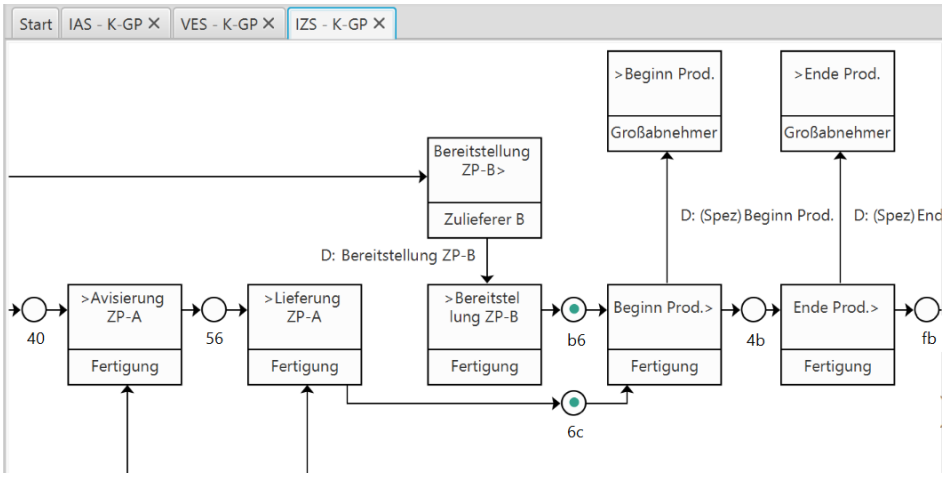



ABBILDUNG 5.12: IZS 3: Beginn des Produktionsvorgangs

In IZS 2 kann die Aufgabe „Beschaffungsdisposition>“ in dem vorliegenden Zustand durch den objektspezifischen Smart Contract von „Makler“ ausgelöst werden, da beide Vorereignisse eingetreten sind. Das IZS-P von „Makler“ enthält eine

entsprechende Pre-Condition (siehe folgender Abschnitt). Weiterhin zeigt das Schema eine innerhalb der gleichen Instanz parallel stattfindende Durchführung von Aufgaben durch „Logistikdienstleister“. Das Schema IZS 3 in Abbildung 5.12 zeigt schließlich den Zustand der Instanz vor der Auslösung des Produktionsvorgangs.

### 5.4.2 Workflow-Instanz

Die Auslösung von Vorgängen und die Nachverfolgung einzelner Instanzen aus der Sicht eines Peers kann auf die zuvor erstellten Vorgangstypschemata zurückgreifen. Ein lokal als VTS-P spezifizierter Peer-Workflow wird bei Instanziierung in ein IZS-P-Schema überführt, das die in den OC erfassten Instanz-Zustände abbildet. Die folgenden IZS-P-Schemata zeigen die zuvor aus der Sicht des kooperativen Prozesses dargestellte Instanz. Abbildung 5.13 zeigt zwei Zustände zu Beginn des Prozesses, die in den IZS-P von „Makler“ und „Auftragsabwicklung“ parallel zu Zustand IZS 1 eintreten (Abbildung 5.10).

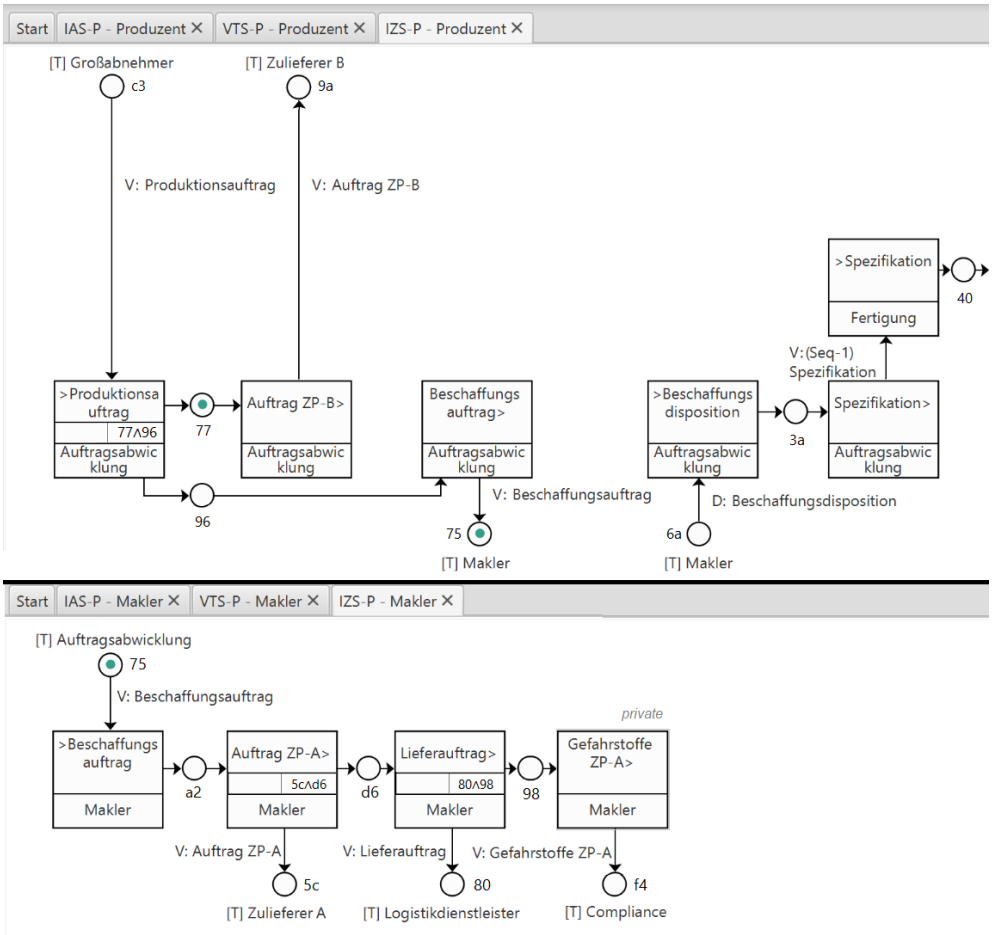



ABBILDUNG 5.13: IZS-P von „Produzent“ und „Makler“

Aus globaler Sicht (IZS 1) ist die betriebliche Transaktion „V: Beschaffungsauftrag“ noch nicht durchgeführt. Die IZS-P von „Makler“ und „Auftragsabwicklung“ zeigen zu diesem Zeitpunkt den Abschluss der Aufgabe „Beschaffungsauftrag>“ durch das zu der Transaktion gehörende T-Ereignis. Eine über die IZS-P-Schemata hinausgehende Zustandsänderung in IZS 1 ergibt sich erst durch das Eintreten des Nachereignisses von „>Beschaffungsauftrag“. Das Ereignis wird in allen Schemata anhand einer UUID identifiziert.

Die zu IZS 2 (Abbildung 5.11) korrespondierenden lokalen IZS-P-Schemata stellt Abbildung 5.14 dar.

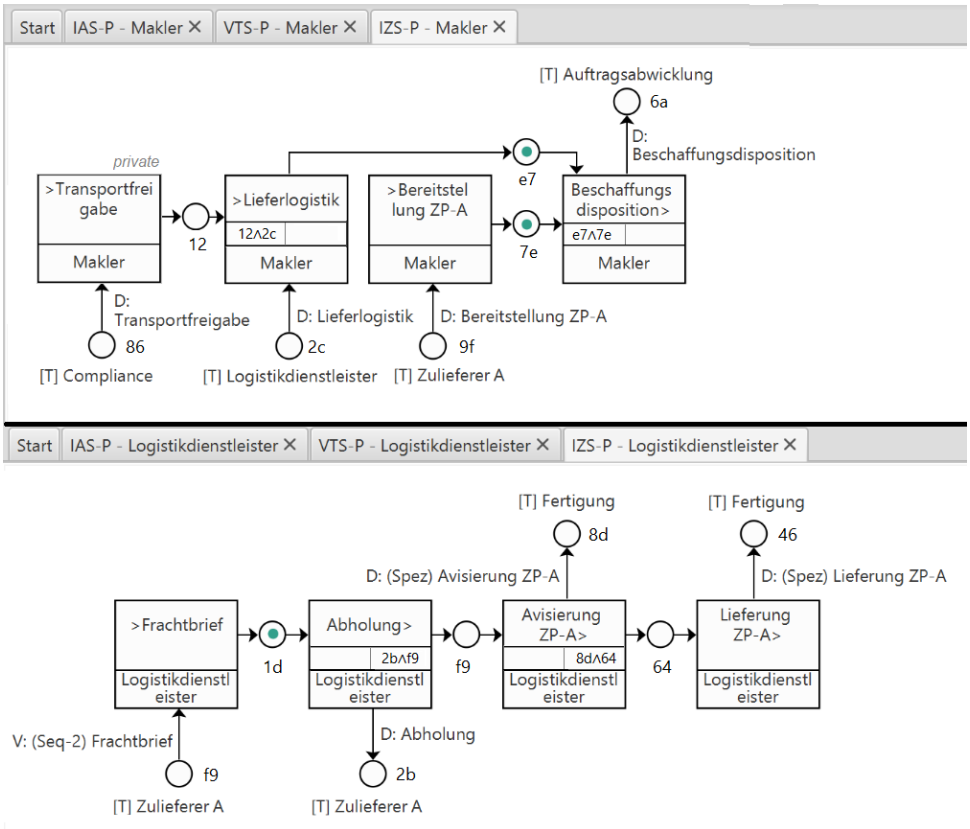



ABBILDUNG 5.14: IZS-P von „Makler“ und „Logistikdienstleister“ während der Vermittlung des Fertigungsauftrages für ZP-A

Die IZS-P von „Makler“ und „Logistikdienstleister“ bilden den Zustand der globalen Instanz ergänzt um Pre- und Post-Conditions ab. Für den bei „Makler“ lokal vorliegenden Zustand ist die Pre-Condition „e7∧7e“ erfüllt, so dass eine von diesem Objekt ausgehende Vorgangsauslösung der Aufgabe „Beschaffungsdisposition>“ erfolgen kann.

Hiervon unabhängig ist der zu diesem Zeitpunkt bei „Logistikdienstleister“ vorliegende Zustand, der sich durch das Eintreten des Ereignisses „1d“ definiert.

Der globale Zustand IZS 3 (Abbildung 5.12) unmittelbar vor Beginn des Produktionsvorgangs stellt sich aus der lokalen Sicht von „Produzent“ gemäß Abbildung 5.15 dar.

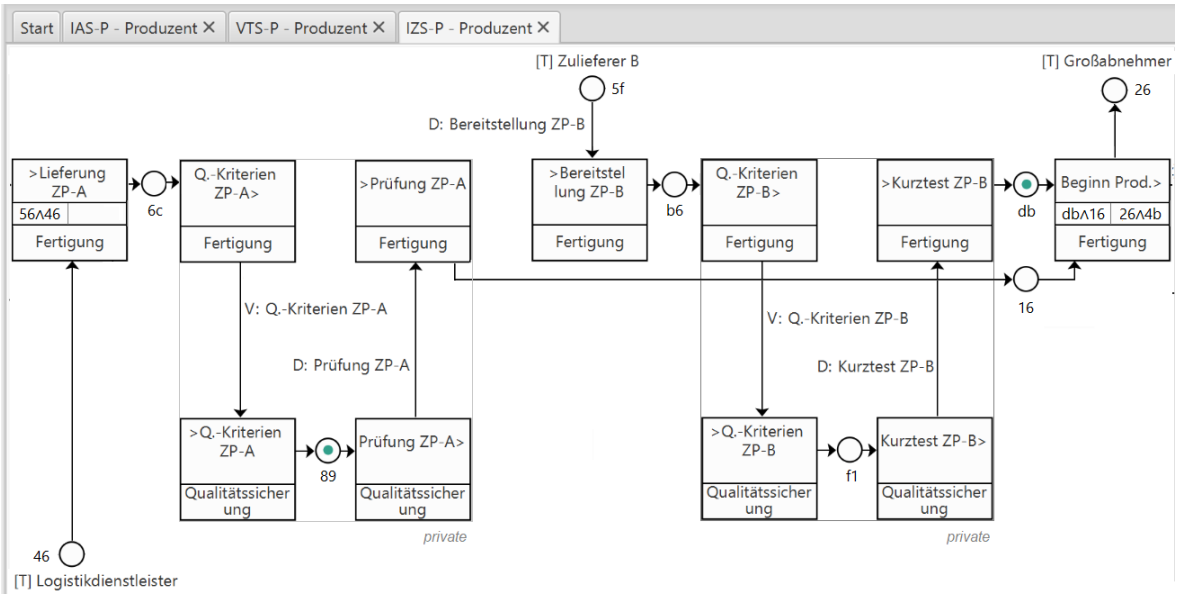



ABBILDUNG 5.15: IZS-P von „Produzent“ vor Beginn des Produktionsvorgangs

Aus globaler Sicht sind die Nachereignisse der Aufgaben „>Lieferung ZP-A“ und „>Bereitstellung ZP-B“ als „6c“ bzw. „b6“ eingetreten. Die lokale Sicht zeigt parallel hierzu die global undefinierten Ereignisse „89“ und „db“, die infolge von „>Qualitätskriterien ZP-A“ bzw. „Kurztest ZP-B“ eingetreten sind.

Die Auslösung des Produktionsvorgangs ist aus globaler Sicht in diesem Zustand möglich, sie erfordert lokal allerdings zunächst die nicht-öffentliche Durchführung eines Vorgangs zur Qualitätsüberprüfung von ZP-A. Das anschließend eintretende Ereignis „16“ erfüllt in Kombination mit Ereignis „db“ die Pre-Condition für den Beginn des Produktionsvorgangs.

### Generierung eines Instanz-Protokolls

Die Generierung eines Instanz-Protokolls durch das DWH-System ist beispielsweise mit der in Quellcode 5.1 angegebenen Anfrage realisierbar. Den Beginn des ermittelten Protokolls zeigt Abbildung 5.16.

```
1 SELECT F.EventID, SourceObjectID, SourceObjectDESC, InstantiatingPeer,
      BlockHeight, F.Timestamp
2   FROM F_InstanceState F
3     INNER JOIN D_Event EV ON F.EventID = EV.EventID
4     INNER JOIN D_Token TK ON F.TokenID = TK.TokenID
5     INNER JOIN D_Time TI ON F.Timestamp = TI.Timestamp
6     INNER JOIN D_Block BL ON F.BlockHash = BL.BlockHash
7   WHERE InstanceID = '319f8ae9-388a-4ed1-8088-ba1e0a282b26'
8   ORDER BY BlockHeight ASC
```

QUELLCODE 5.1: DWH-Anfrage zur Ausgabe eines Instanz-Protokolls

| | EventID | SourceObjectID | SourceObjectDESC | InstantiatingPeer | BlockHeight | Timestamp |
|---|---|---|---|---|---|---|
| 1 | 7742d278-ca2c-4... | 665c4e18-23ff-... | Auftragsabwicklung | 0x6C249D8C7A3A75... | 7773563 | 2019-05-16T20:52:12 |
| 2 | 96ebabc9-ecc2-4... | 665c4e18-23ff-... | Auftragsabwicklung | 0x6C249D8C7A3A75... | 7773563 | 2019-05-16T20:52:12 |
| 3 | 4f22913c-9a2c-4a... | 7df392dd-93fd-... | Zulieferer B | 0x6C249D8C7A3A75... | 7773653 | 2019-05-16T21:11:37 |
| 4 | a201a41d-8ed2-4... | 571ed6d0-8155... | Makler | 0x6C249D8C7A3A75... | 7773664 | 2019-05-16T21:14:30 |
| 5 | 7f670d9a-dbac-43... | e799ab03-1615... | Zulieferer A | 0x13AE9557122D16... | 7773691 | 2019-05-16T21:21:26 |
| 6 | d635b91c-1f17-46... | e799ab03-1615... | Zulieferer A | 0x13AE9557122D16... | 7773691 | 2019-05-16T21:21:26 |
| 7 | 5091e978-ceac-4... | a1a52452-bd63... | Logistikdienstleister | 0x83818ACDCE46B1... | 7773694 | 2019-05-16T21:22:17 |

ABBILDUNG 5.16: Instanz-Protokoll

## 5.5 Veränderung des Prozess-Schemas

Das Szenario sieht Änderungen in der Entwicklung des Prozesses vor. Diese beziehen sich auf die Qualitätssicherung und die Wahrung von Compliance-Richtlinien hinsichtlich der herzustellenden Produkte.

$\delta$1. Innerhalb des kooperativen Prozesses wird eine durch den Produzenten ausgeführte Qualitätsprüfung nach der Anlieferung von ZP-A zunächst ersatzlos entfernt.

$\delta$2. Innerhalb des kooperativen Prozesses kommt ein von Zulieferer A zu verantwortender Bericht zur Qualitätssicherung hinzu, der vor der Auslieferung von ZP-A an den Produzenten übermittelt wird.

$\delta$3. Produzent nimmt innerhalb des privaten Prozesses einen Kurztest angelieferter ZP-A auf.

$\delta$4. Zulieferer A nimmt innerhalb des privaten Prozesses eine nach der Produktion stattfindende Qualitätskontrolle auf, welche $\delta$2 lokal implementiert.

$\delta$5. Zulieferer A sichert die nun durchgeführte Qualitätskontrolle zur Wahrung von Compliance-Richtlinien zu.

Die Änderung $\delta$1 eines Peer-Workflows wird in einer Entwurfstransaktion implementiert:

1. **Modelloperation zu $\delta$1 innerhalb des Peer-Workflows von Produzent:** Entfernen der Aufgaben „Q.-Kriterien ZP-A>“ und „>Prüfung ZP-A“ des Objekts „Fertigung“ sowie der hierzu korrespondierenden Aufgaben „>Q.-Kriterien ZP-A“ und „Prüfung ZP-A>“ des Objekts „Qualitätssicherung“. Die zugehörigen Transaktionen werden ebenso entfernt.
2. **Modell-Management:** Auslösen einer Commit-Operation innerhalb des privaten Branches des Versionsgraphen von Produzent.

Die Änderung $\delta$2 des kooperativen Prozesses wird anhand der folgenden Entwurfstransaktion implementiert:

1. **Modelloperation zu $\delta$2 innerhalb des kooperativen Prozesses:** Hinzufügen der Transaktion „D: Qualitätsbericht“ zwischen den Objekten Zulieferer A und Produzent mit entsprechenden Aufgaben. Die von Zulieferer A ausgehende Aufgabe wird sequenziell nach „>Abholung“ angefügt.

2. **Modell-Management:** Auslösen einer Commit-Operation innerhalb des öffentlichen Branches des Versionsgraphen durch Produzent.
3. **Vereinbarung des Modellsystems:** Auslösen einer Blockchain-Transaktion durch das System zur verbindlichen Vereinbarung des Modells per Zwei-Phasen-Commit. Die Ausführung des Commits bei Produzent, Makler, Logistikdienstleister und Zulieferer A umfasst:
   - Syntaktische und semantische Evaluierung des Modells.
   - Aufruf der Vote-Commit-Funktion des globalen Smart Contracts.
4. Durchführung des Commits in Version 91C13 des Versionsgraphen. Der Hash-Wert dieses Commits ist in Block 7748079 der Ethereum-Blockchain persistiert[3].

Die Änderung $\delta 3$ eines Peer-Workflows wird in einer Entwurfstransaktion implementiert:

1. **Modelloperation innerhalb des Peer-Workflows von Produzent:** Hinzufügen der Aufgaben „Q.-Kriterien ZP-A>“ und „>Kurztest ZP-A“ des Objekts „Fertigung“ sowie der hierzu korrespondierenden Aufgaben „>Q.-Kriterien ZP-A“ und „Kurztest ZP-A>“ des Objekts „Qualitätssicherung“. Entsprechende Transaktionen kommen ebenso hinzu.
2. **Modell-Management:** Auslösen einer Commit-Operation innerhalb des privaten Branches des Versionsgraphen von Produzent.

Die Änderung $\delta 4$ eines Peer-Workflows wird in einer Entwurfstransaktion implementiert:

1. **Modelloperation innerhalb des Peer-Workflows von Zulieferer A:** Hinzufügen der Aufgabe „Qualitätskontrolle>“ des Objekts „Fertigung ZP-A“ sowie der hierzu korrespondierenden Aufgabe „>Qualitätskontrolle“ des Objekts „Auftragsabwicklung ZP-A“. Eine entsprechende Transaktion kommt ebenso hinzu.
2. **Modell-Management:** Auslösen einer Commit-Operation innerhalb des privaten Branches des Versionsgraphen von Zulieferer A.

Anschließend kann die Zusicherung der Einhaltung der Compliance-Richtlinie in $\delta 5$ erfolgen. Diese ist nicht Bestandteil der Modellierung.

Abbildung 5.17 zeigt die globale Sicht des nun bestehenden kooperativen Prozesses.

[3] siehe z.B. Transaktion 0x86a3c7f4df8e8b6e4435b5e7535daa3ed28a29025bd41a9a1b0122fd6dbe7517

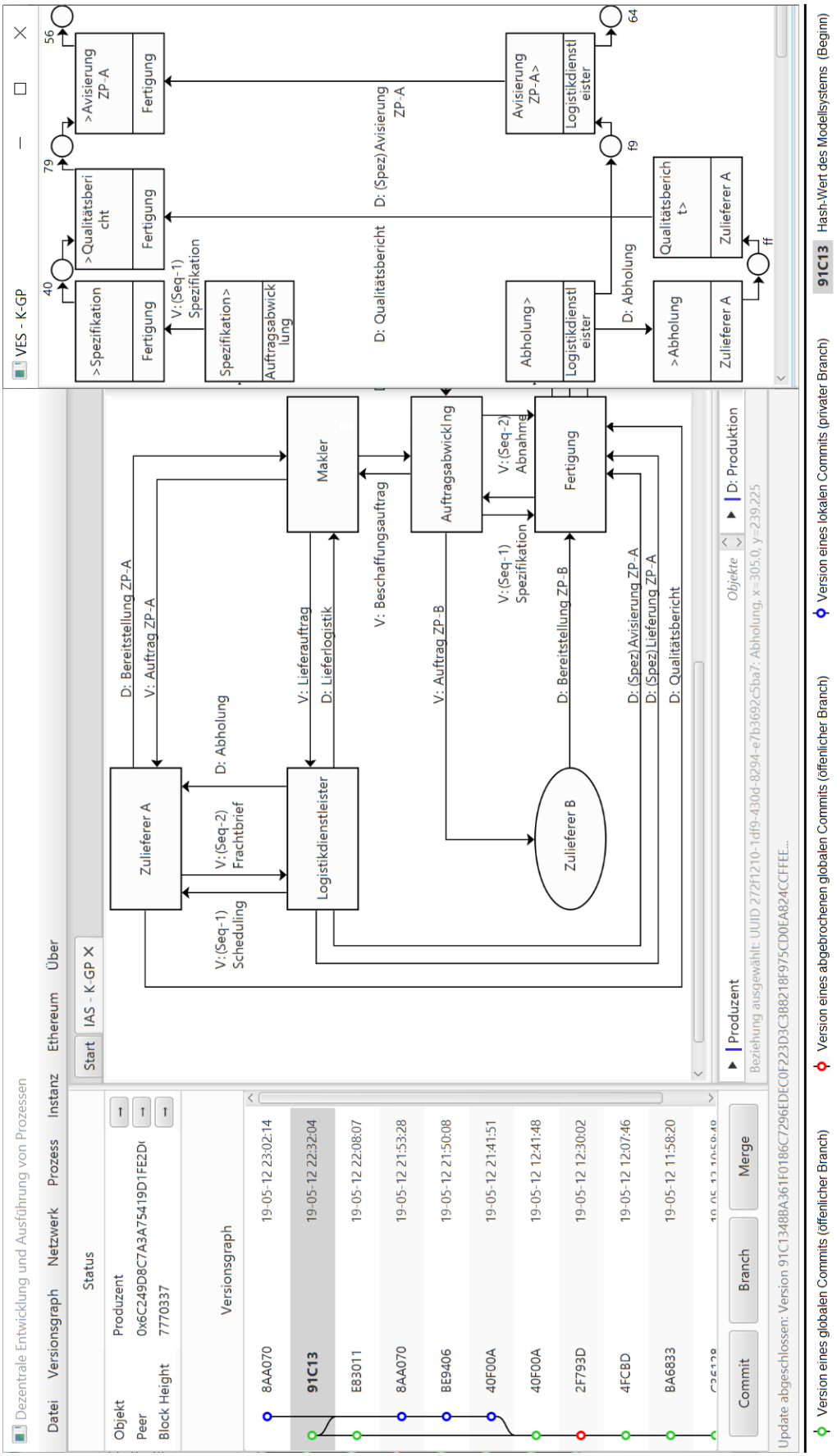



ABBILDUNG 5.17: IAS und VES des kooperativen Prozesses nach $\delta 4$

## 5.6 Ergebnisdiskussion

Dezentrale Systeme sind für interorganisationale Prozesse unter Beteiligung von global verteilten Organisationen von hoher Relevanz. Integrität und Verbindlichkeit bei der Vereinbarung und Ausführung von Geschäftstransaktionen können hier unabhängig von einzelnen Organisationen gewährleistet werden.

Das Supply-Chain-Szenario zeigt eine während der Entwicklung und Ausführung von Prozessen stattfindende Modellierung kooperativer Prozesse, die eine interaktive und gemeinsame Bildung von Modellen und die Übereinkunft hinsichtlich der zur Instanziierung auszuwählenden Modelle unterstützt. Dabei wird die Integration von Prozessmodellen mit der verteilten Erstellung der Modelle sichtbar. So ist die Koordination durch das Protokoll des dezentralen Systems determiniert und auf die innerhalb der Modelle hinterlegte Organisationenabgrenzung bezogen. Die Modellierung und Veränderung der Teilsysteme von Produzent, Zulieferer A und Zulieferer B erfordert jeweils eine nicht-öffentliche Detaillierung und Zerlegung innerhalb der Objekte. Gleichzeitig muss die Integration mit einer globalen Choreographie unterstützt werden.

Die hier entstehenden Modelle unterscheiden sich aufgrund des integrativen Ansatzes von den innerhalb der Fallstudie vorzufindenden öffentlichen und privaten Modellen (Fdhila et al. 2015). So verwendet der hier beschriebene Ansatz keine Mapping-Definitionen zwischen globalen und lokalen Modellen, sondern vereinigt ein globales Modell mit den aus der Organisationenabgrenzung hervorgehenden lokalen Modellen je Objekt anhand eines Objektbaumes. Die modulare Eingliederung und Abtrennung von Modellen, basierend auf Hash-Werten, führt je Objekt zu lokalen Sichten auf das Modell des kooperativen Prozesses. Dies wird während der Veränderung des Prozesses (Abschnitt 5.5) in den Aufgaben zur Qualitätskontrolle (Produzent und Zulieferer A, $\delta 4$.) sichtbar. Die lokalen Sichten beider Objekte umfassen jeweils den Gesamtprozess und enthalten Informationen über die Sichtbarkeit der beteiligten Objekte. Damit liegt eine verteilte Bildung lokaler Sichten vor.

Die Implementierung erlaubt es, Kooperationen zu bilden, Prozesse zu gestalten und deren Ausführung zu überwachen. Dies wird ausschließlich auf der Basis von Modellen realisiert, nicht aber durch andere Aspekte der strategischen Planung oder des operativen Betriebs. Diese Limitation kommt v.a. durch die modellbasierte Überwachung von Instanz-Zuständen zum Ausdruck. Die Implementierung ersetzt damit nicht den Einsatz weitere Systeme für das Supply-Chain-Management oder

das Enterprise-Resource-Planning. Schließlich unterliegt die Erstellung von globalen Commit-Operationen und die Durchführung von Änderungen den Einschränkungen zur Skalierbarkeit von Blockchain-Systemen. Transaktionen erfordern eine Bestätigungszeit von mehreren Minuten und verursachen Transaktionskosten im Bereich von einigen Cent (Härer und Fill 2019b). In Abhängigkeit der Auslastung des Blockchain-Systems können Zeit und Kosten zunehmen.

Die Implementierung zeigt die verbindliche Einhaltung der Abgrenzung zwischen globalen und lokalen Modellen, die durch das Protokoll des dezentralen Systems sichergestellt wird. Die anhand von Smart Contracts implementierten Protokollfunktionen koordinieren die Erstellung von Modellen und deren gesicherte Speicherung in verteilten Versionskontrollsystemen. Damit ist eine dezentrale Koordination gegeben, welche die Einhaltung des Protokolls durch Smart Contracts gewährleistet und auf deren Funktionen beschränkt. Die definitive Einhaltung und die verteilte Validierung des Protokolls, ohne die keinerlei Interaktion möglich ist, unterscheiden dieses System von einem klassischen verteilten System oder einem Datenbank-System. Die Gewährleistung der Korrektheit der Protokollausführung auf technischer Ebene wird schließlich mit der Nachverfolgung der Ausführung auf die fachliche Ebene übertragen.

Die Anwendung des Ansatzes zeigt die Möglichkeit der Entwicklung eines dezentralen Systems ausgehend von den daran beteiligten Komponenten. Die Ausführung der innerhalb des Systems vereinbarten und systemimmanent durchgeführten Transaktionen führt zu einem System, dessen Verhalten transparent für alle Teilnehmer nachvollziehbar und hinsichtlich der Korrektheit der Ausführung validierbar ist.

# Kapitel 6

# Zusammenfassung und Ergebnisdiskussion

## 6.1 Zusammenfassung

### 6.1.1 Geschäftsprozessorientierte Entwicklung betrieblicher Systeme

Der Begriff der Dezentralisation bezieht sich in der Organisationstheorie auf die Verteilung von Merkmalen der Aufbau- und Ablauforganisation. Die sich hierdurch ergebende Struktur von autonomen und lose gekoppelten Teilsystemen ist durch ein objektorientiertes Aufgabenmodell abbildbar. Die lose Kopplung von Objekten und deren nachrichtenbasierte Kommunikation führt zu einem System, das die nichthierarchische Struktur eines Netzes aufweist. Zur Abbildung der dort enthaltenen Prozesse in Struktur und Verhalten besteht die Anforderung, autonome und lose gekoppelte Teilsysteme anhand von Modellen zu erfassen. Auf der Ebene betrieblicher Aufgaben wird die Modellierung von Prozessen in dezentral organisierten Systemen somit durch die SOM-Methodik prinzipiell unterstützt. Workflows der Aufgabenträgerebene sind durch bestehende Erweiterungen und Transformationen ableitbar.

Die Umsetzung einer dezentralen Wertschöpfung über mehrere Organisationen hinweg wirkt sich auf die Prozesse und Paradigmen der Wertschöpfung aus. Eine

durch die Dezentralisierung bedingte Verteilung der Aufbau- und Ablauforganisation in interorganisationalen Prozessen zieht eine kooperative und verteilte Entwicklung und Ausführung von Prozessen nach sich. Die hiermit realisierte Wertschöpfung weist Merkmale einer interaktiven und kooperativen Wertschöpfung auf. Hierdurch ergibt sich die Anforderung, kooperative Merkmale hinsichtlich (a.) der Modellierung von Prozessen und (b.) der Interaktion während der verteilten Modellierung zu berücksichtigen. Dies betrifft (a.) die Modellabbildung von kooperativen Prozessen (Collaborative Processes) in Form von privaten und öffentlichen Prozessen sowie (b.) die Nutzung von Methoden zur verteilten Modellierung, wie z.B. Modell-Repositories und Versionierungssysteme. Eine Abgrenzung von Organisationen wird durch die Bildung von Teilsystemen anhand der SOM-Methodik unterstützt, wobei die Abgrenzung um Aspekte der Sichtbarkeit und des Nachrichtenaustausches innerhalb und außerhalb von kooperierenden Objekten erweitert wird.

Eine Untersuchung verwandter Ansätze zur Abbildung von kooperativen Prozessen führt zu dem Ergebnis, dass die Abbildung von Prozessen in dezentralen Systemen unvollständig unterstützt wird. Insbesondere die fehlende Abbildbarkeit der Systemstruktur und eine Abgrenzung von Teilsystemen über mehrere Abstraktionsebenen werden damit nicht erfasst. Da existierende Ansätze nicht auf dezentrale Systeme ausgerichtet sind, fehlt zudem die Möglichkeit einer lokalen Sichtenbildung für globale Modelle von Prozessen.

### 6.1.2 Blockchain-Systeme

Blockchain-Systeme sind eine mögliche Grundlage zur technischen Realisierung von dezentralen Organisationen.

Ein Blockchain-System umfasst die Komponenten Datenstruktur, Protokoll und Netzwerk. Ein Protokoll regelt die validierbar konsistente Speicherung der aus rückwärts verketteten Blöcken bestehenden Datenstruktur in den Knoten eines Peer-to-Peer-Netzwerks. Das Protokoll stellt global übereinstimmende Systemzustände her, deren Integrität und Verbindlichkeit validierbar ist und damit für Transaktionen zwischen a priori unbekannten Peers herangezogen werden kann. Smart Contracts erweitern dieses Konzept um Programme, die aufgrund ihrer autonomen Ausführung und der Nachvollziehbarkeit der Korrektheit der Ausführung den Charakter verbindlicher Verträge besitzen.

Ein dezentrales System ist ein verteiltes System, das eine von den Systemkomponenten ausgehende, verteilte Koordination unterstützt. Das System wird durch die

validierbare Ausführung eines Protokolls koordiniert. Diese Eigenschaften treffen auf dezentrale Blockchain-Systeme zu, die offene Kommunikationssysteme diesbezüglich erweitern. Extensionen des Protokolls zur Laufzeit werden in einigen Blockchain-Systemen durch Smart Contracts unterstützt. Die Annahme a priori unbekannter Teilnehmer bedingt den Einsatz von Protokollen wie Proof-of-Work, die konsistente Systemzustände im Zeitverlauf herstellen. Die Limitationen des BASE-Paradigmas treten mit zunehmender Knotenzahl in den Hintergrund.

Verbindlichkeit und Integrität ohne zentrale Koordination sind die Basis zur dezentralen Ausführung von betrieblichen Transaktionen.

### 6.1.3 Integrierte Entwicklung und Ausführung von Prozessen

Der vorgeschlagene Ansatz umfasst drei Teilsysteme zur Modellierung, zur kooperativen Vereinbarung von Prozessen sowie zur Überwachung der Ausführung.

**Modellsystem**

Das Modellsystem definiert die innerhalb des dezentralen Systems herangezogenen Modelle (1.) der System-Gestaltung, (2.) der Prozess-Gestaltung und (3.) der Prozess-Ausführung. Die Modelle gliedern sich in eine globale Aufgabenebene sowie eine lokale Aufgabenträgerebene der Modelle einzelner Peers.

Das Modellsystem wird anhand von Metamodellen auf Basis der SOM-Methodik instanziiert. Die System-Gestaltung (1.) bildet die Vernetzung von Peers und die darauf basierende Bildung eines Wertschöpfungsnetzes auf Basis eines globalen Interaktionsschemas ab. Die Prozess-Gestaltung (2.) sieht eine Teilsystembildung als Organisationenabgrenzung vor, welche die kooperativen Prozesse der beteiligten Objekte in Struktur und Verhalten als IAS bzw. VES modelliert.

Diese Modelle sind die Basis für die Ableitung von Peer-Modellen der Aufgabenträgerebene, die eine lokale Detaillierung der zuvor abgegrenzten Teilsysteme beschreiben. Ein Modell eines Peers bildet einen privaten Prozessteil des kooperativen Prozesses ab. Das Modell entspricht einer lokalen Sicht auf den kooperativen Prozess. Die Prozess-Ausführung (3.) erfasst die Zustände einzelner Instanzen während der Ausführung. Hierfür wird die Petri-Netz-Semantik des VES herangezogen.

**Kooperationssystem**

Das Kooperationssystem beschreibt die Koordination der kooperativen und verteilten Erstellung des Modellsystems. Das Teilsystem realisiert die Funktionen (1.)

Modellmanagement und (2.) Vereinbarung des Modellsystems. Das Modellmanagement beschreibt einen Ansatz zur verteilten Verwaltung von Modellen basierend auf einem Versionsgraphen, der eine durch ein Blockchain-System abgesicherte Versionierung von Modellen unterstützt.

Das System ist durch eine verteilte Versionsverwaltung außerhalb der Blockchain skalierbar. Die Vereinbarung von Modellen (2.) betrifft die Durchführung von verteilten Commit-Verfahren, die anhand von Smart Contracts koordiniert werden.

**Ausführungssystem**

Das Ausführungssystem beschreibt die Koordination der Ausführung von Prozessen anhand der Instanz-Modelle des Modellsystems. Das Teilsystem realisiert die Funktion Instanz-Monitoring zur Validierung der Korrektheit der Ausführung. Instanz-Zustände in Form von Modellen werden anhand von objektspezifischen Smart Contracts global verteilt und in die lokalen Sichten einzelner Peers übernommen. Die fachliche Ausführung von verteilt koordinierten Prozessen ist hiermit nachvollziehbar und wird auf Basis der Transaktionshistorie der Blockchain in Protokollen dokumentiert.

**Fallstudie**

Der Ansatz wird anhand einer Fallstudie eines Supply-Chain-Szenarios instanziiert. Hiermit wird die Integration der Teilsysteme des Ansatzes und damit die Integration der Entwicklung und Ausführung von Prozessen in einer dezentralen Organisation gezeigt.

## 6.2 Ergebnisdiskussion

Die Entwicklung und Ausführung von Prozessen in dezentralen Systemen bedingt organisationstheoretisch eine Dezentralisation der Planung, Ausführung und Kontrolle von betrieblichen Aufgaben. Entstehende betriebliche Systeme sind verteilt und unterliegen keiner zentralen Koordination. Historisch zieht dies eine von einzelnen Organisationen ausgehende Verteilung der Aufbau- und Ablauforganisation nach sich. Mit der Dezentralisierung der Entwicklung von betrieblichen Systemen wird der Ansatz verfolgt, die Verteilung ausgehend von den Beziehungen zwischen Organisationen zu gestalten. Damit steht die Gestaltung eines Wertschöpfungsnetzes im Vordergrund, dessen Prozesse nicht der Kontrolle einzelner beteiligter Organisationen unterliegen.

Diese Arbeit zeigt die prinzipielle Möglichkeit der Gestaltung solcher Systeme auf der Basis eines Modellsystems, das innerhalb des zu gestaltenden Systems unter dezentraler Koordination entsteht. Damit entfällt aus der Sicht der beteiligten Organisationen eine explizite Trennung der Gestaltungszeit und Laufzeit, da nicht der Entwurf von Prozessen aus der Sicht einer Organisation, sondern die kooperative und verteilte Entwicklung von Prozessen des interorganisationalen Systems im Vordergrund steht. Prozesse dieses Systems überschneiden sich hinsichtlich der Gestaltungszeit und der Laufzeit. Der vorgeschlagene Ansatz setzt daher eine integrative Entwicklung voraus, die über mehrere Teilsysteme hinweg in Kooperation erfolgt, während innerhalb eines Teilsystems einer beteiligten Organisation eine autonome und lokale Entwicklung von privaten Prozessen stattfindet (H.1, Kapitel 1.3).

Die Integration der Prozesse lokaler Teilsysteme mit den kooperativen Prozessen der globalen Choreographie erfordert die Betrachtung des gesamten Objektsystems und eine hiervon ausgehende Bildung lokaler Prozesse und Sichten. Die Betrachtung und Abbildung voneinander isolierter Teilsysteme (H.2) ist aufgrund der Überschneidung von Gestaltungszeit- und Laufzeitaspekten nicht ausreichend. Die zugrunde liegenden Modelle entstehen auf globaler Ebene ausgehend von kooperativen Prozessmodellen, denen lokale Modelle hierarchisch untergeordnet sind. Somit besteht eine direkte Einbindung lokaler Prozesse in ein globales Netz von Prozessen. Mit dieser Architektur wird ein integrativer Ansatz unterstützt, der eine dezentrale Wertschöpfung über mehrere Ebenen hinweg beschreibt.

Die Gestaltung dieser Prozesse als Teil eines dezentralen Systems erfordert eine Integration des Entwurfs und der Ausführung (H.3), da ein dezentrales System eine Verteilung der Systemkomponenten sowie eine Verteilung der Koordination voraussetzt. In einem solchen System ist die Vereinbarung kooperativer Prozesse ein Bestandteil der dezentralen Koordination, so dass sich die Validierung der Korrektheit einer Prozessausführung auf die zuvor getroffene Vereinbarung stützen muss. Eine Validierung des Eintretens der Instanz-Zustände vereinbarter Prozesse bedingt damit die Nachvollziehbarkeit dieser Zustände für alle Systemteilnehmer. Die transparente Nachvollziehbarkeit ermöglicht damit eine Validierung der Ausführung, die über die syntaktische Überprüfung zugrunde liegender Daten hinausgeht. Geschäftstransaktionen werden unabhängig von beteiligten Organisationen über Unternehmensgrenzen hinweg fachlich nachvollziehbar und validierbar.

Die Entwicklung und Ausführung werden anhand von Modellen koordiniert, deren verbindlicher Charakter durch die Absicherung der Integrität und Verbindlichkeit anhand eines Blockchain-Systems sichergestellt wird. Blockchain-Systeme,

und Distributed-Ledger-Technologien im Allgemeinen, unterstützen insbesondere durch diese beiden Merkmale eine nicht-zentral koordinierte Herausbildung von dezentral organisierten Systemen (H.4). Die Kombination der Verteilung von Systemkomponenten und der Verteilung der Koordination führt damit zu einer systemimmanenten Validierbarkeit des innerhalb des Systems erfassten Verhaltens.

Dezentrale Blockchain-Systeme sind offene Plattformen, die offene Kommunikationsnetzwerke um eine konsistente, globale und validierbare Speicherung und Verarbeitung von Daten erweitern. Die Verfügbarkeit eines globalen Systemzustandes, dessen Integrität von allen Systemkomponenten validierbar ist, führt zu einem vertrauenswürdigen Single-Point-of-Truth. Dieser kann u.a. für die Integration von Anwendungssystemen herangezogen werden, die Teil eines dezentralen Systems werden. Derzeit bestehen Limitationen hinsichtlich der Skalierung des Durchsatzes und der Latenzzeiten bei der Ausführung und Speicherung von Transaktionen. Zudem legt die Vielgestaltigkeit aktueller Blockchain-Systeme eine Standardisierung unter Verallgemeinerung bestehender Implementierungen nahe. Perspektivisch besteht die Möglichkeit, sichere und verteilte Systeme zu realisieren, die Daten ohne Abhängigkeiten zu individuellen Anwendungssystem-Instanzen speichern und verarbeiten.

Zusammenfassend zeigt der vorliegende Ansatz schließlich die prinzipielle Möglichkeit zur Entwicklung komplexer Systeme durch eine modellbasierte und auf Blockchains gestützte Methodik. Insgesamt zeigt sich, dass Blockchain-Technologien geeignet sind, um organisationale Systeme als offene Plattformen bei zunehmender Komplexität und Verteilung ausgehend von autonomen Systemkomponenten zu koordinieren (H.5). Komplexe und hochverteilte soziotechnische Systeme werden ausgehend von deren autonomen Komponenten durch Blockchain-Technologien beherrschbar. Mögliche Auswirkungen sind geringere Transaktionskosten, Disintermediation, eine Öffnung von Märkten, die bisher unter der Kontrolle von Einzelnen stehen, und, perspektivisch, eine höhere gesamtwirtschaftliche Stabilität.

# Anhang A

# Smart Contracts

Die Smart Contracts sind für das Blockchain-System Ethereum in der Sprache Solidity implementiert. Die folgende Quellcode-Auflistung A.1 gibt den globalen Smart Contract an. Der objektspezifische Smart Contract ist Gegenstand von Quellcode-Auflistung A.2.

## A.1 Globaler Smart Contract

```
pragma solidity ^0.5.3;

/// @title Globaler Smart Contract
contract GlobalContract {

  // Anzahl bekannter Objekte
  uint16 public nObjects;

  // Anzahl bekannter Beziehungen
  uint16 public nRelations;

  // Anzahl bekannter Peers
  uint16 public nPeers;

  // Anzahl bekannter kooperativer Prozesse
  uint16 public nCollaborations;

  // Zuordnung: UUID Objekt -> Objekt
  mapping(bytes16 => Object) public knownObjects;

  // Zuordnung: ID relationID -> Beziehung
  mapping(uint16 => Relation) public knownRelations;

  // Zuordnung: ID peerID -> Peer
  mapping(uint16 => Peer) public peers;

  // Zuordnung: Adresse Peer -> Array zugeordneter Objekte
  mapping(address => bytes16[]) public objectMap;

  // Zuordnung: ID collaborationID -> Kooperation
  mapping(uint16 => Collaboration) public knownCollaborations;

  // Zuordnung: ID commitProcedureID -> Two-Phase-Commit-Verfahren
  mapping(uint16 => CommitProcedure) public commitProcedures;

  // Datentyp eines Objekts
  struct Object {
    // Objekt-ID  (sequenziell, ID > 0)
    uint16 objectID;
    // Peer-Zuordnung
    uint16 peerID;
    // Name
```

```
43      bytes32 name;
44      // objektspezifischer Smart Contract
45      address objectSpecificContract;
46      // ID der Kooperation
47      uint16 collaborationID;
48      // ID des laufenden Commit-Verfahrens
49      uint16 commitProcedureID;
50    }
51
52    // Datentyp einer Beziehung
53    struct Relation {
54      bytes16 sourceObject;
55      bytes16 targetObject;
56      bytes16 uuid;
57      bytes32 name;
58      RelationType relType;
59    }
60
61    // Beziehungstypen des IAS-N
62    enum RelationType { ZRelation, RRelation, LRelation }
63
64    // Datentyp eines Peers
65    struct Peer {
66      // Name
67      bytes32 name;
68      // Peer-Adresse
69      address peer;
70      // Versionsgraph URI (privater Prozess)
71      bytes32 versiongraphAddressURI;
72    }
73
74    // Datentyp eines kooperativen Prozesses
75    struct Collaboration {
76      // Anzahl kooperierender Objekte
77      uint16 nObjects;
78      // Modell-Hash-Wert
79      bytes32 modelHashValue;
80      // ID des laufenden Two-Phase-Commit-Verfahrens
81      uint16 commitProcedureID;
82      // Versionsgraph URI (oeffentlicher Prozess)
83      bytes32 versiongraphAddressURI;
84    }
85
86    // Datentyp eines Commit-Verfahrens
87    struct CommitProcedure {
```

```
88      // Ausfuehrungszustand des Commit-Verfahrens
89      CommitProcedureState state;
90      // Ergebnis des Commit-Verfahrens
91      CommitProcedureResult result;
92      // Anzahl Votes des Commit-Verfahrens
93      uint16 nVotes;
94      // Modell-Hash-Wert des laufenden Commit-Verfahrens
95      bytes32 modelHashValue;
96    }
97
98    // Ausfuehrungszustaende des verteilten Commit-Verfahrens
99    enum CommitProcedureState { Init, Wait }
100
101   // Ergebnisse des verteilten Commit-Verfahrens
102   enum CommitProcedureResult { Abort, Commit }
103
104   /// @notice Zuordnen des uebergebenen Objekts zum aufrufenden Peer
105   /// @param object Objekt, von dem der Commit ausgeht
106   /// @param name Name des Objekts (max. 32 byte)
107   function assignObject(bytes16 object, bytes32 name) public {
108     // Existierende Zuordnung nicht ueberschreiben
109     if (knownObjects[object].peerID != 0) {
110       return;
111     }
112     // Peer-ID vergeben oder ermitteln
113     uint16 peerID;
114     if (objectMap[msg.sender].length == 0) {
115       // Peer-ID vergeben
116       peerID = ++nPeers;
117       peers[peerID].peer = msg.sender;
118       peers[peerID].name = name;
119     } else {
120       // Peer-ID ermitteln
121       bytes16 existingObject = objectMap[msg.sender][0];
122       peerID = knownObjects[existingObject].peerID;
123     }
124     // Objekt-ID und -Name vergeben
125     uint16 objectID = ++nObjects;
126     knownObjects[object].objectID = objectID;
127     knownObjects[object].name = name;
128     // Zuordnung
129     knownObjects[object].peerID = peerID;
130     objectMap[msg.sender].push(object);
131     emit ObjectAssigned(objectID, object, name, msg.sender);
132   }
```

```
133
134   // Event bei Zuordnung eines Objekts
135   event ObjectAssigned(uint16 indexed objectID, bytes16 object,
          bytes32 name, address peer);
136
137   /// @notice Ermittelt, ob das angegebene Objekt registriert ist
138   /// @param object Objekt
139   /// @return TRUE: Objekt registriert, FALSE: nicht registriert
140   function objectExists(bytes16 object) public view returns (bool) {
141     if (knownObjects[object].objectID > 0) {
142       return true;
143     }
144     return false;
145   }
146
147   /// @notice Zuordnen eines objektspezifischen Smart Contracts zu
          einem Objekt
148   /// @param objectSpecificContract Adresse des objektspezifischen
          Smart Contracts
149   /// @param object Objekt
150   function assignObjectSpecificContract(address objectSpecificContract
          , bytes16 object) public {
151     // Absender ueberpruefen
152     uint16 peerID = knownObjects[object].peerID;
153     if (peers[peerID].peer != msg.sender) {
154       return;
155     }
156     knownObjects[object].objectSpecificContract =
            objectSpecificContract;
157     emit ObjectSpecificContractAssigned(knownObjects[object].objectID,
            object);
158   }
159
160   // Event bei AEnderung des objektspezifischen Smart Contracts
161   event ObjectSpecificContractAssigned(uint16 indexed objectID,
          bytes16 object);
162
163   /// @notice Setzen des Peer-Namens
164   /// @param name Name des Peers (max. 32 byte)
165   function setPeerName(bytes32 name) public {
166     if (objectMap[msg.sender].length > 0) {
167       bytes16 existingObject = objectMap[msg.sender][0];
168       uint16 peerID = knownObjects[existingObject].peerID;
169       // Festlegung moeglich, sofern der Aufrufer die Adresse des
            Peers besitzt
```

```
170       if (peers[peerID].peer != msg.sender) {
171         return;
172       }
173       peers[peerID].name = name;
174     }
175   }
176
177   /// @notice Setzen eines Objekt-Namens
178   /// @param name Name des Objekts (max. 32 byte)
179   function setObjectName(bytes16 object, bytes32 name) public {
180     uint16 peerID = knownObjects[object].peerID;
181     if (peers[peerID].peer == msg.sender) {
182       knownObjects[object].name = name;
183       emit ObjectNameChanged(knownObjects[object].objectID, object,
              name);
184     }
185   }
186
187   // Event bei AEnderung des Objekt-Namens
188   event ObjectNameChanged(uint16 indexed objectID, bytes16 object,
        bytes32 name);
189
190   /// @notice Setzen eines Beziehungsnamens
191   /// @param name Name der Beziehung (max. 32 byte)
192   function setRelationName(uint16 relationID, bytes32 name) public {
193     bytes16 sourceObj = knownRelations[relationID].sourceObject;
194     uint16 peerID = knownObjects[sourceObj].peerID;
195     if (peers[peerID].peer == msg.sender) {
196       bytes16 uuid = knownRelations[relationID].uuid;
197       knownRelations[relationID].name = name;
198       emit RelationNameChanged(relationID, uuid, name);
199     }
200   }
201
202   // Event bei AEnderung des Beziehugs-Namens
203   event RelationNameChanged(uint16 indexed relationID, bytes16 uuid,
        bytes32 newName);
204
205   /// @notice Zuordnung eines Versionsgraphen zu einem Objekt
206   /// @param versiongraphAddressURI URI des Versionsgraphen (max. 32
        byte)
207   function setPeerVersionGraph(bytes32 versiongraphAddressURI) public
        {
208     if (objectMap[msg.sender].length > 0) {
209       bytes16 existingObject = objectMap[msg.sender][0];
```

```
210       uint16 peerID = knownObjects[existingObject].peerID;
211       // Festlegung moeglich, sofern der Aufrufer die Adresse des
              Peers besitzt
212       if (peers[peerID].peer != msg.sender) {
213         return;
214       }
215       peers[peerID].versiongraphAddressURI = versiongraphAddressURI;
216     }
217   }
218
219   /// @notice Ermitteln der Objekte eines Peers
220   /// @param peer Adresse des Peers
221   /// @return Array der ObjektIDs des Peers
222   function objMap(address peer) public view returns (bytes16[] memory
        objectIDs) {
223     return objectMap[peer];
224   }
225
226   /// @notice Hinzufuegen einer Beziehung
227   /// @param sourceObject Quellobjekt der Beziehung
228   /// @param targetObject Zielobjekt der Beziehung
229   /// @param uuid UUID der Beziehung
230   /// @param name Name der Beziehung (max. 32 byte)
231   /// @param relType Typ der Beziehung (siehe Datentyp RelationTyp)
232   function addRelation(bytes16 sourceObject, bytes16 targetObject,
        bytes16 uuid, bytes32 name, RelationType relType) public {
233     uint16 peerID = knownObjects[sourceObject].peerID;
234     if (peers[peerID].peer != msg.sender) {
235       return;
236     }
237     // Existenz der Objekte ueberpruefen
238     if (knownObjects[sourceObject].objectID == 0 ||
239     knownObjects[targetObject].objectID == 0 ) {
240       return;
241     }
242     // Beziehung registrieren
243     uint16 relationID = ++nRelations;
244     knownRelations[relationID].sourceObject = sourceObject;
245     knownRelations[relationID].targetObject = targetObject;
246     knownRelations[relationID].uuid = uuid;
247     knownRelations[relationID].name = name;
248     knownRelations[relationID].relType = relType;
249   }
250
```

```
251   /// @notice Hinzufuegen eines Objekts zu einer Collaboration, sofern
          das Objekt Teil einer Z-R-Beziehung ist
252   /// @param object Objekt, das zur Collaboration hinzugefuegt werden
          soll
253   /// @param sourceObject Quellobjekt einer bestehenden Z-R-Beziehung
254   /// @param targetObject Zielobjekt einer bestehenden Z-R-Beziehung
255   /// @param zRelationID ID der Z-Beziehung
256   /// @param rRelationID ID der R-Beziehung
257   function addCollaboration(bytes16 object, bytes16 sourceObject,
          bytes16 targetObject, uint16 zRelationID, uint16 rRelationID)
          public {
258     uint16 peerID = knownObjects[object].peerID;
259     if (peers[peerID].peer != msg.sender) {
260       return;
261     }
262     // Objekt bereits Teil einer Kooperation
263     uint16 collaborationID = knownObjects[object].collaborationID;
264     if (collaborationID > 0) {
265       return;
266     }
267
268     // Bestimmung des in Beziehung stehenden Objekts
269     bytes16 relatedObject = 0;
270     if (object == sourceObject) {
271       relatedObject = targetObject;
272     } else if (object == targetObject) {
273       relatedObject = sourceObject;
274     }
275
276     // Bestehende Beziehungen ueberpruefen
277     if (knownRelations[zRelationID].sourceObject == sourceObject &&
278     knownRelations[zRelationID].targetObject == targetObject &&
279     knownRelations[rRelationID].sourceObject == targetObject &&
280     knownRelations[rRelationID].targetObject == sourceObject &&
281     knownRelations[zRelationID].relType == RelationType.ZRelation &&
282     knownRelations[rRelationID].relType == RelationType.RRelation) {
283
284       // ID der Kooperation ermitteln oder vergeben
285       if (knownObjects[relatedObject].collaborationID > 0) {
286         collaborationID = knownObjects[relatedObject].collaborationID;
287       } else {
288         // Kooperation erstellen
289         collaborationID = ++nCollaborations;
290         // Versionsgraph URI auf initiierendes Peer setzen
```

```
291         knownCollaborations[collaborationID].versiongraphAddressURI =
                peers[peerID].versiongraphAddressURI;
292       }
293
294       // Kooperation beitreten
295       knownObjects[object].collaborationID = collaborationID;
296       knownCollaborations[collaborationID].nObjects++;
297     }
298   }
299
300   /// @notice Einleiten eines globalen Commits
301   /// @param hashValue Modell-Hash-Wert des uebergebenen Objekts
302   /// @param object Objekt, von dem der Commit ausgeht
303   function commit(bytes32 hashValue, bytes16 object) public {
304     // Transaktionsabsender ueberpruefen
305     uint16 peerID = knownObjects[object].peerID;
306     if (peers[peerID].peer != msg.sender) {
307       return;
308     }
309     uint16 coID = knownObjects[object].collaborationID;
310     uint16 cpID = knownCollaborations[coID].commitProcedureID;
311     // Commit bei Vorliegen des Zustands Init beginnen
312     if (commitProcedures[cpID].state == CommitProcedureState.Init) {
313       cpID++;
314       commitProcedures[cpID].state = CommitProcedureState.Wait;
315       commitProcedures[cpID].modelHashValue = hashValue;
316       knownCollaborations[coID].commitProcedureID = cpID;
317       emit VoteRequest(coID, cpID, object);
318     }
319   }
320
321   // Event zur UEbermittlung der Vote-Request-Nachricht
322   event VoteRequest(uint16 indexed collaborationID, uint16
          commitProcedureID, bytes16 object);
323
324   /// @notice Abstimmung ueber die Durchfuehrung oder den Abbruch des
          laufenden Commits
325   /// @param object betriebliches Objekt
326   /// @param voteCommit TRUE: Vote Commit, FALSE: Vote Abort
327   function voteGlobalCommit(bytes16 object, bool voteCommit) public {
328     uint16 peerID = knownObjects[object].peerID;
329     if (peers[peerID].peer != msg.sender) {
330       return;
331     }
332     uint16 collaborationID = knownObjects[object].collaborationID;
```

```
333     uint16 commitProcedureID = knownCollaborations[collaborationID].
            commitProcedureID;
334     if (commitProcedures[commitProcedureID].state ==
            CommitProcedureState.Wait &&
335     knownObjects[object].commitProcedureID < commitProcedureID) {
336       commitProcedures[commitProcedureID].nVotes++;
337       knownObjects[object].commitProcedureID = commitProcedureID;
338       uint16 nCollabObjects = knownCollaborations[collaborationID].
              nObjects;
339       uint16 nVotes = commitProcedures[commitProcedureID].nVotes;
340       if (!voteCommit) {
341         abortGlobalCommit(commitProcedureID);
342       } else if (voteCommit && nCollabObjects == nVotes) {
343         executeGlobalCommit(commitProcedureID, collaborationID);
344       }
345     }
346   }
347
348   // Event zur Benachrichtigung ueber den Abbruch eines globalen
        Commits
349   event GlobalAbort(uint16 indexed commitProcedureID);
350
351   /// @notice Abbruch des laufenden Commits
352   /// @param commitProcedureID ID des Commits
353   function abortGlobalCommit(uint16 commitProcedureID) internal {
354     emit GlobalAbort(commitProcedureID);
355     commitProcedures[commitProcedureID].result = CommitProcedureResult
          .Abort;
356     commitProcedures[commitProcedureID].state = CommitProcedureState.
          Init;
357   }
358
359   // Event zur Benachrichtigung der Durchfuehrung eines globalen
        Commits
360   event GlobalCommit(uint16 indexed commitProcedureID, uint16
        collaborationID);
361
362   /// @notice Durchfuehrung und Abschluss eines laufenden Commits (
        interne Funktion)
363   /// @param commitProcedureID ID des Commits
364   /// @param collaborationID ID der Kooperation des Commits
365   function executeGlobalCommit(uint16 commitProcedureID, uint16
        collaborationID) internal {
366     commitProcedures[commitProcedureID].result = CommitProcedureResult
          .Commit;
```

```
367     commitProcedures[commitProcedureID].state = CommitProcedureState.
            Init;
368     knownCollaborations[collaborationID].modelHashValue =
369     commitProcedures[commitProcedureID].modelHashValue;
370     emit GlobalCommit(commitProcedureID, collaborationID);
371   }
372
373   /// @notice Ermitteln der commitProcedureID des letzten
        erfolgreichen Commits
374   /// @param object Objekt, das an dem Commit beteiligt war
375   /// @return ID des Commit-Verfahrens (commitProcedureID)
376   function getCommitProcedureID(bytes16 object) public view returns (
        uint16) {
377     uint16 id = knownObjects[object].commitProcedureID;
378     while (id >= 0 && commitProcedures[id].result !=
            CommitProcedureResult.Commit) {
379       id--;
380     }
381     return id;
382   }
383 }
```

QUELLCODE A.1: Globaler Smart Contract

## A.2 Objektspezifischer Smart Contract

```
1  pragma solidity ^0.5.3;
2
3  /// @title Objektspezifischer Smart Contract
4  contract ObjectSpecificContract {
5
6    // Objekt
7    bytes16 public object;
8
9    // Peer-Zuordnung
10   address public peer;
11
12   // Zuordnung: Objekt-Zerlegungsprodukt -> Versions-Hash-Werte
13   mapping(bytes16 => VersionHashValues) public versionHashValues;
14
15   // Datentyp VersionHashValues zur Speicherung von Hash-Werten je
         Version
16   struct VersionHashValues {
17     uint16 nVersions;                          // Anzahl Versionen
18     mapping(uint16 => bytes32) hashValues;     // Hash-Werte
19     mapping(uint16 => uint16) commitProcedureID; // Commit-IDs
20   }
21
22   /// @notice Einleiten eines lokalen Commits
23   /// @param hashValue Modell-Hash-Wert des uebergebenen Objekts
24   /// @param object Objekt, von dem der Commit ausgeht
25   function commit(bytes32 hashValue, bytes16 object) public {
26     // Transaktionsabsender ueberpruefen
27     if (msg.sender != peer) {
28       return;
29     }
30     // Sequenz-ID der Version (0 < versionSequenceID <= nVersions)
31     uint16 versionSequenceID = ++versionHashValues[object].nVersions;
32     // ID des letzten globalen Commits ermitteln
33     uint16 commitProcedureID = GlobalContract(globalContract).
           getCommitProcedureID(object);
34     // Zuweisung Versions-Hash-Wert
35     VersionHashValues storage vhv = versionHashValues[object];
36     vhv.hashValues[versionSequenceID] = hashValue;
37     vhv.commitProcedureID[versionSequenceID] = commitProcedureID;
38   }
39
```

```
40  // Zuordnung: T-Event-ID -> Transaktionsabsender (zulaessige Peer-
        Adresse)
41  mapping (bytes16 => address) public transactionSenders;
42
43  // Zuordnung: instanceID => Instanz
44  mapping (bytes16 => Instance) public knownInstances;
45
46  // Zuordnung: taskEventAssignmentID => Event-Token-Zuordnung
47  mapping (uint16 => TaskEventAssignment) public taskEventAssignments;
48
49  // Anzahl von Instanz-Ereignis-Zuordnungen
50  uint16 public nTaskEventAssignments;
51
52  // Datentyp einer Instanz
53  struct Instance {
54    bytes16 objectID;
55    bytes16 instanceID;
56  }
57
58  // Datentyp einer Event-Zuordnung
59  struct TaskEventAssignment {
60    bytes16 taskEventID;
61    bytes16 instanceID;
62    bytes16 objectID;
63    bytes32 versionID;
64  }
65
66  // Adresse des globalen Smart Contracts
67  address public globalContract;
68
69  // Konstruktor: initiale Peer-Zuordnung, Zuordnung der Adresse des
        Global Contract
70  constructor(address globalContractAddress, bytes16 objectUUID)
        public {
71    peer = msg.sender;
72    globalContract = globalContractAddress;
73    object = objectUUID;
74  }
75
76  /// @notice Zuordnung des repraesentierten Objekts
77  /// @param objectID Objekt-UUID
78  function setObject(bytes16 objectID) public {
79    if (msg.sender != peer) {
80      return;
81    }
```

```
82      object = objectID;
83    }
84
85    /// @notice Anzahl der verwalteten Versionen / Modellsystem-Hash-
          Werte
86    /// @param objectID Objekt-UUID
87    /// @return Anzahl Versionen
88    function getNVersionHashValues(bytes16 objectID) public view returns
          (uint16) {
89      return versionHashValues[objectID].nVersions;
90    }
91
92    /// @notice Version (Modellsystem-Hash-Wert) ermitteln
93    /// @param objectID Objekt-UUID
94    /// @param versionNr Versionsnummer (sequenzielle Vergabe, versionNr
          >= 1)
95    /// @return Modellsystem-Hash-Wert (32 bit)
96    function getVersionHashValue(bytes16 objectID, uint16 versionNr)
          public view returns (bytes32) {
97      return versionHashValues[objectID].hashValues[versionNr];
98    }
99
100   /// @notice ID eines Commit-Vorgangs ermitteln
101   /// @param objectID Objekt-UUID
102   /// @param versionNr Index der Version (sequenzielle Vergabe,
          versionNr >= 1)
103   /// @return CommitProcedureID (sequenzielle Vergabe,
          CommitProcedureID >= 1)
104   function getVersionCommitProcedureID(bytes16 objectID, uint16
          versionNr) public view returns (uint16) {
105     return versionHashValues[objectID].commitProcedureID[versionNr];
106   }
107
108   /// @notice Letzten bekannten Modellsystem-Hash-Wert des
          uebergebenen Objekts zurueckgeben
109   /// @param objectID Objekt-UUID
110   /// @param versionNr Index der Version (sequenzielle Vergabe,
          versionNr >= 1)
111   /// @return Modellsystem-Hash-Wert (32 bit)
112   function getLatestHashValue(bytes16 objectID) public view returns (
          bytes32) {
113     uint16 versionSequenceID = versionHashValues[objectID].nVersions;
114     VersionHashValues storage vhs = versionHashValues[objectID];
115     return vhs.hashValues[versionSequenceID];
116   }
```

```
117
118   /// @notice Hinterlegen des zulaessigen Absenders einer Transaktion
          zur Vorgangsausloesung
119   /// @param tTaskEventID UUID des Vorgangs
120   /// @param transactionSender Adresse des Peers
121   function addTransactionSender(bytes16 tTaskEventID, address
          transactionSender) public {
122     if (msg.sender != peer) {
123       return;
124     }
125     transactionSenders[tTaskEventID] = transactionSender;
126   }
127
128   /// @notice O-Ereignis-Ausloesung per UEberfuehrung des Tokens
          tokenID in oTaskEventID
129   /// @param oTaskEventID UUID des auszuloesenden
          Transaktionsereignisses
130   /// @param instanceID UUID der Instanz
131   /// @param objectID UUID des instanziierenden Objekts
132   /// @param versionID Hash-Wert des Prozessmodells
133   function triggerTaskExecutionEvent(bytes16 oTaskEventID, bytes16
          instanceID, bytes16 objectID, bytes32 versionID) public {
134     if (msg.sender != peer) {
135       return;
136     }
137     if (knownInstances[instanceID].objectID == 0) {
138       knownInstances[instanceID].objectID = objectID;
139       knownInstances[instanceID].instanceID = instanceID;
140     }
141     uint16 taskEventAssignmentID = ++nTaskEventAssignments;
142     taskEventAssignments[taskEventAssignmentID].taskEventID =
            oTaskEventID;
143     taskEventAssignments[taskEventAssignmentID].versionID = versionID;
144     taskEventAssignments[taskEventAssignmentID].instanceID =
            instanceID;
145     taskEventAssignments[taskEventAssignmentID].objectID = objectID;
146     // Nachricht zur Ausloesung des Ereignisses per Event
147     emit ObjectTaskEvent(oTaskEventID, instanceID, objectID, versionID
          );
148   }
149
150   /// @notice T-Ereignis-Ausloesung: UEbergang tokenID zu tTaskEventID
151   /// @param tTaskEventID UUID des auszuloesenden
          Transaktionsereignisses
152   /// @param instanceID UUID der Instanz
```

```
153   /// @param objectID UUID des instanziierenden Objekts
154   /// @param versionID Hash-Wert der instanziierten Version des
          Prozesses
155   function triggerTransactionEvent(bytes16 tTaskEventID, bytes16
          instanceID, bytes16 objectID, bytes32 versionID) public {
156     // UEberpruefen des Transaktionsabsenders
157     if (msg.sender != peer || msg.sender != transactionSenders[
            tTaskEventID]) {
158       return;
159     }
160     if (knownInstances[instanceID].objectID == 0) {
161       knownInstances[instanceID].objectID = objectID;
162       knownInstances[instanceID].instanceID = instanceID;
163     }
164     uint16 taskEventAssignmentID = ++nTaskEventAssignments;
165     taskEventAssignments[taskEventAssignmentID].taskEventID =
            tTaskEventID;
166     taskEventAssignments[taskEventAssignmentID].versionID = versionID;
167     taskEventAssignments[taskEventAssignmentID].instanceID =
            instanceID;
168     taskEventAssignments[taskEventAssignmentID].objectID = objectID;
169     // Nachricht zur Ausloesung des Ereignisses per Event
170     emit TransactionTaskEvent(tTaskEventID, instanceID, objectID,
            versionID);
171   }
172
173   // Events zur Ausloesung von Ereignis-UEbergaengen
174   event TransactionTaskEvent(bytes16 indexed taskEventID, bytes16
          instanceID, bytes16 objectID, bytes32 versionID);
175   event ObjectTaskEvent(bytes16 indexed taskEventID, bytes16
          instanceID, bytes16 objectID, bytes32 versionID);
176
177 }
178
179 // Interface fuer Aufrufe des globalen Smart Contracts
180 contract GlobalContract {
181   function getCommitProcedureID(bytes16 object) public view returns (
          uint16);
182 }
```

QUELLCODE A.2: Objektspezifischer Smart Contract

# Anhang B

# Fallstudie

Dieser Anhang enthält weitere Modelle der Fallstudie (Kapitel 5), eine beispielhafte Darstellung der Objektbaum-Datenstruktur und die UML-Diagramme des Software-Tools.

## B.1 Modelle

Abbildung B.1 gibt die Zerlegung der Objekte und Transaktionen der Fallstudie an. Die Abbildungen B.2 und B.3 zeigen das Vorgangs-Ereignis-Schema des öffentlichen und globalen Prozesses. Die Vorgangstypschemata von „Produzent“, „Makler“ und „Logistikdienstleister“ sind in Ergänzung zu den in Kapitel 5 abgebildeten VTS-P in den Abbildungen B.4, B.5, B.6 und B.7 angegeben.

ABBILDUNG B.1: Objekt- und Transaktionszerlegung des kooperativen Prozesses

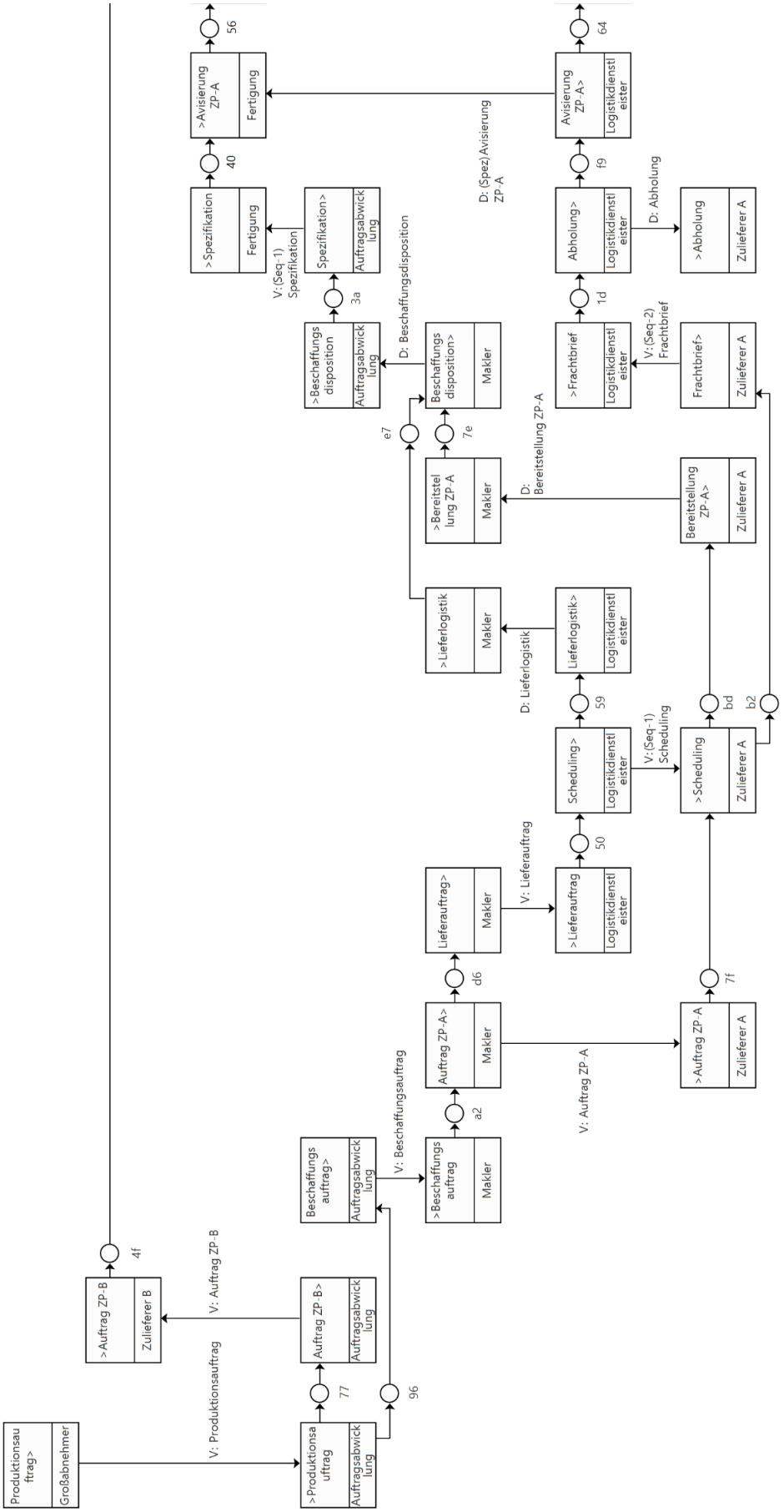



ABBILDUNG B.2: VES des kooperativen Prozesses

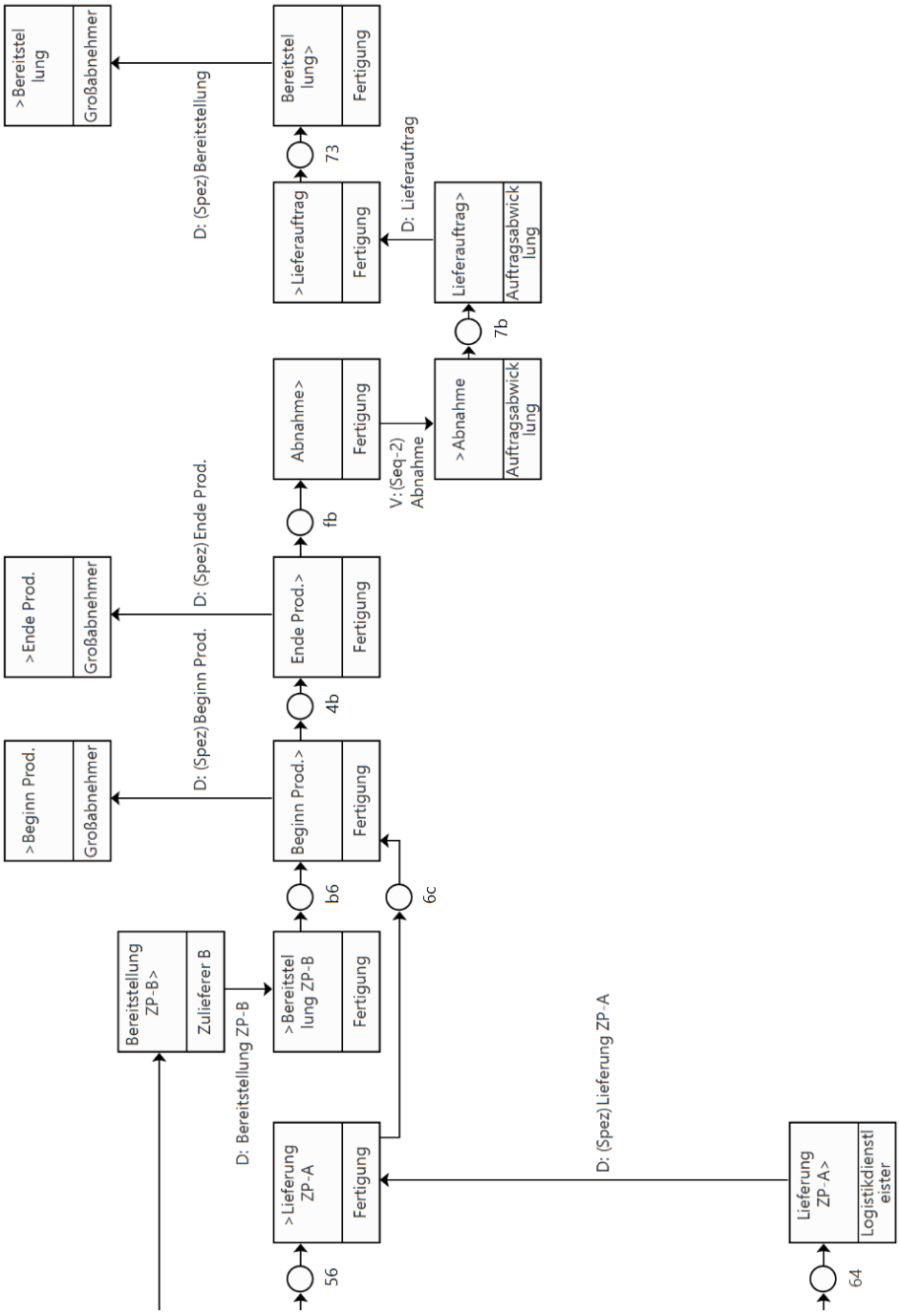



ABBILDUNG B.3: VES des kooperativen Prozesses (Fortsetzung)

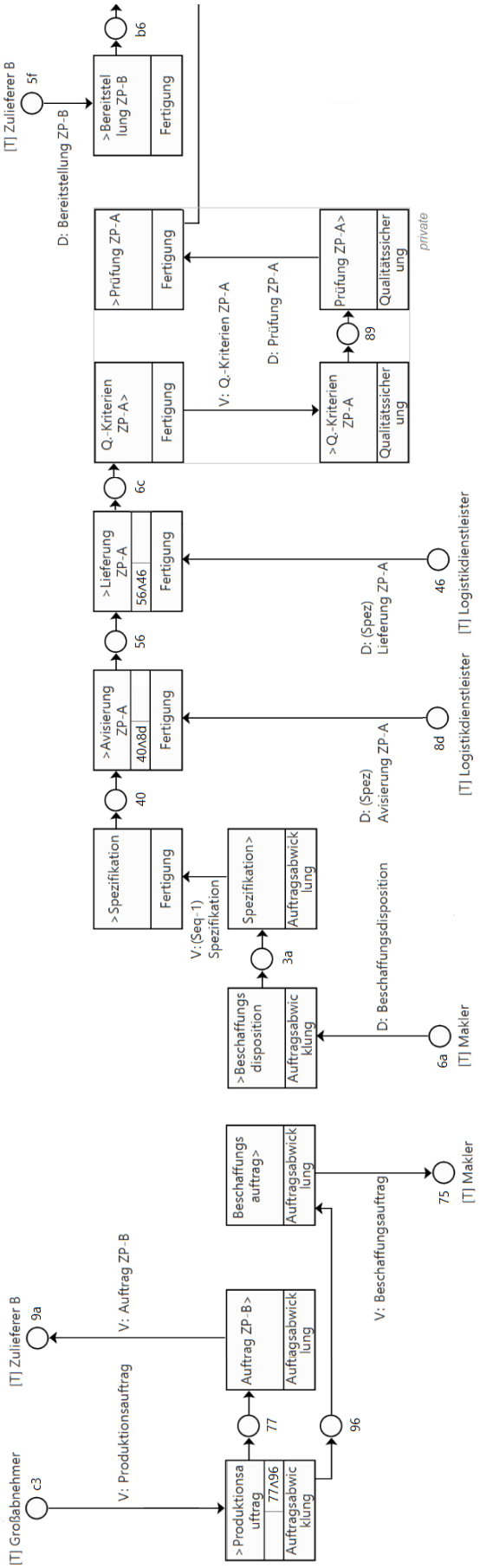



ABBILDUNG B.4: VTS-P von „Produzent“

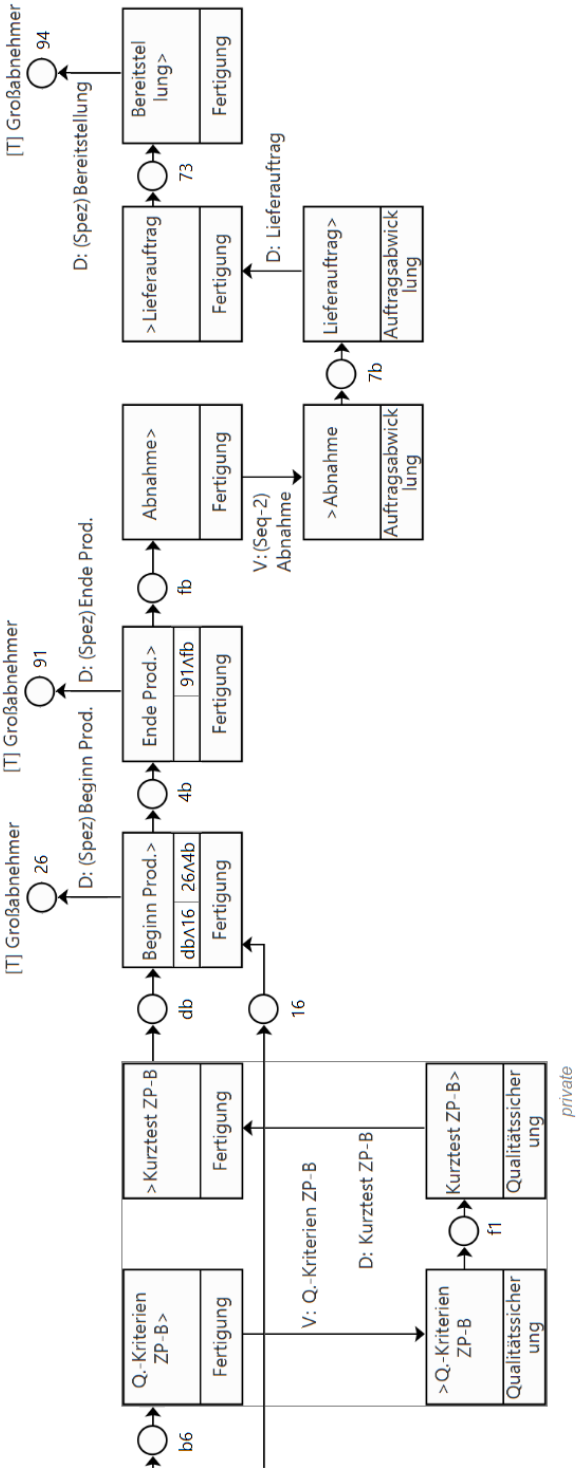



ABBILDUNG B.5: VTS-P von „Produzent“ (Fortsetzung)

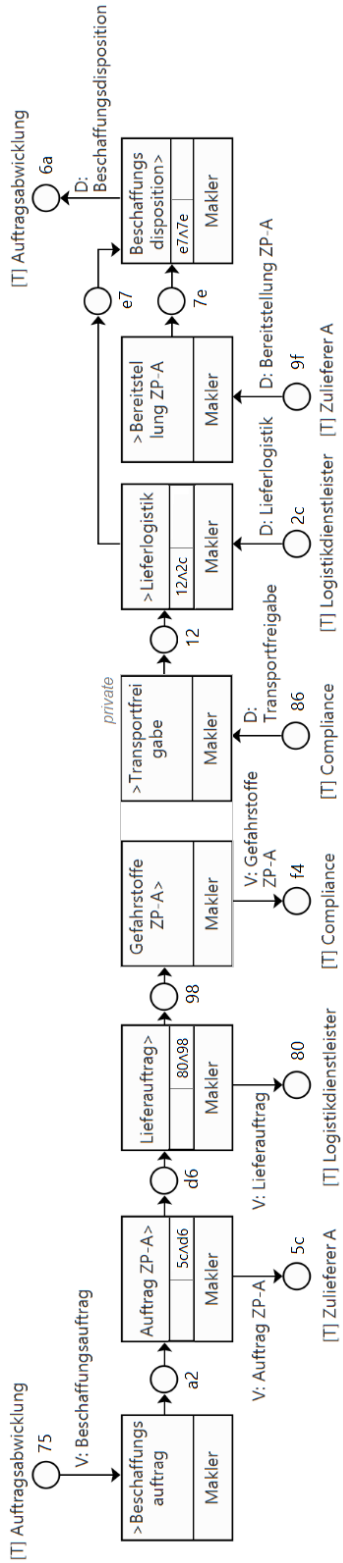



ABBILDUNG B.6: VTS-P von „Makler“

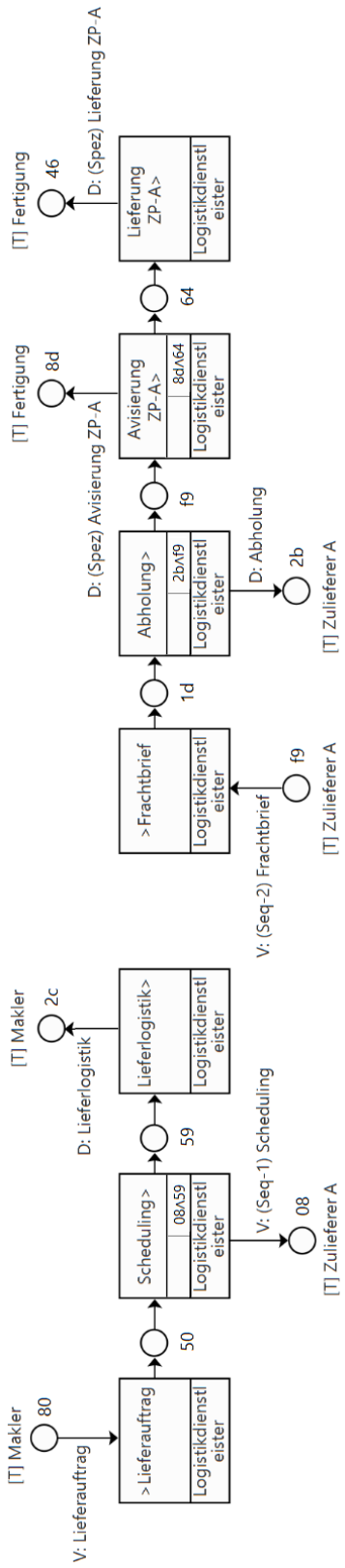



ABBILDUNG B.7: VTS-P von „Logistikdienstleister“

## B.2 Objektbaum-Datenstruktur

Abbildungen B.8 und B.9 zeigen Ausschnitte der Objektbaum-Datenstruktur in der im Versionsgraphen hinterlegten XML-Serialisierung des Software-Tools.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ModelingSystem xmlns="http://seda.wiai.uni-bamberg.de/processmodelingsystem" [...]
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ID="dd805ab3-4ad7-48fd-ae62-307b5581a284">
  <HashValue>C3612840B9A1BFADE0600C944288D69EA5C50A9A4D3A5B0C917F04EA4E3BF277</HashValue>
  <ObjectTree>
    <ObjectTreeNode>
      <HashValue>933C9678F3A80A56CC4DC2BFBADC6AB97AB4B1E7E1D9BD13770EEC05966BE971</HashValue>
      <Object Type="Discourse">
        <Attributes>
          <Attribute Name="UUID" Type="UUID" Value="dd805ab3-4ad7-48fd-ae62-307b5581a284"/> [...]
        </Attributes>
      </Object>
      <ZDecompositionProducts>
        <ZReference>
          <HashValue>090CA084A9B884C64B11F2E025FB9646A0C7A6E4B9F0E6F3270ABFFA3A81C977</HashValue>
          <ReferencedObject>
            <ObjectTreeNode>
              <HashValue>962375CF219033B2C0D9A7C586A6B5068058B0996DCDA1E5D6C124449F1B19C1</HashValue>
              <Object Type="Discourse">
                <Attributes>
                  <Attribute Name="UUID" Type="UUID" Value="5d0d686d-1844-3fef-98a7-3591830d00ae"/>
                  <Attribute Name="Name" Type="String" Value="Produzent"/>
                </Attributes>
                <RelationsOutgoing>
                  <ModelRelations>
                    <ModelRelation Type="Transaction">
                      <Attributes>
                        <Attribute Name="Typ" Type="String" Value="D"/>
                        <Attribute Name="UUID" Type="UUID" Value="8d554743-decc-4f93-8adb-f50e57e8016a"/>
                        <Attribute Name="Name" Type="String" Value="Produktion"/>[...]
                      </Attributes>
                    </ModelRelation>
                  </ModelRelations>
                </RelationsOutgoing>
                <RelationsIncoming>
                  <ModelRelations>
                    <ModelRelation Type="Transaction">
                      <Attributes>
                        <Attribute Name="Typ" Type="String" Value="D"/>
                        <Attribute Name="UUID" Type="UUID" Value="db79bd72-53e0-4fc5-af28-3dbc796470f0"/>
                        <Attribute Name="Name" Type="String" Value="Beschaffung ZP-A"/>[...]
                      </Attributes>
                    </ModelRelation>[...]
                  </ModelRelations>
                </RelationsIncoming>
              </Object>
              <ZDecompositionProducts/>
            </ObjectTreeNode>
          </ReferencedObject>
        </ZReference>
        [...]
```

ABBILDUNG B.8: XML-Serialisierung der Objektbaum-Datenstruktur

```
      [...]
      <ZReference>
        <HashValue>2084D20C25376C7DA8B8A428257C05CE082735AA5FF1B8776CE7F91DA4F1C5C7</HashValue>
        <ReferencedObject>
          <ObjectTreeNode>
            <HashValue>5A6F47D01BE0BB674B8A12AE7DC0AF4069BB74CFD28EEBBF2E8F168ED7119B87</HashValue>
            <Object Type="Discourse">
              <Attributes>
                <Attribute Name="UUID" Type="UUID" Value="571ed6d0-8155-351b-ab6c-8eb9ee34d1b6"/>
                <Attribute Name="Name" Type="String" Value="Makler"/>
              </Attributes>
              <ModelElements/>
              <RelationsOutgoing>
                <ModelRelations>
                  <ModelRelation Type="Transaction">
                    <Attributes>
                      <Attribute Name="Typ" Type="String" Value="D"/>
                      <Attribute Name="UUID" Type="UUID" Value="db79bd72-53e0-4fc5-af28-3dbc796470f0"/>
                      <Attribute Name="Name" Type="String" Value="Beschaffung ZP-A"/>[...]
                    </Attributes>
                  </ModelRelation>
                </ModelRelations>
              </RelationsOutgoing>[...]
            </Object>
            <ZDecompositionProducts/>
          </ObjectTreeNode>
        </ReferencedObject>
      </ZReference>[...]
    </ZDecompositionProducts>
  </ObjectTreeNode>
 </ObjectTree>
 <ModelViews>
  <ModelView Name="K-GP" Type="IAS" UUID="8e1a7e8f-0e6e-4c50-96cc-a9ac15bafb66">
    <ViewElements>
      <ViewElement CSS="" IsVisible="true" UUID="e799ab03-1615-3262-9734-9a6c74e6d5f4" X="200.0" Y="90.0" />[...]
      <ViewElement CSS="" IsVisible="true" UUID="7df392dd-93fd-3aad-b34e-47bb95c08fec" X="195.0" Y="480.0" />[...]
    </ViewElements>
    <ViewRelations>[...]
      <ViewRelation CSS="" IsVisible="true" LineEndType="1" TextX="605.0" TextY="330.0"
         UUID="db79bd72-53e0-4fc5-af28-3dbc796470f0" X="585.0" Y="301.0625">
        <RelationSourceTarget SourceUUID="571ed6d0-8155-351b-ab6c-8eb9ee34d1b6"
         TargetUUID="5d0d686d-1844-3fef-98a7-3591830d00ae"/>
      </ViewRelation>
    </ViewRelations>
  </ModelView>
 </ModelViews>
</ModelingSystem>
```

ABBILDUNG B.9: XML-Serialisierung der Objektbaum-Datenstruktur (Fortsetzung)

## B.3 Architektur des Software-Tools

Dieser Anhang enthält UML-Diagramme für die innerhalb des Software-Tools implementierten Teilsysteme des Ansatzes. Abbildung B.10 zeigt ein Paketdiagramm der Java-Implementierung.

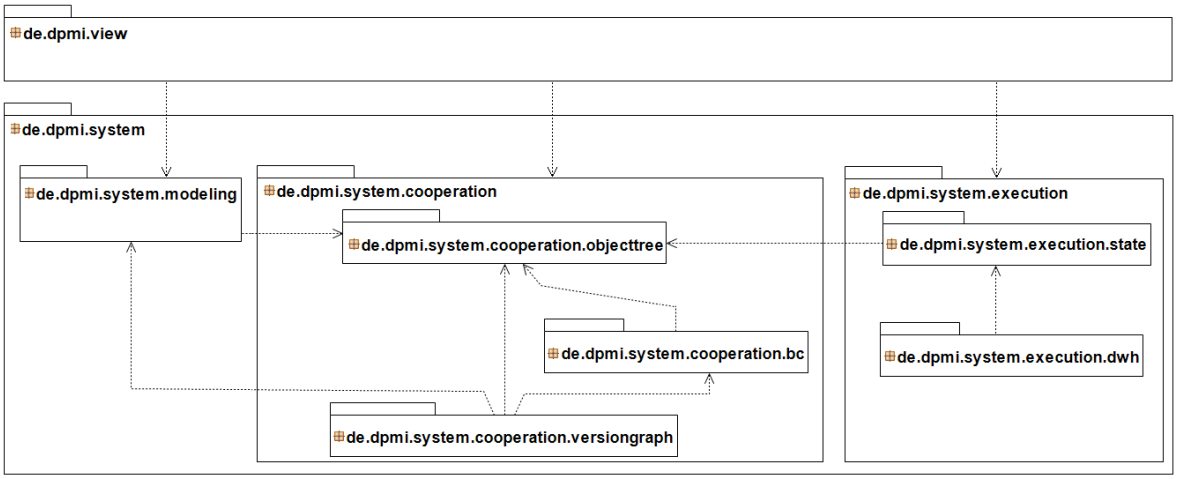



ABBILDUNG B.10: Paketdiagramm des Software-Tools

Das Software-Tool besteht aus den nachfolgenden Java-Paketen.

**de.dpmi.system**

Das Paket enthält die Implementierung der Teilsysteme des Ansatzes. Die Sub-Pakete betreffen das Modellsystem (Sub-Paket modeling), das Kooperationssystem (Sub-Paket cooperation) und das Ausführungssystem (Sub-Paket execution).

- **de.dpmi.system.modeling**: Die Implementierung des Modellsystems enthält Java-Klassen zur Repräsentation von Modellsystemen (Präfix ModelingSystem), Sichten (Präfix ModelView), Input und Output von XML-Dateien (Sub-Paket fileio) sowie die Metamodelle (Sub-Paket metamodel) und Schemata zur Sichtenbildung (Sub-Paket schema). Das Paketdiagramm in Abbildung B.11 stellt den Zusammenhang dar.
  - **de.dpmi.system.modeling.metamodel**: Meta-Metamodell gemäß Abschnitt 2.2.1.3.
    - **de.dpmi.system.modeling.metamodel.network**: Metamodell der Netzwerke zur System-Gestaltung.

    - **de.dpmi.system.modeling.metamodel.som**: Metamodell der Aufgabenebene des semantischen Objektmodells zur Gestaltung kooperativer Prozesse.
    - **de.dpmi.system.modeling.metamodel.workflow**: Metamodell für Workflow-Modelle der Aufgabenträgerebene.
- **de.dpmi.system.modeling.schema**: Implementierung der Schemata des Modellsystems (Abschnitt 4.3).
    - **de.dpmi.system.modeling.schema.network**: Implementierung von IAS-N und NWS-P.
    - **de.dpmi.system.modeling.schema.process**: Implementierung von IAS und VES sowie IAS-P und VTS-P.
    - **de.dpmi.system.modeling.schema.instance**: Implementierung von IZS und IZS-P.
- **de.dpmi.system.cooperation**: Die Implementierung des Kooperationssystems enthält Pakete zur Repräsentation der Objektbaum-Datenstruktur, zur Interaktion mit der Ethereum-Blockchain und zur Versionierung.
    - **de.dpmi.system.cooperation.objecttree**: Implementierung der Objektbaum-Datenstruktur sowie der Berechnung von Hash-Werten und Hash-Bäumen.
    - **de.dpmi.system.cooperation.bc**: Implementierung der Schnittstelle zur Ethereum-Blockchain und der Interaktion mit den Smart Contracts (Sub-Paket contract). Die Schnittstelle wird mit der Bibliothek web3j[1] realisiert.
    - **de.dpmi.system.cooperation.versiongraph**: Implementierung des Versionsgraphen unter Nutzung der Bibliothek JGit[2].
- **de.dpmi.system.execution**: Die Implementierung des Ausführungssystems enthält Klassen zur Verwaltung von Instanz-Zuständen (Sub-Paket state) und zur Erstellung von Instanz-Protokollen anhand eines Data-Warehouse-Systems (Sub-Paket dwh).

[1] https://web3j.readthedocs.io/en/latest/
[2] https://www.eclipse.org/jgit/

* **de.dpmi.system.execution.state**: Klassen zur Verwaltung von Instanz-Zuständen.
* **de.dpmi.system.cooperation.dwh**: Implementierung der Datenbank-Verbindung (DatabaseConnector), des ETL-Prozesses (Präfix ETL) und der Ausführung von Anfragen (Präfix OLAP).

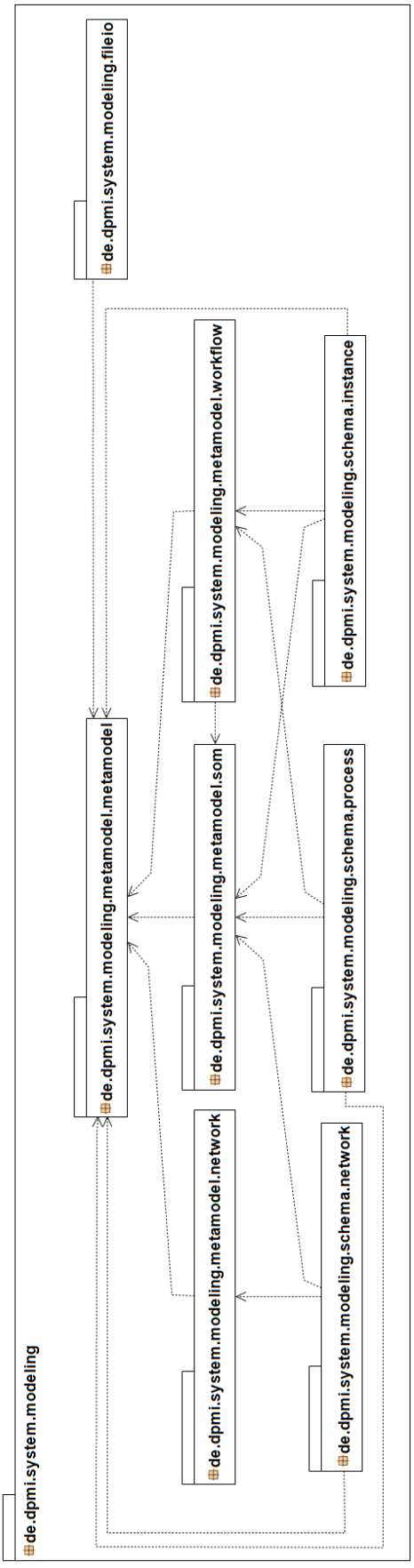



ABBILDUNG B.11: Paketdiagramm zur Implementierung des Modellsystems

Abbildung B.12 zeigt ein UML-Klassendiagramm der zentralen Klassen der Modellsystem-Implementierung. Abbildung B.13 stellt den Zusammenhang zur Implementierung der Objektbaum-Datenstruktur her.

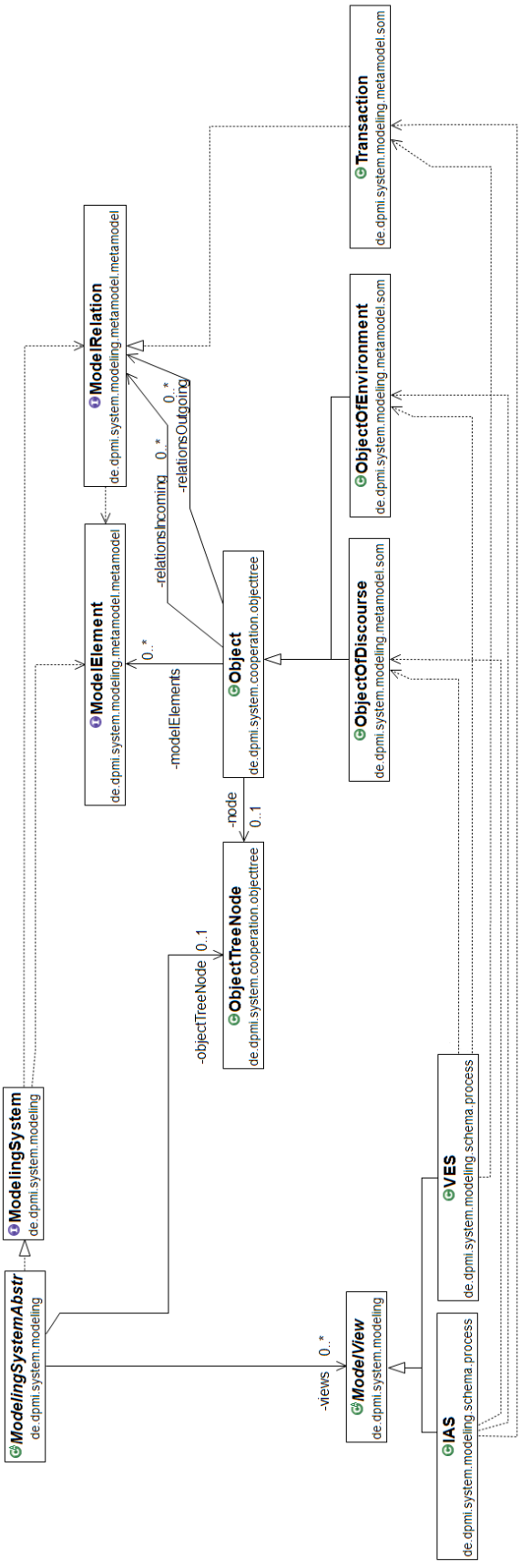



ABBILDUNG B.12: Klassendiagramm der zentralen Klassen zur Implementierung des Modellsystems

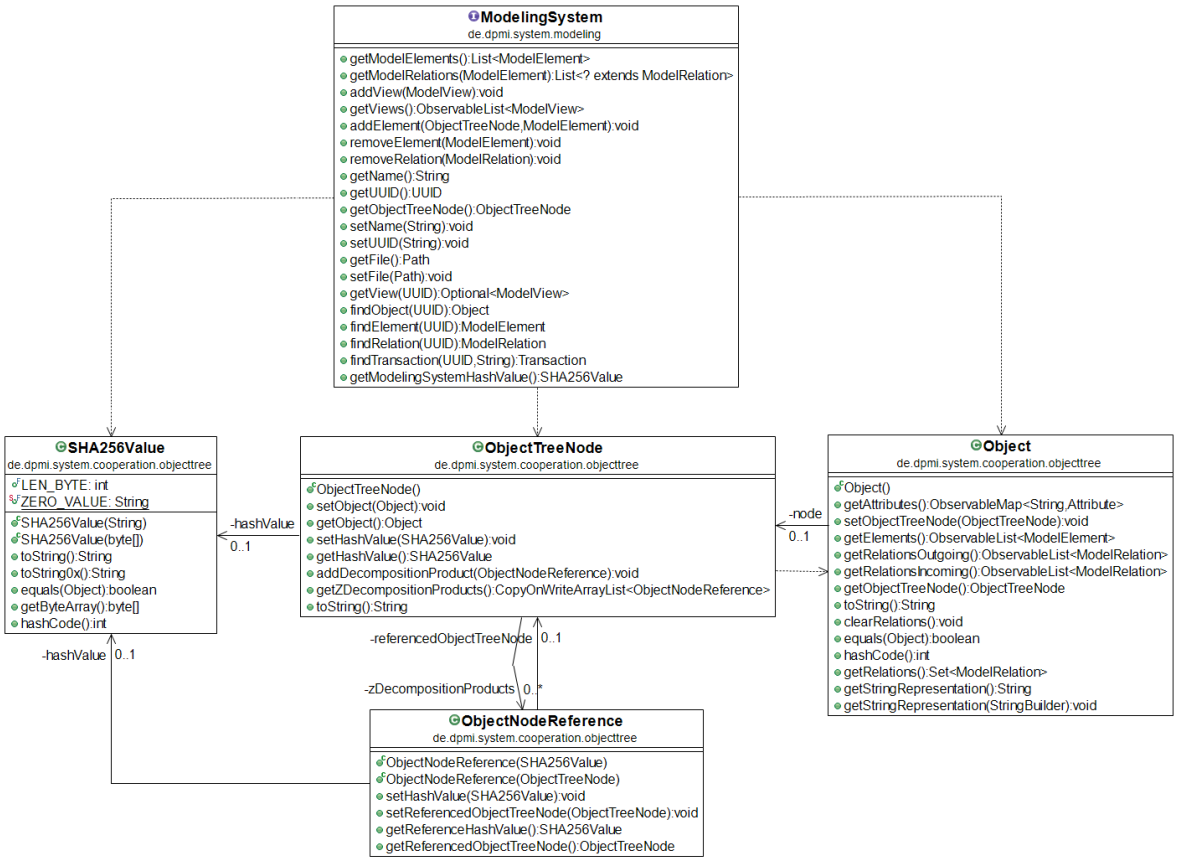



ABBILDUNG B.13: Klassendiagramm zur Implementierung der Objektbaum-Datenstruktur

### de.dpmi.view

Das Paket enthält die Implementierung der Kommunikationsschnittstelle anhand von JavaFX. Das Hauptfenster (Sub-Paket main) startet die Anzeige von Modellen in Tabs oder Fenstern (Sub-Paket modeling), stellt den Versionsgraphen dar (Sub-Paket versiongraph) und zeigt Blockchain-Transaktionen an (Sub-Paket bc). Das Sub-Paket modeling untergliedert sich zur Anzeige der Schemata des Modellsystems (Sub-Paket modeling.schema) analog zu Paket de.dpmi.system.modeling.schema.

# Literatur


Abeyratne, Saveen A. und Radmehr P. Monfared (2016). „Blockchain Ready Manufacturing Supply Chain Using Distributed Ledger“. In: *International Journal of Research in Engineering and Technology* 05.09, S. 1–10. ISSN: 23217308, 23191163. DOI: 10.15623/ijret.2016.0509001. URL: https://ijret.org/volumes/2016v05/i09/IJRET20160509001.pdf, Abruf am 01. 03. 2019.

Aleem, Saiqa, Sanja Lazarova-Molnar und Nader Mohamed (2012). „Collaborative Business Process Modeling Approaches: A Review“. In: *The 21st IEEE International Conference on Collaboration Technologies and Infrastructures (WETICE 2012)*. DOI: 10.1109/WETICE.2012.50.

Allweyer, Thomas (2005). *Geschäftsprozessmanagement*. W3I.

Altmanninger, Kerstin, Martina Seidl und Manuel Wimmer (2009). „A Survey on Model Versioning Approaches“. In: *International Journal of Web Information Systems* 5.3, S. 271–304. ISSN: 1744-0084. DOI: 10.1108/17440080910983556.

Andresen, Gavin (2013). *March 2013 Chain Fork Post-Mortem. BIP 50*. URL: https://github.com/bitcoin/bips/blob/master/bip-0050.mediawiki, Abruf am 21. 11. 2018.

Androulaki, Elli, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukoli, Artem Barger, Sharon Weed Cocco, Jason Yellick, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris und Gennady Laventman (2018). „Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains“. In: *Proceedings of the Thirteenth EuroSys Conference on - EuroSys '18*. The Thirteenth EuroSys Conference. Porto, Portugal: ACM Press, S. 1–15. ISBN: 978-1-4503-5584-1. DOI: 10.1145/3190508.3190538. Abruf am 07. 12. 2018.

Andrychowicz, Marcin und Stefan Dziembowski (2015). „PoW-Based Distributed Cryptography with No Trusted Setup“. In: *Advances in CryptologyCRYPTO 2015*. Springer, S. 379–399. URL: http://eprint.iacr.org/2014/796.pdf.

Antonopoulos, Andreas M. (2017). *Mastering Bitcoin: Programming the Open Blockchain*. Second edition. Sebastopol, CA: O'Reilly. ISBN: 978-1-4919-5438-6.

Antonopoulos, Andreas M. (2018). *Mastering Ethereum: Building Smart Contracts and Dapps*. S.l.: O'Reilly Media. ISBN: 978-1-4919-7191-8.

Apache (2018). *Apache ZooKeeper - Home*. URL: http://zookeeper.apache.org/, Abruf am 03. 12. 2018.

Aragon (2018). *Aragon Software*. URL: https://aragon.org/, Abruf am 25. 11. 2018.

Armknecht, Frederik, Ghassan O. Karame, Avikarsha Mandal, Franck Youssef und Erik Zenner (2015). „Ripple: Overview and Outlook". In: *Trust and Trustworthy Computing - 8th International Conference, TRUST 2015, Heraklion, Greece, August 24-26, 2015, Proceedings*, S. 163–180. DOI: 10.1007/978-3-319-22846-4_10.

Ateniese, Giuseppe, Ilario Bonacina, Antonio Faonio und Nicola Galesi (2013). „Proofs of Space: When Space Is of the Essence". In: Security and Cryptography for Networks (SCN 2014). DOI: 10.1007/978-3-319-10879-7_31.

Augur (2018). *Augur, A Decentralized Oracle & Prediction Market Protocol*. URL: https://www.augur.net, Abruf am 25. 11. 2018.

Bach, Alexander (2009). „Modellbasierte Analyse von Führungsinformationssystemen: ein Ansatz zur Bewertung auf der Grundlage betrieblicher Planungs- und Lenkungsprozesse". Dissertation. Bamberg: Universität Bamberg.

Bach, L. M., B. Mihaljevic und M. Zagar (2018). „Comparative Analysis of Blockchain Consensus Algorithms". In: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Opatija: IEEE, S. 1545–1550. ISBN: 978-953-233-095-3. DOI: 10.23919/MIPRO.2018.8400278. URL: https://ieeexplore.ieee.org/document/8400278/, Abruf am 23. 11. 2018.

Back, Adam (2002). *Hashcash - A Denial of Service Counter-Measure*. URL: http://www.hashcash.org/papers/hashcash.pdf, Abruf am 23. 11. 2018.

Baldwin, Carliss Y. und Eric von Hippel (2010). „Modeling a Paradigm Shift: From Producer Innovation to User and Open Collaborative Innovation: MIT Sloan Research Paper No. 4764-09". In: *Harvard Business School Finance Working Paper* (No. 10-038). DOI: 10.2139/ssrn.1502864.

Ball, Marshall, Alon Rosen, Manuel Sabin und Prashant Nalini Vasudevan (2017). „Proofs of Useful Work". In: *IACR Cryptology ePrint Archive 2017/203*. URL: http://eprint.iacr.org/2017/203, Abruf am 12. 12. 2018.

Balzert, Helmut (2011). *Softwaretechnik - Entwurf, Implementierung, Installation und Betrieb*. 3. Aufl. 2012. Heidelberg: Spektrum Akademischer Verlag. ISBN: 978-3-8274-1706-0.

Balzert, Helmut, Heide Balzert, Rainer Koschke, Uwe Lämmel, Peter Liggesmeyer und Jochen Quante (2009). *Softwaretechnik - Basiskonzepte und Requirements Engineering*. 3. Aufl. 2009. Heidelberg: Spektrum Akademischer Verlag. ISBN: 978-3-8274-1705-3.

Bano, Shehar, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn und George Danezis (2017). „Consensus in the Age of Blockchains". In: arXiv: 1711.03936 [cs]. URL: http://arxiv.org/abs/1711.03936, Abruf am 27. 11. 2018.

Barabási, Albert-László und Réka Albert (1999). „Emergence of Scaling in Random Networks". In: *Science* 286.5439, S. 509–512. ISSN: 00368075, 10959203. DOI: 10.1126/science.286.5439.509.

Barabási, Albert-László, Réka Albert und Hawoong Jeong (2000). „Scale-Free Characteristics of Random Networks: The Topology of the World-Wide Web". In: *Physica A: Statistical Mechanics and its Applications* 281.1, S. 69–77. ISSN: 0378-4371. DOI: 10.1016/S0378-4371(00)00018-2.

Baran, P. (1964). „On Distributed Communications Networks". In: *IEEE Transactions on Communications Systems* 12.1, S. 1–9. ISSN: 0096-1965. DOI: 10.1109/TCOM.1964.1088883.

Barber, Simon, Xavier Boyen, Elaine Shi und Ersin Uzun (2012). „Bitter to Better - How to Make Bitcoin a Better Currency". In: *Financial Cryptography and Data Security*. Springer, S. 399–414. URL: http://eprints.qut.edu.au/69169/1/Boyen_accepted_draft.pdf.

Barbieri, Maurice (2017). „Blockchain  Can This New Technology Really Revolutionize the Land Registry System?" In: *2017 World Bank Conference on Land and Poverty*. Washington DC, USA.

Barjis, Joseph (2009). „Collaborative, Participative and Interactive Enterprise Modeling". In: *Enterprise Information Systems*. Hrsg. von Joaquim Filipe und José Cordeiro. Lecture Notes in Business Information Processing, vol. 24. Springer Berlin Heidelberg, S. 651–662. ISBN: 978-3-642-01347-8.

Bartmann, Dieter, Freimut Bodendorf, Otto K. Ferstl und Elmar J. Sinz (2011). „Merkmale, Systemarchitekturen und Management hochflexibler Geschäftsprozesse". In: *Dienstorientierte IT-Systeme für hochflexible Geschäftsprozesse*. Hrsg. von Elmar J. Sinz, Dieter Bartmann, Freimut Bodendorf und Otto K. Ferstl. Bamberg: University of Bamberg Press, S. 1–13. ISBN: 978-3-86309-009-8. URL: http://nbn-resolving.de/urn:nbn:de:bvb:473-opus-3170, Abruf am 05. 03. 2019.

Bartoletti, Massimo und Livio Pompianu (2017). „An Empirical Analysis of Smart Contracts: Platforms, Applications, and Design Patterns“. In: *Financial Cryptography and Data Security*. Hrsg. von Michael Brenner, Kurt Rohloff, Joseph Bonneau, Andrew Miller, Peter Y.A. Ryan, Vanessa Teague, Andrea Bracciali, Massimiliano Sala, Federico Pintore und Markus Jakobsson. Bd. 10323. Cham: Springer International Publishing, S. 494–509. ISBN: 978-3-319-70277-3. DOI: 10.1007/978-3-319-70278-0_31. Abruf am 04. 12. 2018.

Baruffaldi, Giulia und Henrik Sternberg (2018). „Chains in Chains - Logic and Challenges of Blockchains in Supply Chains“. In: Hawaii International Conference on System Sciences. DOI: 10.24251/HICSS.2018.494. URL: http://hdl.handle.net/10125/50382, Abruf am 03. 12. 2018.

Bashir, Imran (2017). *Mastering Blockchain: Distributed Ledgers, Decentralization and Smart Contracts Explained*. ISBN: 978-1-78712-929-0. URL: http://public.eblib.com/choice/publicfullrecord.aspx?p=4826445, Abruf am 06. 12. 2018.

Bauer, Bernhard, Stephan Roser und Jörg P. Müller (2005). „Adaptive Design of Cross-Organizational Business Processes Using a Model-Driven Architecture“. In: *Wirtschaftsinformatik*. Hrsg. von Otto K. Ferstl, Elmar J. Sinz, Sven Eckert und Tilman Isselhorst. Heidelberg: Physica-Verlag HD, S. 103–121. ISBN: 978-3-7908-1574-0. DOI: 10.1007/3-7908-1624-8.

Bayer, Dave, Stuart Haber und W. Scott Stornetta (1993). „Improving the Efficiency and Reliability of Digital Time-Stamping“. In: *Sequences II*. Hrsg. von Renato Capocelli, Alfredo De Santis und Ugo Vaccaro. Springer New York, S. 329–334. ISBN: 978-1-4613-9323-8.

Bayer, Franz und Harald Kühn, Hrsg. (2013). *Prozessmanagement Für Experten: Impulse Für Aktuelle Und Wiederkehrende Themen*. Gabler Verlag. ISBN: 978-3-642-36994-0. URL: https://www.springer.com/de/book/9783642369940, Abruf am 07. 03. 2019.

Becker, Jörg, Martin Kugeler und Michael Rosemann, Hrsg. (2012). *Prozessmanagement: ein Leitfaden zur prozessorientierten Organisationsgestaltung*. Siebte, korrigierte und erweiterte Auflage. Berlin Heidelberg: Springer Gabler. ISBN: 978-3-642-33843-4.

Benkler, Yochai (2002). „Coase's Penguin, or, Linux and "The Nature of the Firm"“. In: *The Yale Law Journal* 112.3, S. 369–446. ISSN: 0044-0094. DOI: 10.2307/1562247. URL: https://www.jstor.org/stable/1562247, Abruf am 06. 03. 2019.

Benkler, Yochai und Helen Nissenbaum (2006). „Commons-Based Peer Production and Virtue*“. In: *Journal of Political Philosophy* 14.4, S. 394–419. ISSN: 1467-9760. DOI: 10.1111/j.1467-9760.2006.00235.x. Abruf am 06. 03. 2019.

Benshoof, Brendan, Andrew Rosen, Anu G. Bourgeois und Robert W. Harrison (2016). „Distributed Decentralized Domain Name Service“. In: *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. Chicago, IL, USA: IEEE, S. 1279–1287. ISBN: 978-1-5090-3682-0. DOI: 10.1109/IPDPSW.2016.109. URL: http://ieeexplore.ieee.org/document/7530014/, Abruf am 25. 11. 2018.

Berentsen, Aleksander und Fabian Schar (2018). „A Short Introduction to the World of Cryptocurrencies“. In: *Review* 100.1, S. 1–19. ISSN: 00149187. DOI: 10.20955/r.2018.1-16. URL: https://research.stlouisfed.org/publications/review/2018/01/10/a-short-introduction-to-the-world-of-cryptocurrencies, Abruf am 05. 12. 2018.

Bernus, Peter und Martin Nemetz (1994). „A Framework to Define a Generic Enterprise Reference Architecture and Methodology“. In: Proceedings of the International Conference on Automation, Robotics and Computer Vision (ICARCV'94). Singapore.

Beutelspacher, Albrecht, Heike B. Neumann und Thomas Schwarzpaul (2010). „Der diskrete Logarithmus, Diffie-Hellman-Schlüsselvereinbarung, ElGamal-Systeme“. In: *Kryptografie in Theorie und Praxis*. Wiesbaden: Vieweg+Teubner, S. 132–144. ISBN: 978-3-8348-0977-3. DOI: 10.1007/978-3-8348-9631-5_11. URL: http://www.springerlink.com/index/10.1007/978-3-8348-9631-5_11, Abruf am 23. 11. 2018.

Birman, Kenneth (2007). „The Promise, and Limitations, of Gossip Protocols“. In: *ACM SIGOPS Operating Systems Review* 41.5, S. 8. ISSN: 01635980. DOI: 10.1145/1317379.1317382. Abruf am 09. 12. 2018.

— (2012). *Guide to Reliable Distributed Systems*. Berlin Heidelberg: Springer.

Bitansky, Nir, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan und Brent Waters (2016). „Time-Lock Puzzles from Randomized Encodings“. In: *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science - ITCS '16*. The 2016 ACM Conference. Cambridge, Massachusetts, USA: ACM Press, S. 345–356. ISBN: 978-1-4503-4057-1. DOI: 10.1145/2840728.2840745. Abruf am 08. 12. 2018.

Bitcoin (2018). *Bitcoin Entwicklerreferenzen - Bitcoin*. URL: https://bitcoin.org/de/entwickler-referenzen#block-headers, Abruf am 08. 12. 2018.

Bitcoin Core (2017). *Optimized C Library for EC Operations on Curve Secp256k1, Quellcode*. URL: https://github.com/bitcoin/bitcoin/tree/master/src/secp256k1, Abruf am 09. 03. 2019.

Bitcoin Core (2018). *Bitcoin-Pow.Cpp Quellcode*. URL: https://github.com/bitcoin/bitcoin/blob/master/src/pow.cpp, Abruf am 08. 12. 2018.

— (2019). *Bitcoin Quellcode-Repository*. URL: https://github.com/bitcoin/bitcoin, Abruf am 17. 01. 2019.

Bitmain (2018). *AntMiner S9i Spezifikation*. URL: https://shop.bitmain.com/promote/antminer_s9i_asic_bitcoin_miner/, Abruf am 21. 11. 2018.

Blaze, M., J. Feigenbaum und J. Lacy (1996). „Decentralized Trust Management". In: *Proceedings 1996 IEEE Symposium on Security and Privacy*. Proceedings 1996 IEEE Symposium on Security and Privacy, S. 164–173. DOI: 10.1109/SECPRI.1996.502679.

Bleicher, Knut (1991). *Organisation: Strategien - Strukturen - Kulturen*. 2., vollständig neu bearbeitete und erweiterte Auflage. Wiesbaden: Gabler. ISBN: 978-3-322-82919-1.

Blockchain.com (2018). *Bitcoin Charts & Graphs - Blockchain*. URL: https://www.blockchain.com/charts, Abruf am 25. 11. 2018.

Boaro, Lorenzo, Emanuele Glorio, Francesco Pagliarecci und Luca Spalazzi (2011). „Business Process Design Framework for B2B Collaboration". In: *2011 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, S. 633–635. ISBN: 978-1-61284-638-5. DOI: 10.1109/cts.2011.5928749.

Bodendorf, Freimut (2006). *Daten- und Wissensmanagement*. Springer-Verlag. ISBN: 978-3-540-28682-0.

Bodendorf, Freimut, Otto K. Ferstl, Matthias Kurz und Elmar J. Sinz (2012). *Selbstorganisation und Fremdorganisation in hochflexiblen Geschäftsprozessen*. Arbeitsbericht forFLEX-2012-001. Universität Bamberg, Universität Erlangen-Nürnberg.

Bogner, Eva, Ulrich Löwen und Jörg Franke (2018). „Bedeutung der zukünftigen Produktion kundenindividueller Produkte in Losgröße 1". In: *Interdisziplinäre Perspektiven zur Zukunft der Wertschöpfung*. Hrsg. von Tobias Redlich, Manuel Moritz und Jens Wulfsberg. Wiesbaden: Springer Fachmedien, S. 63–75. ISBN: 978-3-658-20265-1. DOI: 10.1007/978-3-658-20265-1_6. Abruf am 08. 03. 2019.

Böhme, Rainer und Paulina Pesch (2017). „Technische Grundlagen und datenschutzrechtliche Fragen der Blockchain-Technologie". In: *Datenschutz und Datensicherheit - DuD* 41.8, S. 473–481. ISSN: 1614-0702, 1862-2607. DOI: 10.1007/s11623-017-0815-y. Abruf am 10. 12. 2018.

Booch, Grady (1993). *Object-Oriented Analysis and Design with Applications*. 2nd edition. Redwood City, Calif: Addison Wesley. ISBN: 978-0-8053-5340-2.

Bork, Dominik und Hans-Georg Fill (2014). „Formal Aspects of Enterprise Modeling Methods: A Comparison Framework Waikoloa, HI, USA, January 6-9, 2014". In:

*47th Hawaii International Conference on System Sciences, HICSS 2014, Waikoloa, HI, USA, January 6-9, 2014*. IEEE Computer Society, S. 3400–3409. ISBN: 978-1-4799-2504-9. DOI: 10.1109/HICSS.2014.422.

Bork, Dominik und Elmar J. Sinz (2013). „Bridging the Gap from a Multi-View Modelling Method to the Design of a Multi-View Modelling Tool“. In: *Enterprise Modelling and Information Systems Architectures* 8.2, S. 25–41. ISSN: 1866-3621. DOI: 10.1007/s40786-013-0003-y. Abruf am 27. 01. 2019.

Bos, Joppe W., J. Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Naehrig und Eric Wustrow (2013). „Elliptic Curve Cryptography in Practice“. In: *Financial Cryptography and Data Security 2014*. Bd. 2014. DOI: 10.1007/978-3-662-45472-5_11.

Brambilla, Marco, Jordi Cabot und Manuel Wimmer (2017). *Model-Driven Software Engineering in Practice*. 2. Aufl. San Rafael, Calif.: Morgan & Claypool. ISBN: 978-1-62705-708-0.

Brauer, Wilfried und Wolfgang Reisig (1996). „Carl Adam Petri and "Petri Nets"“. In: *Informatik Spektrum* 29, S. 369–374. DOI: 10.1142/9781848162914_0007.

Brewer, Eric A. (2000). „Towards Robust Distributed Systems - PODC Keynote“. In: PODC 2000 - ACM Symposium on Principles of Distributed Computing, S. 12.

Brosch, Petra, Uwe Egly, Sebastian Gabmeyer, Gerti Kappel, Martina Seidl, Hans Tompits, Magdalena Widl und Manuel Wimmer (2012). „Towards Semantics-Aware Merge Support in Optimistic Model Versioning“. In: *Models in Software Engineering*. Hrsg. von Jörg Kienzle. Bearb. von David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi und Gerhard Weikum. Bd. 7167. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 246–256. ISBN: 978-3-642-29644-4. DOI: 10.1007/978-3-642-29645-1_24. Abruf am 23. 11. 2018.

Brosch, Petra, Gerti Kappel, Philip Langer, Martina Seidl, Konrad Wieland und Manuel Wimmer (2012). „An Introduction to Model Versioning“. In: *Proceedings of the 12th International Conference on Formal Methods for the Design of Computer, Communication, and Software Systems: Formal Methods for Model-Driven Engineering*. SFM'12. Berlin, Heidelberg: Springer-Verlag, S. 336–398. ISBN: 978-3-642-30981-6. DOI: 10.1007/978-3-642-30982-3_10.

Burchert, Conrad und Roger Wattenhofer (2018). „piChain: When a Blockchain Meets Paxos“. In: DOI: 10.4230/lipics.opodis.2017.0. URL: http://drops.dagstuhl.de/opus/volltexte/2018/8623/, Abruf am 29. 11. 2018.

Buterin, Vitalik (2013). *Ethereum: The Ultimate Smart Contract and Decentralized Application Platform*. URL: http://web.archive.org/web/20131228111141/http://vbuterin.com/ethereum.html, Abruf am 11. 12. 2018.

— (2015). *On Public and Private Blockchains*. URL: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/, Abruf am 11. 12. 2018.

Buterin, Vitalik, Gavin Wood und Jeffrey Wilcke (2014). *Ethereum White Paper: A Next-Generation Smart Contract and Decentralized Application Platform*. URL: https://github.com/ethereum/wiki/wiki/White-Paper, Abruf am 12. 12. 2018.

Cachin, Christian, Rachid Guerraoui und Luís Rodrigues (2011). *Introduction to Reliable and Secure Distributed Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-642-15259-7. DOI: 10.1007/978-3-642-15260-3. Abruf am 28. 11. 2018.

Camunda (2018). *The Camunda BPM Manual*. URL: https://docs.camunda.org/manual/7.10/, Abruf am 08. 03. 2019.

Casati, Fabio und Angela Discenza (2000). „Supporting Workflow Cooperation within and across Organizations“. In: *Proceedings of the 2000 ACM Symposium on Applied Computing - SAC '00*. The 2000 ACM Symposium. Como, Italy: ACM Press, S. 196–202. ISBN: 978-1-58113-240-3. DOI: 10.1145/335603.335742. Abruf am 23. 11. 2018.

Castro, Miguel und Barbara Liskov (2002). „Practical Byzantine Fault Tolerance and Proactive Recovery“. In: *ACM Transactions on Computer Systems* 20.4, S. 398–461. ISSN: 07342071. DOI: 10.1145/571637.571640. Abruf am 30. 11. 2018.

Castro, Miguel, Barbara Liskov et al. (1999). „Practical Byzantine Fault Tolerance“. In: *Operating Systems Design and Implementation (OSDI)*, S. 173–186. URL: http://pmg.csail.mit.edu/papers/osdi99.pdf.

Certicom Research (2010). *SEC 2: Recommended Elliptic Curve Domain Parameters*. Mississauga, ON, Canada: Certicom Research. URL: http://www.secg.org/sec2-v2.pdf.

Chacon, Scott und Ben Straub (2014). *Pro Git*. Second edition. The Expert's Voice. New York: Apress. ISBN: 978-1-4842-0077-3. URL: http://proquest.tech.safaribooksonline.de/9781484200766.

Chen, Peter (1976). „The Entity-Relationship Model - toward a Unified View of Data“. In: *ACM Transactions on Database Systems* 1.1, S. 9–36. ISSN: 03625915. DOI: 10.1145/320434.320440.

Chohan, Usman W. (2018). *Cryptocurrencies as Asset-Backed Instruments: The Venezuelan Petro*. SSRN Scholarly Paper ID 3119606. Rochester, NY: Social Science Research Network. URL: https://papers.ssrn.com/abstract=3119606, Abruf am 25. 11. 2018.

Cicchetti, Antonio, Davide Di Ruscio und Alfonso Pierantonio (2007). „A Metamodel Independent Approach to Difference Representation“. In: *Journal of Object Technology* 6, S. 165–185. DOI: 10.5381/jot.2007.6.9.a9.

Clark, Jeremy, Joseph Bonneau, Edward W. Felten, Joshua A. Kroll und Andrew Miller (2014). „On Decentralizing Prediction Markets and Order Books“. In: *Workshop on the Economics of Information Security (WEIS)*. Pennsylvania State University, State College, Pennsylvania. URL: https://www.econinfosec.org/archive/weis2014/papers/Clark-WEIS2014.pdf, Abruf am 24. 11. 2018.

Clarke Jr., Edmund M., Orna Grumberg und Doron A. Peled (1999). *Model Checking*. Cambridge, MA, USA: MIT Press. ISBN: 978-0-262-03270-4.

Coad, Peter und Edward Yourdon (1991). *Object-Oriented Design*. Upper Saddle River, NJ, USA: Yourdon Press. ISBN: 978-0-13-630070-0.

CoinMarketCap (2018). *Cryptocurrency Market Capitalizations*. URL: https://coinmarketcap.com/, Abruf am 25. 11. 2018.

Collins-Sussman, B., B. W. Fitzpatrick und C. M. Pilato (2004). *Version Control with Subversion*. O'Reilly Media, Sebastopol.

Corda (2018). *Corda Open Source Blockchain Project*. URL: https://github.com/corda/corda, Abruf am 29. 11. 2018.

Core, Chain (2018). *Chain Protocol Whitepaper*. URL: https://chain.com/docs/1.2/protocol/papers/whitepaper, Abruf am 29. 11. 2018.

Coulouris, George, Jean Dollimore, Tim Kindberg und Gordon Blair (2012). *Distributed Systems - Concepts and Design*. 5th-edition. Boston, Massachusetts: Pearson Education.

Courtois, Nicolas und Lear Bahack (2014). „On Subversive Miner Strategies and Block Withholding Attack in Bitcoin Digital Currency“. In: *CoRR* abs/1402.1718. URL: https://www.semanticscholar.org/paper/On-Subversive-Miner-Strategies-and-Block-Attack-in-Courtois-Bahack/822693248834147245d6ff2309192122d1326396, Abruf am 29. 11. 2018.

Crosby, Michael (2016). „BlockChain Technology: Beyond Bitcoin“. In: *Applied Innovation Review* 2, S. 16.

CryptoKitties (2018). *CryptoKitties*. URL: https://www.cryptokitties.co, Abruf am 25. 11. 2018.

Czarnecki, Krzysztof und Simon Helsen (2003). „Classification of Model Transformation Approaches“. In: *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*.

Dadam, Peter und Manfred Reichert (1998). „The ADEPT WfMS Project at the University of Ulm“. In: *Proceedings of the 1st European Workshop on Workflow and Process Management (WPM98)*. Zurich, Switzerland.

Dadam, Peter, Manfred Reichert und Stefanie Rinderle-Ma (2011). „Prozessmanagementsysteme“. In: *Informatik-Spektrum* 34.4, S. 364–376. ISSN: 1432-122X. DOI: 10.1007/s00287-010-0456-0. Abruf am 06. 03. 2019.

Dadam, Peter, Manfred Reichert, Stefanie Rinderle-Ma, Kevin Göser, Ulrich Kreher und Martin Jurisch (2009). „Von ADEPT Zur AristaFlow BPM Suite - Eine Vision Wird Realität: Correctness by Construction Und Flexible, Robuste Ausführung von Unternehmensprozessen“. In: *Ulmer Informatik-Berichte, Universität Ulm*. Bd. 2. Ulm.

Dahl, Ole-Johan und Kristen Nygaard (1966). „SIMULA: An ALGOL-Based Simulation Language“. In: *Communications of the ACM* 9.9, S. 671–678. ISSN: 0001-0782. URL: http://doi.acm.org/10.1145/365813.365819, Abruf am 06. 03. 2019.

Daniel, Gwendal, Gerson Sunyé, Amine Benelallam, Massimo Tisi, Yoann Vernageau, Abel Gómez und Jordi Cabot (2017). „NeoEMF: A Multi-Database Model Persistence Framework for Very Large Models“. In: *Science of Computer Programming*. Special Issue on MODELS'16 149, S. 9–14. ISSN: 0167-6423. DOI: 10.1016/j.scico.2017.08.002. URL: http://www.sciencedirect.com/science/article/pii/S0167642317301600, Abruf am 08. 03. 2019.

Decker, Christian und Roger Wattenhofer (2013). „Information Propagation in the Bitcoin Network“. In: *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference*, S. 1–10.

Diffie, W. und M. Hellman (1976). „New Directions in Cryptography“. In: *IEEE Transactions on Information Theory* 22.6, S. 644–654. ISSN: 0018-9448. DOI: 10.1109/TIT.1976.1055638.

Dinh, T. T. A., R. Liu, M. Zhang, G. Chen, B. C. Ooi und J. Wang (2018). „Untangling Blockchain: A Data Processing View of Blockchain Systems“. In: *IEEE Transactions on Knowledge and Data Engineering* 30.7, S. 1366–1385. ISSN: 1041-4347. DOI: 10.1109/TKDE.2017.2781227.

Dollmann, Thorsten, Constantin Houy, Peter Fettke und Peter Loos (2011). „Collaborative Business Process Modeling with CoMoMod - A Toolkit for Model Integration in Distributed Cooperation Environments“. In: *2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*.

Paris, France: IEEE, S. 217–222. ISBN: 978-1-4577-0134-4. DOI: 10.1109/WETICE.2011.36. URL: http://ieeexplore.ieee.org/document/5990030/, Abruf am 23. 11. 2018.

Duan, Sisi, Michael K. Reiter und Haibin Zhang (2018). „BEAT: Asynchronous BFT Made Practical". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security - CCS '18*. The 2018 ACM SIGSAC Conference. Toronto, Canada: ACM Press, S. 2028–2041. ISBN: 978-1-4503-5693-0. DOI: 10.1145/3243734.3243812. Abruf am 29. 11. 2018.

Dumas, Marlon, Marcello La Rosa, Jan Mendling und Hajo Reijers (2018). *Fundamentals of Business Process Management*. 2. Aufl. Berlin Heidelberg: Springer-Verlag. ISBN: 978-3-662-56508-7. URL: https://www.springer.com/de/book/9783662565085, Abruf am 03. 03. 2019.

Dupont, Quinn (2017). „Experiments in Algorithmic Governance: A History and Ethnography of The DAO, a Failed Decentralized Autonomous Organization". In: *Bitcoin and Beyond: Cryptocurrencies, Blockchains and Global Governance*. Hrsg. von Malcolm Campbell-Verduyn. Oxford, UK: Routledge.

Dwork, Cynthia, Nancy Lynch und Larry Stockmeyer (1988). „Consensus in the Presence of Partial Synchrony". In: *Journal of the ACM* 35, S. 288–323. URL: http://www-usr.inf.ufsm.br/%20ceretta/papers/MITLCSTM270.pdf.

Dwork, Cynthia und Moni Naor (1992). „Pricing via Processing or Combatting Junk Mail". In: *Advances in Cryptology  CRYPTO 92*. Hrsg. von Ernest F. Brickell. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 139–147. ISBN: 978-3-540-48071-6.

Eckert, Claudia (2018). *IT-Sicherheit*. 10. Auflage. Berlin: De Gruyter.

El Ioini, Nabil und Claus Pahl (2018). „A Review of Distributed Ledger Technologies". In: *On the Move to Meaningful Internet Systems. OTM 2018 Conferences*. Hrsg. von Hervé Panetto, Christophe Debruyne, Henderik A. Proper, Claudio Agostino Ardagna, Dumitru Roman und Robert Meersman. Springer International Publishing, S. 277–288. ISBN: 978-3-030-02671-4.

Eskandari, Shayan, Jeremy Clark, Vignesh Sundaresan und Moe Adham (2017). „On the Feasibility of Decentralized Derivatives Markets". In: *Financial Cryptography and Data Security*. Hrsg. von Michael Brenner, Kurt Rohloff, Joseph Bonneau, Andrew Miller, Peter Y.A. Ryan, Vanessa Teague, Andrea Bracciali, Massimiliano Sala, Federico Pintore und Markus Jakobsson. Bd. 10323. Cham: Springer International Publishing, S. 553–567. ISBN: 978-3-319-70277-3. DOI: 10.1007/978-3-319-70278-0_35. Abruf am 24. 11. 2018.

ETH Statistics (2018). *ETH Statistics*. URL: https://bitinfocharts.com/, Abruf am 25. 11. 2018.

Etherchain.org (2018). *Etherchain.Org Statistics*. URL: https://www.etherchain.org/charts/transactionsPerDay, Abruf am 25. 11. 2018.

EtherDelta (2018). *EtherDelta*. URL: https://etherdelta.com/, Abruf am 25. 11. 2018.

Ethereum (2019a). *Ethereum 2.0 Specifications*. URL: https://github.com/ethereum/eth2.0-specs, Abruf am 18. 02. 2019.

— (2019b). *Ethereum Go Quellcode-Repository*. URL: https://github.com/ethereum/go-ethereum, Abruf am 17. 01. 2019.

Fdhila, Walid, Stefanie Rinderle-Ma, David Knuplesch und Manfred Reichert (2015). „Change and Compliance in Collaborative Processes“. In: *2015 IEEE International Conference on Services Computing*. New York City, NY, USA: IEEE, S. 162–169. ISBN: 978-1-4673-7281-7. DOI: 10.1109/SCC.2015.31. URL: http://ieeexplore.ieee.org/document/7207349/, Abruf am 23. 11. 2018.

Ferguson, Niels, Bruce Schneier und Tadayoshi Kohno (2010). *Cryptography Engineering: Design Principles and Practical Applications*. Indianapolis: Wiley Publishing.

Fernández Venero, Mirtha Lina und Flávio Soares Corrêa Da Silva (2017). „Model Checking Multi-Level and Recursive Nets“. In: *Software and Systems Modeling* 16.4, S. 1117–1144. ISSN: 1619-1366. DOI: 10.1007/s10270-015-0509-6. Abruf am 08. 03. 2019.

Ferstl, Otto K. (1979). *Konstruktion Und Analyse von Simulationsmodellen*. Beiträge zur Datenverarbeitung und Unternehmensforschung. Hain.

Ferstl, Otto K. und Elmar J. Sinz (1990). „Objektmodellierung Betrieblicher Informationssysteme Im Semantischen Objektmodell (SOM)“. In: Wirtschaftsinformatik 32.6, S. 566–581.

— (1993). „Geschäftsprozessmodellierung“. In: *Wirtschaftsinformatik* 35.6, S. 589–592.

— (1995). „Der Ansatz Des Semantischen Objektmodells (SOM) Zur Modellierung von Geschäftsprozessen“. In: *Wirtschaftsinformatik* 37.3, S. 209–220.

— (2013). *Grundlagen der Wirtschaftsinformatik*. 7., aktualisierte Aufl. München: Oldenbourg. ISBN: 978-3-486-71917-8.

Fidge, C. J. (1988). „Timestamps in Message-Passing Systems That Preserve the Partial Ordering“. In: *Proceedings of the 11th Australian Computer Science Conference* 10.1, S. 56–66. URL: http://sky.scitech.qut.edu.au/~fidgec/Publications/fidge88a.pdf.

Filecoin (2014). *Filecoin: A Cryptocurrency Operated File Storage Network*. URL: http://filecoin.io/filecoin.pdf, Abruf am 11. 01. 2019.

Filippi, Primavera De und Samer Hassan (2016). „Blockchain Technology as a Regulatory Technology: From Code Is Law to Law Is Code“. In: *First Monday* 21.12. ISSN: 13960466. DOI: 10.5210/fm.v21i12.7113. URL: https://firstmonday.org/ojs/index.php/fm/article/view/7113, Abruf am 06. 03. 2019.

Fill, Hans-Georg, Andreas Eberhart, Andrea Laslop, Ilona Reischl, Thomas Lang und Dimitris Karagiannis (2011). „An Approach to Support the Performance Management of Public Health Authorities Using an IT Based Modeling Method“. In: Proceedings of the 10th International Conference on Wirtschaftsinformatik WI 2.011. Zurich, Switzerland, S. 38–47. URL: https://eprints.cs.univie.ac.at/3054/, Abruf am 08. 03. 2019.

Fill, Hans-Georg und Felix Härer (2018). „Knowledge Blockchains: Applying Blockchain Technologies to Enterprise Modeling“. In: *Proceedings of the 51st Hawaii International Conference on System Sciences (HICSS-51)*. Waikoloa Village, Hawaii, USA, S. 4045–4054. ISBN: 978-0-9981331-1-9. DOI: 10.24251/HICSS.2018.509.

Fill, Hans-Georg und Dimitris Karagiannis (2013). „On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform“. In: *Enterprise Modelling and Information Systems Architectures (EMISAJ)* 8.1, S. 4–25. ISSN: 1866-3621. DOI: 10.18417/emisa.8.1.1. URL: https://emisa-journal.org/emisa/article/view/99, Abruf am 18. 02. 2019.

Fink, Andreas, Gabriele Schneidereit und Stefan Voß (2005). *Grundlagen Der Wirtschaftsinformatik*. 2. Aufl. Physica-Lehrbuch. Physica-Verlag Heidelberg. ISBN: 978-3-7908-0189-7. URL: https://www.springer.com/de/book/9783790801897, Abruf am 07. 03. 2019.

Finney, Hal (2004). *Reusable Proofs of Work (RPOW)*. URL: http://web.archive.org/web/20071222072154/http://rpow.net/, Abruf am 11. 01. 2019.

Fischer, Michael J., Nancy A. Lynch und Michael S. Paterson (1985). „Impossibility of Distributed Consensus with One Faulty Process“. In: *Journal of the ACM* 32, S. 374–382. URL: http://macs.citadel.edu/rudolphg/csci604/ImpossibilityofConsensus.pdf.

Forster, Simon, Jakob Pinggera und Barbara Weber (2012). *Collaborative Business Process Modeling*. Gesellschaft für Informatik e.V. ISBN: 978-3-88579-600-8. URL: http://dl.gi.de/handle/20.500.12116/17744, Abruf am 08. 03. 2019.

Fox, Armando und Eric A. Brewer (1999). „Harvest, Yield, and Scalable Tolerant Systems“. In: *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems*. Proceedings of the Seventh Workshop on Hot Topics in Operating Systems, S. 174–178. DOI: 10.1109/HOTOS.1999.798396.

Frank, Ulrich (1994). *Multiperspektivische Unternehmensmodellierung. Theoretischer Hintergrund Und Entwurf Einer Objektorientierten Entwicklungsumgebung*. München: Oldenbourg.

— (2011). *The MEMO Meta Modelling Language (MML) and Language Architecture. 2nd Edition*. 43. University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB). URL: https://ideas.repec.org/p/zbw/udeicb/43.html, Abruf am 18. 02. 2019.

— (2013). „Domain-Specific Modeling Languages: Requirements Analysis and Design Guidelines“. In: *Domain Engineering, Product Lines, Languages, and Conceptual Models*, S. 133–157. DOI: 10.1007/978-3-642-36654-3_6.

— (2016). *Konstruktionsorientierter Forschungsansatz Enzyklopaedie Der Wirtschaftsinformatik*. URL: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/uebergreifendes/Forschung-in-WI/Konstruktionsorientierter-Forschungsansatz, Abruf am 03. 03. 2019.

Frank, Ulrich, Stefan Strecker, Peter Fettke, Jan Brocke, Jörg Becker und Elmar J. Sinz (2014). „Das Forschungsfeld „Modellierung Betrieblicher Informationssysteme““. In: *Wirtschaftsinformatik* 56.1, S. 49–54. ISSN: 1861-8936. URL: http://dl.gi.de/handle/20.500.12116/11944, Abruf am 06. 03. 2019.

Frese, Erich, Matthias Graumann und Ludwig Theuvsen (2012). *Grundlagen der Organisation: entscheidungsorientiertes Konzept der Organisationsgestaltung*. 10., überarb. und erw. Aufl. Gabler-Lehrbuch. Wiesbaden: Gabler. ISBN: 978-3-8349-3029-3.

Fujimura, S., H. Watanabe, A. Nakadaira, T. Yamada, A. Akutsu und J. J. Kishigami (2015). „BRIGHT: A Concept for a Decentralized Rights Management System Based on Blockchain“. In: *2015 IEEE 5th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*. 2015 IEEE 5th International Conference on Consumer Electronics - Berlin (ICCE-Berlin), S. 345–346. DOI: 10.1109/ICCE-Berlin.2015.7391275.

Gadatsch, Andreas (2012). *Grundkurs Geschäftsprozess-Management: Methoden und Werkzeuge für die IT-Praxis ; eine Einführung für Studenten und Praktiker*. 7. Aufl. Studium. Wiesbaden: Springer Vieweg. ISBN: 978-3-8348-2427-1.

— (2015). *Geschäftsprozesse analysieren und optimieren: Praxistools zur Analyse, Optimierung und Controlling von Arbeitsabläufen*. essentials. Wiesbaden: Springer Vieweg. ISBN: 978-3-658-09109-5.

Garay, Juan und Aggelos Kiayias (2018). „SoK: A Consensus Taxonomy in the Blockchain Era“. In: *IACR Cryptology ePrint Archive, Report 2018/754* 2018, S. 754. URL: https://eprint.iacr.org/2018/754.

Garay, Juan, Aggelos Kiayias und Nikos Leonardos (2015). „The Bitcoin Backbone Protocol: Analysis and Applications“. In: *Advances in Cryptology - EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*. Hrsg. von Elisabeth Oswald und Marc Fischlin. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 281–310. DOI: 10.1007/978-3-662-46803-6.

Garay, Juan, Texas und Aggelos Kiayias (2017). „Blockchain and Consensus from Proofs of Work without Random Oracles“. In: *Cryptology ePrint Archive, Report 2017/775*. URL: https://eprint.iacr.org/2017/775, Abruf am 12. 12. 2018.

Gaevi, Dragan, Dragan Djuri und Vladan Devedi (2009). *Model Driven Engineering and Ontology Development*. 2. Aufl. Berlin Heidelberg: Springer-Verlag. ISBN: 978-3-642-00281-6. URL: https://www.springer.com/de/book/9783642002816, Abruf am 07. 03. 2019.

Gershenfeld, Neil (2007). *Fab: The Coming Revolution on Your Desktop-from Personal Computers to Personal Fabrication*. New Ed. New York, NY: Basic Books. ISBN: 978-0-465-02746-0.

Gerth, Christian, Jochen M. Küster, Markus Luckey und Gregor Engels (2010). „Precise Detection of Conflicting Change Operations Using Process Model Terms“. In: *Model Driven Engineering Languages and Systems*. Hrsg. von Nicolas Rouquette, Øystein Haugen und Dorina C. Petriu. Bd. 6395. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 93–107. ISBN: 978-3-642-16128-5. DOI: 10.1007/978-3-642-16129-2_8. Abruf am 23. 11. 2018.

Gervais, Arthur, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf und Srdjan Capkun (2016). „On the Security and Performance of Proof of Work Blockchains“. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*. The 2016 ACM SIGSAC Conference. Vienna, Austria: ACM Press, S. 3–16. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978341. Abruf am 12. 12. 2018.

Gilbert, Seth und Nancy Lynch (2002). „Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services“. In: *SIGACT News* 33.2, S. 51–59. ISSN: 0163-5700. DOI: 10.1145/564585.564601. Abruf am 09. 03. 2019.

Gipp, Bela, Norman Meuschke und André Gernandt (2015). „Decentralized Trusted Timestamping using the Crypto Currency Bitcoin“. In: *CoRR* abs/1502.04015. arXiv: 1502.04015. URL: http://arxiv.org/abs/1502.04015.

Glaser, Florian und Luis Bezzenberger (2015). „Beyond Cryptocurrencies - A Taxonomy of Decentralized Consensus Systems Germany, May 26-29, 2015“. In: *23rd European Conference on Information Systems, ECIS 2015, Münster, Germany, May 26-29, 2015*. Hrsg. von Jörg Becker, Jan Vom Brocke und Marco de Marco. URL: http://aisel.aisnet.org/ecis2015_cr/57.

Glasersfeld, Ernst von (1995). *Radical Constructivism: A Way of Knowing and Learning*. Falmer Press. ISBN: 978-0-7507-0387-1.

Gluchowski, Peter und Peter Chamoni, Hrsg. (2016). *Analytische Informationssysteme: Business Intelligence-Technologien und -Anwendungen*. 5., vollständig überarbeitete Auflage. Berlin Heidelberg: Springer Gabler. ISBN: 978-3-662-47762-5.

Goldenfein, Jake und Andrea Leiter (2018). „Legal Engineering on the Blockchain: Smart Contracts as Legal Conduct“. In: *Law and Critique* 29.2, S. 141–149. ISSN: 0957-8536, 1572-8617. DOI: 10.1007/s10978-018-9224-0. Abruf am 23. 11. 2018.

Gramoli, Vincent (2017). „From Blockchain Consensus Back to Byzantine Consensus“. In: *Future Generation Computer Systems*. ISSN: 0167739X. DOI: 10.1016/j.future.2017.09.023. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167739X17320095, Abruf am 05. 12. 2018.

Greiner, Martina E. und Hui Wang (2015). „Trust-Free Systems - a New Research and Design Direction to Handle Trust-Issues in P2P Systems: The Case of Bitcoin Rico, August 13-15, 2015“. In: *21st Americas Conference on Information Systems, AMCIS 2015, Puerto Rico, August 13-15, 2015*. Association for Information Systems. URL: http://aisel.aisnet.org/amcis2015/AdoptionofIT/GeneralPresentations/11.

Grochla, Erwin (1972). *Unternehmungsorganisation: neue Ansätze und Konzeptionen*. Betriebswirtschaftslehre 3. Reinbek bei Hamburg: Rowohlt. ISBN: 978-3-499-21003-7.

Gronback, Richard C. (2009). *Eclipse Modeling Project: A Domain-Specific Language Toolkit*. The Eclipse Series. Upper Saddle River NJ: Addison-Wesley. ISBN: 978-0-321-53407-1.

Haber, Stuart und W. Scott Stornetta (1991). „How to Time-Stamp a Digital Document“. In: *Advances in Cryptology-CRYPTO 90*. Hrsg. von Alfred J. Menezes und Scott A. Vanstone. Springer Berlin Heidelberg, S. 437–455. ISBN: 978-3-540-38424-3.

Haerder, Theo und Andreas Reuter (1983). „Principles of Transaction-Oriented Database Recovery“. In: *ACM Comput. Surv.* 15.4, S. 287–317. ISSN: 0360-0300. DOI: 10.1145/289.291. Abruf am 09. 03. 2019.

Harel, David und Bernhard Rumpe (2004). „Meaningful Modeling: What's the Semantics of Semantics?“ In: *Computer* 37.10, S. 64–72. ISSN: 0018-9162. DOI: 10.1109/MC.2004.172.

Härer, Felix (2012). *Vom hochflexiblen Geschäftsprozess zum ausführbaren Workflowschema für AristaFlow*. Bachelorarbeit. Universität Bamberg.

— (2014). *Vom SOM-Geschäftsprozessmodell zum Softwareartefakt - modellgetriebene Systementwicklung mit dem Eclipse Modeling Framework*. Masterarbeit. Universität Bamberg. DOI: 10.20378/irbo-50808.

— (2018). „Decentralized Business Process Modeling and Instance Tracking Secured By a Blockchain“. In: *Proceedings of the 26th European Conference on Information Systems (ECIS)*. Portsmouth, UK. DOI: 10.5281/zenodo.2585717.

Härer, Felix und Hans-Georg Fill (2019a). „A Comparison of Approaches for Visualizing Blockchains and Smart Contracts“. In: *Jusletter IT Weblaw: Tagungsband des 22. Internationalen Rechtsinformatik Symposions 2019* 21 February 2019. ISSN: 1664-848X. DOI: 10.5281/zenodo.2585575.

— (2019b). „Decentralized Attestation of Conceptual Models Using the Ethereum Blockchain“. In: *21st IEEE International Conference on Business Informatics (CBI 2019)*. Moscow, Russia.

Härer, Felix, Andreas Steffan und Alan Herz (2016). „Fallstudienbasierte Einführung von SOM“. In: *Geschäftsprozessorientierte Systementwicklung*. Hrsg. von Thomas Benker, Carsten Jürck und Matthias Wolf. Wiesbaden: Springer Fachmedien. ISBN: 978-3-658-14825-6. DOI: 10.1007/978-3-658-14826-3_8.

Helland, Pat (2015). „Immutability Changes Everything“. In: 7th Biennial Conference on Innovative Data Systems Research (CIDR15), S. 25.

Hermann, Andreas, Hendrik Scholta, Sebastian Bräuer und Jörg Becker (2017). „Collaborative Business Process Management - A Literature-Based Analysis of Methods for Supporting Model Understandability“. In: *Wirtschaftsinformatik 2017 Proceedings*. URL: https://aisel.aisnet.org/wi2017/track03/paper/5.

Herrmannsdoerfer, Markus und Maximilian Koegel (2010). „Towards a Generic Operation Recorder for Model Evolution“. In: *Proceedings of the 1st International Workshop on Model Comparison in Practice - IWMCP '10*. The 1st International Workshop. Malaga, Spain: ACM Press, S. 76. ISBN: 978-1-60558-960-2. DOI: 10.1145/1826147.1826161. Abruf am 23. 11. 2018.

Hess, Thomas (1996). *Entwurf Betrieblicher Prozesse*. Wiesbaden: Deutscher Universitätsverlag. ISBN: 978-3-8244-6284-1. URL: http://dx.doi.org/10.1007/978-3-663-08468-6.

Hewelt, Marcin und Mathias Weske (2016). „A Hybrid Approach for Flexible Case Modeling and Execution Brazil, September 18-22, 2016, Proceedings“. In: *Business Process Management Forum - BPM Forum 2016, Rio de Janeiro, Brazil, September 18-22, 2016, Proceedings*. Hrsg. von Marcello La Rosa, Peter Loos und Oscar Pastor. Lecture Notes in Business Information Processing. Springer, S. 38–54. ISBN: 978-3-319-45467-2. DOI: 10.1007/978-3-319-45468-9_3.

Hilgers, Dennis, Gordon Müller-Seitz und Frank Piller (2010). „Benkler Revisited Venturing Beyond the Open Source Software Arena?“ In: *ICIS 2010 Proceedings*. URL: https://aisel.aisnet.org/icis2010_submissions/97.

Huber, Sebastian (2014). *Informationsintegration in dynamischen Unternehmensnetzwerken*. Wiesbaden: Springer Fachmedien. ISBN: 978-3-658-07747-1. DOI: 10.1007/978-3-658-07748-8. Abruf am 23. 11. 2018.

Huber, Sebastian, Adrian Hauptmann, Matthias Lederer und Matthias Kurz (2013). „Managing Complexity in Adaptive Case Management“. In: *S-BPM ONE - Running Processes: 5th International Conference, S-BPM ONE 2013, Deggendorf, Germany, March 11-12, 2013. Proceedings*. Hrsg. von Herbert Fischer und Josef Schneeberger. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 209–226. ISBN: 978-3-642-36754-0. DOI: 10.1007/978-3-642-36754-0_13.

Huemer, C., P. Liegl, R. Schuster, H. Werthner und M. Zapletal (2008). „Inter-Organizational Systems: From Business Values over Business Processes to Deployment“. In: *2008 2nd IEEE International Conference on Digital Ecosystems and Technologies*. 2008 2nd IEEE International Conference on Digital Ecosystems and Technologies, S. 294–299. DOI: 10.1109/DEST.2008.4635169.

„Bildung Und Entwicklung von Strategischen Allianzen“ (1999). In: *Kooperation Im Wettbewerb: Neue Formen Und Gestaltungskonzepte Im Zeichen von Globalisierung Und Informationstechnologie*. Hrsg. von Harald Hungenberg. 61. Wissenschaftliche Jahrestagung Des Verbandes Der Hochschullehrer Für Betriebswirtschaft e.V., Bamberg, 1999. Gabler Verlag. ISBN: 978-3-322-86919-7. URL: https://www.springer.com/de/book/9783322869197, Abruf am 03. 03. 2019.

Ingalls, Daniel H. H. (1978). „The Smalltalk-76 Programming System Design and Implementation“. In: *Proceedings of the 5th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. New York, NY, USA, S. 9–16. URL: http://doi.acm.org/10.1145/512760.512762, Abruf am 06. 03. 2019.

ISO (2002). *ISO/IEC 13568:2002: Information TechnologyZ Formal Specification NotationSyntax, Type System and Semantics*.

Jablonski, Stefan, Markus Böhm und Wolfgang Schulze (1997). *Workflow-Management*. Heidelberg: Dpunkt. ISBN: 978-3-920993-73-7.

Jacobson, Ivar, Magnus Christerson, Patrik Jonsson und Gunnar Övergaard (1992). *Object-Oriented Software Engineering: A Use Case Driven Approach*. Reading: Addison-Wesley.

Johnson, Don, Alfred Menezes und Scott Vanstone (2001). „The Elliptic Curve Digital Signature Algorithm (ECDSA)“. In: *International Journal of Information Security* 1.1, S. 36–63. ISSN: 1615-5262. DOI: 10.1007/s102070100002. Abruf am 23. 11. 2018.

Kahani, Nafiseh, Mojtaba Bagherzadeh, James R. Cordy, Juergen Dingel und Daniel Varró (2018). „Survey and Classification of Model Transformation Tools“. In: *Software & Systems Modeling*. ISSN: 1619-1366, 1619-1374. DOI: 10.1007/s10270-018-0665-6. Abruf am 23. 11. 2018.

Karagiannis, Dimitris und Harald Kühn (2002). „Metamodelling Platforms“. In: *E-Commerce and Web Technologies*. Hrsg. von Kurt Bauknecht, A Min Tjoa und Gerald Quirchmayr. Bd. 2455. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 182–182. ISBN: 978-3-540-44137-3. DOI: 10.1007/3-540-45705-4_19. Abruf am 18. 02. 2019.

Karagiannis, Dimitris, Heinrich C. Mayr und John Mylopoulos, Hrsg. (2016). *Domain-Specific Conceptual Modeling*. Springer Berlin Heidelberg.

Kaufmann, Petra, Gerti Kappel, Martina Seidl, Konrad Wieland, Manuel Wimmer, Horst Kargl und Philip Langer (2010). „Adaptable Model Versioning in Action“. In: *Modellierung 2010*. Klagenfurt.

Kaulartz, Markus und Jörn Heckmann (2016). „Smart Contracts Anwendungen Der Blockchain-Technologie“. In: *Computer und Recht* 32.9. ISSN: 2194-4172. DOI: 10.9785/cr-2016-0923. URL: http://www.degruyter.com/view/j/cr.2016.32.issue-9/cr-2016-0923/cr-2016-0923.xml, Abruf am 25. 11. 2018.

Keller, Gerhard, Markus Nüttgens und August-Wilhelm Scheer (1992). *Semantische Prozessmodellierung Auf Der Grundlage Ereignisgesteuerter Prozessketten (EPK)*. Hrsg. von August-Wilhelm Scheer. Institut Für Wirtschaftsinformatik Im Institut Für Empirische Wirtschaftsforschung an Der Universität Des Saarlandes, Institut Für Wirtschaftsinformatik (Saarbrücken).

Kelly, Steven und Juha-Pekka Tolvanen (2018). „Collaborative Creation and Versioning of Modeling Languages with MetaEdit+“. In: *Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and*

*Systems Companion Proceedings - MODELS '18*. The 21st ACM/IEEE International Conference. Copenhagen, Denmark: ACM Press, S. 37–41. ISBN: 978-1-4503-5965-8. DOI: 10.1145/3270112.3270132. Abruf am 23. 11. 2018.

Kemper, Alfons und André Eickler (2015). *Datenbanksysteme: Eine Einführung*. De Gruyter.

Kirsch, Werner und Heinz-Karl Klein (1977). *Management-Informationssysteme I*. Kohlhammer. ISBN: 978-3-17-004228-5.

Koblitz, Neal (1987). „Elliptic Curve Cryptosystems". In: *Mathematics of Computation* 48.177, S. 203–209. ISSN: 00255718, 10886842. JSTOR: 2007884.

Korpela, Kari, Jukka Hallikas und Tomi Dahlberg (2017). „Digital Supply Chain Transformation toward Blockchain Integration". In: Hawaii International Conference on System Sciences. DOI: 10.24251/HICSS.2017.506. URL: http://hdl.handle.net/10125/41666, Abruf am 02. 03. 2019.

Kosiol, Erich (1976). *Organisation Der Unternehmung*. 2., durchges. Aufl. Die Wirtschaftswissenschaften. Wiesbaden: Gabler. ISBN: 978-3-409-88454-9.

Kostecki, Jacob und Aashish Sharma (2018). *Crypto Law Review*. URL: https://medium.com/cryptolawreview/about, Abruf am 09. 03. 2019.

Krcmar, Helmut (2015). *Informationsmanagement*. 6. Aufl. Gabler Verlag. ISBN: 978-3-662-45862-4. URL: https://www.springer.com/de/book/9783662458624, Abruf am 07. 03. 2019.

Kreiss, Daniel, Megan Finn und Fred Turner (2011). „The Limits of Peer Production: Some Reminders from Max Weber for the Network Society". In: *New Media & Society* 13, S. 243–259. DOI: 10.1177/1461444810370951.

Kurbel, Karl (2016). *Enterprise Resource Planning und Supply Chain Management in der Industrie: Von MRP bis Industrie 4.0*. Berlin, Boston: De Gruyter. ISBN: 978-3-11-044169-7. DOI: 10.1515/9783110441697. URL: https://www.degruyter.com/view/books/9783110441697/9783110441697/9783110441697.xml, Abruf am 07. 03. 2019.

Kurose, James F. und Keith W. Ross (2013). *Computer Networking: A Top-down Approach*. 6th ed. Boston: Pearson. ISBN: 978-0-13-285620-1.

— (2017). *Computer Networking: A Top-Down Approach*. 7. Aufl. Boston, MA: Pearson.

Kuryazov, Dilshodbek, Andreas Winter und Ralf Reussner (2018). „Collaborative Modeling Enabled By Version Control". In: *Modellierung 2018*. Braunschweig: Gesellschaft für Informatik e.V., S. 183–198.

Kurz, Matthias (2010). „BPM 2.0 - Kollaborative Gestaltung von Geschäftsprozessen". In: Multikonferenz Wirtschaftsinformatik 2010. Hrsg. von Matthias Schumann, Lutz M. Kolbe, Michael H. Breitner und Arne Frerichs. Göttingen, S. 12.

— (2016). „BPMN Model Interchange: The Quest for Interoperability“. In: *Proceedings of the 8th International Conference on Subject-Oriented Business Process Management* (Erlangen, Germany). S-BPM '16. New York, NY, USA: ACM, 6:1–6:10. ISBN: 978-1-4503-4071-7. DOI: 10.1145/2882879.2882886. Abruf am 08. 03. 2019.

Kurz, Matthias und Albert Fleischmann (2011). „BPM 2.0: Business Process Management Meets Empowerment“. In: *Subject-Oriented Business Process Management*. Hrsg. von Albert Fleischmann, Werner Schmidt, Robert Singer und Detlef Seese. Communications in Computer and Information Science. Springer Berlin Heidelberg, S. 54–83. ISBN: 978-3-642-23135-3.

Lamport, Leslie (1978). „Time, Clocks, and the Ordering of Events in a Distributed System“. In: *Communications of the ACM* 21.7, S. 558–565. ISSN: 00010782. DOI: 10.1145/359545.359563. Abruf am 23. 11. 2018.

— (1989). *The Part-Time Parliament*. Arbeitsbericht, Digital Equipment Corporation Systems Research Center.

— (2002). *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 978-0-321-14306-8.

Lamport, Leslie, Robert Shostak und Marshall Pease (1982). „The Byzantine Generals Problem“. In: *ACM Transactions on Programming Languages and Systems* 4.3, S. 382–401. ISSN: 01640925. DOI: 10.1145/357172.357176.

Lee, Sunhwa, Kwangyeol Ryu, Sangil Lee, Jeonghoon Shin und Bohyun Kim (2010). „Extended Collaboration Process Modeling Method Enabling Model Verification“. In: *The 40th International Conference on Computers & Indutrial Engineering*. Industrial Engineering (CIE-40). Awaji City, Japan: IEEE, S. 1–6. ISBN: 978-1-4244-7295-6. DOI: 10.1109/ICCIE.2010.5668235. URL: http://ieeexplore.ieee.org/document/5668235/, Abruf am 23. 11. 2018.

Lemieux, Victoria Louise (2016). „Trusting Records: Is Blockchain Technology the Answer?“ In: *Records Management Journal* 26.2, S. 110–139. ISSN: 0956-5698. DOI: 10.1108/RMJ-12-2015-0042. Abruf am 24. 11. 2018.

Levy, Karen E. C. (2017). „Book-Smart, Not Street-Smart: Blockchain-Based Smart Contracts and The Social Workings of Law“. In: *Engaging Science, Technology, and Society*. Bd. 3, S. 1–15. URL: http://estsjournal.org/article/download/107/61.pdf.

Lewenberg, Yoad, Yonatan Sompolinsky und Aviv Zohar (2015). „Inclusive Block Chain Protocols“. In: *Financial Cryptography and Data Security*. Hrsg. von Rainer

Böhme und Tatsuaki Okamoto. Bd. 8975. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 528–547. ISBN: 978-3-662-47853-0. DOI: 10.1007/978-3-662-47854-7_33. Abruf am 05. 12. 2018.

Linke, Daniel und Susanne Strahringer (2018). „Integration einer Blockchain in ein ERP-System für den Procure-to-Pay-Prozess: Prototypische Realisierung mit SAP S/4HANA und Hyperledger Fabric am Beispiel der Daimler AG". In: *HMD Praxis der Wirtschaftsinformatik* 55.6, S. 1341–1359. ISSN: 1436-3011, 2198-2775. DOI: 10.1365/s40702-018-00472-8. Abruf am 12. 12. 2018.

Linux Foundation (2017). *Hyperledger Architecture, Volume 1*. URL: https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf, Abruf am 30. 11. 2018.

Liu, Chamond (2000). *Smalltalk, Objects, and Design*. iUniverse. ISBN: 978-1-58348-490-6.

Liu, Chengfei, Qing Li und Xiaohui Zhao (2009). „Challenges and Opportunities in Collaborative Business Process Management: Overview of Recent Advances and Introduction to the Special Issue". In: *Information Systems Frontiers* 11.3, S. 201–209. ISSN: 1387-3326, 1572-9419. DOI: 10.1007/s10796-008-9089-0. Abruf am 23. 11. 2018.

Liu, Duen-Ren und Minxin Shen (2003). „Workflow Modeling for Virtual Processes: An Order-Preserving Process-View Approach". In: *Information Systems* 28.6, S. 505–532. ISSN: 03064379. DOI: 10.1016/S0306-4379(02)00028-5. URL: http://linkinghub.elsevier.com/retrieve/pii/S0306437902000285, Abruf am 23. 11. 2018.

Lombrozo, Eric, Lau Johnson und Pieter Wuille (2018). *Bitcoin Improvement Proposal 141*. Bitcoin. URL: https://github.com/bitcoin/bips, Abruf am 25. 11. 2018.

Lua, Eng Keong, and J. Crowcroft, M. Pias, R. Sharma und S. Lim (2005). „A Survey and Comparison of Peer-to-Peer Overlay Network Schemes". In: *IEEE Communications Surveys Tutorials* 7.2, S. 72–93. ISSN: 1553-877X. DOI: 10.1109/COMST.2005.1610546.

Ludwig, Heiko und Keith Whittingham (1999). „Virtual Enterprise Co-Ordinator—Agreement-Driven Gateways for Cross-Organisational Workflow Management". In: *Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration - WACC '99*. The International Joint Conference. San Francisco, California, United States: ACM Press, S. 29–38. ISBN: 978-1-58113-070-6. DOI: 10.1145/295665.295670. Abruf am 23. 11. 2018.

Malischewski, Carsten (1997). „Generierung von Spezifikationen Betrieblicher Anwendungssysteme Auf Der Basis von Geschäftsprozeßmodellen“. Dissertation. Bamberg: Universität Bamberg. URL: http://d-nb.info/952308878.

— (2016). „Modellgetriebene Entwicklung Objektorientierter Anwendungssysteme Auf Basis von SOM“. In: *Geschäftsprozessorientierte Systementwicklung*. Hrsg. von Thomas Benker, Carsten Jürck und Matthias Wolf. Wiesbaden: Springer Fachmedien. ISBN: 978-3-658-14825-6. DOI: 10.1007/978-3-658-14826-3.

Martin, James und James J. Odell (1992). *Object-Oriented Analysis and Design*. First Edition edition. Englewood Cliffs, N.J: Prentice Hall. ISBN: 978-0-13-630245-2.

Mattern, Friedemann (1989). „Virtual Time and Global States of Distributed Systems“. In: *Parallel and Distributed Algorithms*. North-Holland, S. 215–226.

Matthes, Dirk (2011). *Enterprise-Architecture-Frameworks-Kompendium: über 50 Rahmenwerke für das IT-Management*. Xpert.press. Berlin Heidelberg Dordrecht London New York: Springer. ISBN: 978-3-642-12954-4.

McCracken, Daniel D. und Edwin D. Reilly (2003). „Backus-Naur Form (BNF)“. In: *Encyclopedia of Computer Science*. Chichester, UK: John Wiley and Sons Ltd., S. 129–131. ISBN: 978-0-470-86412-8. URL: http://dl.acm.org/citation.cfm?id=1074100.1074155, Abruf am 07. 03. 2019.

Melange (2018). *The Melange Language Workbench*. URL: http://melange.inria.fr/, Abruf am 07. 03. 2019.

Merkle, Ralph C. (1988). „A Digital Signature Based on a Conventional Encryption Function“. In: *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*. CRYPTO '87. London, UK, UK: Springer-Verlag, S. 369–378. ISBN: 3-540-18796-0. URL: http://dl.acm.org/citation.cfm?id=646752.704751.

Mertens, Peter, Freimut Bodendorf, Wolfgang König, Arnold Picot, Matthias Schumann und Thomas Hess, Hrsg. (2012). *Grundzüge der Wirtschaftsinformatik*. 11. Aufl. Springer-Lehrbuch. Berlin: Springer Gabler. ISBN: 978-3-642-30514-6.

MetaCase (2018). *5.3 MERL Generator Definition Language*. URL: https://www.metacase.com/support/45/manuals/mwb/Mw-5_3.html, Abruf am 06. 03. 2019.

Mevius, Marco und Andreas Oberweis (2005). „A Petri-Net Based Approach to Performance Management of Collaborative Business Processes“. In: *16th International Workshop on Database and Expert Systems Applications (DEXA'05)*. 16th International Workshop on Database and Expert Systems Applications (DEXA'05). Copenhagen, Denmark: IEEE, S. 987–991. ISBN: 978-0-7695-2424-5. DOI: 10.1109/DEXA.2005.26. URL: http://ieeexplore.ieee.org/document/1508403/, Abruf am 23. 11. 2018.

Miller, Andrew, Yu Xia, Kyle Croman, Elaine Shi und Dawn Song (2016). „The Honey Badger of BFT Protocols“. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. New York, NY, USA: ACM, S. 31–42. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978399. Abruf am 18. 02. 2019.

Miller, Victor S. (1986). „Use of Elliptic Curves in Cryptography“. In: *Advances in Cryptology CRYPTO 85 Proceedings*. Hrsg. von Hugh C. Williams. Bd. 218. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 417–426. ISBN: 978-3-540-16463-0. DOI: 10.1007/3-540-39799-X_31. Abruf am 26. 11. 2018.

Motahari-Nezhad, H. R. und K. D. Swenson (2013). „Adaptive Case Management: Overview and Research Challenges“. In: *2013 IEEE 15th Conference on Business Informatics*. 2013 IEEE 15th Conference on Business Informatics, S. 264–269. DOI: 10.1109/CBI.2013.44.

Nakamoto, Satoshi (2008a). *Bitcoin: A Peer-to-Peer Electronic Cash System*. URL: https://bitcoin.org/bitcoin.pdf, Abruf am 24. 11. 2018.

— (2008b). *The Proof-of-Work Chain Is a Solution to the Byzantine Generals Problem*. URL: https://www.mail-archive.com/cryptography@metzdowd.com/msg09997.html, Abruf am 27. 11. 2018.

Namecoin (2018). *Namecoin*. URL: https://namecoin.org/, Abruf am 25. 11. 2018.

Narayanan, Arvind, Joseph Bonneau, Edward Felten, Andrew Miller und Steven Goldfeder (2016). *Bitcoin and Cryptocurrency Technologies*. Princeton: Princeton University Press. ISBN: 978-0-691-17169-2.

Narayanan, Arvind und Jeremy Clark (2017). „Bitcoin's Academic Pedigree“. In: *Communications of the ACM* 60.12, S. 36–45. ISSN: 00010782. DOI: 10.1145/3132259. Abruf am 23. 11. 2018.

Nastansky, Ludwig und Wolfgang Hilpert (1994). „The GroupFlow System: A Scalable Approach to Workflow Management between Cooperation and Automation“. In: *Innovationen Bei Rechen- Und Kommunikationssystemen*. Hrsg. von Bernd Wolfinger. Informatik Aktuell. Springer Berlin Heidelberg, S. 473–479. ISBN: 978-3-642-51136-3.

Niehaves, Bjoern und Ralf Plattfaut (2011). „Collaborative Business Process Management: Status Quo and *Quo Vadis*“. In: *Business Process Management Journal* 17.3, S. 384–402. ISSN: 1463-7154. DOI: 10.1108/14637151111136342. Abruf am 23. 11. 2018.

Nielson, Hanne Riis und Flemming Nielson (2007). *Semantics with Applications: An Appetizer*. Undergraduate Topics in Computer Science. London: Springer-Verlag.

ISBN: 978-1-84628-691-9. URL: https://www.springer.com/de/book/9781846286919, Abruf am 06. 03. 2019.

NIST (2015). *Secure Hash Standard (SHS)*. Federal Information Processing Standard (FIPS) 180-4. U.S. Department of Commerce. DOI: https://doi.org/10.6028/NIST.FIPS.180-4. URL: https://csrc.nist.gov/publications/detail/fips/180/4/final, Abruf am 06. 03. 2019.

Nofer, Michael, Peter Gomber, Oliver Hinz und Dirk Schiereck (2017). „Blockchain". In: *Business & Information Systems Engineering* 59.3, S. 6. ISSN: 2363-7005. DOI: 10.1007/s12599-017-0467-3.

Nonaka, Ikujiro und Hirotaka Takeuchi (1995). *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press. ISBN: 978-0-19-987992-2.

Notheisen, Benedikt, Jacob Benjamin Cholewa und Arun Prasad Shanmugam (2017). „Trading Real-World Assets on Blockchain: An Application of Trust-Free Transaction Systems in the Market for Lemons". In: *Business & Information Systems Engineering* 59.6, S. 425–440. ISSN: 2363-7005, 1867-0202. DOI: 10.1007/s12599-017-0499-8. Abruf am 24. 11. 2018.

Oliveira, Luis, Liudmila Zavolokina, Ingrid Bauer und Gerhard Schwabe (2018). „To Token or Not to Token: Tools for Understanding Blockchain Tokens". In: *International Conference of Information Systems (ICIS 2018)*. San Francisco, USA: s.n. DOI: 10.5167/uzh-157908.

OMG (2001). *Model Driven Architecture*. ormsc/2001-07-01. URL: https://www.omg.org/cgi-bin/doc?ormsc/2001-07-01, Abruf am 08. 01. 2019.

— (2003). *MDA Guide Version 1.0.1*. omg/2003-06-01. URL: http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf, Abruf am 08. 01. 2019.

— (2011). *MOF Query/View/Transformation, Version 1.1*. formal/11-01-01. URL: https://www.omg.org/spec/QVT/1.1/PDF, Abruf am 28. 11. 2018.

— (2014). *Business Process Model and Notation (BPMN), Version 2.0.2*. formal/2013-12-09. URL: http://www.omg.org/spec/BPMN/2.0.2, Abruf am 28. 11. 2018.

— (2015). *XML Metadata Interchange, Version 2.5.1*. formal/2013-12-09. URL: https://www.omg.org/spec/XMI/2.5.1/PDF, Abruf am 08. 01. 2019.

— (2016a). *Case Management Model and Notation, Version 1.1*. formal/16-12-01. URL: https://www.omg.org/spec/CMMN/1.1/PDF, Abruf am 08. 01. 2019.

— (2016b). *Meta Object Facility, Version 2.5.1*. formal/16-11-01. URL: https://www.omg.org/spec/MOF/2.5.1/PDF, Abruf am 08. 01. 2019.

— (2017a). *OMG System Modeling Language, Version 1.5*. formal/17-05-01. URL: https://www.omg.org/spec/SysML/1.5/PDF, Abruf am 08. 01. 2019.

OMG (2017b). *Unified Modeling Language, Version 2.5.1*. formal/17-12-05. URL: https://www.omg.org/spec/UML/About-UML/, Abruf am 08. 01. 2019.

— (2018). *Bpmn Tools Tested for Model Interchange*. URL: http://bpmn-miwg.github.io/bpmn-miwg-tools/, Abruf am 08. 01. 2019.

— (2019). *Decision Model and Notation, Version 1.2*. formal/19-01-05. URL: https://www.omg.org/spec/DMN/1.2/PDF, Abruf am 08. 01. 2019.

OpenTimestamps (2018). *OpenTimestamps: A Timestamping Proof Standard*. URL: https://opentimestamps.org/, Abruf am 25. 11. 2018.

Oppl, Stefan (2017). „Supporting the Collaborative Construction of a Shared Understanding About Work with a Guided Conceptual Modeling Technique“. In: *Group Decision and Negotiation* 26.2, S. 247–283. ISSN: 0926-2644, 1572-9907. DOI: 10.1007/s10726-016-9485-7. Abruf am 23. 11. 2018.

Österle, Hubert (1995). *Business Engineering. Prozeß- Und Systementwicklung: Band 1: Entwurfstechniken*. 2. Aufl. Berlin Heidelberg: Springer-Verlag. ISBN: 978-3-540-60048-0. URL: https://www.springer.com/de/book/9783540600480, Abruf am 07. 03. 2019.

Österle, Hubert, Elgar Fleisch und Rainer Alt (2002). *Business Networking in Der Praxis: Beispiele Und Strategien Zur Vernetzung Mit Kunden Und Lieferanten*. Business Engineering. Berlin Heidelberg: Springer-Verlag. ISBN: 978-3-540-41370-7. URL: https://www.springer.com/de/book/9783540413707, Abruf am 08. 03. 2019.

Österle, Hubert, Robert Winter und Walter Brenner, Hrsg. (2010). *Gestaltungsorientierte Wirtschaftsinformatik: ein Plädoyer für Rigor und Relevanz*. Nürnberg: Infowerk. ISBN: 978-3-00-030310-4.

Ouyang, Chun, Marlon Dumas, A. H. m Ter Hofstede und Wil van der Aalst (2006). „From BPMN Process Models to BPEL Web Services“. In: *2006 IEEE International Conference on Web Services (ICWS'06)*. 2006 IEEE International Conference on Web Services (ICWS'06), S. 285–292. DOI: 10.1109/ICWS.2006.67.

Parity (2018a). *Parity Documentation - Getting Synced*. URL: http://wiki.parity.io/Getting-Synced.html, Abruf am 25. 11. 2018.

— (2018b). *Parity Documentation - Private Chains*. URL: http://wiki.parity.io/Private-chains.html, Abruf am 29. 11. 2018.

Pass, Rafael, Lior Seeman und Abhi Shelat (2017). „Analysis of the Blockchain Protocol in Asynchronous Networks“. In: *Advances in Cryptology – EUROCRYPT 2017*. Hrsg. von Jean-Sébastien Coron und Jesper Buus Nielsen. Cham: Springer International Publishing, S. 643–673. ISBN: 978-3-319-56614-6.

Pease, Marshall, Robert Shostak und Leslie Lamport (1980). „Reaching Agreement in the Presence of Faults“. In: Bd. 27. ACM, S. 228–234. URL: http://research.microsoft.com/en-us/um/people/lamport/pubs/reaching.pdf.

Peltz, C. (2003). „Web Services Orchestration and Choreography“. In: *Computer* 36.10, S. 46–52. ISSN: 0018-9162. URL: doi.ieeecomputersociety.org/10.1109/MC.2003.1236471, Abruf am 19. 02. 2019.

Picot, Arnold, Ralf Reichwald und Rolf T. Wigand (2003). *Die Grenzenlose Unternehmung: Information, Organisation Und Management. Lehrbuch Zur Unternehmensführung Im Informationszeitalter*. 5. Aufl. Gabler Verlag. ISBN: 978-3-8349-2162-8. URL: https://www.springer.com/de/book/9783834921628, Abruf am 08. 03. 2019.

Piller, Frank T., Kathrin Möslein, Christoph Clemens Ihle und Ralf Reichwald (2017). *Interaktive Wertschöpfung kompakt: Open Innovation, Individualisierung und neue Formen der Arbeitsteilung*. Lehrbuch. Wiesbaden: Springer Gabler. ISBN: 978-3-658-17513-9.

Plattner, Hasso (2014). *A Course in In-Memory Data Management: The Inner Mechanics of in-Memory Databases*. 2nd ed. 2014. Berlin, Heidelberg and s.l.: Springer Berlin Heidelberg. ISBN: 978-3-642-55270-0. DOI: 10.1007/978-3-642-55270-0. URL: http://dx.doi.org/10.1007/978-3-642-55270-0.

Poon, Joseph und Vitalik Buterin (2017). *Plasma: Scalable Autonomous Smart Contracts*. URL: https://plasma.io/plasma.pdf, Abruf am 25. 11. 2018.

Poon, Joseph und Thaddeus Dryja (2016). *The Bitcoin Lightning Network*. URL: https://lightning.network/lightning-network-paper.pdf, Abruf am 12. 12. 2018.

Porter, Michael E. (2014). *Wettbewerbsvorteile: Spitzenleistungen Erreichen Und Behaupten - Competitive Advantage*. 8., durchges. Aufl. Frankfurt am Main: Campus-Verl. ISBN: 3-593-50048-5.

Prahalad, C. K. und Venkat Ramaswamy (2004). „Co-Creation Experiences: The next Practice in Value Creation“. In: *Journal of Interactive Marketing* 18.3, S. 5–14. ISSN: 1094-9968. DOI: 10.1002/dir.20015. URL: http://www.sciencedirect.com/science/article/pii/S1094996804701073, Abruf am 08. 03. 2019.

Pütz, Corinna und Elmar J. Sinz (2010). „Model-Driven Derivation of BPMN Workflow Schemata from SOM Business Process Models“. In: *Enterprise Modelling and Information Systems Architectures (EMISAJ)* 5.2, S. 57–72. ISSN: 1866-3621. DOI: 10.18417/emisa.5.2.4. URL: https://www.emisa-journal.org/emisa/article/view/72, Abruf am 06. 03. 2019.

Raiden (2018). *Raiden Network*. URL: https://raiden.network/, Abruf am 25. 11. 2018.

Redlich, Tobias und Manuel Moritz (2016). „Bottom-up Economics. Foundations of a Theory of Distributed and Open Value Creation“. In: *The Decentralized and Networked Future of Value Creation: 3D Printing and Its Implications for Society, Industry, and Sustainable Development*. Hrsg. von Jan-Peter Ferdinand, Ulrich Petschow und Sascha Dickel. Progress in IS. Cham: Springer International Publishing, S. 27–57. ISBN: 978-3-319-31686-4. DOI: 10.1007/978-3-319-31686-4_3. Abruf am 08. 03. 2019.

Redlich, Tobias, Manuel Moritz und Stefanie Wulf (2017). „Digitale Produktion: Bottom-up-Ökonomie“. In: *Digitalzeitalter - Digitalgesellschaft: Das Ende des Industriezeitalters und der Beginn einer neuen Epoche*. Hrsg. von Oliver Stengel, Alexander van Looy und Stephan Wallaschkowski. Wiesbaden: Springer Fachmedien, S. 143–167. ISBN: 978-3-658-16509-3. DOI: 10.1007/978-3-658-16509-3_7. Abruf am 06. 03. 2019.

Redlich, Tobias, Manuel Moritz und Jens Wulfsberg, Hrsg. (2019). *Co-Creation: Reshaping Business and Society in the Era of Bottom-up Economics*. Management for Professionals. Springer International Publishing. ISBN: 978-3-319-97787-4. URL: https://www.springer.com/us/book/9783319977874, Abruf am 05. 03. 2019.

Redlich, Tobias und Jens Wulfsberg (2011). *Wertschöpfung in der Bottom-up-Ökonomie*. VDI-Buch. Berlin: Springer. ISBN: 978-3-642-19879-3.

Redlich, Tobias, Jens Wulfsberg und Franz Bruhns (2010). „Open Production wissenschaftliche Fundierung der Wertschöpfungsgestaltung in der Bottom-up-Ökonomie“. In: *Arbeitsgruppe Wertschöpfungssystematik, Helmut Schmidt Universität, Hamburg*, S. 26.

Reisig, Wolfgang (2010). *Petrinetze: Modellierungstechnik, Analysemethoden, Fallstudien*. Vieweg + Teubner.

Rinderle, Stefanie und Peter Dadam (2003). „Schemaevolution in Workflow-Management-Systemen“. In: *Informatik-Spektrum* 26.1, S. 17–19. ISSN: 0170-6012.

Rittgen, Peter (2009). „Collaborative Modeling - A Design Science Approach“. In: *42st Hawaii International International Conference on Systems Science (HICSS-42 2009)*. Waikoloa, Big Island, HI, USA: IEEE Computer Society, S. 1–10. ISBN: 978-0-7695-3450-3. DOI: 10.1109/HICSS.2009.112.

Rivest, R. L., A. Shamir und L. Adleman (1978). „A Method for Obtaining Digital Signatures and Public-Key Cryptosystems“. In: *Communications of the ACM* 21.2, S. 120–126. ISSN: 00010782. DOI: 10.1145/359340.359342. Abruf am 23. 11. 2018.

Rogaway, Phillip und Thomas Shrimpton (2004). „Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance,

Second-Preimage Resistance, and Collision Resistance“. In: *Fast Software Encryption*. Hrsg. von Bimal Roy und Willi Meier. Bd. 3017. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 371–388. ISBN: 978-3-540-22171-5. DOI: 10.1007/978-3-540-25937-4_24. Abruf am 23. 11. 2018.

Rohloff, Michael (2009). *Integrierte Gestaltung von Unternehmensorganisation und IT*. Berlin: Gito. ISBN: 978-3-940019-62-2.

Rumbaugh, James, Michael Blaha, William Premerlani, Frederick Eddy und Bill Lorensen (1991). *Object-Oriented Modeling and Design*. Englewood Cliffs, N.J.: Prentice Hall. ISBN: 978-0-13-629841-0.

Rumbaugh, James, Ivar Jacobson und Grady Booch (2004). *The Unified Modeling Language Reference Manual*, 2 edition. Boston: Addison-Wesley Professional. ISBN: 978-0-321-71895-2.

Ryu, Kwangyeol und Enver Yücesan (2007). „CPM: A Collaborative Process Modeling for Cooperative Manufacturers“. In: *Advanced Engineering Informatics* 21.2, S. 231–239. ISSN: 14740346. DOI: 10.1016/j.aei.2006.05.003. URL: http://linkinghub.elsevier.com/retrieve/pii/S1474034606000279, Abruf am 23. 11. 2018.

Sandkuhl, Kurt, Matthias Wißotzki und Janis Stirna (2013). *Unternehmensmodellierung: Grundlagen, Methode Und Praktiken*. Xpert.Press. Springer Vieweg. ISBN: 978-3-642-31092-8. URL: https://www.springer.com/de/book/9783642310928, Abruf am 06. 03. 2019.

Scheer, August-Wilhelm (1991). *Architektur Integrierter Informationssysteme: Grundlagen Der Unternehmensmodellierung*. Berlin Heidelberg: Springer-Verlag. ISBN: 978-3-642-97333-8. URL: https://www.springer.com/de/book/9783642973338, Abruf am 07. 03. 2019.

— (1998). *ARIS Modellierungsmethoden, Metamodelle, Anwendungen*. 3. Aufl. Springer.

Scheer, August-Wilhelm, Wolfram Jost und Karl Wagner, Hrsg. (2005). *Von Prozessmodellen zu lauffähigen Anwendungen: ARIS in der Praxis*. Berlin: Springer. ISBN: 978-3-540-23457-9.

Schneider, Fred B. (1990). „Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial“. In: *ACM Computing Surveys* 22.4, S. 299–319. ISSN: 03600300. DOI: 10.1145/98163.98167. Abruf am 23. 11. 2018.

Schreyögg, Georg und Daniel Geiger (2016). *Organisation: Grundlagen Moderner Organisationsgestaltung. Mit Fallstudien*. 6. Aufl. Gabler Verlag. ISBN: 978-3-8349-4484-9. URL: https://www.springer.com/de/book/9783834944849, Abruf am 07. 03. 2019.

Seidlmeier, Heinrich (2015). *Prozessmodellierung mit ARIS: eine beispielorientierte Einführung für Studium und Praxis in ARIS 9*. 4., aktualisierte Auflage. Lehrbuch. Wiesbaden: Springer Vieweg. ISBN: 978-3-658-03904-2.

Shapiro, Marc, Nuno Preguiça, Carlos Baquero und Marek Zawirski (2011). „Conflict-Free Replicated Data Types“. In: *Stabilization, Safety, and Security of Distributed Systems*. Hrsg. von Xavier Défago, Franck Petit und Vincent Villain. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 386–400. ISBN: 978-3-642-24550-3.

Simon, Herbert A., Harold Guetzkow, George Kozmetsky und Gordon Tyndall (1954). *Centralization vs. Decentralization in Organizing the Controller's Department.: A Research Study and Report Prepared for Controllership Foundation, Inc.* Series A-1 Controllership: Contracts, Organization. Report,No. 4. New York: Controllership Foundation. xiii, 106 p. URL: `https://catalog.hathitrust.org/Record/005868448`, Abruf am 06. 03. 2019.

Sinz, Elmar J. (1988). „Das Strukturierte Entity-Relationship-Modell (SER-Modell)“. In: *Angewandte Informatik* 30.5, S. 191–202.

— (1995). *Ein Architekturrahmen Für Die Modellierung Betrieblicher Informationssysteme*. Bamberger Beiträge Zur Wirtschaftsinformatik 32. Bamberg.

— (1996). „Ansätze Zur Fachlichen Modellierung Betrieblicher Informationssysteme. Entwicklung, Aktueller Stand Und Trends“. In: *Information Engineering*. Hrsg. von H. Heilmann, L.J. Heinrich und F. Roithmayr. München: Oldenbourg-Verlag.

— (1997). *Architektur Betrieblicher Informationssysteme*. Bamberger Beiträge Zur Wirtschaftsinformatik 40. Bamberg.

— (2002). „Architektur von Informationssystemen“. In: *Informatik-Handbuch*. Hrsg. von Peter Rechenberg und Gustav Pomberger. München: Hanser, S. 1055–1068. ISBN: 978-3-446-21842-0.

— (2010a). „Architektur von Data-Warehouse-Systemen“. In: Ulbrich-vom Ende, Achim. *Analytische Informationssysteme*. Hrsg. von Peter Chamoni und Peter Gluchowski. 4. Aufl. Heidelberg: Springer, S. 175–196.

— (2010b). „Konstruktionsforschung in der Wirtschaftsinformatik: Was sind die Erkenntnisziele gestaltungsorientierter Wirtschaftsinformatik-Forschung?“ In: *Gestaltungsorientierte Wirtschaftsinformatik: ein Plädoyer für Rigor und Relevanz*. Hrsg. von Hubert Österle, Robert Winter und Walter Brenner. Nürnberg: Infowerk. ISBN: 978-3-00-030310-4.

— (2012). *Konzeption Und Implementierung Hochflexibler Geschäftsprozesse*. Bamberg: Projektseminar zur Wirtschaftsinformatik, 2012-10-26, Universität Bamberg.

— (2014). „Konzeptuelle Modellierung der Zustandskonsistenz verteilter betrieblicher Informationssysteme“. In: *Modellierung 2014. Lecture Notes in Informatics (LNI) - Proceedings, Nr. 225*. Hrsg. von Hans-Georg Fill, Dimitris Karagiannis und Ulrich Reimer. Bd. 225. Wien, S. 241–256.

— (2016). *Modellierung Betrieblicher Informationssysteme*. Bamberg: Vorlesung, Wintersemester 2016/2017, Universität Bamberg.

— (2019). „On the Evolution of Methods for Conceptual Information Systems Modeling“. In: *The Art of Structuring: Bridging the Gap Between Information Systems Research and Practice*. Hrsg. von Katrin Bergener, Michael Räckers und Armin Stein. Cham: Springer International Publishing, S. 137–144. ISBN: 978-3-030-06234-7. DOI: 10.1007/978-3-030-06234-7_13.

Sinz, Elmar J., Dieter Bartmann, Freimut Bodendorf und Otto K. Ferstl (2011). *Dienstorientierte IT-Systeme für hochflexible Geschäftsprozesse*. Bamberg: University of Bamberg Press. ISBN: 978-3-86309-009-8.

Sitonio, Camila und Alberto Nucciarelli (2018). „The Impact of Blockchain on the Music Industry“. In: *R&D Management Conference 2018*. Milan, Italy.

Sixt, Elfriede (2017). *Bitcoins und andere dezentrale Transaktionssysteme*. Wiesbaden: Springer Fachmedien. ISBN: 978-3-658-02843-5. DOI: 10.1007/978-3-658-02844-2. Abruf am 06. 12. 2018.

Solihin, Yan (2015). *Fundamentals of Parallel Multicore Architecture*. CRC Press. ISBN: 978-1-4822-1119-1.

Sommerville, Ian (2011). *Software Engineering*. 9th edition. Boston: Pearson. ISBN: 978-0-13-703515-1.

Sousa, J., A. Bessani und M. Vukolic (2018). „A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform“. In: *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, S. 51–58. DOI: 10.1109/DSN.2018.00018.

Stachowiak, Herbert (1973). *Allgemeine Modelltheorie*. Wien, New York: Springer.

Steen, Maarten van und Andrew S. Tanenbaum (2017). *Distributed Systems*. Third edition (Version 3.01 (2017). The Netherlands: Published by Maarten van Steen. ISBN: 978-1-5430-5738-6.

Steinberg, David, Frank Budinsky, Marcelo Paternostro und Ed Merks (2008). *EMF: Eclipse Modeling Framework*. 2nd ed. Eclipse Series. Upper Saddle River, N.J.: Addison-Wesley. ISBN: 978-0-321-33188-5. URL: http://proquest.safaribooksonline.com/9780321331885.

Steinmetz, Ralf und Klaus Wehrle (2005). *Peer-to-Peer Systems and Applications*. Springer.

Stephan, Matthew und James R. Cordy (2013). *A Survey of Methods and Applications of Model Comparison*. Report 2011-582 Rev . 2. Kingston, Ontario, Canada: Queen's University.

Stevens, Marc, Elie Bursztein, Pierre Karpman, Ange Albertini und Yarik Markov (2017). „The First Collision for Full SHA-1". In: *Advances in Cryptology CRYPTO 2017*. Hrsg. von Jonathan Katz und Hovav Shacham. Bd. 10401. Cham: Springer International Publishing, S. 570–596. ISBN: 978-3-319-63687-0. DOI: 10.1007/978-3-319-63688-7_19. Abruf am 28. 02. 2019.

Strahringer, Susanne (1998). „Ein Sprachbasierter Metamodellbegriff Und Seine Verallgemeinerung Durch Das Konzept Des Metaisierungsprinzips". In: *Modellierung 1998*. Münster.

Swan, Melanie (2015). *Blockchain: Blueprint for a New Economy*. First edition. Beijing : Sebastopol, CA: O'Reilly. ISBN: 978-1-4919-2049-7.

Szabo, Nick (1994). *Smart Contracts*. URL: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html, Abruf am 25. 11. 2018.

— (1997). „Formalizing and Securing Relationships on Public Networks". In: *First Monday* 2.9. ISSN: 13960466. DOI: 10.5210/fm.v2i9.548. URL: http://firstmonday.org/article/view/548/469.

— (2008). *Unenumerated: Wet Code and Dry*. URL: http://unenumerated.blogspot.com/2006/11/wet-code-and-dry.html, Abruf am 09. 03. 2019.

— (2018). *World Crypto Economic Forum (WCEF)*. nach https://bctoday.info/2018/01/19/nick-szabo-smart-contracts-p2p-economy/, Abfruf am 22.12.2018.

Szydlo, Michael (2004). „Merkle Tree Traversal in Log Space and Time". In: *Advances in Cryptology - EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings*. Hrsg. von Christian Cachin und Jan L. Camenisch. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 541–554. ISBN: 978-3-540-24676-3. DOI: 10.1007/978-3-540-24676-3_32. URL: http://dx.doi.org/10.1007/978-3-540-24676-3_32.

Taentzer, Gabriele, Claudia Ermel, Philip Langer und Manuel Wimmer (2014). „A Fundamental Approach to Model Versioning Based on Graph Modifications: From Theory to Implementation". In: *Software & Systems Modeling* 13.1, S. 239–272. ISSN: 1619-1366, 1619-1374. DOI: 10.1007/s10270-012-0248-x. Abruf am 23. 11. 2018.

Tendermint (2018). *Tendermint Documentation*. URL: https://tendermint.com/docs/, Abruf am 29. 11. 2018.

Tenorio-Fornés, Antonio, Samer Hassan und Juan Pavón (2018). „Open Peer-to-Peer Systems over Blockchain and IPFS: An Agent Oriented Framework“. In: *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems - CryBlock'18*. The 1st Workshop. Munich, Germany: ACM Press, S. 19–24. ISBN: 978-1-4503-5838-5. DOI: 10.1145/3211933.3211937. Abruf am 09. 12. 2018.

Teusch, Andree (2016). „Vom SOM-Geschäftsprozessmodell zur partiellen SOA“. In: *Geschäftsprozessorientierte Systementwicklung: Von der Unternehmensarchitektur zum IT-System*. Hrsg. von Thomas Benker, Carsten Jürck und Matthias Wolf. Wiesbaden: Springer Fachmedien, S. 261–277. ISBN: 978-3-658-14826-3. DOI: 10.1007/978-3-658-14826-3_17. Abruf am 08. 03. 2019.

Teusch, Andree und Elmar J. Sinz (2012). „Konzeptuelle Modellierung partieller SOA“. In: *Tagungsband der MKWI 2012*. Multikonferenz Wirtschaftsinformatik 2012. Hrsg. von Dirk Mattfeld und Susanne Robra-Bissantz. Braunschweig: GITO mbH Verlag, Berlin, S. 1637–1648.

The Open Group (2017). *ArchiMateő 3.0.1 Specification*. URL: http://pubs.opengroup.org/architecture/archimate3-doc/, Abruf am 18. 02. 2019.

— (2018). *TOGAFő 9.2 Specification*. URL: https://publications.opengroup.org/standards/togaf/specifications/c182, Abruf am 18. 02. 2019.

Thomas, Oliver (2005). *Das Modellverständnis in der Wirtschaftsinformatik: Historie, Literaturanalyse und Begriffsexplikation*. Institut für Wirtschaftsinformatik (IWi) im Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI GmbH), Universität des Saarlandes.

Thommen, Jean-Paul, Ann-Kristin Achleitner, Dirk Ulrich Gilbert, Dirk Hachmeister und Gernot Kaiser (2017). *Allgemeine Betriebswirtschaftslehre*. 8., vollständig überarbeitete Auflage. Lehrbuch. Wiesbaden: Springer Gabler. ISBN: 978-3-658-07767-9.

Tibco (2017). *TIBCO iProcess Engine Release Notes*. URL: https://docs.tibco.com/pub/ipe-oracle/11.6.1/TIB_ipe_windows_oracle_11.6.1_relnotes.pdf?id=4, Abruf am 24. 11. 2018.

Troxler, Peter (2016). „Fabrication Laboratories (Fab Labs)“. In: *The Decentralized and Networked Future of Value Creation*. Hrsg. von Jan-Peter Ferdinand, Ulrich Petschow und Sascha Dickel. Progress in IS. Cham: Springer International Publishing, S. 109–127. ISBN: 978-3-319-31684-0. DOI: 10.1007/978-3-319-31686-4.

Urbach, Nils (2017). *Blockchain Enzyklopaedie Der Wirtschaftsinformatik*. URL: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/daten-wissen/Datenmanagement/Datenbanksystem/blockchain/blockchain/, Abruf am 10. 12. 2018.

Vahs, Dietmar (2015). *Organisation: ein Lehr- und Managementbuch*. 9., überarbeitete und erweiterte Auflage. Stuttgart: Schäffer-Poeschel Verlag. ISBN: 978-3-7910-3437-9.

Van der Aalst, Wil (2000). „Loosely Coupled Interorganizational Workflows: Modeling and Analyzing Workflows Crossing Organizational Boundaries". In: *Information & Management* 37.2, S. 67–75. ISSN: 0378-7206. DOI: 10.1016/S0378-7206(99)00038-5.

Van der Aalst, Wil und Mathias Weske (2001). „The P2P Approach to Interorganizational Workflows". In: *Advanced Information Systems Engineering*. Hrsg. von Klaus R. Dittrich, Andreas Geppert und Moira C. Norrie. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 140–156. ISBN: 978-3-540-45341-3.

Villarreal, Pablo David, Ivanna Lazarte, Jorge Roa und Omar Chiotti (2010). „A Modeling Approach for Collaborative Business Processes Based on the UP-ColBPIP Language". In: *Business Process Management Workshops*. Hrsg. von Wil van der Aalst, John Mylopoulos, Norman M. Sadeh, Michael J. Shaw, Clemens Szyperski, Stefanie Rinderle-Ma, Shazia Sadiq und Frank Leymann. Bd. 43. Lecture Notes in Business Information Processing. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 318–329. ISBN: 978-3-642-12185-2. DOI: 10.1007/978-3-642-12186-9.

Voelter, Markus (2013). *DSL Engineering: Designing, Implementing and Using Domain-Specific Languages*. Lexington, KY: CreateSpace Independent Publishing Platform. ISBN: 978-1-4812-1858-0.

Vom Brocke, Jan und Heinz Lothar Grob (2015). *Referenzmodellierung: Gestaltung und Verteilung von Konstruktionsprozessen*. 2., unveränderte Auflage. Advances in information systems and management science Band 4. Berlin: Logos Verlag. ISBN: 978-3-8325-0179-2.

Von Hippel, Eric (2005). „Democratizing Innovation: The Evolving Phenomenon of User Innovation". In: *Journal für Betriebswirtschaft* 55.1, S. 63–78. ISSN: 1614-631X. DOI: 10.1007/s11301-004-0002-8. Abruf am 08. 03. 2019.

Vorbach, Stefan, Lukas Nadvornik, Christina Müller und Hedwig Höller (2016). „The Co-Creation Square". In: 1. interdisziplinäre Konferenz zur Zukunft der Wertschöpfung. Hrsg. von Jens Wulfsberg, Tobias Redlich und Manuel Moritz. Hamburg: Helmut-Schmidt-Universität Hamburg, S. 391.

Vossen, Gottfried und Jörg Becker, Hrsg. (1996). *Geschäftsprozessmodellierung Und Workflow-Management. Modelle, Methoden, Werkzeuge*. Thomson.

Vukoli, Marko (2016). „The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication“. In: *Open Problems in Network Security*. Hrsg. von Jan Camenisch und Doan Kesdoan. Lecture Notes in Computer Science. Springer International Publishing, S. 112–125. ISBN: 978-3-319-39028-4.

W3C (2007a). *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. URL: https://www.w3.org/TR/soap12-part1/#soapmep, Abruf am 09. 03. 2019.

— (2007b). *Web Services Description Language (WSDL) Version 2.0: Additional MEPs*. URL: https://www.w3.org/TR/wsdl20-additional-meps/, Abruf am 09. 03. 2019.

Wagner, Daniel und Otto K. Ferstl (2011). „Modellierung kontextsensitiver Geschäftsprozesse und Anwendungssysteme“. In: *Dienstorientierte IT-Systeme für hochflexible Geschäftsprozesse*. Hrsg. von Elmar J. Sinz, Dieter Bartmann, Freimut Bodendorf und Otto K. Ferstl, S. 173–191. URL: https://opus4.kobv.de/opus4-bamberg/frontdoor/index/index/docId/51756, Abruf am 08. 03. 2019.

Wagner, Daniel, Christian Suchan, Benjamin Leunig und Jochen Frank (2011). „Methode zur Analyse der Flexibilität von IS“. In: *Dienstorientierte IT-Systeme für hochflexible Geschäftsprozesse*. Hrsg. von Elmar J. Sinz, Dieter Bartmann, Freimut Bodendorf und Otto K. Ferstl, S. 79–106. URL: https://opus4.kobv.de/opus4-bamberg/frontdoor/index/index/docId/51754, Abruf am 08. 03. 2019.

Wattenhofer, Roger (2016). *The Science of the Blockchain*. First edition. CreateSpace Independent Publishing. ISBN: 978-1-5227-5183-0.

Werbach, Kevin (2018). „Trust, but Verify: Why the Blockchain Needs the Law“. In: *Berkeley Tech. L.J.* 33, S. 487. DOI: https://doi.org/10.15779/Z38H41JM9N.

Weske, Mathias (2012). *Business Process Management: Concepts, Languages, Architectures*. 2nd ed. Heidelberg ; New York: Springer. ISBN: 978-3-642-28615-5.

WfMC (2004). *Wf-XML 2.0 - XML Based Protocol for Run-Time Integration of Process Engines*. URL: http://www.wfmc.org/standards/docs/WfXML20-200410c.pdf, Abruf am 28. 11. 2018.

— (2006). *Workflow Reference Model*. tc003v11. URL: http://www.wfmc.org/standards/docs/tc003v11.pdf.

— (2012). *XPDL 2.2 - Process Definition Interface - XML Process Definition Language*. WFMC-TC1025. URL: http://www.xpdl.org/standards/xpdl-2.2/XPDL%202.2%20(2012-08-30).pdf, Abruf am 28. 11. 2018.

Wiefling, Stephan, Luigi Lo Iacono und Frederik Sandbrink (2017). „Anwendung der Blockchain außerhalb von Geldwährungen“. In: *Datenschutz und Datensicherheit - DuD* 41.8, S. 482–486. ISSN: 1614-0702, 1862-2607. DOI: 10.1007/s11623-017-0816-x. Abruf am 25. 11. 2018.

Wilde, Thomas und Thomas Hess (2006). „Methodenspektrum der Wirtschaftsinformatik“. In: *Arbeitsbericht Nr. 2/2006 des Instituts für Wirtschaftsinformatik und Neue Medien der Ludwig-Maximilians-Universität München*, S. 20.

— (2007). „Forschungsmethoden Der Wirtschaftsinformatik“. In: *Wirtschaftsinformatik* 49.4, S. 280–287. ISSN: 1861-8936. DOI: 10.1007/s11576-007-0064-z.

Wolf, Matthias (2015). „Modellbasierte Spezifikation von RESTful SOA Auf Basis Flexibler SOM-Geschäftsprozessmodelle“. Dissertation. Universität Bamberg. URL: http://opus4.kobv.de/opus4-bamberg/frontdoor/index/index/docId/45004.

Wood, Gavin (2014). *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Yellow Paper. URL: https://gavwood.com/paper.pdf, Abruf am 12. 12. 2018.

Wright, Aaron und Primavera De Filippi (2015). *Decentralized Blockchain Technology and the Rise of Lex Cryptographia*. SSRN Scholarly Paper. Rochester, NY: Social Science Research Network. URL: https://papers.ssrn.com/abstract=2580664, Abruf am 09. 03. 2019.

Wüst, Karl und Arthur Gervais (2017). „Do You Need a Blockchain?“ In: *2018 Crypto Valley Conference on Blockchain Technology (CVCBT). IACR Cryptology ePrint Archive 2017/375*. Zug, Switzerland, S. 45–54. URL: https://eprint.iacr.org/2017/375.pdf, Abruf am 18. 11. 2018.

Xie, Tao, Fanbao Liu und Dengguo Feng (2013). „Fast Collision Attack on MD5.“ In: *IACR Cryptology ePrint Archive* 2013/170. URL: https://eprint.iacr.org/2013/170, Abruf am 01. 03. 2019.

Xing, Zhenchang und Eleni Stroulia (2005). „UMLDiff: An Algorithm for Object-Oriented Design Differencing“. In: *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering* (Long Beach, CA, USA). ASE '05. New York, NY, USA: ACM, S. 54–65. ISBN: 978-1-58113-993-8. DOI: 10.1145/1101908.1101919. Abruf am 06. 03. 2019.

Xu, Xiwei, Cesare Pautasso, Liming Zhu, Vincent Gramoli, Alexander Ponomarev, An Binh Tran und Shiping Chen (2016). „The Blockchain as a Software Connector“. In: *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. Venice, Italy: IEEE, S. 182–191. ISBN: 978-1-5090-2131-4. DOI: 10.1109/WICSA.2016.21. URL: http://ieeexplore.ieee.org/document/7516828/, Abruf am 25. 11. 2018.

Xu, Xiwei, Ingo Weber und Mark Staples (2019). *Architecture for Blockchain Applications*. Springer International Publishing. ISBN: 978-3-030-03034-6. URL: https://www.springer.com/gp/book/9783030030346, Abruf am 10. 03. 2019.

Xu, Xiwei, Ingo Weber, Mark Staples, Liming Zhu, Jan Bosch, Len Bass, Cesare Pautasso und Paul Rimba (2017). „A Taxonomy of Blockchain-Based Systems for Architecture Design“. In: *2017 IEEE International Conference on Software Architecture (ICSA)*. Gothenburg, Sweden: IEEE, S. 243–252. ISBN: 978-1-5090-5729-0. DOI: 10.1109/ICSA.2017.33. URL: http://ieeexplore.ieee.org/document/7930224/, Abruf am 23. 11. 2018.

Zamfir, Vlad (2015). *Ethereum Casper*. URL: https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/, Abruf am 25. 11. 2018.

Zyskind, Guy, Oz Nathan und Alex 'Sandy' Pentland (2015). „Decentralizing Privacy: Using Blockchain to Protect Personal Data“. In: *Security and Privacy Workshops (SPW) 2015, IEEE*, S. 180–184. URL: http://web.media.mit.edu/guyzys/data/ZNP15.pdf, Abruf am 14. 12. 2018.

University of Bamberg Press

Die Informationssysteme von dezentralen Organisationen sind heute als Komponenten eines globalen Informationssystems miteinander vernetzt. Organisationen, Individuen und Software agieren autonom und sind zugleich kooperativ an der Erstellung von Leistungen beteiligt. Die Planung und Ausführung der hierfür erforderlichen Interaktionen in Form von Prozessen stehen dabei nicht unter der Kontrolle Einzelner. Zur Entwicklung und Ausführung von Prozessen unter dezentraler Koordination schlägt diese Arbeit einen Ansatz auf der Basis von Blockchain-Technologien vor. Damit wird zum einen die Gestaltung dezentraler Organisationen adressiert und zum anderen die innerhalb von solchen Systemen verteilt und nicht-zentral gesteuerte Koordination der Entwicklung.

Die Arbeit beschreibt die Dezentralisierung und die modellbasierte Entwicklung von prozessorientierten Systemen zunächst aus organisationstheoretischer Sicht. Als Entwicklungs- und Ausführungsplattformen stehen anschließend dezentrale Blockchain-Systeme im Fokus, mit denen die Verbindlichkeit und Integrität von Transaktionen zwischen a priori unbekannten Knoten eines Netzes gewährleistet werden kann. Die Entwicklung und Ausführung von Prozessen werden auf dieser Basis durch einen modellbasierten Ansatz realisiert. Dieser wird durch einen Architekturrahmen zur konzeptuellen Modellierung, zur kooperativen Modellbildung und zur Nachverfolgung von Instanzen etabliert. Die konzeptuelle Modellierung von Netzwerken, Prozessen und Instanzen greift auf das Semantische Objektmodell (SOM) zurück, um die Aufgabenebene dezentraler Organisationen auf eine dezentral realisierte Aufgabenträgerebene abzubilden. Die kooperative Modellbildung und die Nachverfolgung von Instanzen nutzen Technologien zur verteilten Versionsverwaltung in Verbindung mit Smart Contracts des Blockchain-Systems Ethereum. Die Demonstration der Anwendbarkeit des Ansatzes anhand einer Fallstudie und einer Software-Implementierung schließen die Arbeit ab.

eISBN: 978-3-86309-683-0

9 783863 096830

www.uni-bamberg.de/ubp