



Xpert.press



Peter Göhner (Hrsg.)

Agentensysteme in der Automatisierungstechnik



Springer Vieweg

Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals
in den Bereichen Softwareentwicklung,
Internettechnologie und IT-Management aktuell
und kompetent relevantes Fachwissen über
Technologien und Produkte zur Entwicklung
und Anwendung moderner Informationstechnologien.

Peter Göhner
Herausgeber

Agentensysteme in der Automatisierungstechnik



Springer Vieweg

Herausgeber

Peter Göhner

Institut für Automatisierungs- und
Softwaretechnik, Universität Stuttgart,
Stuttgart, Deutschland

ISSN 1439-5428

ISBN 978-3-642-31767-5

DOI 10.1007/978-3-642-31768-2

ISBN 978-3-642-31768-2 (eBook)

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Springer Vieweg

© Springer-Verlag Berlin Heidelberg 2013

Dieses Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags.
Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Gedruckt auf säurefrei und chlorfrei gebleichtem Papier.

Springer Vieweg ist eine Marke von Springer DE. Springer DE ist Teil der Fachverlagsgruppe Springer Science+Business Media
www.springer-vieweg.de

Vorwort

Eine beständige Herausforderung für Ingenieure ist die Entwicklung von Software für moderne Automatisierungssysteme, die häufig im Spannungsfeld unterschiedlichster Anforderungen stattfindet. Dazu gehören die Notwendigkeit zu mehr Flexibilität, Robustheit, Skalierbarkeit, Anpassbarkeit und Integrationsfähigkeit der Automatisierungssoftware, einhergehend mit zunehmender Dezentralisierung und komplexeren Abläufen in Automatisierungssystemen.

Die Automatisierungssoftware muss flexibel und ohne großen Aufwand an Änderungen des Automatisierungssystems oder seiner Umgebung anpassbar sein, um diese Anforderungen zukunftssicher bewältigen zu können. Dabei besteht häufig die Schwierigkeit, die komplexen Abläufe und Verhaltensweisen eines solchen flexiblen Systems zu beherrschen. Hinzu kommt, dass zunehmend auch die Integration des Automatisierungssystems mit heterogenen Komponenten und externen Systemen bewerkstelligt werden muss. Um diesen Herausforderungen zu begegnen, bietet der Einsatz von Softwareagenten viel versprechende Möglichkeiten. Mit ihrer Hilfe kann der nächste Entwicklungsschritt in der Automatisierungstechnik vollzogen werden – weg von hierarchischen, statischen Systemen, hin zu flexiblen, dezentralen Netzwerken, die sich aus autonomen, kooperierenden Elementen zusammensetzen. Jedoch müssen die spezifischen Eigenschaften der Domäne berücksichtigt werden, um für agentenorientierte Software ein breites Anwendungsfeld in der Automatisierungstechnik zu erschließen. Dazu gehören Anforderungen an zeitliches Verhalten der Software, an Zuverlässigkeit, Sicherheit und Verfügbarkeit, aber auch Kommunikationsinfrastrukturen, Hardwareressourcen und nicht zuletzt das Zusammenspiel mit bestehenden Automatisierungsstrukturen.

Mit diesem Band wollen wir einen Beitrag leisten, das Paradigma der agentenorientierten Softwareentwicklung in der Automatisierungstechnik weiter zu verbreiten und so die Entwicklung der Automatisierungssysteme von Morgen zu unterstützen.

Stuttgart, Dezember 2012
Mit freundlichen Grüßen, Ihr
Prof. Dr.-Ing. Dr. h. c. P. Göhner

Inhaltsverzeichnis

Teil I Allgemein

1 Funktionaler Anwendungsentwurf für agentenbasierte, verteilte Automatisierungssysteme	3
Timo Frank, Daniel Schütz und Birgit Vogel-Heuser	
1.1 Einführung	3
1.1.1 Einführungsbeispiel	5
1.1.2 Vorarbeiten zu verteilten Automatisierungssystemen	5
1.1.3 Vorarbeiten zu Agenten in der Automatisierungstechnik	6
1.2 Notation für den Entwurf von verteilten Automatisierungssystemen	7
1.2.1 Sichten der Notation	7
1.2.2 Anforderungen	7
1.2.3 Software-Sicht	8
1.2.4 Hardware-Sicht	10
1.2.5 Sichten Kombination	11
1.3 Funktionaler Entwurf von Softwareagenten	12
1.3.1 Entwurf von Softwareagenten für Anlagenkomponenten	12
1.3.2 Entwurf funktional implementierter Agenten	13
1.3.3 Herausforderungen zukünftiger Architekturen	14
1.4 Evaluation anhand eines Fallbeispiels	15
1.5 Zusammenfassung	17
1.6 Zukünftige Forschungsziele und -schwerpunkte	18
Literatur	18
2 Energiesysteme und das Paradigma des Agenten	21
Andreas Beck, Christian Derksen, Sebastian Lehnhoff, Tobias Linnenberg, Astrid Nieße und Gregor Rohbogner	
2.1 Einleitung	21
2.2 Das elektrische Energiesystem heute	22
2.2.1 Elektrische Energieerzeuger und Speicher	22

2.2.2 Elektrische Netze	23
2.2.3 Elektrische Energieverbraucher	23
2.2.4 Lastermittlung und Netzsteuerung heute	24
2.2.5 Herausforderungen für das zukünftige Energiesystem	25
2.3 Typische Betriebsführungskonzepte und -strukturen in der wissenschaftlichen Literatur	26
2.3.1 Zentral-hierarchische Führungsstruktur	27
2.3.2 Dezentral-hierarchische Führungsstruktur	28
2.3.3 Dezentrale Führungsstruktur	28
2.3.4 Kritische Auseinandersetzung	29
2.4 Spezielle Anforderungen an die Verteilnetzautomatisierung	30
2.4.1 Smart Grid Literatur Architektur	31
2.4.2 Anforderungen an zukünftige Netzteilsysteme	31
2.5 Agenten und Agentenbasierte Betriebsführung	33
2.5.1 Agenten und ihr Einsatz in Smart Grids	33
2.5.2 Agentenbasierte Führungssysteme in der elektrischen Energieversorgung	35
2.5.3 Evaluation agentenbasierter Betriebsführungskonzepte	37
2.6 Forschungsaktivitäten, -fragen und -bedarf	39
2.7 Zusammenfassung	40
Literatur	41

Teil II Konzepte und Methoden

3 Identifikation und Umsetzung von Agenten zur Fabrikautomation unter Nutzung von mechatronischen Strukturierungskonzepten	45
Arndt Lüder und Matthias Foehr	
3.1 Einleitung	45
3.2 Ausgangslage	47
3.2.1 Agentensysteme zur Steuerung von Produktionssystemen	47
3.2.2 Mechatronischer Entwurf von Produktionssystemen	50
3.3 Mechatronikorientierte Agentensysteme	53
3.3.1 Struktur mechatronikorientierter Agenten	54
3.3.2 Entwurfsprozess für mechatronikorientierte Agentensysteme	55
3.3.3 Vorteile der Nutzung mechatronikorientierter Agentensysteme	56
3.4 Zusammenfassung	58
Literatur	59
4 Entwicklungsmethoden für agentenbasierte Systeme	63
Bernhard Bauer	
4.1 Einleitung	63

4.2 Überblick über Entwicklungsmethoden für agentenbasierte Systeme	65
4.2.1 Objektorientierte Ansätze	65
4.2.2 Wissensbasierte Ansätze	68
4.2.3 Agentenbasierte Ansätze	68
4.2.4 Überblick	72
4.3 Vereinheitlichung der Ansätze	72
4.3.1 Vereinheitlichung der Ansätze	72
4.3.2 Prozesslinienansatz für Agenten	74
4.4 Zusammenfassung und Ausblick	75
Literatur	76
5 Middleware für Systeme mobiler Agenten in der Automatisierung	79
Hagen Stanek	
5.1 Einleitung	79
5.2 Grundlegende Anforderungen an eine Middleware	81
5.2.1 Middleware und verteilte Anwendungen	81
5.2.2 Definition mobiler Agenten	81
5.2.3 Bedingungen der Automatisierung	82
5.2.4 Virtuelle Maschinen	83
5.2.5 Zusammenfassung der grundlegenden Anforderungen an eine Middleware	84
5.3 Middleware	84
5.3.1 Existierende Middleware	84
5.3.2 Reduzierung der Middleware-Liste	85
5.4 Zusammenfassung	93
Literatur	94
6 Agentenmodelle in der Anlagenautomation	95
Ulrich Epple	
6.1 Einleitung	95
6.2 Konzept eines Agentensystems	97
6.2.1 Technische Agenten auf Funktionsbausteinbasis	97
6.2.2 Dienstschnittstelle	99
6.2.3 Auftragsschnittstelle	100
6.2.4 Eingangsschnittstelle	101
6.3 Agenten der Anlagenautomatisierung	101
6.3.1 Agenten für die Anlagensteuerung	102
6.3.2 Agenten zur operativen Ausführung von Maßnahmen	103
6.3.3 Agenten zur Ankopplung der Systemdienste	104
6.3.4 Agenten zur Verwaltung von technischen Einrichtungen	104
6.3.5 Agenten zur Verwaltung von Produkten	105
6.3.6 Agenten zur Verwaltung von Modellen	105
6.3.7 Agenten zur Verwaltung von Zusatzfunktionen	105
6.3.8 Agenten zur Unterstützung von Assistenzsystemen	106

6.4 Zusammenfassung	108
Literatur	109
Teil III Agenten im operativen Einsatz	
7 Einsatz von Agentensystemen in der Intralogistik	113
Wilfried Kugler und Dirk Gehlich	
7.1 Einleitung	113
7.2 Ausgangssituation	114
7.3 Konzept	115
7.4 Das agentengesteuerte Kommissioniersystem	117
7.4.1 Aufbau des RFID-Systems	118
7.4.2 Konzept einer agentenbasierten Steuerung	119
7.4.3 Platz-zu-Platz-Steuerung	121
7.4.4 Steuerung der Auftragsreihenfolge	123
7.4.5 Integration der Steuerung	125
7.5 Zusammenfassung	126
Literatur	128
8 Von Softwareagenten zu Cyber-Physical Systems:	
Technologien und Anwendungen	129
Thorsten Schöler, Christian Ego, Johannes Leimer und Rico Lieback	
8.1 Einleitung	129
8.2 Ausgangssituation	130
8.2.1 Dezentralisierung des Energiemarktes	130
8.2.2 Dezentrale Koordination und Steuerung von dynamischen Logistikketten	131
8.2.3 Logik-basierte Fehlererkennung und -analyse	131
8.2.4 Digitales Produktgedächtnis	132
8.2.5 Systemmanagement für Industrieanlagen und Medizingeräte	132
8.2.6 Zusammenfassung	133
8.3 Konzept, Lösung	134
8.3.1 Einordnung der Technologien	134
8.3.2 Objekt-funktionale Plattform für Cyber-Physical Systems .	136
8.3.3 Integration wissensbasierter Komponenten	140
8.4 Evaluierung, Erfahrung mit der Benutzung	142
8.4.1 Leistungsevaluation	142
8.4.2 Typische Einsatzmöglichkeiten	144
8.4.3 Anwendung im Energiemanagement	144
8.5 Zusammenfassung	147
Literatur	148

9	Freie Fahrt – Softwareagenten reduzieren Schadstoffausstoß	151
	Christian Dannegger	
9.1	Einleitung – Zuviel Lärm, zu viel Schadstoffe, zu viel Stillstand	151
9.2	Ausgangssituation/Problemstellung	153
9.3	Stand der Technik	156
9.4	Konzept/Lösung	157
9.4.1	Welches Agentenmodell macht Sinn?	158
9.4.2	Wie die Agenten arbeiten	159
9.4.3	System-Komponenten	159
9.4.4	In drei Schritten zum Ergebnis	160
9.5	Evaluierung/Erfahrung und Nutzen	161
9.6	Zusammenfassung	167
10	Einsatz von Softwareagenten am Beispiel einer kontinuierlichen, hydraulischen Heizpresse	169
	Andreas Wannagat, Daniel Schütz und Birgit Vogel-Heuser	
10.1	Einleitung	169
10.2	Struktur des Agentensystems	171
10.2.1	Prozessagent	173
10.2.2	Steuerungsagenten	174
10.3	Realisierung und Umsetzung	177
10.3.1	Simulation der Presse	177
10.3.2	Implementierung des Agentensystems	179
10.4	Evaluation durch Messungen im Betrieb	183
10.5	Zusammenfassung und Ausblick	184
	Literatur	185
11	Einsatz von Agentensystemen zur Optimierung der Logistik in Produktions- und Agrarprozessen	187
	Holger Kremer und Clemens Westerkamp	
11.1	Einleitung	187
11.2	Ausgangssituation/Problemstellung/Stand der Technik	188
11.2.1	Logistikunterstützung in der Fabrikautomatisierung	188
11.2.2	Logistikunterstützung im landwirtschaftlichen Bereich	188
11.2.3	Problemstellung Fabrikautomatisierung	189
11.2.4	Problemstellung Agrarlogistik	189
11.2.5	Stand der Technik zur Agenten-Orientierten Programmierung	191
11.3	Verteilte Systemarchitektur für Fabrik- und Agraranwendungen	191
11.3.1	Architektur für den Einsatz in der Fabrikautomatisierung	191
11.3.2	Architektur für den Einsatz in der Agrarlogistik (KOMOBAR)	192
11.3.3	Konzeption für Erweiterungen der JADE-Agenten-Plattform	196

11.3.4 Konzept für FIPA konforme, webbasierte Software-Agenten	197
11.4 Ergebnisse	198
11.4.1 Erprobung in der Fabrikautomatisierung	198
11.4.2 Erprobung in der Agrarlogistik	199
11.4.3 JADEAndroid-Prototyp und erste Testergebnisse	202
11.5 Zusammenfassung und Ausblick	203
Literatur	204
12 Agentenbasierte Verteilnetzautomatisierung	207
Sebastian Lehnhoff und Olav Krause	
12.1 Das elektrische Energieversorgungssystem	208
12.1.1 Anforderungen an die Verteilnetzautomatisierung	209
12.1.2 Anforderungen an zukünftige Leitsysteme	210
12.2 Netznutzungskoordination in Verteilnetzen	210
12.2.1 Agentenbasierte Netznutzungskoordination	211
12.3 Netzzustandsberechnung	212
12.3.1 Bestimmung der Räume zur Spannungshaltung	212
12.3.2 Bestimmung der Räume zur Vermeidung zu hoher Leitungs- und Transformatorströme	215
12.3.3 Zusammenführen der Teilergebnisse	216
12.3.4 Klassifizierung der Menge zulässiger Betriebspunkte über Support Vector Machines	219
12.4 Zusammenfassung	221
Literatur	222
13 Agentenbasiertes Selbstmanagement von Automatisierungssystemen	225
Hisham K. Mubarak	
13.1 Einleitung	225
13.2 Selbstmanagement zur Unterstützung der technischen Betriebsbetreuung von Automatisierungssystemen	227
13.2.1 Technische Betriebsbetreuungsaufgaben in Automatisierungssystemen	227
13.2.2 Anforderungen an ein Selbstmanagementsystem in der Automatisierungstechnik	227
13.3 Agentenbasiertes Selbstmanagementsystem für Automatisierungssysteme	230
13.3.1 Agentenorientierte Architektur eines Selbstmanagement-Systems	230
13.3.2 Agenten zur Kontrolle und Überwachung von Selbstmanagement-Funktionalitäten	231
13.3.3 Agenten zur Bereitstellung von Selbstmanagement-Funktionalitäten	233
13.3.4 Agenten zur Anbindung des Automatisierungssystems . .	234

13.4 Anwendungsbeispiel: Selbstheilende Personenaufzugssteuerung	235
13.5 Zusammenfassung	236
Literatur	238
 Teil IV Agentenbasierte Assistenzsysteme	
14 Ready for Take-Off –	
Softwareagenten disponieren Flugcharter-Verkehr	243
Christian Dannegger	
14.1 Einleitung	243
14.2 Ausgangssituation/Problemstellung/Stand der Technik	244
14.3 Konzept/Lösung	247
14.4 Evaluierung/Erfahrung und Nutzen	253
Literatur	254
15 Agentenbasierte Koordination und Überwachung des Designs mechatronischer Systeme	255
Holger Seemüller, Mohammad Chami und Holger Voos	
15.1 Einleitung	255
15.2 Ausgangssituation	257
15.2.1 Modellierung von Entwicklungsprozessen	258
15.2.2 Systemmodellierung	259
15.3 Agentenbasiertes Engineering	259
15.3.1 Ziele	260
15.3.2 Anforderungsanalyse	263
15.3.3 Architektur des Agentensystems	264
15.3.4 Laufzeitkonfiguration des Agentensystems	267
15.4 Anwendungsbeispiel	268
15.4.1 Iterative Entwicklung der Modelle	268
15.5 Zusammenfassung	271
Literatur	272
16 Agentenbasierte dynamische Testfallpriorisierung	275
Christoph Malz	
16.1 Einleitung	275
16.2 Aktuelle Situation und Problematik	
bei der Priorisierung von Testfällen	276
16.3 Grundidee der dynamischen Testfallpriorisierung	277
16.3.1 Informationen zur Testfallpriorisierung aus der Systementwicklung	278
16.3.2 Informationen zur Testfallpriorisierung aus dem Systemtest und dem Systembetrieb	280
16.3.3 Agentenbasierter Ansatz für das dynamische Testfallpriorisierungssystem	282

16.4 Agentenbasierte Testfallpriorisierung	283
16.4.1 Bestimmung der Testwichtigkeit eines Softwaremoduls durch den Softwaremodul-Agent	283
16.4.2 Bestimmung der lokalen Prioritäten und der globalen Priorität eines Testfalls durch den Testfall-Agent	285
16.5 Realisierung und Evaluierung des agentenbasierten dynamischen Testfallpriorisierungssystems	288
16.6 Zusammenfassung	289
Literatur	290
17 Werkzeugunterstützung für die Entwicklung von SPS-basierten Softwareagenten zur Erhöhung der Verfügbarkeit	291
Daniel Schütz und Birgit Vogel-Heuser	
17.1 Einleitung	291
17.2 Erhebung der Anforderungen	293
17.3 Entwicklung von Modellierungssprache und Werkzeug	296
17.3.1 Entwicklung der Modellierungssprache auf Basis der SysML	296
17.3.2 Automatische Übertragung der Modelle in die IEC 61131-3	298
17.4 Evaluation der Projektergebnisse	299
17.5 Zusammenfassung und Ausblick	301
Literatur	302

Teil I

Allgemein

Kapitel 1

Funktionaler Anwendungsentwurf für agentenbasierte, verteilte Automatisierungssysteme

Timo Frank, Daniel Schütz und Birgit Vogel-Heuser

Zusammenfassung Um moderne, komplexe Automatisierungsarchitekturen in der Industrie anwendbar zu machen, fehlen derzeit noch Methoden, die eine Anwendung erleichtern. Das Ziel der hier vorgestellten Arbeit ist es, den Entwickler eines verteilten agentenbasierten Automatisierungssystems dabei zu unterstützen, dieses System unter Berücksichtigung von funktionalen und nicht-funktionalen Anforderungen zu entwerfen. Der Ansatz basiert auf einer auf SysML basierten Notation für eine gezielte Unterstützung des Entwicklers durch eine angepasstes Beschreibungsmittel. Außerdem wird eine Architektur für verteilte Agentensysteme vorgestellt. In diesem Beitrag wird der Entwurf eines Agentensystems mit dem Beschreibungsmittel anhand eines Beispiels dargestellt.

1.1 Einführung

Moderne Ansätze zur Modularisierung im Anlagenbau führen zu einer immer höheren Flexibilität in der Elektrik sowie der Mechanik. Für einen durchgängigen Modulansatz muss die Steuerungstechnik diesen Anforderungen ebenfalls gerecht werden [1]. Unterschiedliche Ansätze wurden entwickelt, um die hohen Flexibilitäts- und Modularitätsanforderungen zu erfüllen. Flexible Manufacturing Systems (FMS) [2] sowie unterschiedliche Software-Agenten-Konzepte [3–6] sind hierfür beispielhafte mögliche Ansätze. Obwohl sich diese und andere Methoden zum Erreichen der gewünschten Modularität und Flexibilität stark in ihren Ansätzen unterscheiden, versuchen sie doch alle, dies mit einer verteilten Steuerungsintelligenz zu erreichen. Die Verteilung erfolgt dabei auf unterschiedliche Steuerungen oder sogenannte „intelligente Geräte“ (im Folgenden als „Knoten“ bezeichnet). Die benötigte größtmögliche Steuerungsautonomie sowie die dennoch benötigte

Timo Frank (✉), Daniel Schütz, Birgit Vogel-Heuser
Automatisierungstechnik und Informationssysteme, Technische Universität München,
Bolzmannstraße 15, 85748 Garching, Deutschland
e-mail: frank@ais.mw.tum.de, schuetz@ais.mw.tum.de, vogel-heuser@ais.mw.tum.de

Kooperation zwischen diesen Knoten führen zu einem erhöhten Kommunikationsbedarf. Zusätzlich kommt es bei räumlich verteilten Fertigungssystemen zu einer logischen Verteilung der Steuerungsintelligenz.

Agentenbasierte Konzepte, die die Flexibilität von Produktionsanlagen durch die Kompensation von Sensorsausfällen erhöhen, wurden in eigenen Vorarbeiten entwickelt [6]. Diese Konzepte verwenden virtuelle Sensoren implementiert in echtzeitfähigen Softwareagenten, die auf üblichen speicherprogrammierbaren Steuerungen (SPS) unter den Beschränkungen hinsichtlich des Speicherbedarfs und der Rechengeschwindigkeit ausgeführt werden können.

Über die in diesen Vorarbeiten vorgeschlagene Architektur hinaus, würde eine Implementierung, in der die Softwareagenten als Repräsentationen für Anlagenfunktionen (z. B. Aufheizen einer Flüssigkeit) im Gegensatz zur Repräsentation von Anlagenkomponenten (z. B. Tank/Reaktor-System) die Flexibilität und Wandelbarkeit eines Produktionssystems durch die Entkopplung der Softwarestruktur vom technischen Aufbau einer Anlage weiter erhöhen.

Aus der Sicht des Systementwicklers besteht die Herausforderung bei verteilten Automatisierungssystemen darin, dass eine Verteilung (Deployment) von Automatisierungsfunktionen auf unterschiedliche Knoten nötig ist, um die geforderten funktionalen Anforderungen unter Berücksichtigung der nicht-funktionalen Anforderungen (NFA) zu erfüllen. Als NFA zählen beispielsweise Echtzeitanforderungen, Interoperabilität, Modularität, Fehlertoleranz und Wiederverwendbarkeit (ISO25010). Hinzu kommt, dass der Entwickler die Einschränkungen der eingesetzten Knoten, zum Beispiel hinsichtlich Speicher oder Rechenleistung, bei der Entwicklung des Gesamtsystems berücksichtigen muss.

Forschungsprojekte im Bereich FMS oder Multiagentensysteme unterstützen den Entwickler üblicherweise nicht bei der Aufgabe, ein verteiltes System unter Berücksichtigung der funktionalen und nicht-funktionalen Anforderungen sowie der Hardwareeinschränkungen zu entwerfen, oder berücksichtigen nur einzelne NFAs, z. B. Zeitanforderungen. Im Gegensatz zu den vorgenannten Ansätzen wird in der hier dargestellten Forschungsarbeit ein Engineering-Ansatz vorgestellt, welcher den Entwickler von den Anforderungen über den Entwurf bis zur Verteilung (Deployment) von verteilten agentenbasierten Automatisierungssystemen unterstützt.

Die Konzepte aus eigenen Vorarbeiten verlangen, dass das Wissen über die analytischen Zusammenhänge zwischen Sensor- und Aktorwerten einer Komponente vollständig in einem einzelnen Agenten für diese Komponente implementiert wird. In einer Architektur, in der Softwareagenten zur Erfüllung einer Anlagenfunktion anstatt zur Steuerung einer Anlagenkomponente implementiert werden, sind Sensoren und Aktoren und damit auch das Wissen bezüglich der Möglichkeiten zur Implementierung von virtuellen Sensoren aufgeteilt auf mehrere Agenten. Zur Erkennung und Kompensation von Sensorsausfällen müssen daher über das Konzept der Vorarbeiten hinaus Mechanismen zur Beschreibung verteilten Wissens im Entwurf und gemeinsamen Auswertung dieses verteilten Wissens durch mehrere kooperierende Softwareagenten eines Agentensystems realisiert werden. Der Weiteren muss das Verhalten eines Agentensystems, in dem mehrere Agenten gemeinsam auf

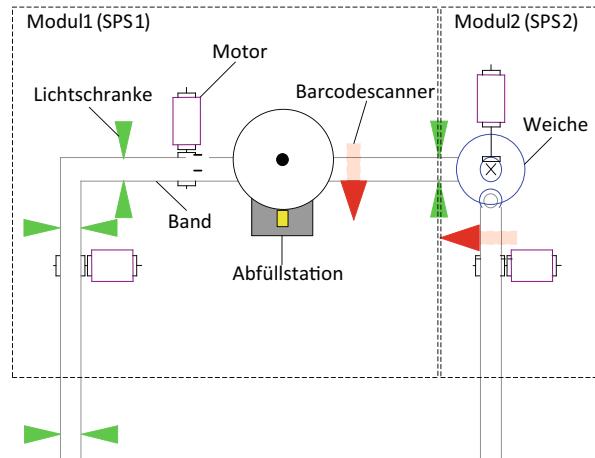


Abb. 1.1 Weiche mit Bändern als Beispiel für ein verteiltes Steuerungssystem

Grundlage ihres jeweiligen lokalen Wissens handeln, durch Mechanismen zur Verifikation abgesichert werden.

1.1.1 Einführungsbeispiel

Als Beispiel für ein verteiltes Automatisierungssystem wird in diesem Beitrag eine Weiche mit zwei angeschlossenen Bändern genutzt. In Abb. 1.1 sind alle Komponenten dargestellt. Die Weiche und das zuführende Band werden dabei durch SPS 2 gesteuert, das abführende Band sowie die Abfüllstation sind durch SPS 1 automatisiert.

Eine zugeführte Flasche wird durch den Barcodescanner erkannt und durch die Weiche auf das Zielband geleitet. Eine ankommende Flasche wird durch eine Lichtschranke in der Weiche erkannt. Hat die Weiche gedreht muss das Abführband eingeschaltet werden und nachdem die Flasche die Weiche verlassen hat muss die Weiche in die Ausgangsposition gedreht werden. Die Flasche wird anschließend befüllt und über das Abfuhrband abtransportiert.

1.1.2 Vorarbeiten zu verteilten Automatisierungssystemen

Um den Engineering Prozess von verteilten Automatisierungssystemen zu optimieren wurde das Projekt FAVA (Funktionaler Anwendungsentwurf für verteilte Automatisierungssysteme) gestartet. Das Ziel war dabei den Entwickler gezielt bei der Erfüllung von Anforderungen, besonders von nicht-funktionalen Anforde-

rungen zu unterstützen [7, 8]. Dafür wurden typische Automatisierungsfunktionen ermittelt [9] und auf deren Basis ein Kommunikationsmodell [10] sowie ein Vorgehensmodell [11] für den Entwurf von verteilten Automatisierungssystemen entworfen [12]. Auf diesen Arbeiten aufbauend ist die in diesem Beitrag vorgestellte Notation entstanden. Alle Aspekte des FAVA Projekts werden mit 60 Probanden (Studenten) evaluiert werden.

1.1.3 Vorarbeiten zu Agenten in der Automatisierungstechnik

In zukünftigen Produktionssystemen werden dezentrale, verteilte Automatisierungskomponenten (neben Steuerungen selbst auch Sensorik und Aktorik) für eine Vielzahl von Anlagenfunktionen genutzt werden, um den Anforderungen an Flexibilität und Veränderbarkeit gerecht zu werden. Darüber hinaus existieren hohe Anforderungen in Bezug auf die Verfügbarkeit von Produktionsanlagen in der Fertigungstechnik sowie in der Verfahrenstechnik.

Konzepte die Verfügbarkeit von Produktionsanlagen durch die Kompensation von Fehlerfällen und insbesondere Sensorausfällen auf Grundlage von Softwareagenten zu erhöhen wurden in Vorarbeiten der Autoren entwickelt und anhand mehrerer Anwendungsfälle positiv hinsichtlich der Kompensation von Sensorausfällen evaluiert. Das Konzept nutzt virtuelle Sensoren implementiert innerhalb von Softwareagenten, die auf handelsüblichen speicherprogrammierbaren Steuerungen (SPS) lauffähig sind und damit Sensorausfälle innerhalb der Echtzeitbedingungen einer Produktionsanlage kompensieren können.

Dazu wurde innerhalb des von der Deutschen Forschungsgemeinschaft (DFG) geförderten Forschungsprojekts AVE die Anwendbarkeit agentenorientierter Software für verteilte, echtzeitfähige Automatisierungssysteme (SPSen) untersucht und prototypische Implementierungen realisiert. Basierend auf durch den Entwickler eines Automatisierungssystems modellierten Wissens über analytische Zusammenhänge zwischen Sensor- und Aktorwerten konnten innerhalb der Agenten Bausteine und Mechanismen für virtuelle Sensoren in den Sprachen der IEC 61131-3 implementiert werden. Die Untersuchung verschiedene Anwendungsfälle [6, 13, 14] zeigte, dass die Softwareagenten in der Lage sind, den Betrieb einer Anlage auch bei Sensorausfällen aufrecht zu erhalten, die mit klassischen Implementierungen zu Stillstandzeiten und/oder gefährlichen Situationen geführt hätten.

Um die Anwendbarkeit des Ansatzes zu unterstützen und dessen Usability zu untersuchen wurde im Rahmen des DFG-Transferprojekts KREAagentuse eine modellbasierte Werkzeugunterstützung basierend auf den Notationen der Systems Modeling Language (SysML) realisiert und mithilfe von Probandenversuchen evaluiert, die es erlaubt die Agentenmodelle modellbasiert zu entwickeln und automatisch in ablauffähigen IEC 61131-3 Code zu übersetzen. Die empirischen Untersuchungen mit Maschinenbau- und Elektrotechnikingenieuren zeigten, dass die eingesetzten Notationen insofern eine bessere Usability besitzen, als dass sie einfacher verständlich sind, als die klassischen Sprachen der IEC 61131-3.

1.2 Notation für den Entwurf von verteilten Automatisierungssystemen

Der Entwurf von verteilten Automatisierungssystemen stellt aufgrund der Anlagenkomplexität eine große Herausforderung dar. In diesem Beitrag wird ein Beschreibungsmittel (Notation) vorgestellt, welches für den Entwurf von verteilten Automatisierungssystemen entwickelt wurde. Die vorgestellte Notation basiert auf der SysML.

1.2.1 Sichten der Notation

Die Notation besteht aus drei unabhängigen Sichten und einer kombinierten Sicht. In dieser sind nur drei dieser vier Sichten spezifiziert. Die Prozess-Sicht ist zum Beispiel ein R&I (engl. P&ID) Fließbild oder eine beliebige andere Prozessdarstellung. Die vier Sichten der Notation sind:

- Prozess-Sicht: nicht Teil der Notation
- Software-Sicht
- Hardware-Sicht
- Deployment-Sicht

1.2.2 Anforderungen

Die Darstellung von Anforderungen in der SysML für verteilte AT-Systeme basiert auf dem SysML Anforderungsdiagramm. Eine Anforderung wird über das Notationselement <<Anforderung>> dargestellt. Der Typ der Anforderung wird als Anforderungstyp eingetragen. Der Name, die Beschreibung sowie eine frei wählbare eindeutige ID können in die vorgesehenen Felder eingetragen werden.

Die Gültigkeit der Anforderung (siehe Abb. 1.2) wird über eine gestrichelte Linie von der Anforderung bis zu dem Punkt der Anforderungsgültigkeit gezogen. Von einer Anforderung können mehrere Gültigkeitslinien gezogen werden. Eine Anforderungsgültigkeit ist eine Beziehung zwischen einer <<Anforderung>> und einem beliebig anderem Notationselement. Die Gültigkeit einer Anforderung entspricht hierbei der Abgrenzung des Lösungsraums in dem eine spätere Lösung entwickelt werden muss. Ist eine <<Anforderung>> mit nur einem Element verbunden muss dieses Element/Modul die Anforderung erfüllen. Hat eine <<Anforderung>> mehrere Verbindungen muss zwischen diesen Verbindungen die <<Anforderung>> erfüllt werden (Lösungsraum). Anforderungen werden zu Beginn des Engineerings in die Prozess-Sicht eingetragen und können so über den gesamten Entwurfsprozess berücksichtigt werden.

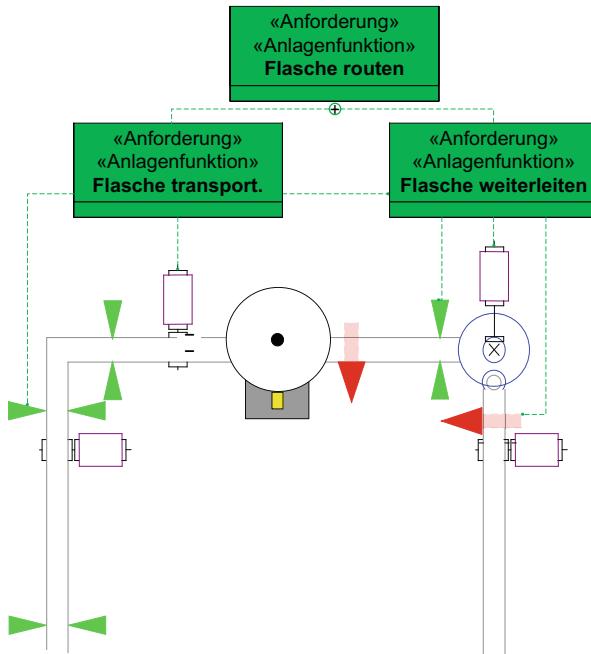


Abb. 1.2 Beispielimplementierung des Notationselement Anforderung vom Typ Anlagenfunktion im Anlagenlayout

Handelt es sich bei der zu entwerfenden Anlage um eine Fertigungstechnische Anlage basiert der Entwurf beispielsweise auf einem Anlagenlayout (Abb. 1.2). In dem gezeigten Beispiel ist die <<Anlagenfunktion>> verfeinert in die <<Anlagenfunktion>> Flasche transportieren und weiterleiten. Für die Erfüllung dieser <<Anforderung>> werden zwei Motoren und drei Lichtschranken benötigt. Die <<Anlagenfunktion>> Flasche transportieren benötigt, wie im Beispiel dargestellt, für den späteren Betrieb ein Startsignal wenn die Weiche fertig gedreht hat. Der Ablauf kann beispielsweise aus einem Aktivitätsdiagramm entnommen werden.

Eine <<Anforderung>> vom Typ <<nicht-funktionale Anforderung>> beschreibt Anforderungen, welche nicht die Funktion der Anlage betreffen. Dieser Anforderungstyp definiert ebenfalls Randbedingungen, die das System später erfüllen muss und wird mit demselben Element beschrieben.

1.2.3 Software-Sicht

Die Software-Sicht bildet das spätere funktionale Verhalten der Anlage ab und unterteilt sich in zwei für unterschiedliche Entwicklungsschritte vorgesehene Nota-

Abb. 1.3 Notationselement
Funktion



Abb. 1.4 Verbindungen
zwischen Funktionen

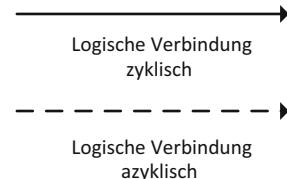


Abb. 1.5 Notationselement
Funktionsblock



tionselemente. Für die frühen Entwicklungsphasen dient die automatisierungstechnische Funktion um eine funktionale Struktur und logische Zusammenhänge und Abläufe zu definieren. Das Element Funktionsblock beschreibt das Verhalten und die genauen Schnittstellen. Im Folgenden sind beide dargestellt.

Automatisierungstechnische Funktionen (AT-Funktion)

Die Funktionalität der Anlage wird auf Basis der Anforderungen in der Software-Sicht entworfen. Eine automatisierungstechnische Funktion wird durch das Notationselement <<Funktion>> dargestellt (Abb. 1.3).

Abbildung 1.4 zeigt zwei unterschiedliche Verbindungen zwischen Funktionen. Die zyklische und die azyklische Verbindung. Die zyklische Verbindung zeigt den späteren Austausch von Messwerten oder Parametern an und führt zu einer parallelen Ausführung von Funktionen. Die azyklische Verbindung hingegen bietet die Möglichkeit logische Abfolgen von Funktionen oder das Starten anderer Funktionen abzubilden.

Funktionsbausteine

Eine <<Funktion>> wird durch einen oder mehrere Funktionsbausteine (FB) implementiert. Das Notationselement <<FB>> (Abb. 1.5) basiert auf dem Element <<Funktion>>. Ein <<FB>> repräsentiert einen späteren Funktionsblock einer SPS und beinhaltet das spätere Verhalten z. B. als Verhaltensmodell der SysML.

Abb. 1.6 Notationselement
Port und Verbindung

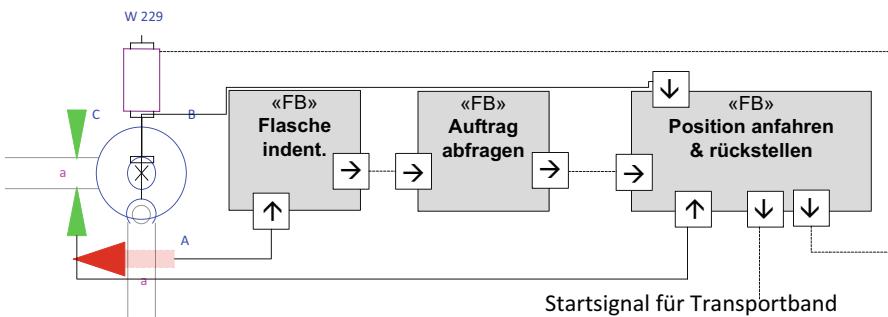
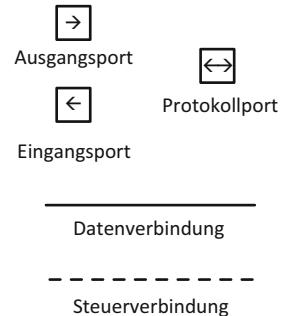


Abb. 1.7 Kombination der Notationselemente Funktionsblock, Port und Verbindung

Beispielhaft ist in Abb. 1.7 die <<Funktion>> Flasche identifizieren in zwei Funktionsblöcke unterteilt und die <<Funktion>> Position anfahren durch einen Funktionsblock.

Ein Port (vgl. Abb. 1.6) definiert ein Interface dieses <<FB>>. Ein <<Port>> repräsentiert eine oder mehrere Variablen/Parameter. Ports untereinander können über eine Steuer-/Datenverbindung miteinander verbunden werden. Der Protokollport repräsentiert ein Interface, welches später ein bestimmtes Protokollverhalten besitzt (z. B. Aufrufen einer Funktion mit Rückgabewert). Wie in Abb. 1.7 dargestellt, wird nach dem Ausführen des <<FB>> „Flasche identifizieren“ die <<FB>> „Auftrag abfragen“ ausgeführt. Außerdem werden die drei Geräte (Lichtschranke, Barcodescanner und Motor) durch die Funktionsblöcke gestartet/genutzt.

1.2.4 Hardware-Sicht

Die automatisierungstechnischen Sensoren und Aktoren einer Anlage werden über das in Abb. 1.8 gezeigte Notationselement <<Knoten>> abgebildet. Ein <<Knoten>> kann dann über den <<Knotentyp>> genauer spezifiziert werden.

Abb. 1.8 Notationselement
Knoten

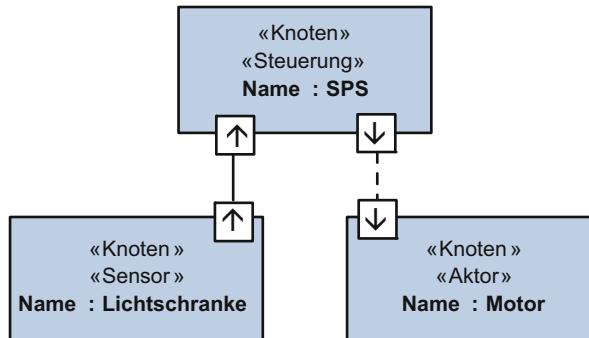


Abb. 1.9 Kombination der Notationselemente Knoten, Port und Verbindung

Knotentypen sind zum Beispiel *<<Sensoren>>*, *<<Aktoren>>* oder auch *<<Steuerungen>>*. Neben dem Namen kann zusätzlich der Gerätetyp angegeben werden, der die Knotentypen weiter unterteilt. Wie bei dem Notationselement *<<Funktion >>* kann an ein Notationselement *<<Knoten>>* beliebig viele Ein-/Ausgangsports angehängt und über Verbindungen in Beziehung gesetzt werden. In Abb. 1.9 sind alle zur Erfüllung der *<<Anlagenfunktion>>* „Flasche transportieren“ nötigen Sensoren und Aktoren dargestellt. Die SPS bekommt den aktuellen Wert der Lichtschranke und steuert den Motor an.

1.2.5 Sichten Kombination

Die unterschiedlichen Sichten (und damit die einzelnen Notationselemente der einzelnen Sichten) können beliebig miteinander kombiniert werden. So können bei kombinierter Software und Hardware *<<Knoten>>*-Ports mit *<<Funktion >>*-Ports verbunden werden. Welche Sichten wann miteinander „sinnvoll“ kombiniert werden können, ergibt sich aus der aktuellen Engineering-Phase mit den entsprechenden Ergebnissen. Je weiter die Entwicklung voranschreitet desto mehr Elemente der Notation werden miteinander kombiniert.

Das Deploymentmodell ist das finale Ergebnis der Konzeptphase. In ihm sind alle Informationen für die Implementierung enthalten. Das Deploymentmodell beinhaltet die Elemente *<<Knoten>>*, *<<FB>>*, Port, Verbindung, Anforderung. Im Deploymentmodell ist festgehalten welcher Funktionsbaustein auf welchem Knoten ausgeführt wird.

1.3 Funktionaler Entwurf von Softwareagenten

Die Konzepte aus eigenen Vorarbeiten zur Implementierung von Softwareagenten basieren auf einer Architektur, in der ein Agentensystem auf mehrere Steuerungen verteilt werden kann, einzelne Agenten, eingesetzt zur Steuerung einer physikalischen Anlagenkomponente, jedoch vollständig auf einer einzelnen Steuerung implementiert werden. Dem entgegen stehen die Bestrebungen hin zu funktionalen Softwarearchitekturen, in denen die implementierten Softwareentitäten als Repräsentationen von in einem Produktionsprozess verwendeten Anlagenfunktionen nicht deckungsgleich mit der Modularisierung der physikalischen Produktionsanlage oder den eingesetzten verteilten Steuerungen sind. Eine Architektur, in der Softwareagenten entkoppelt von physikalischen Modulen für zu realisierende Anlagenfunktionen implementiert werden würde die durch den Ansatz erreichte Flexibilisierung stark erhöhen. Dies zu erreichen müssen jedoch insbesondere die Modellierung und Implementierung der Wissensbasis der Softwareagenten angepasst werden.

In den folgenden Unterabschnitten wird zunächst auf die Entwicklung der Wissensbasis von Softwareagenten eingegangen, die sich für gekapselte physikalische Anlagenmodule implementiert werden. Anschließend wird die Modellierung der Wissensbasis von funktionsorientierten Agenten vorgestellt und Möglichkeiten für eine Implementierung vorgeschlagen. Abschließend werden Herausforderungen bei der Umsetzung des vorgestellten Konzepts vorgestellt.

1.3.1 Entwurf von Softwareagenten für Anlagenkomponenten

Zur Beschreibung der Wissensbasis von Softwareagenten wurde in eigenen Vorarbeiten ein gerichteter Graph vorgeschlagen, der die analytischen Zusammenhänge zwischen Sensor- und Aktorwerten beschreibt (Abb. 1.10, links). In diesem Redundanzmodell repräsentiert jeder Knoten eine Messstelle, die mit einem realen Sensor belegt ist. Ein Qualitätswert an jedem Knoten gibt die Präzision des gemessenen Wertes in Form von dessen Standardabweichung an. Die Kanten des Graphen beschreiben analytische Abhängigkeiten zwischen den Messstellen und werden als Grundlage zur Implementierung von virtuellen Sensoren verwendet.

Für jeden Softwareagenten der Steuerungssoftware eines Produktionssystems kann ein solches Redundanzmodell erstellt werden, das sämtliche Sensoren und Aktoren sowie die bestehenden analytischen Abhängigkeiten der durch den Agenten gesteuerten Komponente erfasst. Das so beschriebene Redundanzmodell kann in eine Matrix von Softwarebausteinen für reale und virtuelle Sensoren übersetzt und innerhalb eines Agenten für die jeweilige Anlagenkomponente implementiert werden. Das damit Implementierte Wissen und die eindeutige Zuordnung von realen Sensoren zu einzelnen Agenten ermöglicht es, dass Defekte von Sensoren durch einen einzelnen Agenten erkannt und kompensiert werden können.

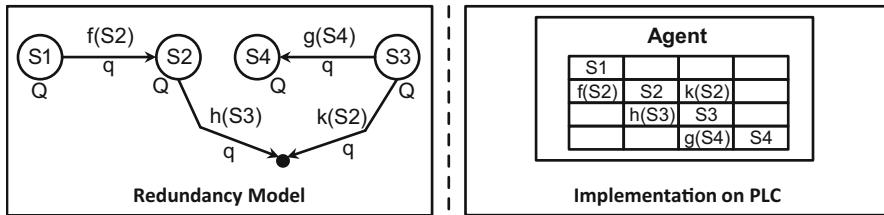


Abb. 1.10 Redundanzmodell (*links*); Implementierung der Wissensbasis (*rechts*) eines Agenten

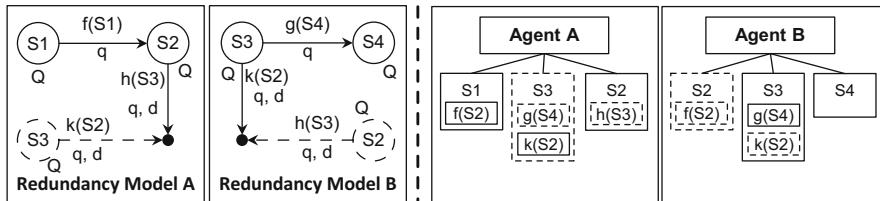


Abb. 1.11 Modellierung (*links*) und Implementierung (*rechts*) von verteiltem Wissen

1.3.2 Entwurf funktional implementierter Agenten

Das Architekturkonzept der funktionalen Implementierung von Softwareagenten erschwert die Realisierung eines Konzepts zur Implementierung von virtuellen Sensoren zur Detektion und Kompensation von Sensorausfällen. Da in einer solchen Architektur Sensor- und Aktorkomponenten mehreren Agenten zugeordnet sind, ist das Wissen über analytische Zusammenhänge zwischen Sensoren ebenfalls aufgeteilt auf mehrere Agenten. Dies erfordert zum einen eine angepasste Modellierung der Wissensbasis (Redundanzmodell) für einzelne Softwareagenten (Abb. 1.11, links) und zum anderen eine angepasste Architektur für die Implementierung von virtuellen Sensoren (Abb. 1.11, rechts).

Der gerichtete Graph, vorgeschlagen in [13–15] muss um die Modellierung von Sensoren erweitert werden, die einem Agenten nicht exklusiv zugeordnet sind und daher nur als Literatur zur Verfügung stehen (gestrichelte Linien in Abb. 1.11, links). Da sich zwei Agenten, in deren Wissensbasis der gleiche Sensor enthalten ist, auf unterschiedlichen, durch einen Feldbus verknüpften Steuerungen befinden können, müssen zusätzlich zu der Präzision eines (virtuellen) Sensorwerts auch Faktoren im Redundanzmodell berücksichtigt werden, die Unsicherheiten (wie z. B. zeitliche Verzögerungen) wiederspiegeln, die, verursacht durch den Feldbus, die Berechnung virtueller Sensoren beeinflussen. Da in der gewählten Architektur die realen und virtuellen Sensoren nicht mehr vollständig in die Matrix eines Agenten implementiert werden können, wird die getrennte Implementierung von Softwareagenten auf der einen Seite und Sensorbausteinen, die virtuelle Sensoren aus den analytischen Zusammenhängen implementieren, auf der anderen Seite vorgeschlagen (Abb. 1.11, rechts).

1.3.3 Herausforderungen zukünftiger Architekturen

In eigenen Vorarbeiten wurden vielversprechende Konzepte zur Steigerung der Verfügbarkeit einer Produktionsanlage vorgeschlagen, die das Wissen über analytische Zusammenhänge zwischen sensor- und Aktorwerten innerhalb von Softwareagenten implementieren. Wie im Vorangegangenen gezeigt, erfordern diese Konzepte jedoch die Implementierung des gesamten für die Kompensation eines Sensorausfalls notwendigen Wissens innerhalb eines einzelnen Softwareagenten. Ansätze für die Aufteilung dieses Wissens auf mehrere Agenten sind zwar vorhanden, werden aber bisher nicht durch ein durchgängiges Konzept gestützt und es fehlen Mechanismen, die es mehreren Agenten in einem Agentensystem erlauben, ihr jeweiliges Wissen zu kombinieren um Sensorausfälle zu erkennen und zu kompensieren. Darüber hinaus wurde die Modellierung der Wissensbasis eines Agenten sowie die Modelle der Softwareagenten bisher nur semi-formal definiert, wodurch eine formale Verifikation des Verhaltens einzelner Agenten und eines Agentensystems nicht möglich ist. Die Erweiterung der Konzepte aus den eigenen Vorarbeiten orientiert sich daher an den folgenden Zielstellungen:

Formalisierung und Verteilung von Wissen auf mehrere Agenten

Die Modelle, welche die Wissensbasis eines einzelnen Agenten beschreiben, werden auf die Anforderungen der Modellierung von verteiltem Wissen angepasst und durch eine formale Basis erweitert werden. An die erweiterte Form der Redundanzmodelle werden zwei wesentliche Anforderungen gestellt: Erstens muss es möglich sein, die Redundanzmodelle einzelner Agenten separat zu entwickeln, sodass mögliche Verknüpfungen untereinander jedoch eine Kombination dieses Wissens durch die Agenten zur Laufzeit ermöglichen. Zweitens müssen die Modelle hinsichtlich der speicherprogrammierbaren Steuerungen als Zielplattform der Agenten mit geringem Speicherbedarf implementierbar sein.

Formale Modelle eines SPS-basierten Agentensystems

Über ein formales Modell des Wissens einzelner Agenten hinaus muss ein formales (probabilistisches) Modell des Verhaltens einzelner Agenten innerhalb eines Agentensystems und des Agentensystems selber entwickelt werden. Insbesondere der Entscheidungsprozess im Rahmen der Erkennung und Kompensation von Sensorausfällen auf der Grundlage von verteiltem und sich gegebenenfalls überschneidendem Wissen, muss durch dieses Modell beschrieben werden. Für diese Modelle werden ebenfalls die Anforderungen bezüglich der Zielplattform gestellt, d. h. das Verhalten der Agenten und die Mechanismen innerhalb des Agentensystems müssen durch Algorithmen realisiert werden können, die auf die Speicher- und Rechenressourcen einer SPS angepasst sind. Dafür müssen Architekturschablonen

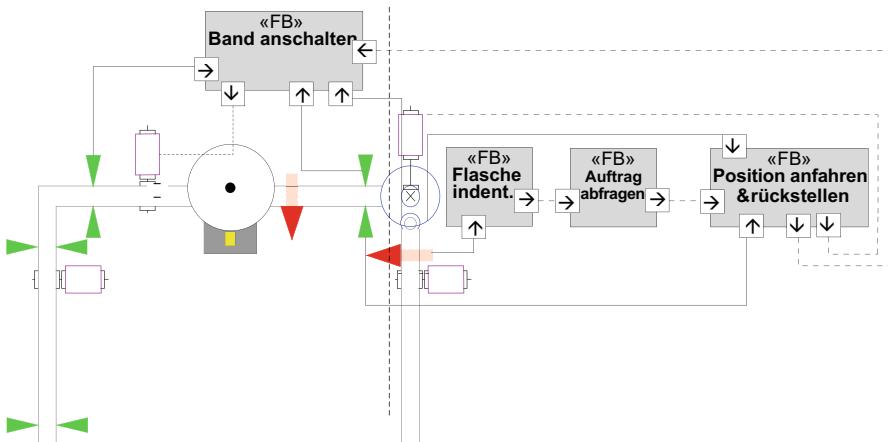


Abb. 1.12 Logische Architektur

und Protokolle definiert werden, die es Agenten eines Agentensystems erlauben, ihr Verhalten unter den Echtzeitbedingungen eines Produktionsprozesses auszuführen.

Verifikation des Verhaltens von Agenten und Agentensystemen

Basierend auf den formalen Modellen des Wissens einzelner Agenten, deren Struktur und Verhalten sowie den Modellen eines Agentensystems müssen Methoden und Konzepte entwickelt werden, welche die formale Verifikation eines implementierten Agentensystems in der Entwicklung wie auch zur Laufzeit ermöglichen. Insbesondere die Anlagenverfügbarkeit und die Wahrscheinlichkeit falscher Entscheidungen durch die Agenten müssen zu jedem Zeitpunkt verifizierbar sein.

1.4 Evaluation anhand eines Fallbeispiels

Für das in Abschn. 1.1 beschriebene Beispiel wird in diesem Abschnitt die Architektur für ein verteiltes, nach funktionalen Aspekten implementiertes Agentensystem vorgeschlagen. Ziel ist es aus der logischen (funktionalen) Architektur in Abb. 1.12 sowie der vereinfachten Deploymentarchitektur in Abb. 1.12 eine Agentenarchitektur zu entwerfen. Wie in Abb. 1.12 dargestellt, werden die Lichtschranke in der Weiche sowie die Lichtschranke am Bandanfang von beiden Modulen benötigt und müssen daher über die Systemgrenzen hinweg verfügbar sein. Abb. 1.13 zeigt, dass die Sensoren jeweils nur auf einer Steuerung physikalisch verfügbar

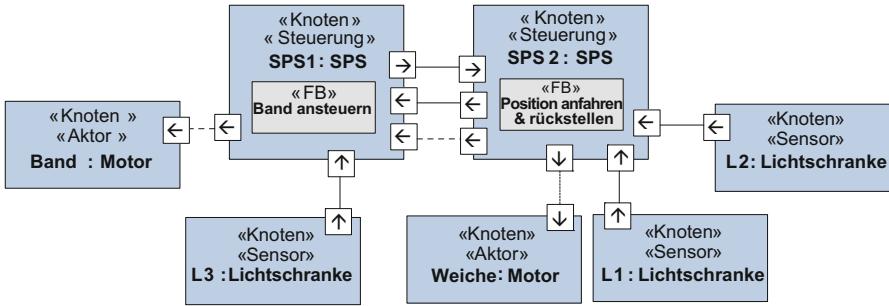


Abb. 1.13 Deploymentarchitektur

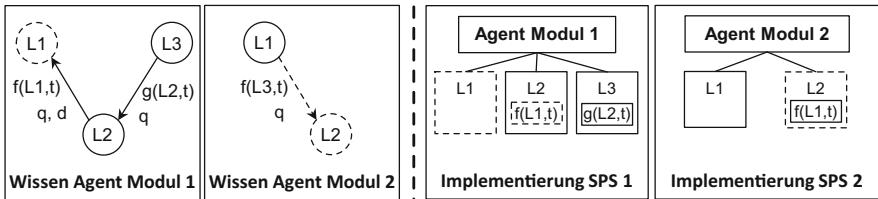


Abb. 1.14 Agentenarchitektur für virtuelle Sensoren L2 und L3

sind. Um einen zuverlässigen Anlagenbetrieb zu gewährleisten sollen virtuelle Sensoren verwendet werden. So soll für Modul 1 (Abb. 1.12, links) die Lichtschranke L1 der Weiche von Modul 2 (Abb. 1.12, rechts) für die Berechnung von virtuellen Sensoren für die in Modul 1 enthaltenen Lichtschranken L2 und L3 verwendet werden. Außerdem ist die Lichtschranke L2 am Beginn des Bandes für Modul 2 verfügbar sein, um das Ausfahren der Flasche auch mit Sensorausfall detektieren zu können.

Aus der in Abb. 1.13 dargestellten Deploymentarchitektur ergibt sich, dass die Agenten zur Ansteuerung des Bandes bzw. der Weiche entsprechend ihren Modulen auf die Knoten SPS1 und SPS2 verteilt werden. Dieser Architektur folgend können Agenten für Modul 1 und Modul 2 anhand der funktionalen Verteilung implementiert werden. Die Entwicklung der Wissensbasis der Agenten als gerichteter Graph ist in Abb. 1.14, rechts dargestellt. Das Wissen des Agenten von Modul 1 (Transportband) umfasst die für die Funktion benötigten Lichtschranken L1, L2 und L3. Da die Lichtschranke L1 entsprechend dem Deployment an den Knoten SPS1 angeschlossen ist, ist sie innerhalb der Wissensbasis des Agenten, welcher auf dem Knoten SPS2 implementiert wird, lediglich als Literatur ausgeführt. Das Wissen des Agenten von Modul 2 (Weiche) beinhaltet lediglich die Lichtschranken L2 und L2 sowie einen möglichen virtuellen Sensor für L2 auf Basis von L1. Eine mögliche Implementierung der Agenten und Bausteine für reale und virtuelle Sensoren ist in Abb. 1.14, rechts skizziert.

1.5 Zusammenfassung

In diesem Beitrag wurde ein Ansatz vorgestellt, verteilte agentenbasierte Automatisierungssysteme zu entwerfen. Dafür bietet die vorgestellte Notation die Möglichkeit sowohl über Sichten als auch über einen sehr begrenzten Notationsumfang die Komplexität im Entwurf beherrschbar zu machen.

Durch die Verwendung von virtuellen Sensoren auf Basis eines Agentenansatzes kann die Verfügbarkeit einer Anlage gesteigert werden. Defekte Sensoren können vom System kompensiert werden. Der Ansatz der vorgestellten verteilten Agentenstruktur ermöglicht es, auch über Systemgrenzen hinweg, Sensorwerte durch deren Berechnung aus anderen noch verfügbaren Sensoren zu ersetzen. Die Komplexität des Entwurfs wird durch den vorgestellten modellbasierten Entwurf reduziert und Fehler im Entwurf verringert.

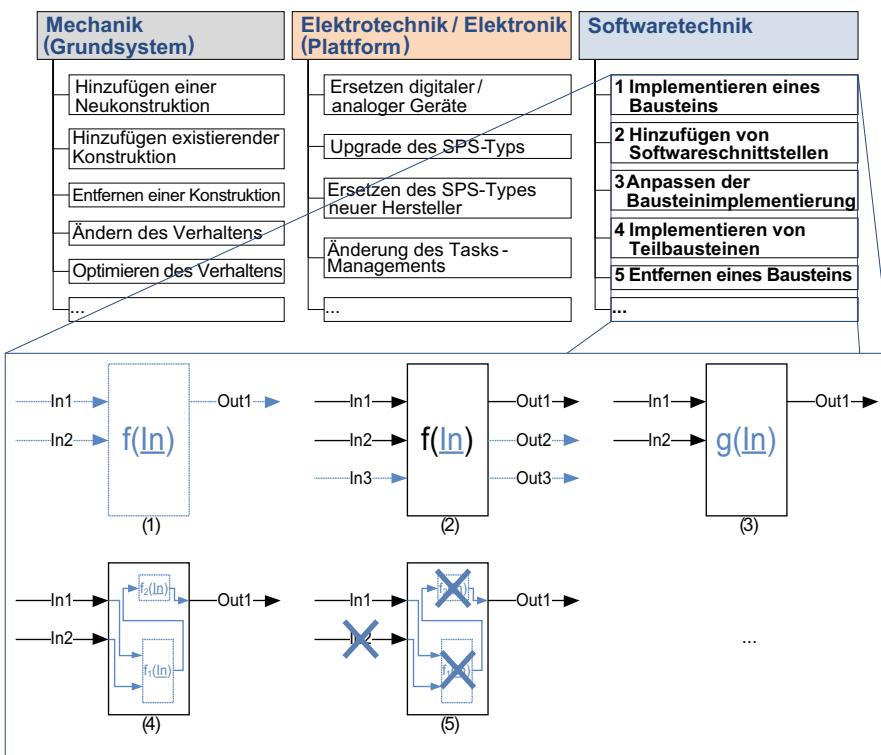


Abb. 1.15 Änderungszenarien in den Disziplinen der Automatisierungstechnik (Mechanik, Elektrotechnik/Elektronik, Softwaretechnik)

1.6 Zukünftige Forschungsziele und -schwerpunkte

In zukünftigen Arbeiten wird eine automatische Transformation des verteilten Entwurfs in eine passende Agentenstruktur angestrebt. Mit den bestehenden Ansätzen zur Codegenerierung von IEC 61131-3 ist dann ein durchgängiger modellbasierter Entwurf von verteilten agentenbasierten Automatisierungssystemen möglich.

Um mit den vorgestellten Ansätzen ebenfalls Rekonfigurationen (Version und Variante) auch während der Betriebsphase einer Maschine oder Anlage durchführen zu können, müssen in weiteren Arbeiten domänentypische Änderungsszenarien identifiziert werden und diese in der vorgestellten Notation abbildbar gemacht werden. Abbildung 1.15, oben, zeigt solche typischen Änderungsszenarien in den verschiedenen Disziplinen (erweitert um Mechanik). Änderungen können sowohl komplex (z. B. Hinzufügen von Neukonstruktionen in der Mechanik oder Ersetzen des SPS-Typs in der Elektrotechnik/Elektronik), als auch einfach sein (z. B. Ersetzen von Sensoren oder Aktoren in der Elektrotechnik/Elektronik oder Hinzufügen von Softwareschnittstellen in der Softwaretechnik). In zukünftigen Arbeiten soll der hier vorgestellte Ansatz um die dargestellten Szenarien erweitert werden. Hierbei können die hierarchische Darstellung sowie die Kombinierbarkeit der Sichten eine Integration erleichtern.

Literatur

- [1] Simon, R., Hänel, M., Riedl, M.: Verteilte Systeme durch Funktionsproxies. 11. Fachtagung für den Entwurf komplexer Automatisierungssysteme (EKA), S. 137–145. Magdeburg (2010)
- [2] Herrero-Pérez, D., Martínez-Barberá, H.: Modeling distributed transportation systems composed of flexible automated guided vehicles in flexible manufacturing systems. IEEE Transactions on Industrial Informatics (TII) **6**(2), 166–180 (2010)
- [3] Lepuschitz, W., Zoitl, A., Valle, M., Merdan, M.: Toward self-reconfiguration of manufacturing systems using automation agents. IEEE Transactions on Systems, Man, and Cybernetics (SMC) **41**(1), 52–69 (2011)
- [4] Theiss, S., Vasyutynskyy, V., Kabitzsch, K.: Software agents in industry: a customized framework in theory and praxis. IEEE Transactions on Industrial Informatics (TII) **5**(2), 147–156 (2009)
- [5] Vogel-Heuser, B., Schütz, D.: Flexible, echtzeitfähige Steuerungssoftware durch Agenten auf IEC 61131 Systemen. Tagungsband SPS/IPC/Drives, S. 411–419. VDI Verlag, Düsseldorf (2010)
- [6] Wannagat, A., Vogel-Heuser, B.: Agent oriented software development for networked embedded systems with real time and dependability requirements the domain of automation, S. 4144–4149. IFAC World Congress, Seoul (2008)
- [7] Frank, T., Eckert, K., Hadlich, T., Fay, A., Diedrich, C., Vogel-Heuser, B.: Dealing with non-functional requirements in distributed control systems engineering. 16th IEEE Conference on Emerging Technologies & Factory Automation (ETFA) (2011)
- [8] Frank, T., Hadlich, T., Eckert, K., Fay, A., Diedrich, C., Vogel-Heuser, B.: Using contact points to integrate discipline spanning real-time requirements in modeling Networked Automation Systems for manufacturing systems. IEEE International Conference on Automation Science and Engineering (CASE) (2012) angenommenes Paper

- [9] Eckert, K., Frank, T., Hadlich, T., Fay, A., Diedrich, C.: Typical automation functions and their distribution in automation systems. 16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) (2011)
- [10] Hadlich, T., Diedrich, C., Eckert, K., Frank, T., Fay, A., Vogel-Heuser, B.: Common communication model for distributed automation systems. INDIN'2011 – IEEE 9th International Conference on Industrial Informatics (2011)
- [11] Frank, T., Eckert, K., Hadlich, T., Fay, A., Diedrich, C., Vogel-Heuser, B.: Workflow and decision support for the design of distributed automation systems. IEEE INDIN: International Conference on Industrial Informatics (2012) angenommenes Paper
- [12] Frank, T., Vogel-Heuser, B., Hadlich, T., Eckert, K.: Erweiterung des V-Modells ® für den Entwurf von verteilten Automatisierungssystemen. Entwurf komplexer Automatisierungssysteme (EKA) (2012)
- [13] Wannagat, A., Vogel-Heuser, B.: Increasing flexibility and availability of manufacturing systems – Dynamic reconfiguration of automation software at runtime on sensor faults. Proc. 9th IFAC Workshop on Intelligent Manufacturing Systems (2008)
- [14] Wannagat, A.: Entwicklung und Evaluation agentenorientierter Automatisierungssysteme zur Erhöhung der Flexibilität und Zuverlässigkeit von Produktionsanlagen. Dissertation, Technische Universität München (2010)
- [15] Wannagat, A., Vogel-Heuser, B.: Agent oriented software-development for networked embedded systems with real time and dependability requirements the domain of automation. Proc. 17th IFAC World Congress, Seoul, South Korea (2008)

Kapitel 2

Energiesysteme und das Paradigma des Agenten

**Andreas Beck, Christian Derksen, Sebastian Lehnhoff, Tobias Linnenberg,
Astrid Nieße und Gregor Rohbogner**

Zusammenfassung Das Paradigma des Agenten findet zunehmend Anwendung in hochdynamischen und komplexen Bereichen, welche koordinierte oder koordinierende Prozesse erfordern. In diesem Beitrag werden neue Anforderungen an die Systeme der Energieversorgung und des Netzbetriebes vorgestellt und diskutiert, inwieweit das Agenten-Paradigma diesen gerecht werden kann.

2.1 Einleitung

Die politischen Vorgaben für die Strukturen der Energieversorgung ab dem Jahrre 2020 und darüber hinaus stellen das Energiesystem vor große Herausforderungen. Die Liste der Aufgabenstellungen reicht dabei von rein technisch, ingenieurwissenschaftlichen Problemen, bis hin zu informationstechnologischen, wirtschaftlichen und soziologischen Fragestellungen. Aus technologischer Sicht liegt die größte Herausforderung in der aktiven Gestaltung eines Transformationsprozesses, der durch die Zunahme volatiler, häufig regenerativer Energieumwandlungsanlagen

Andreas Beck (✉)

IAS, Universität Stuttgart, Pfaffenwaldring 47, 70569 Stuttgart, Deutschland

e-mail: Andreas.Beck@ias.uni-stuttgart.de

Christian Derksen

DAWIS, Universität Duisburg-Essen, Schützenbahn 70, 45127 Essen, Deutschland

e-mail: Christian.Derksen@icb.uni-due.de

Tobias Linnenberg

IfA, Helmut-Schmidt-Universität Hamburg, Holstenhofweg 85, 22043 Hamburg, Deutschland

e-mail: Tobias.Linnenberg@hsu-hh.de

Sebastian Lehnhoff, Astrid Nieße

Institut für Informatik, FuE-Bereich Energie, OFFIS, Escherweg 2,

26121 Oldenburg, Deutschland

e-mail: Sebastian.Lehnhoff@offis.de, Astrid.Niesse@offis.de

Gregor Rohbogner

Fraunhofer-Institut für Solare Energiesysteme ISE, Heidenhofstr. 2, 79110 Freiburg, Deutschland

e-mail: Gregor.Rohbogner@ise.fraunhofer.de

bei gleichzeitiger Reduktion bisher systemstabilisierender Großkraftwerke ausgelöst wird.

Dieser Transformationsprozess beschränkt sich aber nicht allein auf die Umstrukturierung des Kraftwerksparks. Vielmehr umfasst der Wandel sämtliche Prozesse des heutigen Energiesystems: Angefangen bei der Schutz- und Automatisierungstechnik in Nieder- und Mittelspannungsnetzen, über die Betriebsführung von Energieeinspeisern auf allen Netzebenen, dem Lastmanagement flexibler Stromverbraucher bis hin zu den Energiemarkten.

Unter dem Stichwort Smart Grids oder intelligente Stromnetze werden in Forschung und Praxis Ansätze auf Basis der Informations- und Kommunikationstechnik (IKT) entwickelt. Diese sollen in dem angestoßenen Transformationsprozess eine neue Betriebsführung für Stromerzeuger, Speicher, elektrische Verbraucher und im Netz befindliche Betriebsmittel ermöglichen. Den zunehmenden Kapazitätsproblemen im Netz soll damit nicht nur über Ausbaumaßnahmen, sondern ergänzend hierzu durch eine effizientere Nutzung der Netzkomponenten sowie durch Koordinationsansätze in Erzeugung, Verteilung und Verbrauch begegnet werden.

Durch Smart Grids entsteht somit ein informationstechnologisch angereichertes, verteiltes, dynamisches und technisch diversifiziertes System, das aus Automatisierungssicht aus einer Vielzahl von autonomen Teilsystemen besteht. Diese Teilsysteme bedürfen einer Koordination, die auch zukünftig eine sichere, preisgünstige, verbraucherfreundliche, effiziente und umweltfreundliche Energieversorgung gewährleisten kann (§ 1 EnWG).

Der vorliegende Beitrag, der von Mitgliedern der Untergruppe Energie des Fachausschusses Agenten der VDI Gesellschaft für Mess- und Automatisierungstechnik verfasst wurde, stellt das Agenten-Paradigma alten und neuen Herausforderungen des dargestellten Transformationsprozesses gegenüber. Dazu wird zunächst in Abschn. 2.2 ein kurzer Überblick über die heutige Energieversorgung gegeben. Im darauf folgenden Abschnitt werden typische Betriebsführungssstrukturen die in Energieleitsystemen zur Anwendung kommen eingeführt. Abschnitt 2.4 widmet sich den Anforderungen an die Verteilnetzautomatisierung und zukünftigen Leitsysteme, bevor in Abschn. 2.5 das Agenten-Paradigma vorgestellt und auf die vorher definierten Anforderungen hin geprüft wird. Abschließend diskutieren wir den Forschungsbedarf im Schnittbereich von Energietechnik, Elektrotechnik und Informatik, der sich nach Meinung der Autoren aus dem Dargestellten ergibt.

2.2 Das elektrische Energiesystem heute

2.2.1 Elektrische Energieerzeuger und Speicher

Die wichtigsten Energieumwandlungsanlagen sind thermische Großkraftwerke. Diese differenziert man anhand der eingesetzten Primärenergieträger sowie der zeitlichen Verfügbarkeit. So unterscheidet man nukleare und fossile Kraftwerke, welche typischerweise auf der Hochspannungsebene einspeisen. Diese Kraft-

werkstypen können in engen Grenzen geregelt werden. Man nutzt sie daher zur Deckung der Grundlast. Gasturbinen- und Pumpspeicherkraftwerke lassen hingegen eine schnelle Regelung über ein weites Spektrum zu. Daher werden Gasturbinenkraftwerke häufig als notwendiges Gegenstück zu regenerativen Energieumwandlungsanlagen wie Windkraftanlagen oder Solarparks installiert. Diese basieren auf fluktuierenden Primärenergieträgern und können somit nur bedingt zuverlässig Strom erzeugen. Als invariable regenerative Energieumwandlungsanlagen können Biogas- und Wasserkraftwerke angeführt werden.

Es wird versucht die dynamische Einspeisung durch moderne Speichertechnologien zu verstetigen. Insbesondere Pumpspeicherkraftwerke haben seit langem eine systemstabilisierende Rolle in Starklastphasen. Schnell reagierende Schwungmassenspeicher sind parallel hierzu in Pilotprojekten im Einsatz [1, 2].

2.2.2 Elektrische Netze

Die Struktur des kontinentaleuropäischen Elektrizitätsnetzes gliedert sich in zwei Ebenen: Das Übertragungsnetz sowie das Verteilnetz.

Das Transport- bzw. Übertragungsnetz bildet die übergeordnete Struktur. Es wird im Bereich der Hoch- bzw. Höchstspannung (220, 380 oder 400 kV) betrieben, was eine verlustarme Übertragung von Strom aus Großerzeugern ermöglicht. Um lokale Störungen zu kompensieren, ist diese überlagerte Netzebene in den Grenzen Kontinentaleuropas eng vermascht und gut überwacht. Der Stromtausch wird auf der Verteilnetzebene auf Mittelspannungsniveau (10 bis 20 kV) realisiert, hier speisen vermehrt dezentrale und volatile Stromerzeuger ein. Auf Grund des Mangels an Mess-, Steuer- und Regelungstechnik ergeben sich hierdurch zunehmend kritische Situationen, welchen durch eine gezielte Steuerung regelbarer Erzeuger und Verbraucher, auf der tiefsten Stufe des Netzes – dem sternförmigen bzw. radialen Ortsnetz im Bereich der Niederspannung (230 oder 400 V), zukünftig begegnet werden soll. Die Verbindung zwischen den einzelnen Netzebenen bilden Transformatoren in leittechnisch gut bestückten Umspannwerken zwischen Hoch- und Mittelspannungsebene sowie mit rudimentären Schutzmechanismen versehenen Transformatorenstationen zwischen der Mittel- und Niederspannungsebene.

2.2.3 Elektrische Energieverbraucher

Ein elektrischer Energieverbraucher ist ein Gerät oder ein System von Geräten, das einen Teil der elektrischen Energie in eine für den Endnutzer nützliche Energieform umwandelt. Der Energiebedarf eines elektrischen Energieverbrauchers wird durch die auszuführenden Systemfunktionen bestimmt. Dieser ist abhängig von verschiedenen Einflüssen, aus denen sich mögliche Freiheitsgrade für die Optimierung zur Laufzeit ergeben. Einerseits können zwischen elektrischen Energieverbrauchern

oder einzelnen Geräten leitungsgebundene oder räumliche Interaktionen oder Einflüsse bestehen. Das heißt, die einzelnen Energieverbraucher beeinflussen ihren Energiebedarf gegenseitig. Andererseits – und dies ist in der Regel der Haupteinflussfaktor – bestimmt die Bedienung durch den Nutzer und dessen Anforderungen an die ausgeführten Systemfunktionen den Energiebedarf eines elektrischen Energieverbrauchers. Diese Faktoren mit Einfluss auf den Zeitraum und die Intensität stellen die möglichen Freiheitsgrade dar, die zur Optimierung (z. B. hinsichtlich Versorgungssicherheit oder Energiekosten) mit den Einflussfaktoren auf die Energiebereitstellung koordiniert werden müssen. Heutzutage werden bei elektrischen Energieverbrauchern nur einzelne Faktoren optimiert wie z. B. ein zeitabhängig implementierter automatischer Stromsparmodus. Für eine koordinierte Optimierung der elektrischen Energieverbraucher sind allerdings Vernetzung und Kommunikation erforderlich, um die aktuelle Betriebssituation erfassen und die Einflussfaktoren entsprechend anpassen zu können. Hier existieren vor allem im industriellen Umfeld und im Bereich der Hausautomation unterschiedliche Ansätze (z. B. [3–5]).

2.2.4 Lastermittlung und Netzsteuerung heute

Das zurzeit gängige Verfahren in der Auslegung und dem Betrieb elektrischer Netze im Bereich der Nieder- und Mittelspannungsebene basiert auf Bilanzen und Prognosen, welche sich auf Standardlastprofile (SLP) stützen. Dabei handelt es sich um Lastgänge, welche nach unterschiedlichen Konsumenten mit einem Maximalverbrauch von bis zu 100.000 kWh/a differenziert werden. Jedoch liefert dieses Verfahren lediglich eine vereinfachte statische Abschätzung des Gesamtverbrauchs und der damit verbundenen Netzauslastung.

In Anbetracht der wachsenden Durchdringung volatiler, dezentraler Energieumwandlungsanlagen (DEA) in den unteren Spannungsebenen des Elektrizitätsnetzes kommt es vermehrt zu Konstellationen, in welchen eine statische Netlastberechnung keine adäquaten Resultate liefern kann. Ein erster Schritt um dieser Problematik zu begegnen ist die Nutzung von digitalen Stromzählern, mit deren Hilfe der aktuelle Energieverbrauch der Haushalte automatisiert übermittelt werden kann. So ist eine dynamische und kundenspezifische Lastabschätzung für den Energieversorger, aber prinzipiell auch für den Netzbetreiber möglich; zusätzlich ist es beispielsweise auch möglich, dem Kunden sekundengenaues Feedback über die verbrauchte elektrische Energie und Gasmenge zu geben [6].

Das deutsche Energiewirtschaftsgesetz sieht die Installation solcher „Smart Meter“ in Neubauten vor, Energieversorgungsunternehmen haben dabei einen lastvariablen oder tageszeitabhängigen Tarif anzubieten (§ 40 Abs. 5 EnWG). In Ländern wie Italien [7], Kalifornien [8] oder Schweden [9] sind Smart Meter bereits heute weit verbreitet.

Ziel der Anstrengungen ist es, zeitvariable Verbraucher in abnahmeschwache Perioden zu verlagern. In diesen wird die sogenannte „Grundlast“, das absolute Minimum des Energiekonsums, abgenommen. Die Mittellast, welche den darüber

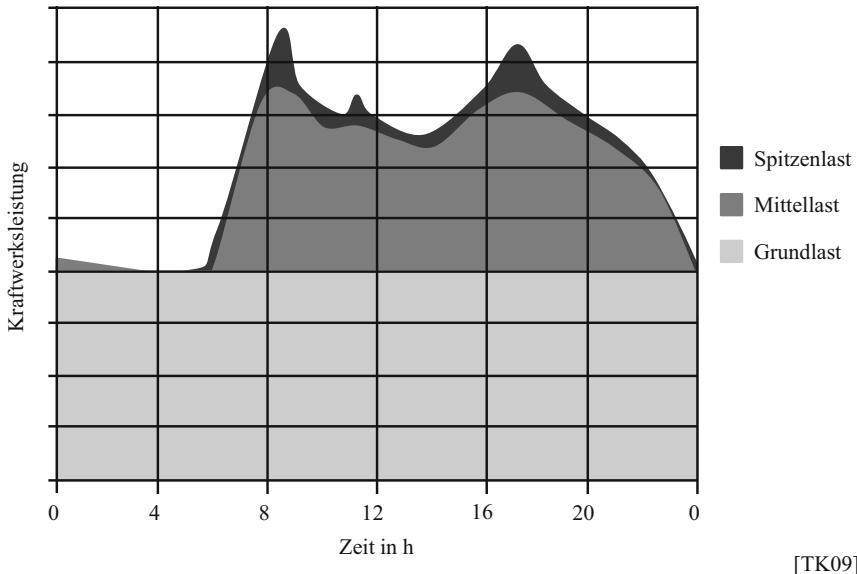


Abb. 2.1 Stromnachfrage und -bereitstellung an einem Durchschnittstag (Quelle: [10])

hinausgehenden Teil der konsumierten Energiemenge darstellt, wird während des Tages benötigt. Im Gegensatz hierzu wird die Spitzenlast in wesentlich geringerem Umfang abgefragt. Sie repräsentiert kurze Lastspitzen, starke Lastanstiege und plötzlichen Energiebedarf. Siehe hierzu Abb. 2.1, welche eine Übersicht der Lastkurve über einen gesamten Tag gibt. Auch industrielle Großverbraucher beteiligen sich durch Verschiebung ihrer Spitzenlasten an diesem Prozess. Durch das Management eigener Energieportfolios sowie der Errichtung von Erzeugungskapazitäten machen sie sich zunehmend unabhängig von Kapazitäts- und Preisschwankungen an den Energiemarkten.

2.2.5 Herausforderungen für das zukünftige Energiesystem

Die Annahme streng hierarchischer Netztopologien in der Energieversorgung ist nicht mehr zeitgemäß. Gleichermaßen gilt für die Anwendung von SLPs, da diese rein statischen Verfahren nicht in der Lage sind, volatile Energieerzeugung und deren Verbrauch zeitlich aufzulösen. Eine Auslegung zukünftiger Netze und Anlagen ist somit erschwert. Für die Entwicklung zukünftiger elektrischer Energiesysteme lassen sich derzeit zwei wesentliche Herausforderungen identifizieren:

- Für die Optimierung der unterschiedlichen Einflussfaktoren auf den Energiebedarf im funktionalen Kontext der elektrischen Energieverbraucher ist eine koordinierte Analyse, Interpretation und Handlungsauslegung von Informationen

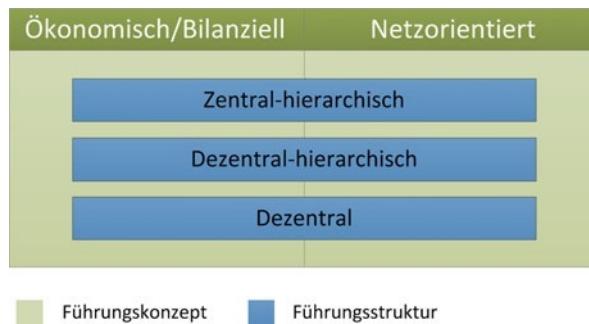


Abb. 2.2 Führungssystem = Führungskonzept (Hintergrund) + Führungsstruktur (Vordergrund)

erforderlich, deren Komplexität durch den Bedarf an quantitativ, zeitlich und räumlich differenzierteren Prognosen ansteigt.

- Mit Hilfe der zu entwickelnden neuen Mess-, Steuer- und Regel-Infrastruktur ist eine Lastverlagerung in die Nachtstunden zu erwirken, um so eine Reduzierung der Spitzenlasten am Tage zu erreichen. Diese Forderung ist unter anderem auf den prognostizierten höheren Energieerzeugungsanteil der Windenergie zurückzuführen.

Ungeachtet dieser Forderungen sind jedoch Faktoren wie zusätzliche Installations- und Wartungskosten, Kompatibilitätsprobleme mit aktuellen und zukünftigen Systemen, sowie die Lebensdauer der verwendeten Komponenten zu beachten.

2.3 Typische Betriebsführungskonzepte und -strukturen in der wissenschaftlichen Literatur

Wie Abschn. 2.2 gezeigt hat, ist die Steuerung der heutigen elektrischen Netze zentralistisch und statisch ausgelegt. Auch wenn dieser Ansatz heute noch angewendet wird, erzwingen die Herausforderungen, die mit den Erneuerbaren Energien erwachsen, perspektivisch eine Veränderung der netzleittechnischen Führungssysteme.

Um in den nachfolgenden Abschnitten die Anwendbarkeit des agentenbasierten Ansatzes im Smart Grid bewerten zu können, wollen wir in diesem Abschnitt die in der Literatur als auch in der Anwendung bekannten Führungssysteme aufzeigen. Dabei unterscheiden wir im Folgenden zwischen der Führungsstruktur und dem Führungskonzept. Beides zusammen ergibt das Führungssystem (vgl. Abb. 2.2). Dabei kann jedes Führungssystem aus einer oder mehreren Führungseinheiten (in Abb. 2.3 als Control System – CS bezeichnet) bestehen, welche jeweils für ein physikalisches System verantwortlich sind. Unter der Führungsstruktur versteht man die Organisation der einzelnen Führungseinheiten mit ihren Beziehungen untereinander. Im Wesentlichen unterscheiden sich die Führungsstrukturen durch die

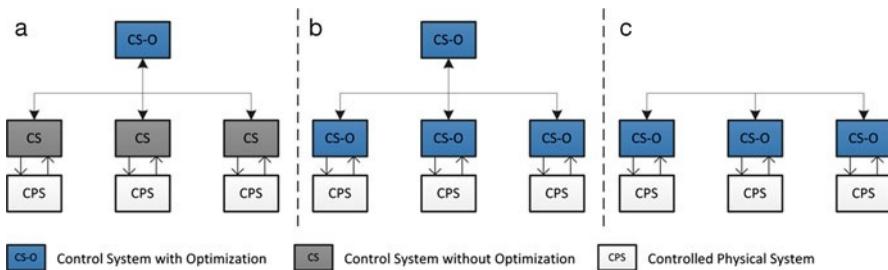


Abb. 2.3 a Zentral-hierarchische, b dezentral-hierarchische und c dezentrale Führungsstruktur

Verortung der „höheren“ Informationsverarbeitung (in den meisten Fällen eine Optimierung) und der Datenhaltung. Das Führungskonzept dagegen beschreibt, mit welcher Zielsetzung bzw. welchem Optimierungsziel das Führungssystem arbeitet. Bei den Führungskonzepten kann im Rahmen der Smart Grid Forschung zwischen „ökonomisch-/bilanzorientiertem“ und „netzorientiertem“ Führungssystem unterschieden werden.

Im Folgenden sollen im Detail nur die Führungsstrukturen betrachtet werden, welche sowohl bei Führungssystemen mit einem „netzorientierten“ Konzept als auch bei „ökonomisch/bilanziellen“ Konzepten zur Anwendung kommen können.

2.3.1 Zentral-hierarchische Führungsstruktur

Sowohl in der industriellen Automatisierung als auch für den Betrieb elektrischer Netze werden Führungsstrukturen in aller Regel zentral-hierarchischen ausgelegt. Abbildung 2.3a zeigt diese Struktur. Ein zentral-hierarchisches System besteht dabei aus zwei oder mehreren Ebenen, bei welchen die jeweils unterlagerte Führungsebene die Sollgrößen der überlagerten zentralen Führungseinheit erhält und umsetzt. Erste Forschungsprojekte, die sich mit einem Energie Management System in den Verteilnetzen beschäftigt haben, verfolgten im Wesentlichen eine zentral-hierarchische Struktur. In „Dispower“, als einem der frühen Forschungsprojekte in diesem Kontext, wird ein Energiemanagement für Niederspannungsnetze mittels einer zentralen Einheit und dezentralen Schnittstellen an den im Niederspannungsnetz angeschlossenen Verbraucher und Erzeuger umgesetzt. Während die zentrale Einheit die Optimierung des gesamten Netzes in Hinblick auf die Netzstabilität ausführt, erhalten die dezentralen Komponenten nur einen fertigen Fahrplan, welcher die Sollgrößen für die Erzeuger und Verbraucher enthält [11]. Ähnliche Konzepte, mit ökonomischen und weniger netzoptimierenden Hintergrund, welche eine zentral-hierarchische Struktur umsetzten, sind unter dem Namen „Virtuelles Kraftwerk“ bekannt. Als Beispiel kann das von der Firma Schmack Biogas AG, Solarworld AG und Enercon GmbH entwickelte „Kombikraftwerk 1“ genannt werden. Hier werden über eine zentrale Führungseinheit verteilte Erneuerbare-Energien-

Kraftwerke verschiedener Art (Windkraftanlagen, Photovoltaikanlagen, Biogas oder Wasserkraft) mittels Informations- und Kommunikationstechnologie vernetzt und gesteuert [12]. Während die zentrale Führungseinheit Daten speichert und verarbeitet, fungieren die dezentralen Recheneinheiten an den Erzeugern als Kommunikationschnittstellen, welche „nur“ für die Umsetzung der zentral errechneten Sollgröße verantwortlich sind.

2.3.2 Dezentral-hierarchische Führungsstruktur

Eine der zentral-hierarchischen Struktur sehr ähnliche, ist die dezentral-hierarchische Struktur. Wie Abb. 2.3b zeigt, besteht dieses Struktur jedoch aus verteilten Führungseinheiten, welche nicht nur Sollgrößensignale umsetzen, sondern eine „höherwertige“ Informationsverarbeitung besitzen. Diese „höherwertige“ Informationsverarbeitung kann durch lokal vorhandene Optimierungsalgorithmen oder als lernendes System umgesetzt sein. Neben der lokalen bzw. dezentralen Informationsverarbeitung kann auch die Datenhaltung in dezentraler Weise erfolgen. Zur Sicherstellung der Erreichung eines globalen Optimums (zum Beispiel für ein betrachtetes Teilnetz), ganz gleich ob in Hinblick auf eine stabile Netzführung oder aus ökonomischem Antrieb, erfolgt dabei der Informationsaustausch zwischen den dezentralen Elementen immer über eine zentrale Einheit. Eine Vielzahl von Forschungsprojekten der letzten Jahre hat diese Struktur umgesetzt. Gerade auch solche, die einen agentenbasierten Ansatz verfolgen. In Abschn. 2.5.2 wollen wir darauf näher eingehen und hier lediglich ein kurzes Beispiel anführen. Im Rahmen eines E-Energy¹ Projekts „eTelligence“ wurde für die Modellregion Cuxhaven ein Stromvermarktungssystem entworfen. Es handelt sich dabei um dezentrale Führungseinheiten, welche, verantwortlich für den wirtschaftlichen Betrieb ihnen zugeordneter physikalischer Systeme (z. B. Schwimmbad, Kläranlage), mit einer zentralen Einheit Strompreise aushandeln [13].

2.3.3 Dezentrale Führungsstruktur

Als letzte Struktur von Führungssystemen kann die dezentrale Führungsstruktur genannt werden. Heute noch im Bereich der Leittechnik der Energieversorgung weitgehend im Forschungs- und Entwicklungsstatus, entspricht diese Führungsstruktur wohl am ehesten den Konzepten und Grundgedanken von Agentensystemen. So gibt es in dieser Struktur keine dedizierte zentrale Führungseinheit mehr.

¹ Ein Förderprogramm des Bundesministeriums für Wirtschaft und Technologie für ein Informations- und Kommunikationstechnologie basiertes Energiesystem der Zukunft (www.e-energy.de).

Alle verteilten Führungseinheiten arbeiten, neben der lokalen Informationsverarbeitung, kooperativ für die Erreichung eines Gesamtziels zusammen.

2.3.4 Kritische Auseinandersetzung

In einem zukünftigen Smart Grid ist nicht davon auszugehen, dass sich ein dediziertes Führungssystem für das Smart Grid durchsetzen wird. Vielmehr wird es zu einer Ergänzung der jetzigen Führungssysteme kommen und hier zu einem Nebeneinander verschiedener Führungsstrukturen. Welche Struktur im Einzelnen zielführend ist, muss im jeweiligen Einsatzfall entschieden werden. Im Folgenden wollen wir die Führungsstrukturen deshalb beispielhaft an ein paar wesentlichen Eigenschaften kritisch gegenüberstellen, um sie später gegen Anforderungen von Smart Grid Anwendungen zu prüfen².

Steht die Optimierung eines geringfügig varianten Gesamtsystems³ (z. B. ein Niederspannungsnetz) im Fokus der Führungsbestrebungen, so scheint die zentral-hierarchische Struktur am geeignetsten. Alle Daten die in unterlagerten Führungseinheiten gesammelt wurden, sind einfach und mit einem direkt Zugriff für die zentrale Einheit verfügbar. Diese zentrale Verarbeitung bringt zwar einen erheblichen Datentransfer mit sich, führt aber zu einem vollständigen Systemmodell, mit welchem ein globales Optimum (im betrachtet Teilnetz) gefunden werden kann. Um das Kommunikationsaufkommen zu reduzieren und die Datensicherheit zu gewährleisten, eröffnet die dezentral-hierarchische Struktur die Möglichkeit der Speicherung und Verarbeitung von Informationen und Daten an der Stelle, an der sie anfallen. Daten und Informationen werden also schon vorverarbeitet und erst dann abstrahiert verteilt. Dies mindert die zu übermittelnde Datenbreite und erhöht die Menge an Daten, die im Einflussbereich des unterlagerten Systems bleiben (Datenkapselung). In vielen der durchgeführten Smart Grid Projekte (Dezent, PowerMatcher, eTelligence, um nur einige zu nennen) wurden Strompreise und -mengen als Führungsgröße verwendet. Nachteil dieser Dezentralisierung ist jedoch eine mögliche Verlängerung der Antwortzeit (bedingt durch dezentrale Berechnungen und Datenverarbeitung) zwischen zentraler und dezentraler Einheit. Neben dem Vorteil, dass die zu übertragene Datenmenge reduziert wird, bei gleichzeitiger Sicherstellung einer globalen Optimierung, bleibt jedoch der „Single Point of Failure“, wie im zentral-hierarchischen System, bestehen. Abhilfe würde hier das dezentrale System, ohne dedizierte zentrale Einheit, schaffen. Nachteil dieser Struktur ist jedoch, der mit der Dezentralisierung steigende Kommunikationsaufwand, da der Austausch

² Diese kritische Auseinandersetzung erhebt keinen Anspruch auf Vollständigkeit, sondern stellt einen Teil der Erfahrungen des Autorenteams vor.

³ Geringfügig variant ist hier im Regelungstechnischen Sinne zu verstehen und spielt auf ein an nähernd zeitlich invariantes System (z. B. Teilnetz) an. Es kommt also nur sehr selten zu einer Veränderung der Netzzusammensetzung, bei der ein neuer Verbraucher oder Erzeuger dazukommt oder wegfällt.

nicht nur zwischen zwei Einheiten sondern ungerichtet zwischen einer unbestimmten Zahl von Führungseinheiten stattfindet. Auch die Koordination (z. B. Zeitsynchronisierung) stellt eine Herausforderung dar, welche sich im Wesentlichen nur bei dezentralen Strukturen ergibt. Ist jedoch die Skalierbarkeit, sowie die Robustheit gegen Streckenänderungen und die Fehlertoleranz gegenüber dem Ausfall der Kommunikationsinfrastruktur oder einer der Führungseinheiten von entscheidender Bedeutung, ist das dezentrale System dem zentral- und dezentral-hierarchischen System überlegen.

2.4 Spezielle Anforderungen an die Verteilnetzautomatisierung

Im Umfeld elektrischer Energieversorgungssysteme wird zwischen Primärtechnik (Transformatoren, Kabel, Schalter etc.) und Sekundärtechnik unterschieden. Die Sekundärtechnik umfasst das Messen, Steuern und den Betrieb von Energieversorgungssystemen bis hin zu Funktionen, die den Energiemarkt betreffen. Die Prozesse der Schutz- und Leittechnik, die den Systembetrieb wirtschaftlich-technisch optimieren und insbesondere störungssicher machen sollen, bilden einen zentralen Bestandteil dieser Sekundärtechnik. Besondere Anforderungen an die Schutz- und Leittechnik reichen dabei von einer systemweiten Überwachung von Netzen großer räumlicher Ausdehnung (z. B. das europäische (Verbund-)Netzgebiet) bis hin zu Reaktionen und Prozessabläufen in kürzesten zeitlichen Intervallen (wenige Millisekunden) bei auftretenden Störungen. Wesentlich ist hierbei die Forderung nach größtmöglicher Selektivität, was bedeutet, dass Schutz- und Leittechnikkomponenten eine Störung durch eine möglichst minimale, lokal beschränkte Versorgungsunterbrechung beseitigen bzw. minimal stabilisierend auf unerwünschte Betriebspahnomene reagieren. Diesen besonderen und extremen Anforderungen an die Schutz- und Leittechnik wird bislang mit proprietären (zumeist nicht interoperablen) und auf einzelne spezielle Anwendungsfälle hin entwickelten teuren Lösungen begegnet.

Mit steigender Zahl von DEA muss zwangsläufig die Anzahl aktiver schutz- und leittechnischer Komponenten steigen, welche auf Seiten der DEA aber auch innerhalb der bestehenden Netzinfrastruktur installiert werden, um notwendige Schutz- und Steuerungsfunktionen für die Versorgungssicherheit in den Netzen zu gewährleisten.

In den Verteilnetzen der Mittelspannung erfordert der Zuwachs an DEA mit unvorhersehbarer fluktuierender Einspeisung (zusätzlich zu den bereits vorhandenen prognoseunsicheren Verbrauchsprozessen) eine kontinuierliche Neubewertung von, der Prozesssteuerung und Schutztechnik zugrunde liegenden, Netzmodellen und Parametern. Getrieben werden diese Änderungen im Wesentlichen durch den Zu- bzw. Ausbau von Windkraftanlagen und Blockheizkraftwerken größerer Leistung in den Mittelspannungsnetzen sowie PV-Anlagen in den Niederspannungsnetzen, aber auch die zu erwartende Integration der Elektromobilität. Während in der Vergangenheit eine nahezu konstante Kurzschlussleistung und ein gerichte-

ter Lastfluss von den höheren zu den niedrigeren Spannungsebenen als Berechnungsgrundlage für die Parametrierung der Schutzgeräte in den Schaltanlagen (z. B. Überstrom-Zeit-Schutz) dienten, müssen neuartige Schutz- und Leitsysteme adaptiv auf hochdynamische Betriebssituationen reagieren. Eine Koordination zwischen sekundärtechnischen Komponenten der (Verteil-)Netzautomatisierung ist hier unbedingt erforderlich, um eine rechtzeitige automatisierte Regelung bei größtmöglicher Selektivität zu gewährleisten und somit ein hohes Maß an Versorgungssicherheit zu erreichen. Ein Mangel an adäquaten Hard- und Softwarekomponenten, die dank offener Kommunikationsschnittstellen austauschbar eine herstellerunabhängige Kommunikation ermöglichen würden, stellt hier jedoch derzeit noch ein Hindernis für Anwendungen und den Einsatz derartiger koordinierter Automatisierungsfunktionen in den elektrischen Verteilnetzen dar.

2.4.1 Smart Grid Literatur Architektur

Derartige neue Prozesse sind nur erste Anwendungen von komplexen Funktionen in zukünftigen Smart Grids. Weitaus anspruchsvollere Prozesse sind dann umzusetzen, wenn viele verschiedene Anlagen, Netzkomponenten und Marktteilnehmer ggf. über Branchengrenzen hinweg gemeinsam eine Funktion im elektrischen Energieversorgungssystem realisieren müssen.

Zur Abbildung und Klassifizierung derartiger Funktionen arbeiten europäische Normungsorganisationen im EU Smart-Grid-Mandat M/490 an einer europäischen Literaturarchitektur im Bereich Smart Grid. Ziel dieser Architektur ist es, die Identifikation und Bearbeitung gemeinsamer komplexer sowie innovativer Funktionen zu erleichtern. So lassen sich mittels Funktionsbeschreibungen wichtige Schnittstellen definieren und auch Komplexität und nicht funktionale Anforderungen bestimmter Aufgaben ableiten. Letztere sind gerade vor dem Hintergrund des Entwurfs agentenbasierter Energiemanagement- und Leitsysteme von großer Bedeutung.

2.4.2 Anforderungen an zukünftige Netzleitsysteme

Die Aufgabe eines Netzleitsystems ist die Koordination und die Verwaltung einer großen Anzahl dezentraler Akteure und Messstellen und der damit verbundenen Prozesse und Aufgaben im elektrischen Energieversorgungssystem mit dem Ziel eines sicheren und zuverlässigen Betriebs des Gesamtsystems. Bis dato beschränken sich elektrische Netzleitsysteme auf die übersichtliche Darstellung des komplexen zeitlichen und räumlichen Systemzustands, um einen steuernden menschlichen Eingriff zu unterstützen oder überhaupt erst zu ermöglichen. Wie bereits dargestellt, wird sich jedoch die Qualität und Quantität der Steuer-, Regel- und allgemeiner der Automatisierungsaufgaben und -prozesse im zukünftigen Smart Grid drastisch verändern. Die Erfüllung der energiepolitischen Anforderungen Versorgungssicher-

heit, Wirtschaftlichkeit und Nachhaltigkeit kann dabei durch folgende Schritte sichergestellt werden:

- Versorgungssicherheit durch Redundanz im Netz: der Verletzung von Betriebsgrenzen oder dem Ausfall von aktiven Komponenten ist durch (zu minimierenden) Einsatz vorgehaltener Reserven und z. T. durch Restrukturierungsmaßnahmen zu begegnen, so dass die Versorgungsqualität im Idealfall unbeeinträchtigt bleibt. Die Versorgungsstabilität ist bei fortschreitender Verdrängung konventioneller Kraftwerke durch die Übernahme netzstabilisierender Systemdienstleistungen durch dezentrale Netzkomponenten und -akteure zu gewährleisten.
- Wirtschaftlichkeit durch günstige IKT-Integration und effiziente Nutzung: der wirtschaftliche Einsatz von IKT soll durch einfache und etablierte (off-the-shelf) Systeme und Komponenten ermöglicht werden. Hierzu sind möglicherweise Rückgriffe auf existierende Komponenten und bewährte Lösungen der Industrieautomatisierung möglich, welche bereits heute hohe Anforderungen an Echtzeit, Verfügbarkeit und Sicherheit erfüllen müssen.
- Nachhaltigkeit durch effiziente Energienutzung und Integration Erneuerbarer Energien: aktive Komponenten sollen ihre internen Zustände kontinuierlich anpassen, um eine effiziente und damit nachhaltigere Nutzung der zur Verfügung stehenden Ressourcen zu ermöglichen. Dabei ist auch hier zu erwarten, dass sich dezentrale, agentenbasierte Ansätze deutlich flexibler und damit effizienter und wirtschaftlicher in hochdynamischen Umgebungen betreiben lassen als konventionelle zentrale Organisationsstrukturen.

Für ein zukünftiges IKT-basiertes Netzleitsystem lassen sich aus diesen Anforderungen folgende nicht-funktionale Eigenschaften ableiten:

- Restrukturierungsfähigkeit: erlaubt die transparente Integration, Segregation und Substitution von Systemkomponenten in das IKT-System.
- Skalierbarkeit: die Eigenschaft des IKT-Systems, eine Vielzahl dezentraler Erzeuger und Verbraucher zu integrieren, um ihre Potentiale nutzbar zu machen sowie bestehende Aggregationsformen wie virtuelle Kraftwerke um neue Komponenten zu erweitern.
- Robustheit: strebt die Vermeidung eines Single-Point-of-Failure durch die Verteilung kritischer Systemfunktionen auf mehrere redundante dezentrale IKT-Komponenten an.
- Verfügbarkeit: Vorhersagbarkeit des aggregierten Leistungspotenzials fluktuiender dezentraler Akteure. Abhängig von Populationsgrößen lassen sich so Verfügbarkeiten, trotz des stochastischen Charakters individueller fluktuiender Erzeuger und Verbraucher, in bestimmten Grenzen garantieren.
- Echtzeitfähigkeit: muss für dezentral bereitgestellte Dienstleistungen gelten, die zur Netzstabilisierung und zur Versorgungssicherung notwendig sind und daher innerhalb fester betrieblich bedingter Reaktionszeiten aktiviert werden müssen.

- Fähigkeit zur Relokation von Funktionen: ermöglicht es dem System, (Teil-)Funktionen auf andere, mächtigere IKT-Komponenten, z.B. IEDs⁴ neuerer Technologie, zu verlagern. Die unterschiedlichen Lebenszeiten der IKT-Komponenten und die damit einhergehenden unterschiedlichen Fähigkeiten machen dies zu einem vielversprechenden Konzept für Smart Grids.

Diese Anforderungen/Eigenschaften stellen somit die anzusetzenden Kriterien bei der Bewertung der Eignung eines agentenbasierten Ansatzes für ein zukünftiges Netzeleitsystem dar. Hierzu werden nachfolgend agentenbasierte Betriebsführungs-konzepte betrachtet und im Anschluss hinsichtlich dieser Kriterien diskutiert.

2.5 Agenten und Agentenbasierte Betriebsführung

Agenten und Agentensysteme sind bereits seit vielen Jahren in verschiedenen Disziplinen fester Bestandteil der wissenschaftlichen Forschung. Die Idee ihrer automatisierungstechnischen Anwendung im Bereich der Energieversorgung hingegen ist vergleichsweise jung und steht im Zusammenhang mit der Dezentralisierung von Energieerzeugern und -verbrauchern. Dieser Abschnitt erläutert den Begriff des Agenten sowie die Eignung dieses Modellierungs- und Implementierungsansatzes auf die Anforderungen zukünftiger Energiesysteme. Ergänzend hierzu werden Konzepte der agentenbasierten Betriebsführung aus der wissenschaftlichen Literatur vorgestellt. Der Abschnitt schließt mit einer Betrachtung agentenbasierter Simulationen, die zur systematischen Evaluation bei der Weiterentwicklung einer intelligenten Energieversorgung genutzt werden können.

2.5.1 Agenten und ihr Einsatz in Smart Grids

Das Paradigma des Agenten stellt einen Modellierungs- und Implementierungsansatz für Softwaresysteme zur Verfügung: Spezielle Eigenschaften geben dieser Art von Software erweiterte Fähigkeiten und Freiheitsgrade, die einerseits einen erweiterten Handlungsspielraum ermöglichen, es jedoch gleichzeitig erlauben, Szenarien aus der Realität intuitiver zu modellieren und in ein Softwaresystem zu überführen.

Allgemein formuliert ist ein Agent ein Softwareartefakt oder ein Computer System, welches in eine Umgebung eingebettet ist und dabei in der Lage ist, in dieser Umgebung autonom zu agieren, um so seine (ggf. vorgegebenen) Ziele zu verfolgen. Diese Beschreibung orientiert sich weitestgehend an der in der wissenschaftlichen Literatur häufig zitierten Definition von Wooldridge und Jennings [14].

⁴ IED: Intelligent Electronic Device – nach IEC 61850 ein prozessorbasiertes Controller, vereinfacht eine Rechner-Einheit, die einer Komponente des elektrischen Energieversorgungssystems auf der Feldebene zugeordnet ist und eine Kommunikationsschnittstelle zur Komponente bietet.

Ergänzend hierzu wird in einem automatisierungstechnischen Kontext ein technischer Agent als Kombination von Hard- und Software verstanden [15].

Neben der genannten Eigenschaft der Autonomie werden Agenten häufig weitere Merkmale zugeschrieben. Zu diesen gehören die kommunikativen und kooperativen Fähigkeiten, das reaktive oder proaktive Verhalten sowie ggf. Lernfähigkeit und Mobilität [16]. Diese Eigenschaften stellen jedoch keine notwendigen Befähigungen eines Agenten dar. Vielmehr können sie, abhängig vom konkreten Einsatzzweck des Agenten, optional hinzugefügt werden. Als Mindestvoraussetzung zur Klassifizierung eines Softwareartefakts als Agent seien an dieser Stelle Autonomie und Kommunikationsfähigkeit als minimale Ausprägung der kooperativen Fähigkeiten genannt.

Kommen in einem Szenario mehrere Agenten zum Einsatz, die in gemeinsam abgestimmten Prozessen eine bestimmte Aufgabe lösen sollen, wird vom Einsatz eines Multi-Agentensystems bzw. eines Agentensystems gesprochen.

Eine grundsätzliche Möglichkeit der Klassifizierung von Agenten besteht in der Art, wie ein Agent die Entscheidung für seine nächste Handlung trifft. Hier wird im Wesentlichen in zwei Kategorien unterschieden: den reaktiven Agenten und den deliberativen (abwägenden) Agenten. Während reaktive Agenten auf Grund einer einfachen Musterunterscheidung auf Anfragen oder Veränderungen in ihrer Umgebung reagieren, erfolgt innerhalb eines deliberativen Agenten ein interner Abwägungsprozess, der nach der besten nächsten Handlungsoption für den Agenten sucht. Hierbei sind diese Abwägungsprozesse stark abhängig von der konkreten Aufgabe des Agenten, was zu einer Vielzahl von Implementierungen führen kann (Padgham und Winikoff 2004). Das aus Agentensicht bekannteste Konzept ist das sogenannte BDI-Konzept (Belief, Desire, Intention), auf das an dieser Stelle jedoch nur verwiesen werden soll [17, 18]. Im Gegensatz zum reaktiven Agenten, können Abwägungsprozesse in deliberativen Agenten ein zeitlich nicht-deterministisches Verhalten verursachen, welches abhängig vom Einsatzzweck eines Agenten jedoch unerwünscht sein kann.

Betrachtet man die Strukturen, die Anforderungen und die verschiedenen Konzepte der Betriebsführung zukünftiger aber auch heutiger Energiesysteme, zeigen sich eine Reihe von Vorteilen, die sich aus einem agentenbasierten Modellierungs- und Implementierungsansatz ergeben. Diese Vorteile lassen sich insbesondere in Bezug auf die Entwicklung hochgradig verteilter Anwendungen beschreiben:

- Im Gegensatz zu einem objektorientierten Modellierungsansatz ist die Abbildung verteilter Probleme mit einem agentenbasierten Ansatz direkt abbildbar, indem unterschiedliche und eigenständige Akteure koordiniert zusammenarbeiten, um ein gemeinsames Ziel zu erreichen. Ein denkbares Ziel könnte es hierbei z. B. sein, lokale Netzstabilität zu gewährleisten.

Ebenso wird bei der Implementierung eines agentenbasierten Systems der verteilte Charakter eines Problems direkt berücksichtigt. Dabei kommen in der Regel Agentenplattformen oder -Frameworks zum Einsatz, die die Kapselung der Agenten als autonomen Prozess bereits vorsehen und ein situationsabhängiges Verhalten unterstützen.

- Durch die Delegation von Aufgaben auf aktive und eigenständige Agenten können zentrale Koordinations- und Monitoring-Prozesse kalkulatorisch und kommunikativ entlastet werden. Überwachungs-, Auswertungs- und Planungsfunktionen werden dabei in erster Linie vor Ort ausgeführt. Ein kommunikativer Austausch findet nur bei Bedarf statt.
- Durch die selbstüberwachenden und selbstkonfigurierenden Fähigkeiten von Agenten können sich diese dynamisch an Veränderungen in ihrer Umgebung anpassen. Im Kontext eines sich verändernden Strom- bzw. Energienetzes ist diese Eigenschaft von besonderem Vorteil, da so das Netz über eine geeignete Agenten-Struktur selbst in die Lage versetzt wird, auf veränderte Systemzustände zu reagieren, ohne das ein menschlicher Eingriff erforderlich ist.
- Im Gegensatz zu den Investitionskosten bei automatisierungstechnischen und reglungstechnischen Anlagen lassen sich Agenten bereits auf kostengünstigen Hardwarestrukturen ausführen. Dieser Aspekt ist besonders für den Haushalts- und Endkundenbereich interessant und erlaubt somit einen breiten und flächen-deckenden Einsatz der Agententechnologie.

Mit den vorab genannten Eigenschaften und Aspekten ist eine agentenbasierte Führungsstruktur nicht nur für die Entwicklung zukünftiger Energiesysteme geeignet. Vielmehr ermöglicht eine agentenbasierte Führung elektrischer Systeme eine Konsolidierung und Restrukturierung der aktuellen Systeme, die an den heutigen Stand der Technik anknüpfen kann. Hieraus ergibt sich eine Vielzahl an neuartigen Fragestellungen und Aufgaben, die sowohl technischer, als auch ökonomischer Natur sind.

2.5.2 Agentenbasierte Führungssysteme in der elektrischen Energieversorgung

Agentenbasierte Ansätze haben in der elektrischen Energiewirtschaft erst in den letzten 10 bis 15 Jahren eine zunehmende Beachtung in der Forschung gefunden. Hier muss zwischen zwei grundlegenden Optimierungsfeldern bzw. Führungskonzepten (vgl. Abschn. 2.3) unterschieden werden.

Bei den bilanziellen Ansätzen wird typischerweise ein Aggregationskonzept umgesetzt, dass eine ökonomische Optimierung vorsieht und die so an Energiemarkten erzielten Gewinne an die Betreiber der aggregierten Anlagen weitergibt. Dem Gegenüber stehen netzorientierte Betriebsführungskonzepte, die den Betrieb der elektrischen Verteil- und Übertragungsnetze optimieren bzw. die Reaktion auf kritische Netzzustände innerhalb vorgegebener Echtzeitanforderungen ermöglichen. Bilanzielle Ansätze stellen eine Erweiterung bestehender Konzepte der Energiewirtschaft bzw. eine Anpassung an eine geänderte zunehmend dezentrale Energieumwandlungsstruktur dar, während die netzoptimierenden Ansätze auf neue Herausforderungen in der Systemführung der Netze Antworten geben wollen, die über Netz- und Kraftwerksausbau hinausgehen.

Für den Bereich der Inhouse-Optimierung werden ebenfalls agentenbasierte Konzepte entwickelt, die eine Steuerung dezentraler Erzeuger und Verbraucher innerhalb eines Haushaltes umsetzen. Diese Ansätze sind aber meist Bestandteil eines darüber gelagerten Konzeptes (sei es nun aus bilanzieller oder netzoptimierender Sicht), daher werden sie an dieser Stelle nicht dediziert ausgeführt.

Im Folgenden wird ein kurzer Überblick über exemplarische Ansätze der beiden erstgenannten Kategorien gegeben.

Mit dem PowerMatcher-Ansatz [19] wurde bereits früh ein auktionsbasiertes Agentensystem für das dezentrale Energiemanagement entwickelt. Jeder Anlage ist ein lokaler Anlagenagent zugeordnet, der als Kontrollagent eine ökonomische Optimierung für diese Anlage anstrebt. Dieser Agent kauft oder verkauft gemeinsam mit den Agenten seines Verbundes Elektrizität an elektronischen Märkten. Wurde das Konzept zunächst für den lokalen Abgleich von Erzeugung und Verbrauch konzipiert, so findet sich in mit der Erweiterung um den objective agent die Möglichkeit, andere Zielfunktionen zu hinterlegen, die z. B. die Geschäftslogik eines virtuellen Kraftwerks abzubilden. Der PowerMatcher-Ansatz wird bereits in mehreren Feldversuchen erfolgreich eingesetzt [20].

Ein ähnlicher Ansatz wird in DEZENT verfolgt [21, 22]: Hier schließen sich dezentrale und teilweise regenerative stochastische Energieumwandlungsanlagen unter Berücksichtigung technischer, wirtschaftlicher und ökologischer Randbedingungen zu regionalen Bilanzkreisen zusammen. Mit einem so gearteten verteilten agentenbasierten Verhandlungssystem für Verbraucher und Erzeuger sollen kurzfristige Freiheitsgrade in der Erzeugung und beim Verbrauch für eine Glättung der Lastprofile nutzbar gemacht werden. Ebenso wie bei PowerMatcher handelt es sich um ein System ohne prädiktive Planung, d. h. es wird nur für die nächste Verhandlungsperiode (hier: 500 ms) gehandelt. Eine Berücksichtigung der Netztopologie wird vorgesehen, indem die Hierarchie des Netzes über den Agententyp Bilanzgruppenmanager als aggregierende Einheit an den jeweiligen Transferstellen der Spannungsebenen abgebildet wird.

[23] widmet sich insbesondere der Frage, wie in dezentralen agentenbasierten Betriebsführungskonzepten die lokalen Anlagenrestriktionen informationstechnisch abgebildet werden können. In dieser Arbeit wird ein agentenbasiertes Energiemanagement mit einer Abbildung lokaler Restriktionen mittels Support Vector Machines (SVM) kombiniert. Ebenfalls unter besonderer Beachtung anlagenlokaler Restriktionen wurde das agentenbasierte Energiemanagementkonzept holonischer virtueller Kraftwerke entwickelt [24], deren Besonderheit in der dynamischen Reorganisation der internen Agenten-Organisation zur Minimierung des Kommunikationsaufwandes besteht. Im Gegensatz zu den anderen Arbeiten ergänzen sich hier prädiktive und reaktive Einsatzplanung zur Integration in die Konzepte der heutigen Energiewirtschaft.

Eine echte Substitution thermischer Kraftwerke durch dezentrale (erneuerbare) Energieumwandlungsanlagen kann nur dann erfolgen, wenn auch systemstabilisierende Aufgaben durch dezentrale Anlagen übernommen werden. Die Arbeiten aus dem Bereich agentenbasierter netzbezogener Betriebsführungskonzepte orientieren sich häufig an diesem Anspruch und können weiter nach detaillierten betrieblichen

Führungsgrößen und -zielen unterteilt werden wie z. B. Frequenz- und Spannungsregelung, Engpassmanagement, Microgrid-Betrieb und Schwarzstartfähigkeit. Die folgende Darstellung kann daher nur einen Ausschnitt der Arbeiten darstellen.

In [25] wird ein Ansatz zu agentenbasierten Steuerung der Frequenz vorgestellt, der auf einem ähnlichen Prinzip wie DEZENT basiert und diesen um die lokale Messung und Anpassung der Frequenz erweitert. In Simulationen kleinerer Netzbereiche zeigt der Ansatz gute Ergebnisse; die Dämpfung eines zu erwartenden überregionalen Schwingungsverhaltens ist noch nicht abgebildet, soll jedoch in weiteren Arbeiten berücksichtigt werden.

Die Spannungshaltung muss – im Gegensatz zur Frequenzhaltung – lokal erfolgen, weshalb dezentrale Anlagen dafür sehr geeignet sind, sofern sie die technischen Grundvoraussetzungen (Fähigkeit zur Blindleistungsbereitstellung) erfüllen. Im Niederspannungsnetz wird bisher die Spannung weder erfasst noch geregelt, was immer häufiger zu Problemen im Netzbetrieb führt, die erst dann erkennbar werden, wenn sich entsprechend gegen Über- bzw. Unterspannung geschützte Komponenten vom Netz trennen. In der Dissertation von Richardot [26] wurde ein agentenbasierter Ansatz zur hierarchisch koordinierten Anpassung der Blindleistungslieferung durch dezentrale Komponenten vorgestellt. Dabei werden Netzbezirke zu Regelungszonen zusammengefasst, innerhalb derer dedizierte Messpunkte (sogenannte pilot busses) zur Ermittlung der erforderlichen Blindleistung platziert werden. Insbesondere die optimale Messstellenkonfiguration stellt dabei eine große Herausforderung dar [27].

Unter der Thematik der koordinierten Lastflusssteuerung werden Arbeiten zusammengefasst, die im Rahmen eines Engpassmanagements eine gleichmäßige Nutzung der Netze innerhalb zulässiger Betriebsgrenzen verfolgen. So können z. B. die angeschlossenen Energieumwandlungsanlagen so gesteuert werden, dass das Netz innerhalb der zulässigen Grenzen betrieben wird. In [28] wird ein agentenbasierter und verteilter Ansatz für ein solches Verfahren beschrieben. Möglich ist es weiterhin, die Leistungsflüsse innerhalb des Netzes mit Hilfe sogenannter Leistungsflussregler (z. B. FACTS-Geräte oder konventionelle Querregler) durch Anpassung von Spannung oder Impedanz zu beeinflussen und so die Netze möglichst gleichmäßig zu beladen. Ein agentenbasierter und verteilter koordinierter Ansatz zur Lastflusssteuerung im Übertragungsnetz wird in [29] beschrieben.

2.5.3 Evaluation agentenbasierter Betriebsführungskonzepte

Die vorgestellten Arbeiten zur agentenbasierten Betriebsführung unterstreichen das große Interesse an der Agententechnologie und deren Einsatz in der Energieversorgung. Viele der angeführten Arbeiten betrachten jedoch häufig nur einzelne Teilprobleme oder stellen eine proprietäre Individuallösung dar, was insbesondere auf den Haushalts- und Endkundenbereich zutrifft. Eine Vielzahl von unterschiedlichen und untereinander nicht abgestimmten Einzellösungen kann aber bei einem konsequenten Umbau der Energieversorgung hin zu einem dezentral geführten Sys-

tem zu ungewollten Nebeneffekten führen, die letztlich die Versorgungssicherheit gefährden können. Entsprechend ist der Einsatz geeigneter Methoden und Werkzeuge notwendig, die die Charakteristik der zukünftig „smarten“ bzw. „intelligenten“ Energieversorgung repräsentieren können und einen wertfreien und kostengünstigen Versuchsraum für die systematische Entwicklung zukünftiger Regeln und Vereinbarungsmechanismen bieten. Einen vielversprechenden Ansatz stellen hier agentenbasierte Simulationen dar, da sie dezentrale Systeme inhärent repräsentieren.

Grundsätzlich besteht eine agentenbasierte Simulation aus der Modellierung eines Szenarios als Agentenmodell. Aktive Einheiten im Modell werden dabei als Agenten abgebildet, die mit ihren Verhaltensweisen die simulierte Umgebung beeinflussen können, in der sie eingebettet sind [30]. Wie bei anderen Simulationsmethoden auch sollen mit dieser Art der Simulation Schlüsse gezogen werden können, die dem Verhalten eines realen Systems entsprechen und die das empirische bzw. normative Verständnis für vorgegebene Szenarien erhöhen können.

Jedoch gibt es im Vergleich zu konventionellen Simulationsmethoden einen entscheidenden Unterschied, der in Hinblick auf die Simulation einer dezentralen Energieversorgung sehr vorteilhaft ist. Dieser besteht darin, dass das hohe Maß an Individualität und Unabhängigkeit der beteiligten Teilsysteme berücksichtigt werden kann, was einen nahezu beliebigen Detaillierungsgrad erlaubt. So lassen sich verschiedene Aspekte und Verhaltensweisen der beteiligten Komponenten (bzw. der Agenten) im Kontext eines Gesamtsystems vergleichen und beurteilen [31].

Die Entwicklung einer agentenbasierten Simulation stellt Entwickler aber auch vor Herausforderungen bei der Modellierung, der Implementierung und der Ausführung des entwickelten Agentensystems. Dies beginnt bei der schon erwähnten Frage nach dem notwendigen Detaillierungsgrad der einzelnen Komponenten und der Frage, welches Verhaltensweisen hinreichend bzw. notwendig sind. Der Aufwand zur Entwicklung einer agentenbasierten Simulation ist bezüglich der funktionalen Anforderungen vergleichbar mit der Entwicklung eines in der Realität eingesetzten agentenbasierten Systems. Hinzu kommen Aufwände zur Schaffung einer virtuellen Umgebung und der simulierten Verhaltensweisen einzelner Komponenten [32].

Die Idee und der Einsatz agentenbasierter Simulationen in der Energiewirtschaft ist ebenfalls nicht neu, wobei in den wissenschaftlichen Arbeiten jedoch zwischen ökonomischen und technischen Simulationen differenziert werden muss: Während in ökonomischen Simulationen häufig deliberative Eigenschaften von Agenten verwendet werden, um die Entscheidungsprozesse einzelner Marktteilnehmern abzubilden, konzentrieren sich vor allem die aktuellen technischen Arbeiten auf die Entwicklung von Smart Grids. Eine kurzen Überblick, Hinweise zu weiterführender Literatur sowie Betrachtungen zu eingesetzten Software-Werkzeugen für technische agentenbasierte Simulationen finden sich in [33].

Im Kontext der Entwicklung agentenbasierter Betriebsführungssysteme erscheint die agentenbasierte Simulation als nützliches Mittel, um eine Evaluation und Validierung neuer Betriebsführungskonzepte zu gewährleisten bevor diese in der Realität eingesetzt werden. Überdies bietet eine agentenbasierte Simulati-

on, durch die mögliche Abbildung von Betriebsmitteln aus realen Szenarien auf Agenten in simulierten Szenarien eine systematische Vorgehensweise an, die eine Übertragung von zuverlässigen Verhaltensweisen auf in der Realität agierende Agenten erlaubt.

2.6 Forschungsaktivitäten, -fragen und -bedarf

Die vorangegangen Abschnitte haben aufgeführt welche Herausforderungen im Bereich der zukünftigen Energieversorgung bestehen und welches Potenzial ein agentenbasiertes Energiemanagementsystem besitzt. Dennoch besteht weiterer Forschungsbedarf in der Herausarbeitung des Nutzens von Agentensystemen und -technologien unter realen, zumeist sicherheitskritischen Anforderungen in zukünftigen Energiesystemen. Eine wesentliche Herausforderung ist dabei der Entwurf sowie die Analyse solcher Systeme mit aus der Informations- und Kommunikationstechnologie stammenden und dort etablierten Modellierungs- und Implementierungstechniken sowie die Akzeptanz innerhalb eines interdisziplinären Smart Grid Forschungskreises. Damit dies gelingt sind weitere Forschungsbemühungen sowohl im Bereich der (Selbst-) Organisation und Interaktion von Agenten, als auch im Bereich der hardwarenahen Implementierung notwendig.

Das Antwortzeitverhalten bei der Interaktion von Agenten wird hierbei eine entscheidende Rolle spielen. Zuvor ist jedoch zu klären, was Echtzeitfähigkeit im Kontext von Smart Grid Anwendungen zu bedeuten hat und wie bzw. ob diese mit vertretbarem Aufwand und unter realen Bedingungen umsetzbar ist. Diese Frage gilt es insbesondere in Hinblick auf die vorhandene bzw. benötigte Kommunikationsinfrastruktur hin zu überprüfen. Weiter stellt sich in diesem Zusammenhang die Frage, in wie weit sich Reichweite und Daten- bzw. Informationsmengen bei der Kommunikation zwischen Agenten minimieren lassen.

Auch stellen sich Fragen hinsichtlich der Mechanismen und der Semantik kommunizierender Agenten. Standards der FIPA⁵ und des IEEE dienen schon heute vielen Agentensystemen als Kommunikationsgrundlage. Hier ist zu klären, ob diese als Grundlage für ein agentenbasiertes Smart Grid genügen oder ob es hier einen Weitentwicklungsbedarf gibt: Lassen sich diese Standards beispielsweise mit etablierten Automatisierungsstandards (z. B. OPC-UA) kombinieren?

Auch besteht Forschungsbedarf bei der Harmonisierung von Datenstrukturen und Ontologien. Als Grundlage für eine Abstimmung könnten dabei die derzeit in der Energietechnik zur Anwendung kommenden Standards wie IEC61850, IEC61970/61968 (CIM⁶) oder EDI@Energy dienen [34]. Diese Abstimmung scheint gerade vor dem Hintergrund des interdisziplinären Charakters notwendiger Forschung und Entwicklung im Smart Grid sowie dem Vorhaben verschiedene Energienetze für Strom, Wärme und Gas systemtechnisch zu integrieren, eine

⁵ Foundation for Intelligent Physical Agents.

⁶ Common Information Modell.

entscheidende Rolle zu spielen. Eine mögliche Selbstdaption verteilter agentenbasierter Systeme erfordert hierzu ein hohes Maß an Standardisierung von Informationen.

Auch bei der Implementierung und dem Design von Agenten besteht noch Forschungsbedarf. Zwar werden in vielen agentenbasierten Systemen Lern- und Anpassungsfähigkeit implementiert, doch fehlen noch Ansätze die eine einfache automatisierte Umgebungsmodellanpassung auf das dem Agenten unterlagerte physikalische System erlauben. Gerade im privaten Anwendungsbereich ist die automatisierte Anpassung des Umgebungsmodells und die damit verbundene individuelle Agentenstrategien von großer Bedeutung für die Akzeptanz durch den Nutzer [35].

Abhängig vom konkreten lokalen Anwendungsfall für ein smartes Mittel- oder Niederspannungsnetz gilt es qualitative und quantitative Kriterien zu finden, die eine objektive Bewertung hinsichtlich der in Abschn. 2.3 erläuterten Führungsstrukturen erlauben. Hierbei muss z. B. die Frage beantwortet werden, welche Vor- bzw. Nachteile dezentrale agentenbasierte Systeme gegenüber zentralen oder dezentral-hierarchisch organisierten Systemen aufweisen.

2.7 Zusammenfassung

Dieser Artikel zeigt potenzielle Anwendungsfelder von Agentensystemen in der Energietechnik auf und diskutiert bereits entworfene und implementierte Ansätze. Hierfür werden einführend die grundlegenden Eigenschaften des elektrischen Energiennetzes und seiner Komponenten sowie die daraus resultierenden Herausforderungen an die Automatisierungstechnik diskutiert. Als mögliche Strukturen für die Betriebsführung werden zentral-hierarchische, dezentral-hierarchische oder vollständig dezentrale Führungsstrukturen gegenübergestellt. Weiter diskutiert dieser Artikel funktionale und nichtfunktionale Anforderungen an zukünftige Netzeleitsysteme, verschiedene Optimierungsansätze und -ziele sowie Ansätze zur Evaluierung von agentenbasierten Betriebsführungsstrukturen.

Vor dem Hintergrund der avisierten Modifikationen des deutschen – sowie des europäischen Energieversorgungssystems werden hierbei unter anderem folgende Anforderungen an ein zukünftiges Netzeleitsystem definiert, die im Besonderen durch Agentensysteme abgedeckt werden können: Restrukturierungsfähigkeit, Skalierbarkeit, Robustheit, Verfügbarkeit, Echtzeitfähigkeit und Fähigkeit zur Re-Allokation einzelner Software-Fragmente. In Hinblick auf Echtzeitaspekte, Kommunikationsstandards, Normierungen und die Kooperationsfähigkeit von Agentensystemen wird ein wesentlicher Forschungsbedarf gesehen.

In zukünftigen Veröffentlichungen der Untergruppe Energie des GMA Fachausschusses 5.15 sollen die Bewertung agentenbasierter Betriebsführungskonzepte sowie die Themenfelder Gas und Wärme mit in die Betrachtungen integriert werden. Darüber hinaus wird sich die Gruppe der Identifikation und Evaluierung weiterer potenzieller Anwendungsgebiete aus der Energietechnik widmen.

Literatur

- [1] Lazarewicz, M.: Status of flywheel storage operation of first frequency regulation plants. Technical report, Beacon Power Corporation, S. 10 (2011)
- [2] Crow, M.L., Arsoy, A., Liu, Y., Ribeiro, P.F., Johnson, B.K.: Energy storage systems for advanced power applications. Proc. IEEE **89**, 1744–1756 (2001)
- [3] PROFIBUS Nutzerorganisation e. V. Profienrgy. online, 5 (2012)
- [4] RWE. Rwe smart home. online, 5 (2012)
- [5] What is KNX? Knx association [official website]. online, 05 (2012)
- [6] EWE AG. Ewe trio smartbox, 04 (2012)
- [7] ENEL. Smart metering system. online, 4 (2012)
- [8] Southern California Edison. Edison's smarter meter. Technical report, Southern California Edison, 04 (2012)
- [9] Iiro Rinta-Jouppi. Smart meter – a field report from sweden, 5 (2009)
- [10] Kästner, A., Kießling, T.: Energie in 60 Minuten: Ein Reiseführer durch die Stromwirtschaft. VS Verlag für Sozialwissenschaften, Wiesbaden (2009)
- [11] Strauss, P., Degner, T., Schmid, J. (Hrsg.): Dispower Distributed Generation with High Penetration of Renewable Energy Sources (2006)
- [12] Iwes, kombikraftwerk, 05 (2012)
- [13] Erge, T., Hollinger, R.: Integrative energy market as system integrator of decentralized generators (2012)
- [14] Wooldridge, M., Jennings, N.R.: Intelligent agents: theory and practice. Knowledge Eng. Rev. (1994). Submitted to Revised
- [15] VDI/VDE 2653 Blatt 1–3. Agentensysteme in der Automatisierungstechnik – Grundlagen, Entwicklung, Anwendung (2010–2012)
- [16] Wooldridge, M.: An Introduction to MultiAgent Systems, 2. Aufl. Wiley & Sons, Hoboken (New Jersey) (2009)
- [17] Rao, A.S., Georgeff, M.P.: Bdi agents: from theory to practice. In: Lesser, V.R., Gasser, L. (Hrsg.) ICMAS, S. 312–319. The MIT Press, Cambridge (Massachusetts) (1995)
- [18] Sudeikat, J., Braubach, L., Pokahr, A., Lamersdorf, W., Renz, W.: Validation of bdi agents. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (Hrsg.) PROMAS. Lecture Notes in Computer Science, Bd 4411, S. 185–200. Springer, Berlin Heidelberg (2006)
- [19] Kok, K., Warmer, C., Kamphuis, R., Mellstrand, P., Gustavsson, R.: Distributed control in the electricity infrastructure. Proc. Int. Conf. Future Power Syst. **2005** (2005)
- [20] Kamphuis, R., Roossien, B., Bliek, F., van der Noort, A., van der Velde, J., de Wit, J., Eijgelaar, M.: Architectural design and first results evaluation of the PowerMatching city field test. 4th International Conference on Integration of Renewable and Distributed Energy Resources, EPRI. Albuquerque (2010)
- [21] Lehnhoff, S.: Dezentrales vernetztes Energiemanagement – Ein Ansatz auf Basis eines verteilten Realzeit-Multiagentensystems. Vieweg + Teubner, Wiesbaden (2010)
- [22] Lehnhoff, S., Krause, O., Rehtanz, C., Wedde, H.F.: Dezentrales autonomes Energiemanagement Distributed Autonomous Power Management. at – Automatisierungstechnik **3**, 167–179 (2011)
- [23] Platt, G.: The decentralised control of electricity networks – intelligent and self-healing systems. Grid-Interop-Forum **2007**, 1–6 (2007)

- [24] Tröschel, M.: Aktive Einsatzplanung in holonischen Virtuellen Kraftwerken. Universität Oldenburg, Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften (2010)
- [25] Linnenberg, T., Wior, I., Schreiber, S., Fay, A.: A market-based multi-agent-system for decentralized power and grid control. 16th IEEE International Conference on Emerging Technologies and Factory Automation (IFTA2011), Toulouse, France (2011)
- [26] Richardot, O.: Réglage Coordonné de Tension dans les Réseaux de Distribution à l'aide de la Production Décentralisée. PhD thesis, Grenoble, France (2006)
- [27] Richardot, O., Bésanger, Y., Radu, D., Hadjsaid, N.: Optimal location of pilot buses by a genetic algorithm approach for a coordinated voltage control in distribution systems. 2009 IEEE Bucharest PowerTech, S. 1–7. Bucharest, Romania (2009)
- [28] Miller, S., Ramchurn, S.D., Rogers, A.: Optimal decentralised dispatch of embedded generation in the smart grid. *AAMAS* **2012**(June), 4–8 (2012)
- [29] Häger, U., Lehnhoff, S., Rehtanz, C.: Verteilte koordinierte Lastflusssteuerung in elektrischen Energieübertragungsnetzen. *at – Automatisierungstechnik* **3**, 153–160 (2011)
- [30] Drogoul, A., Vanbergue, D., Meurisse, T.: Multi-agent based simulation: Where are the agents? In: Simão Sichman, J., Bousquet, F., Davidsson, P. (Hrsg.) *MABS. Lecture Notes in Computer Science*, Bd. 2581, S. 1–15. Springer, Berlin Heidelberg (2002)
- [31] Klügl, F.: A validation methodology for agent-based simulations. In: Wainwright, R.L., Hadad, H. (Hrsg.) *SAC*, S. 39–43. ACM (2008)
- [32] Derksen, C., Branki, C., Unland, R.: Agent.gui: A multi-agent based simulation framework. In: Ganzha, M., Maciaszek, L.A., Paprzycki, M. (Hrsg.) *FedCSIS*, S. 623–630 (2011)
- [33] Derksen, C., Branki, C., Unland, R.: A framework for agent-based simulations of hybrid energy infrastructures. In: Ganzha, M., Maciaszek, L.A., Paprzycki, M. (Hrsg.) *FedCSIS*, page to be published (2012)
- [34] Davidson, E.M., Catterson, V., Dimeas, A.L., Hatziargyriou, N.D., Ponci, F., Funabashi, T., McArthur, S.D.J.: Multi-agent systems for power engineering applications—part ii: Technologies, standards, and tools for building multi-agent systems (2007)
- [35] Rohbogner, G., Fey, S., Hahnel, U.J.J., Benoit, P., Wille-Haussmann, B.: What the term agent stands for in the smart grid definition of agents and multi-agent systems from an engineer's perspective. *FedCSIS*, page to be published (2012)

Teil II

Konzepte und Methoden

Kapitel 3

Identifikation und Umsetzung von Agenten zur Fabrikautomation unter Nutzung von mechatronischen Strukturierungskonzepten

Arndt Lüder und Matthias Foehr

Zusammenfassung Zum Zweck der Verbesserung von Effizienz und Korrektheit beim Entwurf von flexiblen Produktionssystemen wurden in den letzten Jahren verstärkt Forschungs- und Entwicklungsanstrengungen unternommen, die die unterschiedlichsten Resultate erbracht haben. Zwei Ergebnisse sind agentenbasierte Steuerungsarchitekturen und mechatronische Entwurfsmethoden. Sie entstammen unterschiedlichen Forschungsrichtungen, sind jedoch gemeinsam nutzbringend einsetzbar. Im Rahmen dieses Beitrages wird kurz angerissen, wie der Stand der Entwicklung in den genannten Forschungs- und Entwicklungsbereichen ist. Auf dieser Grundlage wird dargestellt, wie beide Gebiete kombiniert und gemeinsam genutzt werden können. Es wird aufgezeigt, welche Vorteile und welche Konsequenzen diese Kombination besitzen kann.

3.1 Einleitung

Produktionssysteme der verschiedenen Industriebereiche stehen heute vor einem gemeinsamen Problem. Sie sollen möglichst flexibel an sich ändernde Bedingungen der Anlagen Nutzung anpassbar sein [1]. Dabei sollen sie sich je nach Anwendungsbereich flexibel an sich ändernde Produktsortimente und Ausbringungsmengen, genutzte Produktionsressourcen und anwendbare Produktionstechnologien anpassen und die effizienteste Produktionsweise sicherstellen [2]. Diese Anforderungen haben umfassenden Einfluss sowohl auf die Ausprägung der Produktionssysteme selbst, als auch auf ihren Entwurf.

Arndt Lüder (✉), Matthias Foehr
Fakultät Maschinenbau, Institut für Mobile Systeme und Institut für Arbeitswissenschaften,
Fabrikautomatisierung und Fabrikbetrieb, Otto-v.-Guericke Universität, Universitätsplatz 2,
30106 Magdeburg, Deutschland
e-mail: arndt.lueder@ovgu.de, matthias.foehr@ovgu.de

Bedeutender Bestandteil der Produktionssysteme sind ihre Steuerungssysteme. Sie steuern auf verschiedenen Ebenen und unter Berücksichtigung unterschiedlicher Abstraktionsgrade und Steuerungsmodelle die Produktion von der Zuweisung von Personal, Material und Ressourcen zu einem Produktionsauftrag bis hin zur Ansteuerung von Sensoren und Akten [3, 4]. Auch auf die Steuerungssysteme wirken sich die Anforderungen an Produktionssysteme aus. Sie müssen die notwendige Flexibilität der Produktionssysteme sicherstellen [5] sowie in den Entwurfs- und Entstehungsprozessen möglichst effizient und fehlerfrei entworfen und erstellt werden können [6].

Für den Entwurf von Steuerungssystemen sind sowohl die Architektur des zu steuernden Systems, die Architektur der Steuerung als auch der Entwurfsprozess und die in ihm ausgeführten Entwurfsschritte und genutzten Entwurfswerkzeuge von Bedeutung [7]. Dieses Problem ist allgemein bekannt und hat dazu geführt, dass die unterschiedlichsten Forschungs- und Entwicklungsvorhaben unterschiedliche Bereiche aus Steuerungsarchitektur und Steuerungsentwurf sowie die sie einrahmenden Problembereiche untersuchten und unterschiedliche Steuerungsarchitekturen, Steuerungstechnologien und -geräte, sowie Steuerungsentwurfsprozesse entwickelt haben.

Auf eine ausführliche Analyse der Gesamtheit der gemachten Entwicklungen soll auf Grund ihres Umfangs dieser Stelle verzichtet werden. Zwei der nach Ansicht der Autoren bedeutenden und zukunftsträchtigen Entwicklungen jedoch sollen an dieser Stelle betrachtet und zusammengeführt werden. Dies sind die auf Agentensystemen basierenden Steuerungsarchitekturen [8] und die auf der Mechatronik aufbauende Architektur für Produktionssysteme [9] und der dazugehörige Entwurfsprozess [10]. Es soll aufgezeigt werden, wie beide Entwicklungsrichtungen zusammenpassen, wie sie voneinander profitieren können, welchen Nutzen Anwender von mechatronikorientierten Agentensystemen erwarten können und wie diese Kombination auch andere Architekturen (z. B. Holone [11] oder PABADIS Architekturen [7]) abdecken kann.

Insbesondere soll in diesem Beitrag herausgestellt werden, das mechatronikorientierte Agentensysteme einen neuen Grad der Wiederverwendung von Entwurfsergebnissen von Produktionssystemen ermöglichen, der sowohl in den Phasen des Entwurfsprozesses von Produktionssystemen durch die einfache Einbindung von Whiteboxes oder Blackboxes, als auch in der Installation und Inbetriebnahme von Produktionssystemen durch „Plug-and-Participate“-Fähigkeiten und garantierter Interoperabilität deutlich wird.

Zu diesem Zweck gliedert sich dieser Beitrag wie folgt. Im nachfolgenden zweiten Abschnitt werden die Grundlagen dieser Arbeit betrachtet. Dabei werden der Stand der Technik agentenbasierter Steuerungssysteme für Produktionssysteme und der Stand der Technik des mechatronischen Entwurfs von Produktionssystemen dargestellt. Diesen Betrachtungen schließen sich die Beschreibung der Struktur, des Entwurfsprozesses und der Vorteile mechatronikorientierter Agentensysteme an. Mit einer kurzen Zusammenfassung endet dieser Beitrag.

3.2 Ausgangslage

Nachfolgend sollen zunächst der Stand der Forschung in den Bereichen Agentensysteme zur Steuerung von Produktionssystemen sowie mechatronischer Entwurf von Produktionssystemen analysiert und bewertet werden.

3.2.1 *Agentensysteme zur Steuerung von Produktionssystemen*

Die Anwendung von Agentensystemen zur Steuerung von Produktionssystemen wird seit den letzten zwei Jahrzehnten des 20. Jahrhunderts intensiv untersucht [8, 12]. Dabei ist eine Vielzahl an Architekturen entstanden, unter denen die durch Agenten umgesetzten Holonen Steuerungsarchitekturen [11] und die PABADIS Steuerungsarchitekturen [7] nach Ansicht der Autoren eine weite Verbreitung gefunden haben (siehe z. B. [8, 13–16]).

Alle diese Steuerungsarchitekturen gehen von Agenten als Entitäten aus, die als reine Softwareagenten oder als Kombination aus Software und Hardware gestaltet sind und wichtige Eigenschaften besitzen [17]. Dabei werden immer wieder die Implementierung des Agentenverhaltens auf Basis eines Umweltnutzungsmodells sowie die Reaktivität, die Proaktivität und die Mobilität von Agenten hervorgehoben, wenn es um ihre Bedeutung für Produktionssysteme geht.

Grundlage des Verhaltens von Agenten in Produktionssystemen ist ein dediziertes, im Agenten implementiertes Modell seiner Umwelt. Er kann unter Nutzung dieses Modells das Verhalten seiner Umwelt erkennen und entsprechend seiner Ziele und seines Handlungsspielraumes darauf reagieren. Zur Umsetzung dieses Verhaltens wird im Bereich der agentenbasierten Steuerungsarchitekturen die sogenannte Believe-Desire-Intention (BDI) Architektur für Agenten [18] verwendet. Nimmt die Umwelt eines Agenten einen bestimmten Zustand ein, der für den Agenten eine bestimmte Bedeutung besitzt, so wird er durch eigene Aktivitäten darauf reagieren (Reaktivität). Dies ermöglicht in Produktionssystemen zum Beispiel die Implementierung von Agenten, die auf Anforderung ihrer Umwelt ein bestimmtes Verhalten zeigen und eine Bearbeitung ausführen. (Ressourcenagenten). Ebenso können Agenten eigene Ziele anstreben/verfolgen und zur Erreichung dieser ein (zielgerichtetes) spezielles Verhalten aufzeigen (Proaktivität). Dies ermöglicht in Produktionssystemen die Implementierung Agenten, die selbstständig Aufträge, die von außerhalb des Produktionssystems kommen, gezielt auszuführen und dazu Verhalten von anderen Agenten abzufordern. Ein Beispiel hierfür sind Agenten, die die Ausführung von Produktionsaufträgen steuern (Auftragsagenten) [7].

Mobilität spielt bei Agenten als Eigenschaft eine besondere Rolle. Agenten können auf verschiedenste Art und Weise mobil sein. So könne sie diese Eigenschaft durch ihre Hardware erreichen (Agentifizierte Transporteinheit/AGV). Ebenso können auch reine Softwareagenten auf der Basis ihrer Softwaretechnologie mobil gemacht werden (Migration). Eine Klassifikation bezüglich der Beweglichkeit von Agenten ist in Abb. 3.1 dargestellt.

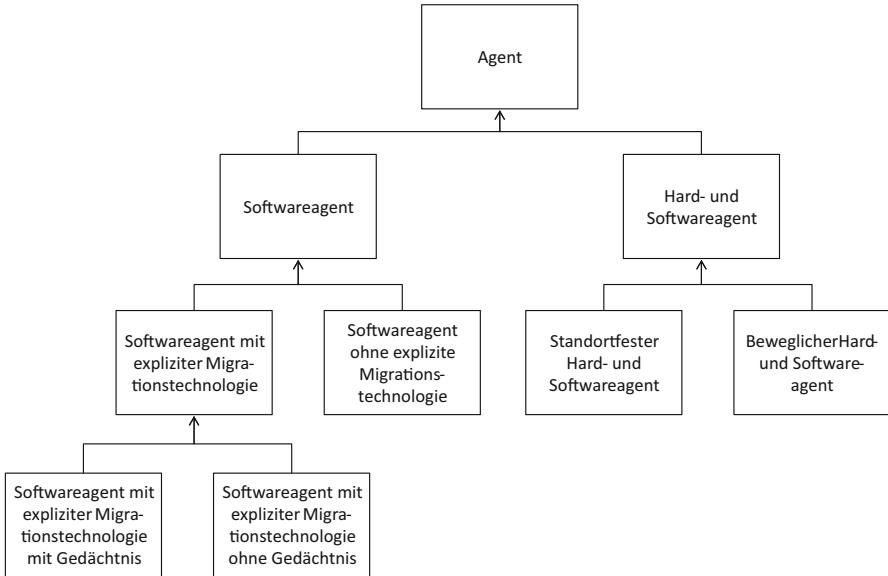


Abb. 3.1 Agentenklassifikation bezüglich Mobilität

In den verschiedenen Agentenarchitekturen für die Steuerung von Produktionsystemen sind Agenten, die für die verschiedenen Steuerungsentscheidungen notwendig sind, ähnlich gestaltet und besitzen vergleichbare Eigenschaften. So gibt es in den meisten Architekturen Agenten, die für die Produktionsaufträge und ihre Steuerung zuständig sind. Diese Agenten sind immer proaktiv und teilweise mobil. Ressourcenagenten dienen demgegenüber der Steuerung von Produktionsressourcen und sind reaktiv und nicht mobil. Genau so sind die Agenten zur Verwaltung und Bereitstellung von Produktwissen (insbesondere Material und Fertigungsschritte) sowie die Agenten zur Unterstützung von Planungsaufgaben reaktiv und nicht mobil.

In Tab. 3.1 wird noch einmal gegenübergestellt, wie diese vier Agententypen in einigen Agentenarchitekturen ausgeprägt sind. Hierbei wurde neben Agentenarchitekturen, die Strukturierungsprinzipien mittels Holonen (HMS Multi-Agentensysteme) folgen, auch solche gegenübergestellt, die PABADIS Strukturierungsprinzipien folgen. Siehe dazu auch [19]. Die Aufzählung strebt nicht nach Vollständigkeit.

Zur Sicherstellung der Flexibilität von Produktionssystemen wird von Agentensystemen erwartet, dass sie sich ebenfalls an die sich ändernden Aufgaben anpassen können. Dazu gibt es grundsätzlich mehrere Möglichkeiten.

Die bisher verbreitetste Variante der Flexibilisierung in Agentensystemen basiert auf der Nutzung generischer Agententypen mit einem generischen Basis-Umweltmodell, dass für die entsprechenden Aufgaben mit speziellem Wissen zielgerichtet erweitert wird. Im Allgemeinen werden Agenten nach dem BDI Prin-

Tab. 3.1 Agententypen und ihre Ausprägung in verschiedenen Architekturen

Agentenarchi- tekturen	PROSA [20] [21]	PABADIS [21]	ADACOR [22]	GRACE [23]	Eigenschaften
Produktions- auftragsagent	Auftragsholon	Auftragsagent	Auftragsholon	Produktagent	proaktiv, teilweise mobil
Ressourcen- agent	Ressourcen- holon	Maschinen- agent	Unterneh- mensholon	Ressourcen- agent	reakтив, nicht mobil
Agent für Pro- duktwissen	Produktholon	Auftragsagent	Produktholon	Produkttypen- agent	reakтив, nicht mobil
Planungs- agent	Unterstüt- zungsholon	Anlagenma- nagemen- tagent	Leiterholon	Ressour- cenagent, Unabhängiger Meta Agent	reakтив, nicht mobil

zip entwickelt, d. h. ihr Verhalten gründet sich auf ein Umweltmodell und eine Menge von kurz- und langfristigen Zielen, die unter Berücksichtigung des Umweltmodells angestrebt werden. Dabei können diese Ziele und das Umweltmodell sowohl statische als auch dynamische (sich im Lebenszyklus des Agenten ändernde) Anteile besitzen. Die dynamischen Anteile können durch den Agenten selbst, andere Agenten oder eine außerhalb des Agentensystems liegende Instanz geändert werden. Diese Änderungsfähigkeit kann genutzt werden, um den Agenten flexibel an sich ändernde Anforderungen anzupassen und dabei Eigenschaften wie self-adaptation oder self-learning umzusetzen.

Die erste Möglichkeit wird zum Beispiel in der PABADIS'PROMISE Architektur [7] oder in der GRACE Architektur [23] genutzt. So werden in der PABADIS'PROMISE Architektur zum einen generische Auftragsagenten mit der speziellen Beschreibung ihres Produktes und ihres Produktionsauftrages ausgestattet. Dies schließt auch spezifischen Steuerungscode für Produktionsressourcen mit ein. Zum einen kann das Umweltmodell zur Agentenlaufzeit durch andere Agenten oder von außerhalb des Systems geändert werden. Ebenso werden bei Veränderungen der Umweltbedingungen neue Agenten mit angepasster Erweiterung des Umweltmodells gestartet und alte, nicht mehr notwendige Agenten terminiert. Dem gegenüber stehen Maschinenagenten, die in ihrem Umweltmodell eine Menge von Steuerungscode für ihre Ressource besitzen. Gemeinsam können dann Auftrags- und Maschinenagenten eine jeweils für ein spezifisches Produkt optimierte Steuerungsapplikation für die Produktionsressourcen erstellen und ausführen lassen [24, 25].

Weitere Möglichkeit zur Flexibilisierung besteht in der Veränderbarkeit der Agenten selbst. Hier könnten modernste Softwaretechniken, die eine Veränderung des Agentenquellcodes ermöglichen, genutzt werden. Beispiele dafür sind das Klassenladen von Java. Derzeit sind jedoch keine Agentenarchitekturen bekannt, die diese Möglichkeit nutzen. Hier besteht noch weiteres Untersuchungspotenzial.

Für den Entwurfsprozess von agentenbasierten Produktionssteuerungsapplikationen existieren verschiedene Ansätze wie die GAIA Methode [26], die GAIA UML Methode [27] und die DACS Methode [28]. Die DACS Methode ist dabei die nach Ansicht der Autoren die Methode, die mit der mechatronischen Betrachtungsweise am besten vereinbar ist.

Die DACS Methode (Design of Agent-based Control Systems) hat zum Ziel, ein Agentensystem für Produktionssysteme zu entwerfen und zu spezifizieren [28–30]. Sie basiert dabei auf einem Vorgehen aus drei Schritten zur Analyse der Menge der notwendigen Steuerungentscheidungen, zur Identifikation von Agenten und zur Spezifikation der Interaktion zwischen Agenten.

Der erste Schritt der DACS Methode zielt auf die Analyse und Beschreibung der Menge von notwendigen Steuerungentscheidungen im Produktionssystem ab. Ausgehend von einer Festlegung der zu untersuchenden und zu steuernden Ebene in der Steuerungspyramide des Produktionssystems wird analysiert, welche Steuerentscheidungen für die korrekte Funktionsweise des Produktionssystems auf der betrachteten Ebenen notwendig sind. Ebenso werden die Abhängigkeiten identifiziert, die zu Interaktionen zwischen Steuerungentscheidungen führen müssen. Es wird dabei analysiert, welche Informationen in den Interaktionen ausgetauscht werden müssen. Im Ergebnis des ersten Schrittes entsteht ein Entscheidungsmodell.

Im zweiten Schritt der Methode erfolgt die Spezifikation der Agentenmenge für das Steuerungssystem. Zu diesem Zweck werden die einzelnen Steuerungsentcheidungen des Entscheidungsmodells gruppiert und einzelnen Agenten zugeordnet. Entsprechend ergeben sich notwendige Interaktionen zwischen den Agenten aus den Interaktionen zwischen Steuerungentscheidungen im Entscheidungsmodell. Bei der Definition von Agenten und Interaktionen muss darauf geachtet werden, welche Agenteneigenschaften für das korrekte Steuerungsverhalten notwendig sind, d. h. ob die Agenten eher als Anbieter von Funktionalitäten für andere Agenten (reakтив) oder als Nutzer von Funktionalitäten anderer Agenten (proaktiv) gestaltet sein müssen. Im Ergebnis des Schrittes entsteht ein Agentenmodell.

Im dritten und letzten Schritt der DACS Methode werden die notwendigen Interaktionen zwischen den Agenten konkretisiert. Dazu werden die Anforderungen der Interaktionen erhoben und mit den Möglichkeiten von Interaktionsprotokollen aus einer Bibliothek verglichen. Für jede Interaktion wird das am besten geeignete Protokoll ausgewählt und angepasst. Im Ergebnis des Schrittes wird das Agentenmodell um Interaktionsprotokolle erweitert.

Auf der Basis des Entscheidungsmodells und des Agentenmodells kann nun ein Steuerungssystem für ein Produktionssystem mit geeigneten Agentenplattformen wie JADE [31] oder AMES [32] implementiert werden.

3.2.2 Mechatronischer Entwurf von Produktionssystemen

Mechatronische Denk- und Arbeitsweisen sind im Bereich des Entwurfs technischer Systeme bereits seit den siebziger und achtziger Jahren des letzten Jahrhun-

derts verbreitet. Anfänglich unter dem Begriff der Feinwerkmechanik hat sich später der aus Japan stammende Begriff Mechatronik als die Ingenieurwissenschaft der methodischen Kombination von Mechanik und Elektrik/Elektronik durchgesetzt. In den letzten Jahren wird er durch die spätere Hinzunahme weiterer Entwurfsdisziplinen ergänzt [33, 34].

Initial wurde die Mechatronik für die Entwicklung von Produkten betrachtet, bei der die gezielte Kombination der verschiedenen ingenieurwissenschaftlichen Disziplinen einen Mehrwert für die Produkte erbringen sollte [35, 36]. Diese Kombination hat sich sehr schnell auch für die Strukturierung und den Entwurf von Produktionssystemen (und darüber hinaus) als nutzbringend herausgestellt [37–39].

In den letzten Jahren hat sich für den Begriff der Mechatronik ein Konsens in der Begriffsdefinition gebildet [33–39]. Dem entsprechend gilt:

Eine mechatronische Einheit ist ein abgeschlossenes System, das unter Nutzung von Sensoren, Aktoren und Steuerungsgeräten innerhalb einer geschlossenen Steuer- bzw. Regelkette ein definiertes (meist physikalisches) Verhalten innerhalb eines Produktionssystems bereitstellt. Dabei kombiniert die mechatronische Einheit zum einen Software (für die Steuerung) und Hardware (Mechanik, Elektrik, Elektronik, ...) sowie unterschiedliche weitere ingenieurwissenschaftliche Gebiete wie die Fluidik u. a. für eine optimale Funktionserbringung.

Ein mechatronisches System wird durch die gezielte Kombination/Vernetzung von mechatronischen Einheiten und mechatronischen Systemen über verschiedene Hierarchieebenen hinweg erstellt. Dabei besitzt jedes mechatronische System seine eigene steuerungstechnisch genutzte Informationsverarbeitung, die das optimale Zusammenwirken und die optimale Nutzung der unterlagerten mechatronischen Einheiten und Systeme sichert.

Die Unterscheidung von mechatronischen Einheiten und mechatronischen Systemen ergibt sich aus der Betrachtung der Hierarchie mechatronischer Einheiten und mechatronischer Systeme. Als mechatronische Einheiten werden üblicherweise die Blätter der Hierarchie betrachtet wogegen die weiteren Hierarchieebenen durch mechatronische Systeme gebildet werden.

Die Struktur und Vernetzung von mechatronischen Einheiten und mechatronischem System ist in Abb. 3.2 beispielhaft für eine Hierarchie aus zwei Ebenen dargestellt. In realen Anwendungen kann die Anzahl der Hierarchieebenen weitaus größer sein.

Nach [37, 39] kann die Struktur von mechatronisch organisierten Produktionssystemen als eine sechs Schichten umfassende Hierarchie betrachtet werden. Die unterste Schicht bilden mechanische oder elektrische Bauteile wie Metallschienen oder Kabel. Diese sind in Unterfunktionsgruppen enthalten, die in Kombination mit anderen Unterfunktionsgruppen einen Teil einer Funktion des Produktionssystems erbringen. Unterfunktionsgruppen werden dann zu Funktionsgruppen kombiniert. So werden beispielsweise Spanner zu Spannergruppen kombiniert um die Produktionsfunktion „Material festhalten“ zu realisieren oder einzelne Motoren und Getriebe zu Antriebssträngen um die Produktionsfunktion „Bewegen“ auszuführen. Funktionsgruppen stellen bereits größere Produktionsfunktionen bereit, die für

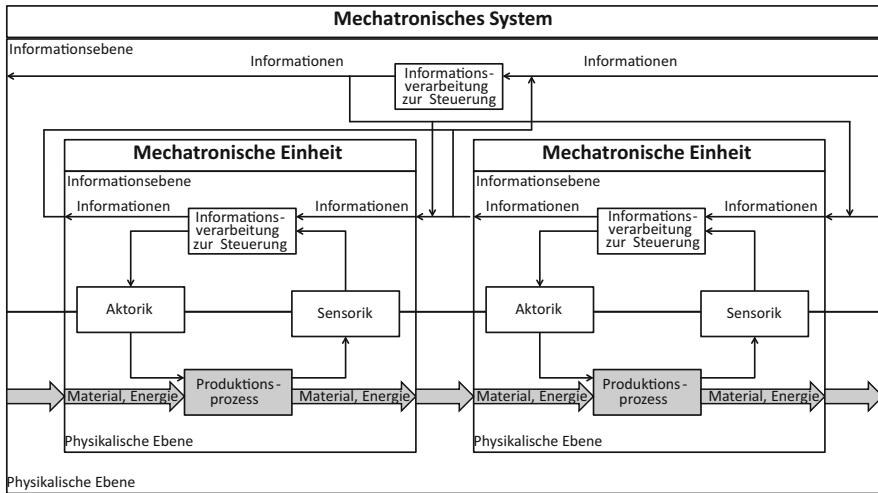


Abb. 3.2 Mechatronische Strukturierung

einen Produktionsschritt von Bedeutung sind. Ganze Produktionsschritte werden durch Hauptgruppen erbracht. Diese kombinieren Funktionsgruppen, wie es bei der Kombination von Antriebsträngen und Spannergruppen mit weiteren Funktionsgruppen innerhalb einer Werkzeugmaschine der Fall ist. Diese können gemeinsam eine Fräsfunktion an einem Werkstück erbringen. Hauptfunktionen können wiederum zu Zellen kombiniert werden, in denen mehrere Produktionsschritte erfolgen. So kann eine Werkzeugmaschine mit Fräsfunktionen und ein Roboter mit Handlungsfunktion zu einer Fräszelle kombiniert werden. Letztendlich werden dann mehrere Zellen zu Anlagen kombiniert, so wie etwa mehrere Fräszellen eine Motorenfertigung umsetzen können.

Üblicherweise werden Anlagen, Zellen, Hauptgruppen und Funktionsgruppen als mechatronische Systeme und Zellen, Hauptgruppen, Funktionsgruppen und Unterfunktionsgruppen als mechatronische Einheiten aufgefasst. Hier ist der Betrachtungswinkel wichtig, der die unterste betrachtete Ebene der mechatronischen Einheiten definiert. Dies ist in Abb. 3.3 dargestellt.

Der Entwurfsprozess für mechatronisch strukturierte Produktionssysteme erfolgt unter direkter Nutzung mechatronischer Einheiten und mechatronischer Systeme in der oben beschriebenen hierarchischen Struktur [10, 35, 40]. Dabei gibt es zwei Prozessteile. Der erste Prozessteil hat zum Ziel, ein auf eine bestimmte Produktionssaufgabe ausgerichtetes Produktionssystem zu erstellen. Dabei werden bereits vorentworfene mechatronische Einheiten und mechatronische Systeme (bzw. Teile von diesen) als Ausgangspunkt für das Entwurfsvorgehen aus einer Bibliothek entnommen und genutzt. Der zweite Prozessteil dient der Erstellung von getesteten mechatronischen Einheiten bzw. Systemen (sowohl physischer als auch virtueller) zur Wiederverwendung im ersten Prozessteil. Dies erfolgt zum Teil unter Abstraktion der Ergebnisse des ersten Prozessteiles sowie unter Einbeziehung von Wissen

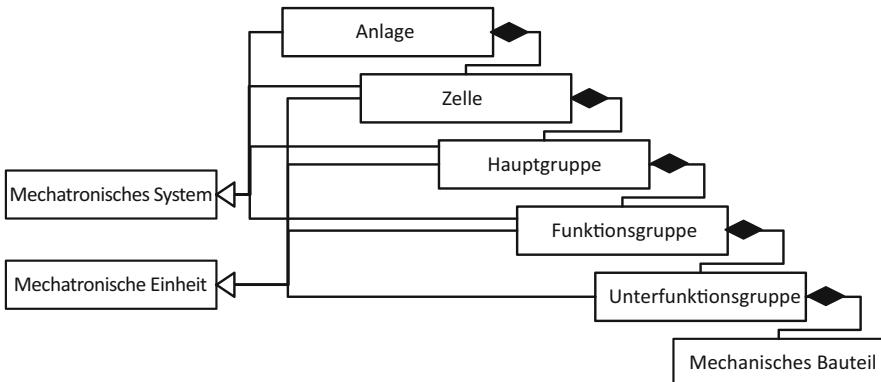


Abb. 3.3 Mechatronische Hierarchie von Produktionssystemen

über die Industriedomaine, in denen die erstellten Produktionssysteme genutzt werden sollen [41].

Die Produktionssteuerungsapplikationen ist in einem mechatronisch strukturierten Produktionssystem auf die verschiedenen Informationsverarbeitungen der mechatronischen Einheiten und Systeme verteilt. Diese Verteilung kann sowohl real auf verschiedenen Hardwaresystemen sein, als auch virtuell in einer Steuerungshardware. Dabei orientieren sich die einzelnen Ebenen mit den sie betreffenden Steuerungssentscheidungen teilweise an den einzelnen, sie betreffenden Ebenen der Automatisierungspyramide. So können auf Anlagenebene auch ERP Funktionen und auf Zellenebene MES Funktionen relevant sein [7]. Für den Entwurfsprozess dieser Steuerungssapplikationen, aber auch des gesamten mechatronisch strukturierten Produktionssystems, ist es sinnvoll eine feste Schnittstellenstruktur für die Informationsverarbeitung zu besitzen, wie sie in Abb. 3.4 dargestellt ist [39].

Mechatronische Systeme einer höheren Hierarchieebene können über ihre Geräteschnittstellen mit den Ausführungsschnittstelle der mechatronischen Systeme einer tieferen Hierarchieebene interagieren und somit auf die Produktionsfunktionen der untergeordneten mechatronischen Systeme zugreifen. Damit ergibt sich eine klare Steuerungshierarchie.

3.3 Mechatronikorientierte Agentensysteme

Betrachtet man eine mechatronische Einheit oder ein mechatronisches System aus Sicht der Implementierung von Steuerungsapplikationen und der dazu notwendigen Schnittstellen, wie es in Abb. 3.4 dargestellt ist, dann wird schnell deutlich, dass diese Implementierung auch durch einen Agenten erfolgen kann. Dabei kann ausgenutzt werden, dass ein Agent einen Teil oder die Gesamtheit der Informationsverarbeitung einer mechatronischen Einheit übernimmt. Dies ist in Abb. 3.5 wiedergegeben.

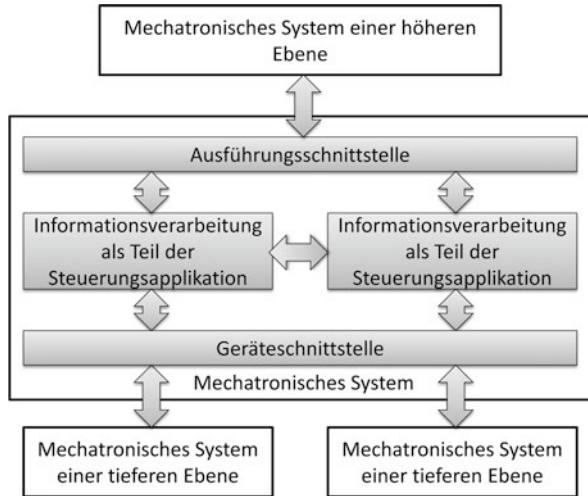


Abb. 3.4 Schnittstellenstruktur einer mechatronischen Einheit

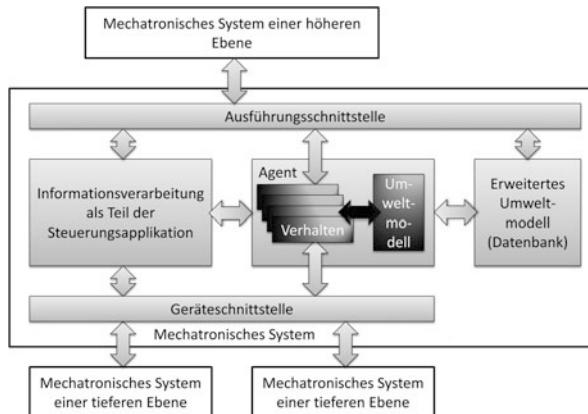


Abb. 3.5 Struktur mechatronikorientierter Agenten für die Implementierung von Produktionssystemsteuerungsapplikationen

3.3.1 Struktur mechatronikorientierter Agenten

Agenten besitzen dabei verschiedene Verhaltensweisen, die für reaktives und proaktives Verhalten notwendig sind. Zudem besitzen sie ein Umweltmodell, das für den Zweck der Flexibilitätssicherung veränderbar sein muss.

Um die Verhaltensweisen des Agenten richtig und vollständig nutzen zu können und dabei die Kapselung von Agentenverhalten beizubehalten, greifen überlagerte mechatronische Einheiten bzw. Systeme direkt über die Ausführungsschnittstelle auf die verschiedenen Verhaltensweisen des Agenten zu. Sie werden mit entspre-

chenden Parametern versehen, gestartet und gestoppt. Ebenso interagieren die Verhaltensweisen über die Geräteschnittstelle mit den unterlagerten mechatronischen Einheiten bzw. Systemen.

Für die Veränderbarkeit des Umweltmodells wird dieses in zwei Teilmodelle zerlegt. Das Basisumweltmodell ist im Agenten fest implementiert. Es erhält Informationen für seine Arbeitsweise aus einem erweiterten Umweltmodell, das in einem Datenspeicher zusätzliche Verhaltensinformationen bereitstellt.

Unter Nutzung dieser Struktur können nun drei Typen von Kombinationen von mechatronischen Einheiten bzw. Systemen und Agenten betrachtet werden, die unterschiedlich im Rahmen der in Abb. 3.5 dargestellten Hierarchie von Produktionssystemen genutzt werden können. Dies sind mechatronische Einheiten/Systeme ohne Agent, mechatronische Einheiten/Systeme mit Agent und Agenten ohne mechatronische Einheiten.

Mechatronische Einheiten/Systeme ohne Agent (BME/Basis-mechatronische Einheiten) können zum Beispiel für die Implementierung von Unterfunktionen, Funktionen, und/oder Hauptfunktionen eingesetzt werden. Sie ermöglichen die Entwicklung statischer Basisfunktionen (Equipment function) für Produktionssysteme wie sie in [42] vorgeschlagen werden. Derartige Basisfunktionen implementieren die Steuerung von Funktionen, die direkt auf der Nutzung der Anlagenphysik aufsetzen und grundlegende physikalische Prozesse bereitstellen.

Mechatronische Einheiten/Systeme mit Agent (AME/Agentifizierte mechatronische Einheiten) können bei der Implementierung von Funktionen, Hauptfunktionen, Zellen und Anlagen genutzt werden. Sie ermöglichen die Umsetzung von Koordinierungssteuerungen (Control functions), siehe [42], und können weitreichende Flexibilität, Reaktivität und Proaktivität bereitstellen. Sie verwenden die Basisfunktionen als Grundlage der von Ihnen angesteuerten komplexeren Produktionsprozesse. AMEs nutzen BMEs für die Ausführung ihrer Funktionen indem sie Basisfunktionen abrufen und koordinieren und somit Produktionsprozesse einer hohen Komplexität anstreben.

Agenten ohne mechatronische Einheiten (SAA/Stand Alone Agenten) dienen der Implementierung von Teilen der Steuerungsapplikation der obersten Hierarchieebenen aus Abb. 3.3, die nicht direkt einer ausführenden Hardware zugeordnet werden müssen. Dies können Planungsfunktionen, Monitoringfunktionen, Dokumentationsfunktionen, usw. sein. Sie interagieren dabei mit den AMEs.

Erste Anwendungen mechatronikorientierter Agentensysteme sind in [43] zu finden.

3.3.2 Entwurfsprozess für mechatronikorientierte Agentensysteme

Die Kombination von Agenten und mechatronischen Einheiten verändert den Entwurfsprozess der in Produktionssystemen enthaltenen Steuerungssysteme. Dies

lässt sich durch die sich ergebende Veränderung der DACS Entwurfsmethode für Agentensysteme am deutlichsten beschreiben.

Im ersten Schritt der DACS Methode wird auf der Basis der Festlegung der zu steuernden Ebene der Automatisierungspyramide die Menge der notwendigen Steuerungsentscheidungen analysiert. Es werden alle Steuerungsentscheidungen im Entscheidungsmodell gesammelt.

Jedes Produktionssystem beinhaltet eine Menge von Produktionsressourcen, die über mechatronische Einheiten implementiert werden können und eine Teilmenge aller Steuerungsentscheidungen umsetzen. Jede mechatronische Einheit, die über einen mechatronikorientierten Agenten gesteuert wird und die in der Bibliothek mechatronischer Einheiten enthalten ist, besitzt eine interne Menge von Steuerungsapplikationsbausteinen. Diese Menge setzt die für die mechatronische Einheit notwendigen Steuerungsentscheidungen um. Dementsprechend erfordert die Menge der Produktionsressourcen ein Teil der Steuerungsentscheidungen. Bei der Identifikation der Steuerungsentscheidungen können somit als erstes alle notwendigen mechatronischen Einheiten identifiziert werden. Für diese muss unterschieden werden, welche als BMEs und welche als AMEs gestaltet werden müssen. Darüber hinaus müssen dann noch alle Steuerungsentscheidungen identifiziert werden, die nicht von Ressourcen realisiert werden können/müssen.

Im zweiten Schritt der DACS Methode werden die Steuerungsentscheidungen aus dem Entscheidungsmodell verschiedenen Agenten zugeordnet. Für die AMEs ist dies bei der Auswahl der mechatronischen Einheiten bereits erfolgt. Entsprechend muss noch die nicht aus mechatronischen Einheiten stammende Entscheidungsmenge auf AMEs oder SAAs aufgeteilt werden. Bei der Übernahme in AMSS kann dies zum Teil durch Erweiterung des Umweltmodells erfolgen.

Im letzten Schritt werden die Interaktionsprotokolle zwischen den Agenten festgelegt. Hier können und müssen die Schnittstellen der mechatronischen Einheiten als Leitlinien und Begrenzung der Möglichkeiten dienen.

Die entsprechende Nutzung mechatronischer Einheiten und mechatronikorientierter Agenten in der DACS Methode ist in Abb. 3.6 grafisch dargestellt.

Der entscheidende Vorteil dieser Erweiterung der DACS Methode ist ihre direkte Einbettbarkeit in den Entwurfsprozess für Produktionssysteme. Gemäß [7, 9] würde die Auswahl der Menge der notwendigen Steuerungsentscheidungen in den Entwurfsschritten der Anlagenlayoutplanung und der Steuerungsprogrammierung erfolgen.

3.3.3 Vorteile der Nutzung mechatronikorientierter Agentensysteme

Mechatronikorientierte Agentensysteme können die Eigenschaft der Wiederverwendbarkeit von Entwurfsergebnissen auf agentenbasierte Steuerungssysteme für Produktionssysteme erweitern. Für Produktionsressourcen der verschiedenen Ebenen der mechatronischen Hierarchie von Produktionssystemen können mecha-

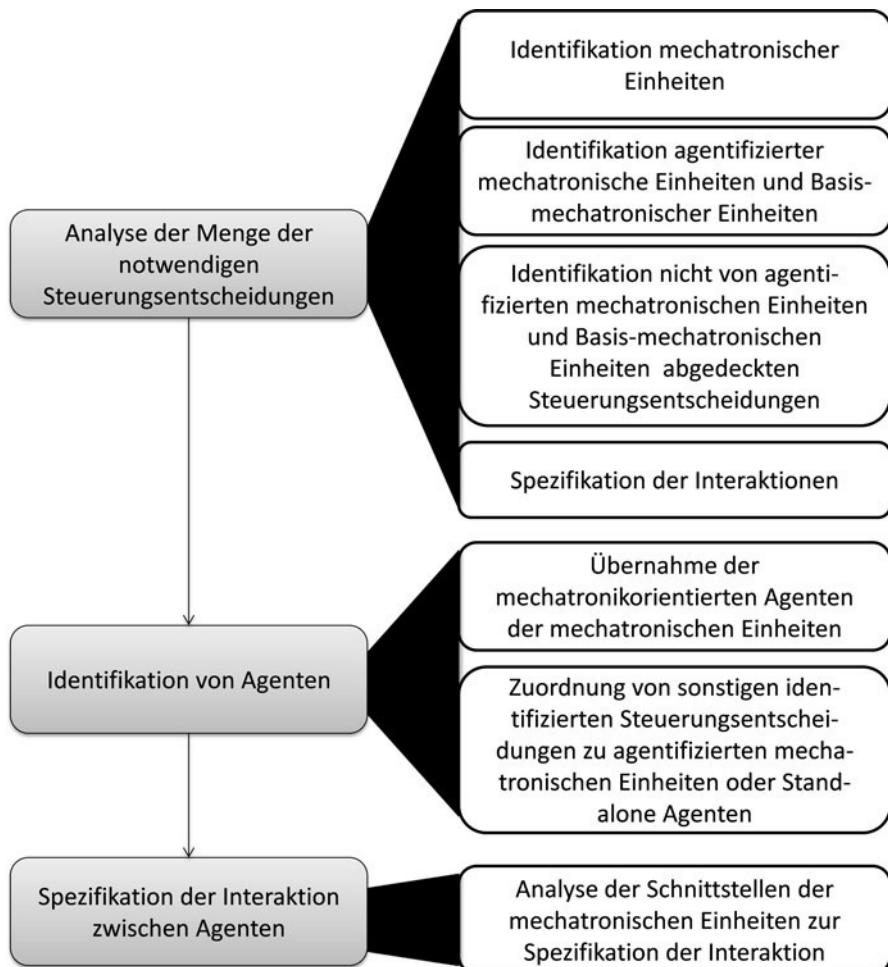


Abb. 3.6 DACS Erweiterung für mechatronikorientierte Agenten

tronische Einheiten entworfen werden, die über mechatronikorientierte Agenten gesteuert werden. Insbesondere scheinen hier die Ebenen Zelle und Hauptgruppe sowie teilweise die Ebene Funktionsgruppe von Bedeutung. Sind diese Ressourcen entwickelt und in einer Bibliothek gespeichert, so können sie im Entwurfsprozess in Kombination der DACS Methode und des PABADIS’PROMISE Entwurfsprozesses als Whiteboxes oder Blackboxes genutzt und damit die Effizienz des Prozesses erhöht bzw. Fehler vermieden werden.

Ein weiterer Vorteil dieser mechatronikorientierten Agentensysteme ist die Möglichkeit, über sie Plug-and-Participate Fähigkeiten für Produktionsressourcen zu implementieren [44] die dann eine Interoperabilität der verschiedenen Produktionsressourcen auf der Steuerungsebene gewährleisten.

Bezogen auf die Agenteneigenschaften Reaktivität, Proaktivität und Mobilität besitzen mechatronikorientierte Agenten keine Einschränkungen. Es ist jedoch zu erwarten, dass Produktionsressourcen, die ihre Fähigkeiten in einem Produktionsystem als Services anbieten auch durch reaktive Agenten gesteuert werden. Die entsprechenden mechatronikorientierten Agenten müssen also reaktiv gestaltet sein. Es sind jedoch auch Ressourcen vorstellbar, die proaktiv umgesetzt werden müssen. Ein Beispiel hierfür sind AGVs, die Produktionsaufträge exklusiv befördern, wie es zum Beispiel in der Halbleiterindustrie der Fall ist. Sie wären proaktiv und mobil.

3.4 Zusammenfassung

Im vorliegenden Fachbeitrag wurden mechatronikorientierte Agentensysteme als Kombination der Ideen der Mechatronik und der agentenbasierten Steuerung von Produktionssystemen vorgestellt. Es wurde aufgezeigt, welche Struktur sie besitzen, wie sie im Entwurfsprozess von Produktionssystemen verwendet werden können und welche Vorteile sich daraus ergeben.

Bisher sind mechatronikorientierte Agenten jedoch Zukunftsmusik. In ersten Forschungs- und Entwicklungsvorhaben wie dem GRACE Projekt werden ihre Möglichkeiten untersucht und der Einfluss auf den Entwurfsprozess und die Nutzung von Produktionssystemen abgeschätzt. Es ist jedoch heute schon klar, dass sie vor dem Hintergrund der Anwendung des Internets der Dinge in der Produktion große Bedeutung erlangen können.

Die vorgestellte Architektur mechatronikorientierter Agenten besitzt jedoch über die bisher beschriebenen Aspekte hinaus einen vereinheitlichenden Charakter. Es ist möglich auf ihrer Basis Architekturen aus dem Bereich der holonen Systeme, der PABADIS Systeme und darüber hinaus zu vereinheitlichen, da sie eine generellere Struktur der Kombination von Hardware und Software auf der einen und Funktionalität und Informationsverarbeitung/Steuerung auf der anderen Seite beinhaltet. Dies könnte es ermöglichen, Eigenschaften und Funktionalitäten der einzelnen Architekturen auf andere Architekturen zu übertragen und dort nutzbar zu machen.

Die mechatronikorientierten Agenten könnten noch einen weiteren Nutzen erbringen. Die Anwendung von Agentensystemen zur Steuerung von Produktionssystemen ist in der industriellen Praxis noch stark beschränkt. Dies gründet sich vor allem auf die Komplexität der Umsetzung derartiger Steuerungssysteme als auch auf einem eingeschränkten Vertrauen in seine Funktionsfähigkeit. Ausgehend von bestehenden Umsetzungen von Steuerungssystemen, wie sie zum Beispiel im Teil 2 von [17] beschrieben werden, können erste Bibliotheken von AMEs und BMEs entstehen. Auf ihnen aufsetzend können sowohl Entwurfsmethoden für agentenbasierte Steuerungssysteme entwickelt werden [14] als auch erfolgreiche Anwendungsfälle zu größeren Systemen kombiniert und damit das Anwendervertrauen gesteigert werden. Einige Beispiele dazu sind unter [14] dokumentiert.

Literatur

- [1] Kühnle, H.: Post mass production paradigm (PMPP) trajectories. *Journal of manufacturing technology management* **18**, 1022–1037 (2007)
- [2] Wünsch, D., Lüder, A., Heinze, M.: Flexibility and re-configurability in manufacturing by means of distributed automation systems – an overview. In: Kühnle, H. (Hrsg.) *Distributed Manufacturing*, S. 51–70. Springer-Verlag, London (2010)
- [3] Lüder, A.: Strukturen zur verteilten Steuerung von Produktionssystemen. Habilitationsschrift. Fakultät Maschinenbau, Otto-von-Guericke Universität Magdeburg (2006)
- [4] Lunze, J.: *Automatisierungstechnik*, 2. Aufl. Oldenbourg Verlag, München (2008)
- [5] Terkaj, W., Tolio, T., Valente, A.: Focused flexibility in production systems, in changeable and reconfigurable manufacturing systems. Springer Series in Advanced Manufacturing **2009**(I), 47–66 (2009)
- [6] Wagner, T., Haußner, C., Elger, J., Löwen, U., Lüder, A.: Engineering Processes for Decentralized Factory Automation Systems, *Factory Automation* 22, In-Tech. Austria (2010). <http://www.intechopen.com/articles/show/title/engineering-processes-for-decentralized-factory-automation-systems>.
- [7] Ferrarini, L., Lüder, A. (Hrsg.): *Agent-Based Technology Manufacturing Control Systems*. ISA Publisher (2011)
- [8] Shen, W., Hao, Q., Yoon , H., Norrie, D.H.: Applications of agent systems in intelligent manufacturing: an update review. *Int. J. Adv. Eng. Inf.* **20**(4), 415–431 (2006)
- [9] Hundt, L.: Durchgängiger Austausch von Daten zur Verhaltensbeschreibung von Automatisierungssystemen., Dissertation. Fakultät für Maschinenbau, Otto-von-Guericke Universität Magdeburg (2012)
- [10] Lüder, A., Foehr, M., Hundt, L., Hoffmann, M., Langer, Y., Frank, S.: Aggregation of engineering processes regarding the mechatronic approach. 16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2011). Toulouse, France (2011). Proceedings-CD.
- [11] Deen, S.M. (Hrsg.): *Agent-Based Manufacturing – Advances in the Holonic Approach*. Springer, Berlin Heidelberg (2003)
- [12] Lüder, A., Peschke, J., Klostermeyer, A., Kühnle, H.: Design pattern for distributed agent based factory automation. *IMS International Forum 2004 – Global Challenges in Manufacturing*, Proceedings S. 783–791. Cernobbio, Italy (2004)
- [13] Barata, J., Camarinhamatos, L., Boissier, R., Leitão, P., Restivo, F., Raddadi, M.: Integrated and distributed manufacturing, a multi-agent perspective. *Workshop on European Scientific and Industrial Collaboration*, Proceedings, S. 27–29 (2001)
- [14] GRACE consortium: Grace Project website. <http://grace-project.org/> (2011)
- [15] IDEAS consortium: IDEAS project website. <http://www.ideas-project.eu/index.php/home> (2012)
- [16] Leitão, P., Restivo, F.: ADACOR: A holonic architecture for agile and adaptive manufacturing control. *Computers in Industry* (57), 121–130 (2006)
- [17] Verein Deutscher Ingenieure: *Agentensysteme in der Automatisierungstechnik*. VDI Richtlinie, Bd. 2653. Beuth Verlag, Berlin (2010)
- [18] Weiss, G.: *Multiagent Systems – A modern approach to distributed artificial intelligence*. MIT Press, Cambridge (1999)
- [19] Lüder, A., Peschke, J., Klostermeyer, A., Bratoukhine, A., Sauter, T.: Distributed Automation: PABADIS vs. HMS. *IEEE Transactions on Industrial Informatics* **1**(1), 31–38 (2005)

- [20] VanBrussels, H., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P.: Reference architecture for holonic manufacturing systems – PROSA. *Computers in Industry* **37**, 255–274 (1998)
- [21] Lüder, A., Peschke, J., Sauter, T., Deter, S., Diep, D.: Distributed intelligence for plant automation based on multi-agent systems – the PABADIS approach, special issue on application of multiagent systems to PP&C. *J. Prod. Planning Control* **15**(2), 201–212 (2004)
- [22] Colombo, A., Schoop, R., Leitao, P., Restivo, F.: A Collaborative Automation Approach to Distributed Production Systems, 2nd International Conference on Industrial Informatics (INDIN'04), Proceedings, S. 27–32. Berlin, Germany (2004)
- [23] Leitão, P., Rodrigues, N.: Multi-Agent System for On-demand Production Integrating Production and Quality Control. In: Marík, V., Vrba, P., Leitão, P. (Hrsg.) *HoloMAS 2011*, LNAI 6867. S 84–93. Springer, Heidelberg (2011)
- [24] Ferrarini, L., Veber, C., Lüder, A., Peschke, J., Kalogerias, A., Gialelis, J., Rode, J., Wünsch, D., Chapurlat, V.: Control architecture for reconfigurable manufacturing systems – the PABA-DIS'PROMISE approach. 11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA2006), Proceedings. Prague, Czech Republic (2006)
- [25] Lüder, A., Peschke, J., Bratukhin, A., Treytl, A., Kalogerias, A., Gialelis, J.: Order oriented manufacturing control – the PABADIS'PROMISE approach. In: Raabe, M., Mihok, P. (Hrsg.) *New technologies for the Intelligent Design and Operation of Manufacturing Networks*, S. 105–124. Fraunhofer IRB Verlag, Stuttgart (2007)
- [26] Wooldridge, M., Jennings, N., Kinny, D.: The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems* **3**(3), 285–312 (2000)
- [27] Lüder, A., Peschke, J.: Incremental design of distributed control systems using GAIA-UML. 12th IEEE International Conference on Emerging Technologies and Factory Automation (ET-FA2007), Proceedings CD. Patras, Greece, (2007).
- [28] Bussmann, S., Jennings, N.R., Wooldridge, M.: Multiagent systems for manufacturing control: A design methodology. Series on Agent Technology. Springer-Verlag, Berlin, Germany (2004)
- [29] Bussmann, S., Jennings, N.R., Wooldridge, M.: On the Identification of Agents in the Design of Production Control Systems. In: Ciancarini, P., Wooldridge, M.J. (Hrsg.) *Agent-Oriented Software Engineering*, LNCS 1957, S. 141–162. Springer-Verlag, Berlin, Germany (2001)
- [30] Bussmann, S., Jennings, N.R., Wooldridge, M.: Re-use of interaction protocols for agent-based control applications. Proc. of the 3rd Int. Workshop on Agent-Oriented Software Engineering held at the 1st Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems, Bologna, Italy (2002). Also published in Giunchiglia, F., Odell, J., Weiß, G. (Hrsg.), *Agent-oriented software engineering III*. LNCS 2585, S. 73–87. Springer-Verlag, Berlin, Germany (2003)
- [31] Bellifemine, F., Caire, G., Greenwood, D.: Developing multi-agent systems with JADE. Wiley series in agent technology. Wiley, Hoboken (New Jersey) (2007)
- [32] Theiss, S., Vasyutynskyy, V., Kabitzsch, K.: Software agents in industry: a customized framework in theory and praxis. *IEEE Transactions on Industrial Informatics* **5**(2), 147–156 (2009)
- [33] Harashima, F., Tomizuka, M., Fukuda, T.: Mechatronics – What Is It, Why, and How? An Editorial. *IEEE/ASME Transactions on Mechatronics* **1**, 1–4 (1996)
- [34] Tomizuka, M.: Mechatronics: from the 20th to 21st Century. *Control Engineering Practice* **10**(8), 877–886 (2004)
- [35] Verein Deutscher Ingenieure: Entwicklungsmethodik für mechatronische Systeme. VDI Richtlinie, Bd. 2206. VDI Verlag, Düsseldorf (2004)
- [36] Czichos, H.: *Mechatronik – Grundlagen und Anwendungen technischer Systeme*. Vieweg Verlag, Wiesbaden (2006)

- [37] Kiefer, J.: Mechatronikorientierte Planung automatisierter Fertigungszellen im Bereich Karosseriebau, Dissertation, Universität des Saarlandes, Schriftenreihe Produktionstechnik, Bd. 43. (2007)
- [38] Thramboulidis, K.: Challenges in the development of mechatronic systems: the mechatronic component. 13th IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA'08), Proceedings. Hamburg, Germany (2008)
- [39] Lüder, A., Hundt, L., Foehr, M., Wagner, T., Zaddach, J.-J.: Manufacturing system engineering with mechatronical units. 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2010), Proceedings-CD. Bilbao, Spain (2010)
- [40] Verein Deutscher Ingenieure: Engineering von Anlagen – Evaluieren und optimieren des Engineerings. VDI Richtlinie, Bd. 3695. VDI Verlag, Düsseldorf (2009)
- [41] Maga, C., Jazdi, N., Göhner, P., Ehben, T., Tetzner, T., Löwen, U.: Mehr Systematik für den Anlagenbau und das industrielle Lösungsgeschäft – Gesteigerte Effizienz durch Domain Engineering. at – Automatisierungstechnik **9**, 524–532 (2010)
- [42] Lüder, A., Peschke, J., Sanz, R.: Design Patterns for Distributed Control Applications. In: Kühnle, H. (Hrsg.) Distributed Manufacturing, S. 155–176. Springer-Verlag, London (2010)
- [43] Leitao, P., Foehr, M., Wagner, T.: Integrating Mechatronic Thinking and Multi-agent Approaches. 38th Annual Conference of the IEEE Industrial Electronics Society (IECON 2012), Proceedings. Montreal, Kanada (2012)
- [44] Heinze, M., Peschke, J., Lüder, A.: Resource management and usage in highly flexible and adaptable manufacturing systems. 13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA2008), Proceedings-CD. Hamburg, Germany (2008)

Kapitel 4

Entwicklungsmethoden

für agentenbasierte Systeme

Bernhard Bauer

Zusammenfassung Neben den Konzepten und Plattformen für agentenbasierte Systeme spielen Entwicklungsmethoden eine entscheidende Rolle bei der Realisierung von Systemen. Daher betrachtet dieses Kapitel den aktuellen Stand der Wissenschaft im Bereich Software Engineering für agentenbasierte Systeme und bringt diese in Zusammenhang mit Automatisierungstechnik. Dabei wird insbesondere auch versucht ausgehend von den existierenden Methoden eine vereinheitlichte Sichtweise auf die Entwicklung von agentenbasierten Systemen für die Automatisierungstechnik zu erlangen.

4.1 Einleitung

Ausgereifte Methoden des Software Engineerings sind die Grundvoraussetzung für eine erfolgreiche Durchführung von Softwareprojekten. Ohne ausreichende methodische Unterstützung ist es so gut wie unmöglich die Komplexität eines herkömmlichen Softwareentwicklungsprozesses zu bewältigen. Insbesondere im Automatisierungsumfeld mit seinen Sicherheitsanforderungen und der Forderung nach Nachvollziehbarkeit sind gute Entwicklungsprozesse ein entscheidender Faktor für die erfolgreiche Durchführung eines Projekts.

Charakterisiert wird eine Softwaremethode nach VDI/VDE 2653 typischerweise durch

- die Festlegung von Konzepten mit ihrer Semantik,
- die Formalisierung in einer speziellen Modellierungssprache oder Notation,

Prof. Dr. Bernhard Bauer (✉)

Fakultät für Angewandte Informatik, Institut für Informatik, Universität Augsburg,

Universitätsstraße 14, 86135 Augsburg, Deutschland

e-mail: bernhard.bauer@informatik.uni-augsburg.de

- die Beschreibung einer Vorgehensweise oder eines Entwicklungsprozesses, die den Ablauf der durchzuführenden Entwicklungsaktivitäten beinhaltet und festlegt, wie die einzelnen Aktivitäten durchgeführt werden sollen.

Ein Softwareentwicklungsprozess definiert insbesondere die Aktivitäten für Prozess- und Projektmanagement sowie die Qualitätskontrolle. Jede Aktivität liefert ein oder mehrere Ergebnisse oder Artefakte – wie zum Beispiel Spezifikationsdokumente, Analysemodelle, Designdokumente, Software-Code, Testspezifikationen, Testberichte, Performanceauswertungen, etc. die als Eingabe für darauf folgende Aktivitäten dienen.

Analyse, Design und Implementierung sind die drei Phasen, die in fast jedem Softwareentwicklungsprozess zu finden sind. Im klassischen Wasserfallmodell sind dies die einzigen Phasen. Heutige Softwareentwicklungsprozessmodelle bieten „round trip Engineering“-Ansätze, bei dem eine Wiederholung in kleineren Granularitätszyklen möglich ist, damit Modelle, die in einem früheren Entwicklungszeitraum verfeinert wurden später noch einmal angepasst werden können.

Agenten zeichnen sich durch Situationsbewusstsein, intelligente Verhaltensweisen, hohes Maß an Verteilung sowie Mobilitätsunterstützung aus. Durch die verbesserte Automatisierung von Routineprozessen und die Unterstützung von Benutzern mit proaktiver und intelligenter Assistenz, die auf Grundsätzen der Adaption und Selbstorganisation basieren, hat die Agententechnologie das Potenzial eine Schlüsseltechnologie in der Automatisierung zu werden. Um jedoch Agententechnologie erfolgreich in industriellen Anwendungen integrieren zu können, benötigt man industrietaugliche Softwaremethoden und Tools, die die Spezifika von Agenten abdecken. Diese Spezifika, die wir in diesem Beitrag betrachten werden, sind insbesondere der interne Zustand des Agenten (z. B. BDI), seine Ziele und Pläne, sowie die Kooperation mit seiner Umgebung, etwa anderen Agenten oder der realen Welt.

Die Agenten-Forschungsgemeinschaft unternimmt beträchtliche Anstrengungen um Methoden und Tools für die Analyse und das Design von komplexen agentenbasierten Softwaresystemen zu entwickeln. Seit 1996 richtet sich etwa der Fokus der ATAL Workshop-Reihe auf agentenbasierte Softwareentwicklung; Seit 2000 wurde der agentenorientierte Software Engineering Workshop (AOSE) zum bedeutendsten Forum für Forschung, die auf diese Thematik abzielt, ebenso die ATOP Workshops. Verschiedene Forscher haben Methoden für Agentendesign entwickelt, wie z. B. die GAIA Methodik [1] oder das umfangreiche Programm der „Free University of Amsterdam“ für Anforderungen [2], Design [3] und Verifizierung [4], ebenso etwa am DFKI [5]. In [6, 7] Kinny et al. wird eine Technik zur Modellierung für BDI Agenten vorgeschlagen, die die Ähnlichkeiten zwischen Designmechanismen, angewandt für agentenbasierte Systeme und solchen für objektorientierte Systeme untersuchen, diese Kombination von Ansätzen wird von einigen Autoren geteilt und es gibt eine ganze Reihe an Veröffentlichungen mit UML als Basis etwa [8].

Diese Beiträge zur Forschung bauen oft auf etablierten Technologien wie Objektorientierung, serviceorientierten Technologien und weit verbreiteten Modellie-

rungsnotationen wie UML auf, um alle Schritte eines Softwarelebenszyklus zu unterstützen. Diese standardisierten Metamodelle und Modellierungsnotationen werden vor allem von Tool-Entwicklern benötigt, um kommerzielle Toolkits anzubieten, die wiederum Entwicklerteams für industrielle Agentensystementwicklung benötigen.

Dieser Beitrag gliedert sich wie folgt: Abschn. 4.2 gibt einen Überblick über Entwicklungsmethoden für agentenbasierte Systeme. Ein Versuch die unterschiedlichen Ansätze zu vereinheitlichen wird in Abschn. 4.3 unternommen. Abschnitt 4.4 gibt eine Zusammenfassung und Ausblick.

4.2 Überblick über Entwicklungsmethoden für agentenbasierte Systeme

Dieser Abschnitt beschreibt unterschiedliche Entwicklungsmethoden für agentenbasierte Systeme, nämlich wissensbasierte, agenten- und objektorientierte Ansätze.

4.2.1 Objektorientierte Ansätze

Die ersten Ansätze [6, 7] bauen auf BDI Agenten auf, die objektorientiert modelliert werden und entwickeln dafür eine Methode. BDI-Agenten sind gekennzeichnet durch:

- Weltbild (*beliefs*), d. h. es wird der aktuelle Zustand seiner Umgebung, sein eigener Zustand und weitere Informationen hinterlegt,
- In den Zielen (*desires*) sind die Hauptziele des Agenten spezifiziert, nach welchen er sein Verhalten richtet.
- Die Absichten oder Pläne (*intentions*) beschreiben hierarchisch organisierte Pläne, die zum Erreichen der Ziele ausgeführt werden.

Oft unterscheidet man zwischen der internen und der externen Sichtweise. Die interne Sicht ist repräsentiert durch BDI, während die externe Sicht die Identifikation der Rollen der Applikationsdomäne und der Interaktionen der Rollen bzw. Agenten beschreibt. Das Agentenmodell besteht aus einem Modell von Agentenklassen und Agenteninstanzen. Außerdem wird die Agentenkommunikation in einem Interaktionsmodell festgelegt.

UML findet in den letzten Jahren eine starke Verbreitung. Entweder wird direkt UML verwendet [8] oder spezielle Profile oder Metamodelle entwickelt. Diese Ansätze sind in den letzten Jahren Standard im Agentenumfeld geworden.

Agent UML [9] war ein Startpunkt für die Modellierung von Agenten-basierten Systemen mit UML. Der Schwerpunkt lag auf der Spezifikation von Interaktionsprotokollen sowie der Beschreibung des Verhaltens von Agenten. Die Ergebnisse im Bereich der Interaktionsmodellierung flossen in UML 2 Sequenzdiagramme ein.

und nach dem erscheinen von UML 2.0 wurde von den Autoren vorgeschlagen UML 2 direkt für die Spezifikation von agenten-basierten Systeme zu verwenden (siehe [8]). Einige der unten aufgeführten Ansätze verwenden Agent UML.

PASSI [10] beschreibt das Systemanforderungsmodell mit der Identifikation der Agenten, Rollen und Aufgaben, der Festlegung der Domänen- und Kommunikationsontologie sowie der Rollenbeschreibung. Die Agentenstruktur sowie das Agentenverhalten werden durch das Agentenimplementierungsmodell beschrieben. Das Codemodell und das Verteilungsmodell runden diese Vorgehensweise ab.

Einen iterativen Ansatz verfolgt Prometheus [11, 12], der den Software Engineering Prozess von der Systemspezifikation über das Architekturentwurf bis hin zum detaillierten Design unterstützt. Insbesondere sollen damit BDI-Agenten mit ihren Weltbildern, Zielen, Plänen und Ereignissen entwickelt werden. Das Ergebnis ist dabei eine Spezifikation, die leicht auf existierende Agentenplattformen mit BDI Semantik abgebildet werden kann.

MESSAGE [13, 14] legt den Schwerpunkt auf die Analysephase und definiert fünf Modelle:

- Das *Organisationsmodell* für die Beschreibung der allgemeinen Struktur und dem Verhalten von mehreren Agenten um ein gemeinsames Ziel zu erreichen.
- Das *Ziel/Aufgabenmodell* definiert die Ziele und Teilziele des gesamten Systems.
- Das *Agenten/Rollenmodell* besteht aus einer Menge von individuellen Agenten und Rollen, sowie der Beschreibung des Zusammenspiels von Rollen und Agenten.
- Das *Informations-* oder *Domänenmodell* beschreibt die Domäne als Mischung aus objektorientierter und relationaler Modellierung.
- Das *Interaktionsmodell* beschreibt die Interaktion von sehr abstrakt bis sehr konkret.

Eine Erweiterung von MESSAGE stellt INGENIAS dar, welches den Entwicklungsprozess von Anforderung bis hin zum Test unterstützt.

Eine weitere Methode ist Tropos [15, 16]. Hier liegt der Schwerpunkt der Modellierung auf: Frühe und späte Anforderungen, Architekturentwurf, detailliertes Design sowie Implementierung. Dazu definiert Tropos folgende Modelle:

- Das Akteur-/Abhängigkeitsmodell beschreibt die sozialen Akteure, die Systemakteure und deren Ziele.
- Daraus leiten sich die Ziel- und Planmodelle ab.
- Das Fähigkeitsdiagramm beschreibt die Fähigkeiten eines Agenten.
- Interaktionsprotokolle werden durch AgentUML Sequenzdiagramme spezifiziert.

Aufbauend auf dem Unified Process und UML unterstützt die MaSE Methode [17] die Analyse und das Design: Ausgehend von den ersten Anforderungen werden Ziele festgelegt. Klassische Use Cases werden durch Sequenzdiagramme verfeinert, ebenso Rollen und Aufgaben. Daneben werden Agentenklassen und deren Interaktionsprotokolle spezifiziert. Ebenso werden das interne Agentenverhalten sowie das Systemdesign beschrieben. Eine Erweiterung der MaSE-Methode

ist die MaSE-RT Methode [18], welche zusätzlich die Besonderheiten eingebetteter Echtzeitsysteme berücksichtigt. Sie erlaubt die durchgängige Spezifikation von Zeitanforderungen über alle Entwicklungsphasen hinweg. Hierzu wurde eine zusätzliche Anforderungsdefinitionsphase eingeführt und die agententypischen Artefakte wie Ziele, Rollen und Interaktionen um Zeitanforderungen ergänzt. O-MaSE [19] ist eine Erweiterung, um Organisationsaspekte besser modellieren zu können. Dazu werden Modellierungskonzepte für Ziele, Rollen, Agenten, Domänenmodelle und Policies angeboten.

Aktuelle Ansätze wie z. B. [20] bauen teilweise auf SOAML auf und/oder definieren für die Agenten-basierten Systeme eigene Profile und Metamodelle.

AML (Agent Modeling Language) [21] und ADEM (Agent-oriented Development Methodology) [22] ist in die UML 2 Welt eingebettet und versucht die oben aufgeführten Ansätze zu integrieren. Das Hauptanwendungsgebiet liegt in der Spezifikation von Agentensystemen für Geschäftsanwendungen. Die von AML [23] unterstützten Konzepte sind Agenten, Ressourcen, Umgebungen, Rollen, Ontologien, Agentenverteilung, Verhaltensbeschreibungen, Dienste und Dienstprotokolle, Perzeptoren und Effektoren, Mobilität von Agenten sowie die BDI-Konzepte. ADEM bietet eine auf AML basierende Vorgehensweise und somit eine komplette Methodik an.

Die Agent-Object-Relationship Modeling Language (AORML) [24] definiert in seinem Profil Konzepte wie Events, Aktionen, Ziele, Vereinbarungen oder aber auch normale Objekte. Neben einfachen Agenten können auch institutionelle Agenten definiert werden. Interaktionsprotokolle werden mit AUML spezifiziert. Es fehlt das mentale Modell, wie Ziele oder auch das proaktive Verhalten von Agenten.

Atelier de Développement de Logiciels à Fonctionnalité Emergente (ADELF-E) [25, 26] spezifiziert adaptive MAS durch ein eigenes Metamodell und unterstützt in der Version 2 auch den modell-getriebenen Softwareentwicklungsansatz. Die Konzepte sind dabei Perzeption, Agenten, Wissen, Sensoren sowie Aktionen und Kommunikationsaktionen. Es unterstützt von Methodikseite frühe und späte Anforderungen, insbesondere Benutzeranforderungen, Umgebungsbeschreibungen sowie Use Cases. In der Analysephase wird die Domäne analysiert, die Agententypen identifiziert und deren Interaktion. Darauf aufbauend wird in der Designphase die Architektur detailliert und das Multi-Agenten-Modell analysiert. Zum Schluss erlaubt das Implementierungsmodell die Transformation des Modells in Code.

DSML4MAS [20] erlaubt es Agentensystem auf eine plattformunabhängige und graphische Art und Weise zu spezifizieren. Dem model-getriebenen Ansatz folgend definiert es eine abstrakte, konkrete Syntax und ihre Semantik. Darüber hinaus wird eine Methode entwickelt, die es erlaubt von einer abstrakten Spezifikation semi-automatisch bis zur konkreten Implementierung zu verfeinern und alternative Softwareparadigmen wie Service-orientierte Architekturen zu berücksichtigen. So mit stellt sie aktuell eine relativ neue, umfassende Methode für die Entwicklung von Agenten-basierten Systemen zur Verfügung. DSML4MAS definiert dabei verschiedene Sichtweisen, nämlich Multi-Agenten-, Agenten-, Organisations-, Rollen-, Interaktions-, Verhaltens-, Umgebungs- und Deployment-Viewpoints.

4.2.2 Wissensbasierte Ansätze

Die frühesten Ansätze um agenten-basierte Systeme zu spezifizieren waren inspiriert durch die Modellierung von wissensbasierten Systemen. Die drei einflussreichsten Methoden, die die Grundlage für weitere Forschungsarbeiten darstellen, waren: CommonKADS, CoMoMAS und MAS-CommonKADS.

CommonKADS ist eine reine Methode für wissensbasierte Systeme, d. h. die Schwerpunkte liegen u. a. auf der Methodik zum Wissenserwerb, der Wissensrepräsentation und existierendes Wissen zur Verfügung zu stellen. Nachfolger von CommonKADS wurden entwickelt, um agenten-spezifische Aspekte zu berücksichtigen: CoMoMAS und MAS-CommonKADS. Im Folgenden werden wir nur MAS-CommonKADS diskutieren, für CoMoMAS verweisen wir auf [27].

MAS-CommonKADS [28] fügt verschiedene Erweiterungen zu CommonKADS hinzu und umfasst folgende Aspekte: Für die Spezifikation von Agentensystemen bietet MAS-CommonKADS Konzepte wie Organisationsmodell, Aufgabenmodell, Agentenmodell, Wissensmodell, Kommunikationsmodell und Designmodell an. Die Konzeptualisierung erfolgt basierend auf Use-Cases. Ebenso werden Codierung, Test, Integration und Betrieb in dieser Methode betrachtet, CommonKADS eine umfassende und gute Basis für die Entwicklung von Agentensystemen dar.

Neben der Verwendung im Agentenumfeld wird diese Methode im Wissensmanagementkontext weiterentwickelt und eingesetzt.

4.2.3 Agentenbasierte Ansätze

Ein Nachteil rein wissensbasierter Methoden ist, dass diese Methoden nicht für agentenbasierte Systeme entwickelt und somit die Modellierung und Realisierung von Agentensystemen nicht von Anfang an unterstützt wurden.

Daher wurden rein agentenbasierte Ansätze entwickelt. Typische Beispiele für agentenorientierte Ansätze sind Gaia [1, 29], deren Erweiterung ROADMAP [30] sowie SODA [31].

Gaia und die Erweiterung durch ROADMAP

Gaia ist eine Methode für agentenorientierte Analyse und Design. ROADMAP erweitert Gaia, um die Anforderungsanalyse und die Umgebung offener Systeme zu spezifizieren. Darüber hinaus liegt der Fokus mehr auf den Spezifikationen von Interaktionsprotokollen, basierend auf AUML [9]. In der nachfolgenden Beschreibung werden beide Ansätze gemeinsam präsentiert.

In der Analysephase werden die folgenden Modelle erstellt (siehe auch [29] und [30]):

Use Case Model Wie auch in der objektorientierten Welt werden Use Cases verwendet um die Anforderung effektiv und ausreichend auszudrücken. Vergleichbar mit UML beinhaltet das Model eine grafische Darstellung und bestimmte textuelle Szenarien. Die Semantik der Use Cases ist jedoch abweichend von traditionellen OO Ansätzen, denn durch die Wissensbasis des Agenten wird der bestmögliche Dienst angeboten und nicht die erstbeste Methode aufgerufen. Man könnte auch sagen, ein Agent ist ein Objekt, das „nein“ sagen kann.

Das *Umgebungsmodell*, das aus dem Use Case Modell abgeleitet wird, bietet eine ganzheitliche Beschreibung der Systemumgebung. Dies wurde eingeführt, da der ROADMAP Ansatz auf komplexe, offene Systeme abzielt, die typischerweise in hoch dynamischen und heterogenen Umgebungen eingebettet sind. Das Modell ist die Wissensbasis um Änderungen auf der Umgebung zu berechnen und anschließend auf der realen Welt auszuführen. Ebenfalls wird das Wissensmodell aus dem Use Case Modell abgeleitet. Es bietet eine ganzheitliche Beschreibung des im System verwendeten Domänenwissens. Das Modell besteht aus einer Hierarchie von Wissenskomponenten und einer dazugehörigen Beschreibung, wodurch das benötigte Wissen identifiziert wird, um das Verhalten des Agenten zu beschreiben. Jeder Wissenskomponente sind Rollen zugeordnet und dadurch mit dem Use Case Modell und der Umgebung verbunden.

Das *Rollenmodell* identifiziert die Individuen, Gruppen oder Organisationen des Systems. Sie sind durch vier Attribute charakterisiert, nämlich

- *Verantwortlichkeiten*, die die Funktionalität eines Agenten definieren,
- *Befugnisse*, die durch Rechte definiert sind, welche Informationen und Wissensressourcen von der Rolle benutzt werden können sowie Ressourcenbeschränkungen in denen der Ausführende der Rolle operieren darf.
- *Aktivitäten* einer Rolle sind nicht-öffentliche Aktionen, die eine Rolle ausführen kann ohne Interaktion mit anderen Agenten.
- *Protokolle* definieren wie eine Rolle mit anderen Rollen interagiert. In der Designphase werden die Protokolle ausführlicher beschrieben.

Das *Interaktionsmodell* wurde in Gaia ursprünglich für Protokolle definiert. Es beschreibt Abhängigkeiten und Beziehungen zwischen unterschiedlichen Rollen („Pattern of interaction“). In ROADMAP wird dies Protokollmodell genannt und zusätzlich als Interaktionsmodell beschrieben, das auf AUML Interaktionsdiagrammen basiert. Gaia beschreibt ein Protokoll durch

- die *Zielsetzung*: eine kurze textuelle Beschreibung der Interaktion;
- den *Initiator des Protokolls*: die Rolle(n), die die Konversation begonnen haben;
- die *Antwortenden*: alle Rollen, die in diesem Protokoll teilnehmen; sowie
- die *Eingaben und Ausgaben*: alle Informationen die vom Protokollinitiator und den Antwortenden verwendet und zur Verfügung gestellt werden, sowie eine kurze textuelle Beschreibung der Handlungen des Protokollinitiators.

In der Designphase werden die abstrakten Modelle der Analysephase in ausreichend konkrete Modelle verfeinert, die anschließend implementiert werden können. Im Gegensatz dazu hat die Designphase von Gaia das Ziel, einen Abstraktionslevel

zu erreichen, der ausreichend spezifisch ist, um traditionelle Designtechniken für die Implementierung der Agenten zu erreichen. Gaia und ROADMAP definieren dabei folgende Modelle:

- Das *Agentenmodell* identifiziert die Agententypen, die das System beschreiben, d. h. die Mengen von Agentenrollen und Agenteninstanzen, die aus diesen Rollen instanziert werden.
- Das *Servicemodell* charakterisiert die Dienste eines Agenten, durch Angabe seiner Eingaben, Ausgaben, Vor- und Nachbedingungen eines Dienstes um eine speziellen Rolle zu realisieren, vergleichbar mit einer Servicebeschreibung in einem Service-orientiertem Umfeld. Ein Service leitet sich aus der Liste der Protokolle, Aktivitäten, Verantwortlichkeiten und Liveness-Eigenschaften einer Rolle ab.
- Das *Acquaintancemode*ll dokumentiert die Informationswege zwischen verschiedenen Agenten um Engpässe, die zu Problemen während der Laufzeit führen können, zu identifizieren.
- Das *Interaktionsmodell* bietet eine detaillierte Definition der Interaktion zwischen verschiedenen Rollen oder individuellen Agenten unter Anwendung von AUML Interaktionsdiagrammen.

SODA

Eine andere agentenorientierte Softwareentwicklungsmethodik, die ihr Hauptaugenmerk auf Organisationen – ähnlich zu Gaias Organisationen – gelegt hat, ist SODA (Societies in Open and Distributed Agent spaces). Wie ROADMAP adressiert SODA einige Aspekte, die nicht von Gaia behandelt werden, wie etwa offene Systeme oder selbst-interessierte Agenten. Darüber hinaus berücksichtigt SODA die Umgebung von Agenten und bietet Mechanismen für besondere Abstraktionen und Vorgehensweisen für das Design von Agenteninfrastrukturen. Basierend auf der Analyse und dem Design von Agentengesellschaften und Agentenumgebungen bietet SODA Unterstützung für die Modellierung von Inter-Agentenaspekten, d. h. zwischen dem Agenten und seiner Umgebung. Deshalb ist SODA keine komplettte Methodik. Ihr Ziel ist es ein schlüssiges, konzeptuelles Programmiergerüst und eine umfassende Vorgehensweise zur Softwareentwicklung zu definieren. Sie unterstützt dabei die Analyse und das Design von individuellen Agenten, sowie die Modellierung der Agentenorganisation und der Agentenumgebung.

Während der Analysephase, wird die Anwendungsdomäne studiert und modelliert. Zusätzlich werden die verfügbaren Ressourcen und technologischen Beschränkungen festgelegt und die grundlegenden Anwendungsziele und Vorgaben definiert. Das Resultat der Analysephase wird typischerweise in Anforderungen auf hohen Abstraktionslevel und mit ihren gegenseitigen Abhängigkeiten ausgedrückt, in dem sie dem Designer eine formale oder halbformale Beschreibung der beabsichtigten Anwendungsstruktur- und Organisation anbieten. Die SODA-Analysephase benutzt dabei drei Modelle: Die Anwendungsziele werden hinsichtlich zu er-

reichender Aufgaben modelliert, die mit Rollen und Gruppen (*Rollenmodell*) verbunden sind. Aufgaben werden hinsichtlich der Verantwortlichkeiten, in die sie involviert sind, den Kompetenzen, die sie benötigen und den Ressourcen, von denen sie abhängig sind, spezifiziert. Verantwortlichkeiten werden in Bezug auf das Weltmodell ausgedrückt. Aufgaben werden entweder als eigenständig oder sozial eingestuft. Soziale Aufgaben benötigen typischerweise eine Anzahl von verschiedenen Kompetenzen und den Zugriff auf verschiedene Ressourcen, wohingegen eigenständige Aufgaben eher eine gut abgegrenzte Ressource und begrenzte Kompetenz benötigen. Jede eigenständige Aufgabe ist einer eigenständigen Rolle zugeordnet, welche in Bezug auf deren eigenständige Aufgabe, ihrer Zugriffsrechte, der Ressourcen und dem entsprechenden Interaktionsprotokoll definiert wird. Analog sind soziale Aufgaben Gruppen zugeordnet. Eine Gruppe wird bzgl. ihrer sozialen Aufgaben, ihrer Zugriffsrechte auf die Ressourcen, den beteiligten sozialen Rollen und der entsprechenden Interaktionsrolle definiert. Eine soziale Rolle beschreibt die Rolle, die von einem einzelnen in einer Gruppe eingenommen wird und kann sich entweder mit einer bereits definierten (individuellen) Rolle überschneiden oder kann sich ex-novo definieren. Die Anwendungsumgebung wird bzgl. der verfügbaren Services modelliert, die mit dem abstrakten Ressourcen (*Ressourcenmodell*) verbunden sind. Eine Ressource wird definiert durch

- die angebotenen Services,
- die Zugangsmodalitäten,
- den gewährten Zugriffsrechten für Rollen und Gruppen, die den Services nutzen wollen und
- dem entsprechenden Interaktionsprotokoll.

Wenn eine Aufgabe, die einer Rolle oder Gruppe zugeordnet ist, einen vorgegebenen Service benötigt, werden die Zugangsmodalitäten für den Abruf der Ressource, die für diesen Services verantwortlich ist, festgelegt und definiert. Zugriffe sind mit Rollen oder Gruppe verbunden. Die Interaktion (*Interaktionsmodell*), die Rollen, Gruppen und Resources werden in Bezug auf Interaktionsprotokolle modelliert. Sie spezifizieren alle benötigten Informationen, die von Rollen und Ressourcen bereitgestellt werden, um einzelne Aufgabe zu erfüllen. Darüber hinaus werden Interaktionsregeln, die die Interaktion zwischen Rollen und Ressourcen festlegen, definiert.

SODA erzielt eine Designabstraktion durch folgende Modelle: Das *Agentenmodell* modelliert die Agentenklassen. Eine Agentenklasse wird als Menge von einer oder mehreren individuellen oder sozialer Rollen definiert. Infolgedessen wird eine Agentenklasse durch die Aufgaben, die Zugriffsrechte und die Interaktionsprotokolle, mit dem die Rollen verbunden sind, beschrieben. Agentenklassen können des Weiteren auch in Bezug auf andere Eigenschaften gekennzeichnet werden: die Kardinalität (die Anzahl von Agenten dieser Klasse), ihre Lage (in Bezug auf ihr topologisches Model), ihre Herkunft (Systemzugehörigkeit – wenn man von offenen Systemen ausgeht).

4.2.4 Überblick

Um einem besseren Überblick über die unterschiedlichen Ansätzen zu erhalten wird eine Charakterisierung der existierenden Methoden präsentiert, die folgendes Schema verwendet (siehe Tab. 4.1):

- **Zielsetzung:** Methode, Vorgehensweise oder Modellierungssprache, d. h. ist der Ansatz eine Methodik bestehend aus Vorgehensweise und Notation; nur ein Vorgehensmodell ohne Notation; eine Modellierungssprache wie z. B. UML
- **Basis:** was wurde als Basis für die Methodik verwendet, z. B. rein agentenorientierter Ansatz, formaler Ansatz, Objektorientierung oder wissensbasierter Ansatz
- **Phasen:** welche Phasen des Softwareentwicklungsprozesses werden unterstützt (Frühe Anforderungen, Analysis, Design, Implementierung,...)
- **Syntax/Semantik:** wird die Syntax und/oder die Semantik definiert, z. B. durch modelltheoretische Ansätze, natürliche Sprache oder Logik
- **Anwendungsbereiche:** was sind die Hauptanwendungsbereiche der Methodik (z. B. Internetagenten, mobile Agenten, Fertigung, Geschäftsumfeld)
- **Agentenunterstützung:** welche Konzepte von Agenten werden durch die Methodik unterstützt

4.3 Vereinheitlichung der Ansätze

In diesem Abschnitt wird ausgehend von den oben vorgestellten Methoden zunächst versucht die vorgestellten Ansätze zu vereinheitlichen und anschließend eine Idee für einen Prozesslinienansatz für Agenten entwickelt.

4.3.1 Vereinheitlichung der Ansätze

Die Entwicklung agentenbasierter Automatisierungssysteme stellt besondere domänenbezogene Anforderungen an eine Entwicklungsmethode. Qualitätsattribute (oft auch *nicht funktionale Eigenschaften* genannt) etwa Zeit, Sicherheit (Safety und Security) sowie Normen und Standards werden in agenten-basierten Ansätzen meistens nicht berücksichtigt, sind aber im Automatisierungsumfeld essentiell.

Daher ist es notwendig Zeitanforderungen wie Anfangszeiten, Zeitintervalle und Deadlines explizit zu spezifizieren. Möglichkeiten bieten hier z. B. SysML, EAST-ADL, AADL oder klassisches UML.

Automatisierungssysteme müssen funktionale Sicherheitsanforderungen (Safety) ebenso wie Security Eigenschaften in geeigneter Weise berücksichtigen, um einen verlässlichen Betrieb zu gewährleisten. Diese Anforderungen müssen während des gesamten Entwicklungsprozesses spezifizierbar und nachvollziehbar sein.

Tab. 4.1 Vergleich der Ansätze

Ansatz	Zielsetzung	Basis	Phasen	Syntax/Semantik	Anwendungsbereiche	Agentenunterstützung
CommonKADS CoMoKADS MAS- CommonKADS	Methodologie KM	Analyse/Design; CommonKADS: Impl.	Syntax/teilweise Semantik	Wissenszentrierte Applikationen	Organisationen, Aufgaben, Agenten, Wissensaustausch	
Gai/ROADMAP	Methodologie AO	Analyse/high-level Design (Gai)	Syntax/teilweise Semantik	IT Systeme	Rollen, Agenten, Wissensaustausch, Services	
SODA	Hauptsächlich AO Prozess	Analyse/Design	Syntax	Offene Systeme	Rollen, Agenten, Ressourcen, Socie- ties, Interaktionen	
Kinny et al.	Methodologie OO & BDI	Analyse/Design Agenten	Syntax/teilweise Semantik	BDI Agenten	Agenten, Interaktion, Beliefs, Ziele, Pläne	
MESSAGE	Methodologie OO & RUP	hauptsächlich Analyse	Syntax/teilweise Semantik	IT Systeme	Organisation, Ziele, Aufgaben, Agen- ten, Rollen, Wissen, Interaktionen	
Tropos	Methodologie OO & BDI	Analyse/ Design/impl.	Syntax/teilweise Semantik	BDI Agenten	Aktoren, Ziele, Pläne, Ressourcen, Fähigkeiten, Interaktion	
Prometheus	Methodologie OO & BDI	Analyse/ Design/Impl.	Syntax/teilweise Semantik	BDI Agenten	Ziele, Beliefs, Pläne, Events, Agen- ten, Interaktionen, Fähigkeiten	
MaSE	Methodologie OO & RUP	Analyse/ Design/impl.	Syntax/teilweise Semantik	Heterogene MAS	Ziele, Rollen, Interaktionen, Agenten	
PASSI	Methodologie OO & RUP	Analyse/ Design/Impl.	Syntax/teilweise Semantik	Hauptsächlich Ro- botik	Societies, Agenten, Rollen, Wissen.	
AML	Methodologie OO & RUP	Analyse/ Design/Impl.	Syntax/Object-Z Semantik	Geschäftsanwen- dungen, Robotik, etc.	Agenten, Ressourcen, Umgebungen, Rollen, Ontologien, Agentendeploy- ment, Verhaltensbeschreibungen, Dienste, Dienstprotokolle, Perzep- toren, Effektoren, Mobilität, Beliefs, Ziele, Pläne	

KM: wissensbasiert (knowledge management)

OO: objektorientiert

BDI: Belief, Desires, Intention

RUP: Rational Unified Proces

MAS: Multi-Agenten-Systeme

Während in Agentenmethoden die Umgebung spezifiziert werden kann, muss darüber hinaus für Automatisierungssysteme die Hardware-Infrastruktur modelliert werden können (etwa durch UML Deployment-Diagramme) und mit der Agentenumgebung kombinierbar sein. Dies umfasst auch die Kommunikationsinfrastruktur, etwa Bussysteme, inklusive Zeitverhaltens und Integration mit den Interaktionsprotokollen der Agenten unter Berücksichtigung von Prioritäten auf den Nachrichten, etwa durch eine Nothalt-Nachricht.

Unter dem Aspekt des Einsatzes von Multicore-Prozessoren und der auch durch die Verteilung von Aufgaben ist eine Synchronisation der parallel ablaufenden Aktivitäten zu modellieren und zu analysieren, etwa durch UML Aktivitätsdiagramme und Datenabhängigkeitsanalysen auf diesen Modellen. Solche Aspekte werden in den meisten agentenbasierten Methoden zurzeit nicht berücksichtigt, obwohl Agententechnologie inhärent parallel ist.

Die Systemarchitektur kann etwa durch eine saubere agentenbasierte Beschreibung der Organisation, der Agenten, Rollen etc. erreicht werden, um so einen Gesamtüberblick über das System zu bekommen. Hier können die oben aufgeführten Methoden Hilfestellung geben.

Die Unterstützung einer verteilten Struktur wird als fester Bestandteil der agentenorientierten Methoden angesehen. Da die betrachteten Automatisierungssysteme über verschiedene Plattformen und Hardwarekomponenten verteilt sind, kann dieser Aspekt sehr gut mit Agentensystemen modelliert werden.

Daneben müssen die Agenteneigenschaften wie Ziele, Rollen, Fähigkeiten, Pläne, innerer Aufbau definiert werden können und ein Verhaltenskorridor, der sicherstellt, dass das System in gewünschten Bereich arbeitet.

Ebenso ist es im Automatisierungsumfeld wichtig die Systemumgebung und ihre Dynamik z. B. neue Agenten oder Komponenten kommen hinzu oder werden entfernt zu spezifizieren.

4.3.2 Prozesslinienansatz für Agenten

Wenn man sich die unterschiedlichen Notationen, Modellierungen und Methoden im agentenorientierten Umfeld ansieht stellt man fest, dass diese Methoden viele Gemeinsamkeiten haben und sich oft nur an bestimmten Stellen unterscheiden oder andere Schwerpunkte setzen. Gerade in letzter Zeit verstärken sich die Anstrengungen nicht nur in der Industrie sondern auch in der Wissenschaft, immer komplexer werdenden Projektsituationen von der Prozessseite aus flexibel zu begegnen. Metamodelle wie SPEM [32] unterstützen bereits die Möglichkeiten Prozesse variabel zu verwalten. Parallel dazu entwickeln sich Forschungszweige, wie das „Situational Method Engineering“ (SME) [33], Techniken, um die Anpassungsmöglichkeiten von Prozessen und Methoden an die jeweilige Anforderungen eines bestimmten Projekts oder einer bestimmten Situation weiter zu verbessern.

Um durch Hinzufügen und Entfernen von Methodenbausteinen einen situationsspezifischen Zielprozess abzuleiten, beschreiben z. B. Karlsson et. al. in [34] die

Verwendung mehrere Konfigurationen eines Basisprozesses, wodurch die Wieder-verwendung in allgemein auftretenden Projektsituationen erleichtert werden soll. In [35] definieren Xu et.al. die Softwareprozess-Konfiguration als wissens-intensive Aktivität, und analysieren die Vorteile in diesem Bereich. Ein anderer Ansatz im Rahmen von SME adressiert mit seinen „Families of Method-oriented Architectures“ [36] das Erstellen, die Konfiguration und die Veröffentlichung von variablen Prozessen auf Basis von Prinzipien des Software Product Line Engineering (SPL) und von Service-orientierten Architekturen (SOA). In [37] wird ein modell-getriebener Ansatz beschrieben, wobei mithilfe eines sogenannten Context Modells projektspezifische Variablen entlang bestimmter Dimensionen formalisiert werden, um damit einen mittels Modelltransformationen ein Tailoring des Softwareentwicklungsprozesses zu erreichen.

Dieser Ansatz wurde bisher für agentenbasierte Systeme noch nicht im Detail untersucht, sollte aber auch gerade im Hinblick an Anpassbarkeit und Flexibilität betrachtet werden.

4.4 Zusammenfassung und Ausblick

Dieser Beitrag gab einen Überblick über Agentenmethoden und ihre unterschiedlichen Ausprägungen. Es wurde versucht für die Automatisierungstechnik wichtige Aspekte zu identifizieren und mit den Agentenmethoden zusammenzubringen. Für die Anwendung von Agenten im Bereich der Automatisierung lassen sich folgende Schlussfolgerungen ziehen und auch mögliche Forschungsaspekte identifizieren:

- Ähnlich dem beschriebenen Prozesslinienansatz ist es notwendig wieder verwendbare Standard-Modellierungsbauusteine zu entwickeln. Hier bieten die existierenden Agentenmethoden einen guten Startpunkt.
- UML sollte als Basis für die Modellierung verwendet werden und ähnlich zu DSML4MAS sollte Metamodelle und Profile entwickelt werden, die eine Entwicklung mit Standard UML-Tools ermöglichen.
- Es sollte ein modell-getriebener Ansatz verfolgt werden, um Modell zu analysieren, Modelltransformationen durchzuführen und im besten Fall sogar Code aus dem Modell zu erzeugen.

Somit kann die Agententechnologie ein möglicher Lösungsansatz für die Automatisierung der Zukunft sein. Der Einsatz von situationsspezifisch angepassten Entwicklungsprozessen und Unterstützung mit geeigneten Modellierungsnotationen bietet hier einen entscheidenden Beitrag zur Verwendung der Agententechnologie in der Automatisierung.

Literatur

- [1] Wooldridge, M., Jennings, N.R., Kinny, D.: The gaia methodology for agent-oriented analysis and design. *J. Aut. Agents Multi-Agent Sys.* **3** (2000)
- [2] Herlea, D.E., Jonker, C.M., Treur, J., Wijngaards, N.J.E.: Specification of behavioural requirements within compositional multi-agent system design. Proc. of Ninth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, S. 8–27. Springer, Berlin Heidelberg (1999)
- [3] Brazier, F.M.T., Jonkers, C.M., Treur, J.: Principles of Compositional Multi-Agent System Development. Proceedings 15th IFIP World Computer Congress, WCC'98, Conference on Information Technology and Knowledge Systems, IT&KNOWS'98, S. 347–360. Chapman and Hall, London (1998)
- [4] Jonker, C.M., Treur, J.: Compositional verification of multi-agent systems: a formal analysis of pro-activeness and reactivity. Proc. International Workshop on Compositionality (COMPOS'97). Springer, Berlin Heidelberg (1997)
- [5] http://www.dfki.de/asr/index_mas_de.html
- [6] Kinny, D., Georgeff, M., Rao, A.: A Methodology and Modelling Technique for Systems of BDI Agents. 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96), S. 56–71. Springer, Berlin Heidelberg (1996)
- [7] Kinny, D., Georgeff, M.: Modelling and Design of Multi-Agent Systems. Intelligent Agents. Bd III. Springer, Berlin Heidelberg (1996)
- [8] Bauer, B., Odell, J.: UML 2.0 and Agents: How to Build Agent-based Systems with the new UML Standard. Special issue on Agent-oriented Software Development of the EAAI Journal (2005)
- [9] Bauer, B., Müller, J.P., Odell, J.: A formalism for specifying multiagent software systems. *IJSEKE* **11**(3), 1–24 (2001)
- [10] PASSI website: www.csai.unipa.it/passi
- [11] Padgham, L., Winikoff, M.: Prometheus: a methodology for developing intelligent agents. Proceedings of AOSE 2002. Springer, Berlin Heidelberg (2002)
- [12] Padgham, L., Winikoff, M.: Prometheus: a pragmatic methodology for engineering intelligent agents. Proc. of the workshop on Agent-oriented Methodologies at OOPSLA 2002 (2002)
- [13] MESSAGE web site: <http://www.eurescom.de/public/projects/P900-series/p907/>
- [14] Caire, G., Coulier, W., Garijo, F., Gomez, J., Pavon, J., Massonet, P., Leal, F., Chainho, P., Kearney, P., Stark, J., Evans, R.: Agent Oriented Analysis using MESSAGE/UML, Proceedings AOSE 2001. Springer, Berlin Heidelberg (2001)
- [15] Mylopoulos, J., Kolp, M., Castro, J.: UML for agent-oriented software development: the tropos proposal. Proc. of the 4th Int. Conf. on the Unified Modeling Language UML'01. Toronto, Canada (2001)
- [16] Tropos web site: <http://www.cs.toronto.edu/km/tropos/>
- [17] Deloach, S.A., Wood, M.F., Sparkman, C.H.: Multiagent systems engineering. *IJSEKE* **11**(3), 231–258 (2001)
- [18] Julian, V., Botti, V.: Developing real-time multi-agent systems. *Integrated Computer-Aided Engineering*, Bd. 11, S 135–149. IOS Press 135 (2004)
- [19] <http://macr.cis.ksu.edu/omase>
- [20] Warwas, S., Hahn, C.: The DSML4MAS Development Environment. In: Decker, K., Sichman, J., Sierra, G., Castelfranchi, C. (Hrsg.). Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems. International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2009), May 10–15, S. 1379–1380. Budapest, Hungary IFAAMAS (2009)

- [21] <http://www.whitestein.com/aml>
- [22] <http://www.whitestein.com/adem>
- [23] Bauer, B.: UML class diagrams revisited in the context of agent-based systems. Proceedings AOSE 2001, Montreal. Springer, Berlin Heidelberg (2001)
- [24] <http://oxygen.informatik.tu-cottbus.de/aor/?q=node/1>
- [25] Bernon, C., Gleizesm M.-P., Peyruqueou, S., Picard, G.: ADELFE, a methodology for adaptive multi-agent systems engineering. Third International Workshop „Engineering Societies in the Agents World“ (ESAW-2002), S. 7. Madrid (2002)
- [26] Bernon, C., Gleizes, M.-P., Picard, G., Glize, P.: The Adelfe Methodology For an Intranet System Design. Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002), 27–28 May 2002, Toronto (Ontario, Canada) at CAiSE’02 (2002)
- [27] Glaser, N.: Contribution to Knowledge Modelling in a Multi-Agent Framework (the Co-MoMAS Approach). PhD thesis. L’Université Henri Poincaré, Nancy I, France (1996)
- [28] Iglesias, C.A., Garijo, M., Gonzalez, J.C., Velasco, J.R.: A methodological proposal for multi-agent systems development extending CommonKADS. Proc. of 10th KAW. Banoe, Canada (1996)
- [29] Wooldridge, M., Jennings, N.R., Kinny, D.: The gaia methodology for agent-oriented analysis and design. *J. Aut. Agents Multi-Agent Sys.* 3(3), 285–312 (2000)
- [30] Juan, T., Pearce, A., Sterling, L.: ROADMAP: extending the gaia methodology for complex open systems. Proc. of AAMAS. ACM Press, New York (2002)
- [31] Omicini, A.: SODA: Societies and infrastructures in the analysis and design of agent-based systems. Proc. of AOSE. Springer, Berlin Heidelberg (2000)
- [32] OMG. Software Process Engineering Meta-Model, version 2.0. <http://www.omg.org/spec/SPEM/2.0/>, 2008.
- [33] Henderson-Sellers, B., Ralyté, J.: Situational method engineering: state-of-the-art review. *J. Univ. Comp. Sci.* (2010)
- [34] Karlsson, F., Egerfalk, P.J.A.: Method configuration: adapting to situational characteristics while creating reusable assets. *Inform. Soft. Technol.* **46**, 9 (2004)
- [35] Xu, P.: Knowledge support in software process tailoring. Proceedings of HICSS’05 – Track 3, Bd. 3. IEEE Computer Society, Washington, DC, USA (2005)
- [36] Asadi, M., Mohabbati, B., Gasevic, D., Bagheri, E.: Developing families of method-oriented architectures. IFIP WG8.1 Working Conference on ME (ME 2011). Springer, Berlin Heidelberg (2011)
- [37] Alegría, J., Bastarrica, M., Quispe, A., Ochoa, S.: An MDE Approach to Software Process Tailoring. International Conference on Software and Systems Process ICSSP. ACM Press, New York (2011)
- [38] Bauer, B., Bergenti, F., Massonet, P., Odell, J.: Agents and the UML: a unified notation for agents and multi-agent systems. Proc. AOSE 2001, Montreal. Springer, Berlin Heidelberg (2001)
- [39] Kinny, D., Georgeff, M., Rao, A.: A methodology and modeling technique for systems of BDI agents. Proc. of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW 96), LNAI 1038. Springer, Berlin Heidelberg (1996)
- [40] Kinny, D. and Georgeff, M.: Modelling techniques for BDI agent systems. Technical Report 54, Australian Artificial Intelligence Institute, Melbourne, Australia (1995)
- [41] AUML web-site: <http://www.auml.org>
- [42] Bauer, B., Bergenti, F., Massonet, P., Odell, J.: Agents and the UML: a unified notation for agents and multi-agent systems. Proc. AOSE 2001, Montreal. Springer, Berlin Heidelberg (2001)
- [43] Bauer, B., Müller, J.P., Odell, J.: An extension of UML by protocols for multiagent interaction. Proc. Fourth International Conference on MultiAgent Systems, ICMAS 2000, Boston. IEEE Computer Society (2000)

Kapitel 5

Middleware für Systeme mobiler Agenten in der Automatisierung

Hagen Stanek

Zusammenfassung Das Ziel dieses Artikels ist es, eine Middleware zu finden, die es ermöglicht, darauf basierend Systeme mobiler Agenten, d. h. solche, die ihren Zustand und Kode zu einem anderen Computer transferieren können, in der Automatisierung zukunftsträchtig zu entwickeln und zu betreiben. Keinesfalls geschieht das unter der Annahme, schon „die“ Lösung zu finden. Stattdessen möchte dieser Artikel einen Anstoß zu reger Diskussion und zu realen Entwicklungen bzw. Weiterentwicklung existierender Middleware bzw. Plattformen für Agenten-Systeme mit der Middleware als Basis geben.

5.1 Einleitung

Derzeit gibt es eine Reihe von Frameworks, die sich als Middleware anbieten. Dieser Artikel analysiert die vielversprechendsten Ansätze systematisch. Vor- und Nachteile einer Technik werden unter Berücksichtigung der Anforderungen der Automatisierungstechnik beleuchtet. Dabei werden auch weichere Gründe für eine Technik diskutiert. Hierzu zählt etwa die tatsächliche Entwicklungsgeschwindigkeit, mit der die Möglichkeiten einer Middleware oder deren bisherige Verbreitung in kommerziellen Anwendungen erweitert werden können. Diese Aspekte dürfen nicht unbeachtet bleiben und müssen gewichtet werden, will man nachhaltige Software-Entwicklung bei niedrigem Kostenaufwand betreiben.

Mit Middleware ist in diesem Artikel eine Software gemeint und keine Hardware. Gerade jetzt entwickeln sich im Umfeld des Cloud-Computing und im Bereich der Telefonie interessante Lösungen. Die Erwartung ist, dass eine solche Software aufgrund einer weiten Verbreitung eine höhere Stabilität, bessere Dokumentation und stärkere Werkzeugunterstützung aufweist. Das wird einer Plattform für Sys-

Hagen Stanek (✉)
genRob GmbH, Würmhalde 31, 71134 Aidlingen, Deutschland
e-mail: stanek@genRob.com

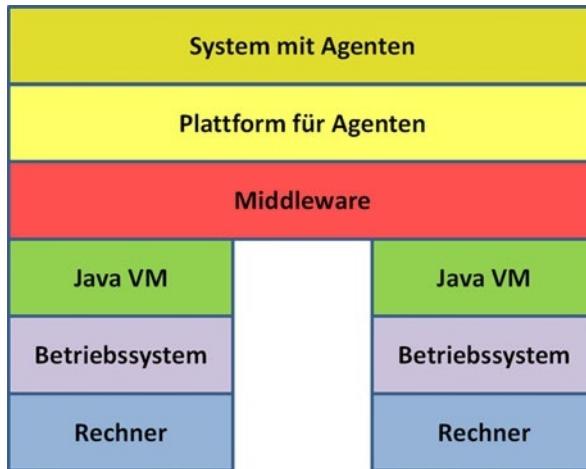


Abb. 5.1 Einordnung von Middleware, Plattform für Agenten und System mit Agenten beispielhaft für 2 Rechner

teme von Agenten, die darauf basiert, eine größere Stabilität verleihen und sie einfacher warten und weiterentwickeln lassen (vgl. Abb. 5.1).

Abschnitt 5.2 erläutert, wie eine Middleware systematisch auf ihre Tauglichkeit geprüft wird. Zunächst wird der Begriff Middleware geklärt. Dann ziehen wir die Definition von mobilen Agenten heran. Aus dieser leiten wir grundlegende Anforderungen ab, die durch eine Middleware erfüllt sein müssen. Natürlich reicht die Definition mobiler Agenten alleine nicht aus. Die Automatisierungsindustrie ist ein etablierter wirtschaftlicher Zweig mit vielen Standards, Normen und Vorschriften, die helfen, einen sicheren und stabilen Betrieb von Anwendungen zu gewährleisten. Zusammen mit den Einsatzbedingungen der Automatisierung, ergeben sich weitere Anforderungen an eine Middleware.

Um in einer kompakten Form eine Beurteilung bestehender Middleware-Ansätze vorzunehmen, werden in Abschn. 5.3 die wichtigsten Anforderungen zusammengeführt. Daraus ergibt sich ein Schema, nach dem im folgenden Abschn. 5.3.2 die einzelnen Techniken untersucht und beurteilt werden. Die Schlussfolgerungen aus diesen Einzelanalysen zeigen, welche Techniken derzeit die vielversprechendsten Ansätze für Systeme mobiler Agenten sind. Offene Fragen, Probleme und Herausforderungen bleiben, auf deren Lösung am Ende dieses Textes im Abschn. 5.4 gedeutet wird.

5.2 Grundlegende Anforderungen an eine Middleware

5.2.1 *Middleware und verteilte Anwendungen*

Im allgemeinsten Sinne ist Middleware (dt. Diensteschicht, Vermittlungs-Software) eine Computer-Software, welche anwendungsneutrale Dienste für Software-Anwendungen anbietet, die über die vom Betriebssystem Angeboteten hinausgehen [1, 2]. Dies kann in Form von Software-Bibliotheken geschehen und auch über dienstanbietende Anwendungen (Server). Die Middleware nutzt ihrerseits die Dienste des Betriebssystems oder anderer Middleware und verbirgt damit Komplexität. Ist die Middleware für verschiedene Betriebssysteme verfügbar, so werden zusätzlich Unterschiede zwischen den Betriebssystemen und der unterliegenden Hardware ausgeglichen [2].

Eine Middleware hat allgemein ein breiteres Einsatzgebiet, weshalb eine größere Praxiserprobtheit, geringere Fehlerquellen und bessere Werkzeuge für die Entwicklung und den Einsatz vorliegen.

In Bezug auf Middleware kann ein Software-System mit mobilen Agenten als verteilte Anwendung aufgefasst werden. Eine verteilte Anwendung ist eine Software, die in einem verteilten System, also auf mehreren Rechnern/Prozessoren, abläuft und unter diesen Informationen austauscht. Zur Erfüllung der Gesamtaufgabe müssen alle Anwendungsteile der verteilten Anwendung mitwirken und untereinander kommunizieren [3]. In verteilten Anwendungen bedeutet Middleware meist eine Software, welche Kommunikation und Datenverwaltung vermittelt [1].

5.2.2 *Definition mobiler Agenten*

Generell wird ein Agent als mobil angesehen, wenn er mit seinem Programmcode und Laufzeitdaten selbstständig zu einem anderen Computer wechseln kann, um dort seine Ausführung fortzusetzen [4].

Ein mobiler Agent ist eine Form von Software-Agent und ist daher darüber hinaus autonom (unabhängig von Benutzereingriffen), proaktiv (Aktionen aufgrund eigener Initiative), reaktiv (reagiert auf Änderungen der Umgebung), robust (kompenziert äußere und innere Störungen), adaptiv (ändert aufgrund der eigenen Zustände und der Zustände der Umgebung seine eigenen Einstellungen), kognitiv (lernfähig und lernt aufgrund zuvor getätigter Entscheidungen bzw. Beobachtungen) und sozial (kommuniziert mit anderen Agenten) [5].

Mehr softwaretechnisch gesehen ist ein mobiler Agent eine Art Prozess, welcher seinen Zustand von einer Umgebung in eine andere unter Erhalt seines Zustandes transferieren kann. Dabei kann der Agent selbst entscheiden wohin und wann transferiert wird. Transferiert wird durch Zustandsfeststellung in Form transferierbarer Daten, Transportieren dieser Daten in die neue Umgebung, Zustandswiederherstellen

lung aus diesen Daten und Fortführung des Agenten mit dem wiederhergestellten Zustand in der neuen Umgebung [4].

Nimmt man die Idee mobiler Programme hinzu, so transferiert ein mobiler Agent zusätzlich zu seinem Zustand auch noch seinen Kode zur neuen Umgebung. In diesem Falle muss die neue Umgebung den Kode des mobilen Agenten a priori nicht kennen, aber verstehen und ausführen.

Autonomie, Proaktivität, Reaktivität, Robustheit, Adaptivität, Kognitivität und Sozialität wird im Wesentlichen möglich, wenn eine freie Programmierbarkeit z. B. in Form einer höheren Programmiersprache vorliegt. Zusätzlich muss ein passender Ablaufmechanismus für die resultierenden Programme gegeben sein, sodass die Programme geladen, gestartet, beendet oder angehalten und schließlich entladen werden können.

Reaktivität erfordert zusätzlich einen Zugriff auf die Umgebung und damit normalerweise auf die passenden Ressourcen in Form von Hard- oder Software-Diensten. Sobald solche Ressource-Zugriffe durch mehrere Agenten parallel erfolgen könnten, müssen sie dann auch synchronisiert sein. Aus der Sozialität leitet sich weiterhin ab, dass besondere Dienste zur Verfügung stehen müssen, die den Agenten sozialen Kontakt ermöglichen. Das sind normalerweise Kommunikationsdienste lokal und für das Netz. Damit ein Agent sich selbst transferieren kann, muss sein Zustand und sein Kode transferiert werden können. Dazu sind ebenso spezielle Kommunikationsdienste erforderlich. Diese sind dann mit den o. g. Ablaufmechanismen gekoppelt. Und letztlich ergibt sich noch aus der Tatsache, dass die Ressourcen wie erwähnt parallel genutzt werden können und Agenten über die Ablaufmechanismen unerwartet enden können, dass eine sehr zuverlässige Laufzeitumgebung existieren muss, welche die Ablaufmechanismen bereitstellt und die Ressourcen verwaltet.

5.2.3 *Bedingungen der Automatisierung*

Als hauptsächliche Gründe für die Automatisierung werden die Verbesserung und Vergleichsmäßigung der Produktqualität, die Erhöhung der Durchsatzleistung, der Ersatz von Baugruppen, für die keine Ersatzteile mehr verfügbar sind, die Einsparung von Personalkosten und die Entlastung des Menschen von schwerer körperlicher oder monotoner Arbeit genannt [6].

Die Automatisierungstechnik, die zum Ziel hat, Maschinen oder Anlagen automatisiert, also selbstständig und ohne Mitwirkung von Menschen, zu betreiben, befasst sich dabei mit Methoden und Lösungen, wie Messen, Steuern, Regeln, Kommunikation, Bedienen, Sicherheit und Implementierung. Die Grenze für den Einsatz der Automatisierung ergibt sich heutzutage meistens aus der Wirtschaftlichkeit [7].

In der Praxis haben in der Automatisierungstechnik eine große Zahl von Anbietern und lange Zeiträume zu einer stark heterogenen Welt geführt. Diese Entwicklung wird sich fortsetzen. Aus Sicht der Informationsverarbeitung betrifft diese Heterogenität der Zukunft deshalb sämtliche Bereiche, wie Peripherie, Rechentechn-

nik, Betriebssystem u. a. m. und auch besonders dann, wenn komplette Prozesse von Web über Leitsystem hin zu Automatisierungsgerät einheitlich abbildbar sein sollen. Eine Middleware für Systeme mobiler Agenten muss eine Brücke über die Heterogenität spannen. Nur dann kann damit ein System frei transferierender mobiler Agenten geschaffen werden. Letztlich führt diese Anforderung zu den virtuellen Maschinen.

5.2.4 Virtuelle Maschinen

Um den Kode eines mobilen Agenten in einer heterogenen Welt nutzbar machen zu können, muss virtualisiert werden. Auf irgendeiner Ebene muss derart abstrahiert werden, sodass ein solcher Agent laufen kann. Zum einen muss der Kode grundsätzlich interpretiert werden können und zum anderen der Zugriff auf die Peripherie einheitlich möglich sein. Generell werden solche Konstruktionen virtuelle Maschinen genannt.

Eine virtuelle Maschine (VM) ist eine Software-Implementierung einer Maschine, welche Kode wie eine physische Maschine ausführt. Als ein wesentliches Merkmal solcher Maschinen wird der ausgeführte Kode auf die Ressourcen begrenzt, die die virtuelle Maschine bereitstellt [8].

Virtuelle Maschinen lassen sich abhängig von ihrem Einsatz und Grad der Übereinstimmung mit existierenden physischen Maschinen in System-virtuelle Maschinen und Prozess-virtuelle Maschinen einteilen. System-virtuelle Maschinen bieten eine zum Teil perfekte Übereinstimmung mit physischen Maschinen und werden eingesetzt, um vollständige Betriebssysteme zu virtualisieren. Prozess-virtuelle Maschinen laufen demgegenüber als ein einzelner Prozess in einem Betriebssystem und bieten den Programmen eine von physischen Maschinen unabhängige Maschinendefinition [8].

In der Praxis laufen System-virtuelle Maschinen nur jeweils auf gewissen Arten physischer Maschinen und Betriebssystemen, bieten ihrerseits dann aber wieder die Virtualisierung weiterer physischer Maschinen. Die lange Zeiträume in der Automatisierung und die nötigen Mindestanforderungen sprechen gegen den Einsatz System-virtueller Maschinen. Lange Zeiträume können zum Problem werden, wenn gewisse Arten physischer Maschinen einmal nicht mehr oder nur noch in geringen Stückzahlen gebaut werden und dann dadurch Ersatzteileinschränkungen auftreten. Die Größe solcher virtuellen Maschinen bedingt normalerweise auch eine gewisse größere Mindestgröße für das physische System mit entsprechendem Stromverbrauch, Wärmeentwicklung etc. Schließlich wächst mit der Größe statistisch auch die Fehlerzahl.

Die Prozess-virtuellen Maschinen sind der Idee nach demgegenüber nicht auf physische Maschinen und darauf laufende Betriebssysteme eingeschränkt. Dass der zu interpretierende Kode in der Regel nicht direkt von der physischen Maschine verarbeitet werden kann und deshalb interpretiert werden muss, gleichen spezielle Techniken aus.

Die vermutlich mit Abstand verbreitetsten Prozess-virtuellen Maschinen sind die Java-Virtual-Machine und die Virtual-Machine der Common-Language-Runtime als Teil des.NET-Framework. Die Verbreitung des.NET-Frameworks beschränkt sich Stand heute im Wesentlichen auf Windows-Systeme mit einer relativen hohen Grundleistung. Da in der Automatisierung aber möglichst viele Betriebssysteme und Hardware-Leistungsstufen für die mobilen Agentensysteme potenziell zum Einsatz kommen sollen, wird im Folgenden nur die Java-Virtual-Machine (JVM) betrachtet werden. Für die gesuchte Middleware bedeutet es daher zwingend, dass sie auf Java als Technologie-Plattform aufsetzen muss.

5.2.5 Zusammenfassung der grundlegenden Anforderungen an eine Middleware

Zusammenfassend kann man daher folgende grundlegende Anforderungen ableiten, die eine Middleware erfüllen muss, damit ein System mobiler Agenten möglich wird: Es muss eine zuverlässige Laufzeitumgebung für die mobilen Agenten basierend auf der Java-Virtual-Machine existieren, die den Zugriff für jeden mobilen Agenten auf Hard- und Software-Dienste, insbesondere Kommunikationsdienste und Transferdienste inklusive Ablaufmechanismen bereitstellt.

5.3 Middleware

In den vergangenen Jahren sind eine Reihe von Frameworks entstanden, die auf den ersten Blick als Middleware für mobile Agenten geeignet erscheinen. Dieser Abschnitt beschreibt im Folgenden, um welche Frameworks es sich handelt und was sie zu bieten haben.

5.3.1 Existierende Middleware

Die im nachfolgenden vorgestellte Middleware wurde bei der Suche im Internet mit Begriffen wie „distributed“, „computing“, „java“, „parallel“, „concurrent“, „framework“, „middleware“, „operating system“, „grid“ und den entsprechenden deutschen Begriffen gefunden. Die Suche fand im Mai 2012 statt. Es wurden nur die spontan glaubwürdigsten Kandidaten ausgewählt, da es sehr viele kleinere Middleware gibt, die z. B. einfach im Rahmen von Arbeiten von Studenten entstanden und keine repräsentativen Anwendungen zu erkennen sind. Weiterhin wurde die Middleware weggelassen, für die gar kein kommerzieller Einsatz erlaubt ist.

Tabelle 5.1 zeigt das Ergebnis der Suche in alphabetischer Reihenfolge.

Tab. 5.1 Die existierende Middleware

Middleware	Update	URL
Akka	22.5.2012	http://akka.io
Blitz JavaSpaces	7.5.2012	http://www.dancres.org/blitz
Cajo	21.2.2011	http://java.net/projects/cajo
Cleversave	Produkt	http://www.cleversafe.com
Clutch	2008	http://clutch.sourceforge.net
Cougaar	21.8.2007	http://www.cougaar.org
DAC	2009	http://www.dacframe.org
Globus Toolkit	26.4.2012	http://www.globus.org
GridEngine	Oracle-Produkt	http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html
Gridgain	14.5.2012	http://www.gridgain.com
Hadoop	16.5.2012	http://hadoop.apache.org
Hazelcast	2.5.2012	http://www.hazelcast.com
Ibis	9.12.2009	http://www.cs.vu.nl/ibis
JADIF	6.2010	http://jadif.sourceforge.net
Java CoG Kit	2006	http://wiki.cogkit.org
JavaParty	30.3.2007	http://svn.ipd.uni-karlsruhe.de/trac/javaparty
JCAF	18.5.2011	http://www.daimi.au.dk/~bardram/jcaf
JPPF	1.4.2012	http://www.jppf.org
Korus	2.2010	http://code.google.com/p/korus
P-GRADE portal	3.5.2011	http://portal.p-grade.hu
Pegasus	3.2012	http://pegasus.isi.edu
ProActive	12.4.2007	http://proactive.inria.fr
Red dwarf server	27.10.2010	http://www.reddwarfserver.org
Roblet	1.3.2012	http://roblet.org
nebulaframework	1.9.2008	http://code.google.com/p/nebulaframework
UNICORE	13.4.2012	http://www.unicore.eu

5.3.2 Reduzierung der Middleware-Liste

Eine Middleware sollte sich in der Weiterentwicklung befinden. Dies kennzeichnet eine aktuelle Nutzung. Zu diesem Zweck wurde die Liste der existierenden Middleware auf die eingeschränkt, die innerhalb der letzten 12 Monate einen neuen Software-Stand veröffentlicht haben: Akka, Blitz JavaSpaces, Cleversave, Globus Toolkit, GridEngine, Gridgain, Hadoop, Hazelcast, JPPF, Pegasus, Roblet und UNICORE.

Bewertungskriterien

Für eine sinnvolle und vergleichbare Bewertung werden acht Kriterien vorgeschlagen, anhand derer eine Middleware für die Umsetzung eines Systems mobiler Agenten für die Automatisierung beurteilbar wird.

Zusätzlich wird noch das vom Hersteller publizierte Anwendungsziel seiner Middleware angegeben. Erfahrungsgemäß können zu breite Zielsetzungen der Benutzbarkeit schaden; zu enge Zielsetzungen können Anwendungsfälle ausschließen oder für deren Umsetzung unsaubere Nutzungen eines Frameworks erfordern.

Java

Wie zuvor diskutiert, ist die Nutzung von Java und einer JVM als virtuelle Maschine in der Automatisierung der vielversprechendste Ansatz, um verteilte heterogene Systeme wirtschaftlich zu verbinden. Wie unterstützt die Middleware den Einsatz von Java? Residiert die diskutierte Middleware direkt in der JVM als Java-Programm oder kommt ein Java-Container zum Einsatz, in dem die Middleware zum Laufen gebracht wird?

Kommunikationsdienst

Kommunikation ist ein wichtiger Bestandteil eines Agentensystems. Wie weiter oben beschrieben, ist er nötig, damit die einzelnen Agenten miteinander kommunizieren können. Gängig ist hier das Austauschen von Nachrichten in Form von serialisierbaren Objekten. Die dazu nötigen Mechanismen sind oftmals nicht trivial, da sich in allgemeineren Fällen keine direkte Verbindung zwischen den Komponenten, auf denen die kommunizierenden Agenten residieren, herstellen lässt. Vielmehr muss dann ein effizienter Routing-Mechanismus einspringen, der auch mit Fehlersituationen im Netz klarkommt. An dieser Stelle sei betont, dass es nicht reicht, wenn die Rechner sich alle in einem IP-Netz befinden bzw. die Routing-Mechanismen von IP eine Verbindung herstellen können. Es muss ein Routing auf Agenten-Ebene stattfinden.

Transferdienst

Mobile Agenten müssen sich transferieren können. Dabei sind meist nicht nur Daten, sondern auch Kode zu verschieben. Unterstützt die Middleware direkt einen Transfer und in welcher Form?

Umgebungszugang

In der Automatisierungstechnik sollen mobile Agenten auf die Automatisierungsanlagen zugreifen können. Dabei sollen Zustandsinformationen über eine Anlage erfassbar sein und Aktorik angesprochen werden können. Wie wird ein solcher Zugriff auf dedizierte Hardware durch die Middleware unterstützt? Sichert die Middleware die Hardware-Schnittstellen bei parallel laufenden Agenten, in dem sie z. B. den Zugriff synchronisiert bzw. Mechanismen dafür bereitstellt?

Laufzeitumgebung

Laufzeitumgebung beschreibt den Teil einer Middleware, der den Zugriff für jeden mobilen Agenten auf Hard- und Software-Dienste, insbesondere Kommunikationsdienste und Transferdienste regelt und Ablaufmechanismen bereitstellt. Im Falle unkooperativen Verhaltens eines mobilen Agenten muss über die Ablaufmechanismen dieser abgebrochen oder alternativ die Laufzeitumgebung neu gestartet werden können.

Kode-Größe

Die Kode-Größe beeinflusst das Zeitverhalten einer verteilten Anwendung sowohl beim (Neu-)Start der verteilten Komponenten, als auch beim Verteilen der Agenten. Je größer der Kode, umso mehr muss möglicherweise beim Start einer Anwendung verteilt werden. Das muss dann bei möglicherweise begrenzter oder unzuverlässiger Bandbreite mit berücksichtigt werden. Dieser Einfluss erstreckt sich von der Entwicklungsphase bis zum Produktiv-Einsatz und weiter hin zur Fehlersuche in produktiven Systemen.

Dokumentationsgrad

Um ein System mobiler Agenten auf Basis einer Middleware zu erstellen, ist eine gute Dokumentierung von großer Bedeutung. Neben einer Einführung und Schnittstellenbeschreibung sind Beispiele und Tutorials – auch in Form von Videos – in der Praxis sehr hilfreich. Weiterhin sind auch die Webseiten der verschiedenen Middleware verschieden übersichtlich aufgebaut. Insgesamt sollte der Aufwand, den die Einarbeitung in die Middleware bedingt, so gering wie möglich bleiben.

Werkzeuge

Neben den notwendigen Programmierschnittstellen einer Middleware sind Werkzeuge für Betrieb und Entwicklung hilfreich. Werkzeuge reichen von einzelnen

Dateien zur Einbindung in Entwicklungsumgebungen bis zu Programmen, die verteilte Komponenten überwachen.

Bewertung

Anhand der zuvor aufgestellten Kriterien lassen sich nun die aktuellen Entwicklungsstände der unterschiedlichen Middleware beurteilen. Es sind nur die Kriterien kommentiert, zu denen sich klare Aussagen aufgrund der verfügbaren Dokumentation machen lassen.

Weitere Kriterien, wie Größe zur Laufzeit, Boot-Up-Zeit in verschiedenen Umgebungen, Netzwerk-toleranz, Medienbrüche, Sicherheit und Stabilität, bedürfen einer tieferen Untersuchung und waren nicht im Rahmen dieses Artikels möglich.

Akka

Ziel: Entwicklung und Betrieb verteilter Anwendungen mithilfe eines Aktor-Konzepts vereinfachen

1. Java: als Teil eines AppServers oder hauseigener AppServer
2. Kommunikationsdienst: Inter-Aktor-Kommunikation über Nachrichten
3. Transferdienst: vermutlich keine eingebaute automatische Kode-Verteilung
4. Umgebungszugang: vermutlich direkt, unsynchronisiert per Java
5. Laufzeitumgebung: höchstwahrscheinlich JVM
6. Kode-Größe: 20 MByte – umfasst Scala-Teile
7. Dokumentationsgrad: gut und übersichtlich – mindestens zur Einführung – stets Scala- und Java-Beispiele
8. Werkzeuge: Integrationshilfe für verschiedene Entwicklungsumgebungen

Schlussfolgerung: Ist wegen Aktor-Konzept ein guter Kandidat, aber leider sehr groß.

Blitz JavaSpaces

Ziel: Implementierung von „Tupelräumen“, d. h. eines verteilten assoziativen Speichers.

1. Java: Reines Java

Schlussfolgerung: Die Grundzüge der Idee sind sehr weit weg von möglichen Agenten-Konzepten.

Cleversave

Ziel: Aufteilung von Daten in unkenntliche Datenstücke und Verteilung derselben über sichere Verbindungen in ein verteiltes Speichernetz.

1. Java: Vermutlich reines Java

Schlussfolgerung: Die Grundzüge der Idee sind sehr weit weg von möglichen Agenten-Konzepten.

Globus Toolkit

Ziel: Web-Service-basiertes System zur Bereitstellung von Grid-Infrastruktur.

1. Java: Java nur in zweiter Ebene, d. h. Java nur als Teilkonzept.

Schlussfolgerung: Die Grundzüge der Idee sind sehr weit weg von möglichen Agenten-Konzepten.

GridEngine

Ziel: System zur Verwaltung verteilter Ressourcen zur Erreichung höchstmöglicher Rechenauslastung

Vormals SGE (Sun Grid Engine)

1. Java: Vermutlich reines Java

Schlussfolgerung: Die Grundzüge der Idee sind sehr weit weg von möglichen Agenten-Konzepten.

Gridgain

Ziel: Verarbeitung großer Datenmengen in Echtzeit im Speicher und inkl. HPC

1. Java: Java + Groovy + Scala + AOP + REST + JUnit + u. a.m. – C#-Client
2. Kommunikationsdienst: Messaging
3. Transferdienst: eingebaute automatische Kode-Verteilung
4. Umgebungszugang: vermutlich direkt, unsynchronisiert per Java
5. Laufzeitumgebung: JVM
6. Kode-Größe: 85 MByte – umfasst Scala- und Groovy-Teile u. v. a. m.
7. Dokumentationsgrad: gut und übersichtlich – mindestens zur Einführung – stets Scala/Groovy- und Java-Beispiele
8. Werkzeuge: Erhältlich über die „Professional“-Version

Schlussfolgerung: Sehr vielseitige Technik, die als allgemeine Grundlage dienen könnte, aber leider sehr groß.

Hadoop

Ziel: Zuverlässiges verteiltes Arbeiten auf Daten – dateibasierte Datenverarbeitung

1. Java: Gewisse Kernteile sind für Linux; Rest der Implementierung ist Java; Datenarbeit beliebige Anwendungen

Schlussfolgerung: Die Grundzüge der Idee sind sehr weit weg von möglichen Agenten-Konzepten.

Hazelcast

Ziel: In-Memory-Data-Grid

1. Java: rein
2. Kommunikationsdienst: Möglicherweise über Daten-Verteilungsmechanismus realisierbar
3. Transferdienst: vermutlich keine eingebaute automatische Kode-Verteilung
4. Umgebungszugang: höchstwahrscheinlich direkt, unsynchronisiert per Java, da nicht thematisiert
5. Laufzeitumgebung: höchstwahrscheinlich JVM
6. Kode-Größe: 2 MByte
7. Dokumentationsgrad: gut und übersichtlich – mindestens zur Einführung
8. Werkzeuge: Programm zur Anzeige von Daten

Schlussfolgerung: Mit einer anderen Zielstellung entwickelt, aber möglicherweise passend weiterentwickelbar.

JCAF

Ziel: Entwicklung kontextsensitiver Anwendungen

1. Java: Reines Java
2. Kommunikationsdienst: Reines RMI lt. Dokumentation, REST u. a. lt. beiliegender Bibliotheken
3. Transferdienst: nach eigenen Angaben unsichere Kode-Verteilung per RMI bzw. da Android auch unter den Bibliotheken ist, wohl aktuell gar nicht mehr
4. Umgebungszugang: vermutlich direkt, unsynchronisiert per Java
5. Laufzeitumgebung: höchstwahrscheinlich JVM
6. Kode-Größe: 2,3 MByte
7. Dokumentationsgrad: grundlegende Einführung
8. Werkzeuge: keine vorhanden

Schlussfolgerung: Mit einer anderen Zielstellung entwickelt, aber möglicherweise passend weiterentwickelbar.

JPPF

Ziel: Ausgerichtet auf verteiltes Rechnen im Grid

1. Java: Reines Java
2. Kommunikationsdienst: Nicht vorgesehen – ausgerichtet auf verteiltes Rechnen
3. Transferdienst: eingebaute automatische Kode-Verteilung
4. Umgebungszugang: vermutlich direkt, unsynchronisiert per Java
5. Laufzeitumgebung: höchstwahrscheinlich JVM
6. Kode-Größe: 1,5 MByte bei Optimierung über das Dateisystem – ein vielfaches, wenn nicht optimiert
7. Dokumentationsgrad: gut und übersichtlich – mindestens zur Einführung
8. Werkzeuge: Graphisches Monitoring-Werkzeug

Schlussfolgerung: Mit einer anderen Zielstellung entwickelt, aber möglicherweise passend weiterentwickelbar.

Pegasus

Ziel: Konfigurierbares System zur Verteilung und Ausführung abstrakter Workflows in einer großen Anzahl Ausführungsumgebungen einschließlich Laptops, Rechenzentren, Grids und Clouds

1. Java: Teilweise Java mit vielen System-spezifischen Anteilen

Schlussfolgerung: Die Grundzüge der Idee sind sehr weit weg von möglichen Agenten-Konzepten.

Roblet

Ziel: Verteilte heterogene Systeme insbesondere der autonomen mobilen Robotik

1. Java: Reines Java
2. Kommunikationsdienst: Über eine technikeigene-Kommunikation implementierbar
3. Transferdienst: eingebaute automatische Kode-Verteilung
4. Umgebungszugang: synchronisiert implementierbar
5. Laufzeitumgebung: Sub-JVM – über „protection domains“ und „Einheiten“ von der Basis-JVM und untereinander abgeschirmte Roblets
6. Kode-Größe: 0,5 MByte
7. Dokumentationsgrad: mindestens zur Einführung
8. Werkzeuge: Grundlegende Kommandozeilenprogramme

Schlussfolgerung: Dem Konzept nach auf die verteilte Ausführung von Kode mit Datenübertragung ausgelegt, was gut als Basis für Agenten-Systeme dienen könnte.

Tab. 5.2 Zusammenfassung der Vergleich der Middleware

	Akka	Gridgain	Hazelcast	JCAF	JPPF	Roblet
1) Java	(-)	(-)	(+)	(+)	(+)	(+)
2) Kommunikationsdienst	(++)	(+)	(-)	(o)	(-)	(o)
3) Transferdienst	(-)	(++)	(-)	(-)	(+)	(++)
4) Umgebungszugang	(o)	(o)	(o)	(o)	(o)	(+)
5) Laufzeitumgebung	(o)	(o)	(o)	(o)	(o)	(++)
6) Kode-Größe	(-)	(-)	(+)	(+)	(+)	(++)
7) Dokumentationsgrad	(++)	(++)	(+)	(o)	(+)	(+)
8) Werkzeuge	(+)	(+)	(+)	(-)	(+)	(o)

UNICORE

Ziel: HPC- und Grid-Computing – Ist darauf ausgelegt, dateibasiert Werte an Anwendungen zu geben und nach Abarbeitung wieder zu übernehmen.

1. Java: Reines Java, Eclipse-Java

Schlussfolgerung: Die Grundzüge der Idee sind sehr weit weg von möglichen Agenten-Konzepten.

Zusammenfassung

Die Bewertungen zeigen, dass mehr als die Hälfte der Middleware eher für das HPC- oder Grid-Computing gedacht sind. Letztendlich wird das sicherlich durch die momentan starken Fortschritte im Cloud-Computing und den damit auf einfacher Weise verfügbaren Ressourcen begünstigt.

Für die Automatisierung bleiben nur noch 6 Middleware übrig, die in Tab. 5.2 zusammengefasst sind. Dabei wird eine qualitative Beurteilung in Form von (++) , (+), (o), (-) und (-) vorgenommen, die damit von sehr gut bis eher negativ reicht.

Die meisten Produkte benutzen direkt die JVM zur Ausführung. Akka benötigt einen (möglicherweise nur einfachen) Anwendungs-Container. Bei Gridgain ist AOP im Einsatz, was teilweise Kompilationen zur Laufzeit in einer separaten JVM zur Folge hat.

Akka bietet wegen des Aktor-Konzeptes einen hervorragenden Kommunikationsdienst. Bei Gridgain kann ein einfacher Kommunikationsdienst vermutlich sofort realisiert werden. Bei JCAF und Roblet lässt sich ein solcher Dienst architektonisch begünstigt sicher realisieren. Dabei wird der Aufwand etwas größer sein, als bei Gridgain, aber vermutlich ist dann auch die Flexibilität höher. Hazelcast und JPPF scheinen nicht auf Kommunikation ausgelegt.

Kode-Transfer in praktischer Form, nämlich im Grunde transparent auf der Ebene von Java-Objekten, bieten nur Gridgain und Roblet. Die Middleware, die keinen Kode-Transfer einsetzt, ist kritisch einzuschätzen. Die Fähigkeit zum Kode-

Transfer bedingt, dass die Middleware intern darauf ausgerichtet ist. Eine Software lässt sich mit derartiger Technologie erfahrungsgemäß nur mit hohem Aufwand nachrüsten.

Der Zugang auf die Umgebung des jeweiligen Rechners ist nur bei Roblet mit Mechanismen versehen, die eine Synchronisation erzwingen können. Bei allen anderen Middleware müssen eigene Bibliotheken erstellt werden und die Entwickler müssen die in der Praxis auch nutzen. Das bedingt eine gewisse Unsicherheit, die mit möglicherweise Prüf-Programmen kompensiert werden müsste.

Bei den Laufzeitumgebungen ist nur bei Roblet ein Mechanismus vorhanden, der die zukünftigen Agenten sicher von der unterliegenden JVM und anderen Agenten trennt. Unkooperative Agenten lassen sich in dieser Laufzeitumgebung beenden, ohne dass die JVM neu gestartet werden muss. Die Agenten können dazu schadlos von den Ressourcen getrennt werden bzw. diese können passend auf eine Abtrennung reagieren. Das hat zur Folge, dass andere Agenten in dieser JVM ohne Unterbrechung weiterlaufen können. Würde man die JVM neu starten müssen, so muss die Startzeit der JVM, der Middleware, der Netzwerkaufbau, das Laden und Starten der Agenten u. a. m. in einem solchen Szenarium mit berücksichtigt werden.

Hinsichtlich der Kode-Größe sind Akka und GridGain eher schlecht einzustufen. Alle anderen sind gut bis sehr gut. Kode-Größe macht sicher spätestens in produktiven Systemen bemerkbar, wenn zur Suche von Fehlern neue Software-Versionen auf alle Komponenten verteilt werden müssen. Auch der Start der Middleware geht langsamer vorstatten.

Die Dokumentation war bei keiner Middleware schlecht. Akka und Gridgain fielen durch viel Aufwand und hohe Qualität auf.

Werkzeugunterstützung ist äußerst unterschiedlich und nur JCAF scheint gar nichts zu bieten.

5.4 Zusammenfassung

Theoretisch wäre Akka der Idee mobiler Agenten am nächsten. Viele Problemstellungen, die entstehen, wenn man verteilte Anwendungen erstellt, die eher nicht homogen sind, werden durch den Aktoren-Ansatz aufgegriffen. Dazu gehört auch der ausgereifte Kommunikationsdienst mit Adressierungsschema u. a. Aber Akka unterstützt von Haus aus keinen Kode-Transfer und im Grunde zu groß. Den Kode-Transfer bieten nur Gridgain und Roblet praktisch ausgereift. GridGain ist für die Anwendung in der Automatisierung auch zu groß und hat als Laufzeitumgebung nur die JVM. Für Roblet spricht, dass die Laufzeitumgebung und der Umgebungszugang auf den einzelnen Agenten ausgelegt ist. Im Grunde wäre deshalb eine Entwicklung anzustreben, die alle Vorteile von Akka mit denen von Roblet verbindet. Das bedeutet, die Erfahrungen mit den Aktoren wären mit der Kompaktheit und Stabilität der Roblet-Technik zu kombinieren.

Literatur

- [1] Wikipedia (2012). Middleware. <http://en.wikipedia.org/wiki/Middleware>. Zugegriffen: 31. Mai 2012
- [2] Wikipedia (2012). Middleware. <http://de.wikipedia.org/wiki/Middleware>. Zugegriffen: 31. Mai 2012
- [3] Wikipedia (2012). Verteilte Anwendung. http://de.wikipedia.org/wiki/Verteilte_Anwendung. Zugegriffen: 31. Mai 2012
- [4] Wikipedia (2012). Mobile agent. http://en.wikipedia.org/wiki/Mobile_agent. Zugegriffen: 31. Mai 2012
- [5] Wikipedia (2012). Software-Agent. <http://de.wikipedia.org/wiki/Software-Agent>. Zugegriffen: 31. Mai 2012
- [6] Wikipedia (2012). Automatisierung. <http://de.wikipedia.org/wiki/Automatisierung>. Zugegriffen: 31. Mai 2012
- [7] Wikipedia (2012). Automatisierungstechnik. <http://de.wikipedia.org/wiki/Automatisierungstechnik>. Zugegriffen: 31. Mai 2012
- [8] Wikipedia (2012). Virtual machine. http://en.wikipedia.org/wiki/Virtual_machine. Zugegriffen: 31. Mai 2012

Kapitel 6

Agentenmodelle in der Anlagenautomation

Ulrich Epple

Zusammenfassung Ein Agentensystem kann nur dann erfolgreich eingesetzt werden, wenn es sich in seiner „Denke“ und in seiner Grundstruktur harmonisch in bestehende Konzepte der Anwendungsdomäne einbetten lässt. In vielen Fällen, in denen versucht wurde Agentensysteme ohne Rücksicht auf bestehende Strukturen einer Anwendungsdomäne als neues Paradigma überzustülpen, war der Einsatz nicht nachhaltig und hat zu keiner weiteren Verbreitung geführt. Im Folgenden wird daher untersucht, wie eine agentenorientierte Modellierung in die bestehenden Strukturen der Anlagenautomatisierung integriert werden kann. Ausgangspunkt ist die in der Anlagenautomation weit verbreitete Funktionsmodellierung und Softwaredirektorierte Funktionsbausteine. Durch schrittweise, intuitiv naheliegende Erweiterungen entsteht aus der Funktionsbausteintechnik eine Plattform, die sich als Grundlage für ein Agentensystem eignet.

6.1 Einleitung

Die Möglichkeit, bestimmte komplexe Systeme als Netzwerk aus autonomen Akteuren zu modellieren, die untereinander kommunizieren und jeweils eigene Ziele verfolgen, wurde schon frühzeitig in der KI angedacht. 1977 diskutierte Hewitt [1] ein System von Kontrollkomponenten, die durch Nachrichtenverbindungen miteinander verbunden sind. In den 80er Jahren entstand dann der Begriff des Agenten [2]. In den 90er Jahren wurden Softwareagenten ein zentrales Paradigma der Informatik [3]. In der Anlagenautomatisierung, insbesondere in der Prozesstechnik, hat die Komponentendanke uralte Wurzeln aus der Einzelgerätetechnik. Diese Komponentendanke wurde in die Leitsystemära übernommen. Allerdings fehlte und fehlt bis heute eine durchgängige nachrichtenorientierte Kommunikation. Schon

Ulrich Epple (✉)
RWTH Aachen, 52064 Aachen, Deutschland
e-mail: epple@plt.rwth-aachen.de

im Jahr 2000 gab es erste Ansätze die Bausteintechnik durch Einführung einer nachrichtenorientierten Kommunikation und einer internen Wissensbasis zu einem agentenorientierten Konzept zu erweitern [4]. Damals gab es jedoch sowohl für die Einführung einer nachrichtenorientierten Kommunikation als auch für die Einführung eines agentenorientierten Konzepts in der Anlagenautomatisierung noch keine Akzeptanz. Das Thema blieb jedoch in der Automatisierungstechnik aktuell und wurde weiterverfolgt [5, 6]. 2005 wurde der GMA Fachausschuss „Agenten in der Automatisierungstechnik“ gegründet. Mit der von diesem Fachausschuss im Jahr 2009 veröffentlichten VDI/VDE-Richtlinien 2653 wurden erstmals die Begrifflichkeiten geklärt und für die Automatisierungstechnik standardisiert verfügbar gemacht.

Dies war dringend erforderlich, da auf die Anlagenautomatisierung neue Herausforderungen zukommen, die mit den klassischen Techniken nicht mehr effizient zu lösen sind. Als besondere Herausforderung für den operativen Betrieb sind dabei anzusehen:

- Integration der leittechnischen Systemdienste

Leittechnische Systemdienste wie z. B. *Melden, Alarmieren, Archivieren* usw. werden auf jeder Ebene benötigt. Es ist daher sinnvoll, für jeden dieser Dienste eine einheitliche Lösung zu schaffen, die dann über alle Ebenen hinweg genutzt werden kann. Als Grundlage können die von den klassischen Leitsystemen angebotenen leittechnischen Systemfunktionen dienen.

- Virtualisierung der Infrastruktur

Betrachtet man die Prozessleitebene und die MES-Ebene zusammen, dann kann man von vornherein von einer verteilten heterogenen Systemlandschaft ausgehen. Die leittechnischen Dienste und Funktionen werden sich auf verschiedene Komponenten verteilen. Die Funktionsverteilung und die Konfiguration des Kommunikationsnetzes wird sich während des Lebenszyklus der Anlage mehrfach ändern. Insbesondere in flexiblen Anlagen ist es daher erforderlich, die Verteilung und Kommunikationskonfiguration automatisch vorzunehmen oder zumindest durch eine Virtualisierungsschicht gegenüber der Anwendung zu verbergen. Die Virtualisierungsschicht ermöglicht es auch zukünftige IT-Architekturen, wie z. B. das Cloud-Computing für bestimmte Funktionsbereiche in der MES-Ebene einzusetzen.

- Standardisierung des HMI

In Zukunft wird der Anlagenfahrer immer umfassendere und komplexere Aufgabenstellungen mit übernehmen müssen. Neben dem „einfachen“ Fahren der Anlage wird die ständige Überwachung der Performance-Indikatoren, die Optimierung der Energie- und Stoffverbräuche und eine effiziente Unterstützung einer flexiblen Produktion zunehmend zu seinen Aufgaben gehören. Dazu muss er die vielfältigen Funktionen und Tools der MES-Ebene genauso sicher beherrschen wie die klassischen Prozessführungsfunctionen. Dies kann jedoch nur gelingen, wenn ihm die gesamte Funktionspalette in einer durchgängigen, standardisierten und intuitiv erfassbaren Oberfläche angeboten wird. Neben der grafischen Oberflächengestaltung ist dabei insbesondere wichtig, dass prinzipiell

gleichartige Prozessabläufe auch durch prinzipiell gleichartige Eingabeprozeduren bedient werden.

- **Funktionsadaption zur Laufzeit**

Sowohl in flexiblen Produktionsanlagen als auch in klassischen starren Produktionsprozessen wird die Funktionsadaption zur Laufzeit eine immer wichtigere Forderung. Die Implementierung temporärer Diagnose- und Auswertefunktionen zur effizienten Klärung und Lösung auftretender Probleme, die Aufrüstung auf neue Softwarestände, die Implementierung verbesserter Regel- und Optimierungsfunktionen usw. sind erforderlich, um eine Anlagen verfügbar und auf dem Stand der Technik zu halten. In flexiblen Produktionsanlagen muss zusätzlich ständig ein Teil der operativen Führungsfunktionen rekonfiguriert werden.

Um den dargestellten Herausforderungen begegnen zu können, genügen die klassischen Strukturierungskonzepte und Standards der Automatisierungstechnik nicht mehr. Sie müssen durch neue Konzepte und Standards ergänzt und abgerundet werden. In diesem Beitrag wird ein Literaturmodell als Grundlage für eine zukünftige Systemarchitektur vorgestellt. Das Modell baut auf dem *Komponentengedanken* auf. Es nutzt die Vorteile einer *Serviceorientierung* [7, 8] und das kybernetische Paradigma der *Autonomen Manager* [9]. Auf diesen Grundpfeilern wird ein einfaches, robustes Agentensystem und zur Unterstützung des Bedieners ein effizientes und flexibles Assistenzsystem definiert.

6.2 Konzept eines Agentensystems

6.2.1 Technische Agenten auf Funktionsbausteinbasis

Technische Agenten sind Funktionseinheiten, die sich durch folgende Eigenschaften auszeichnen [VDI/VDE 2653-1]:

- Autonomie
- Interaktionsfähigkeit
- Kapselung
- Persistenz
- Reaktivität

Sie können als selbstständige, klar von ihrer Umwelt abgegrenzte Komponenten angesehen werden, die in der Lage sind, Aufträge eigenverantwortlich abzuwickeln, und die mit diesen Aufträgen verbundenen Ziele proaktiv zu verfolgen. Diese „Komponentendenke“ entspricht den Vorstellungen der Prozessautomatisierung und ist intuitiv verständlich. Die Funktionsbausteintechnik als eine traditionelle Form zur Modellierung automatisierungstechnischer Funktionalität, besitzt schon eine Reihe der angesprochenen Charakteristika, so z. B. die Kapselung, die Persistenz und die Reaktivität. Auch die Autonomie und Interaktionsfähigkeit ist bereits angelegt, allerdings nur rudimentär vorhanden. Dies röhrt aus der strikten Signal-

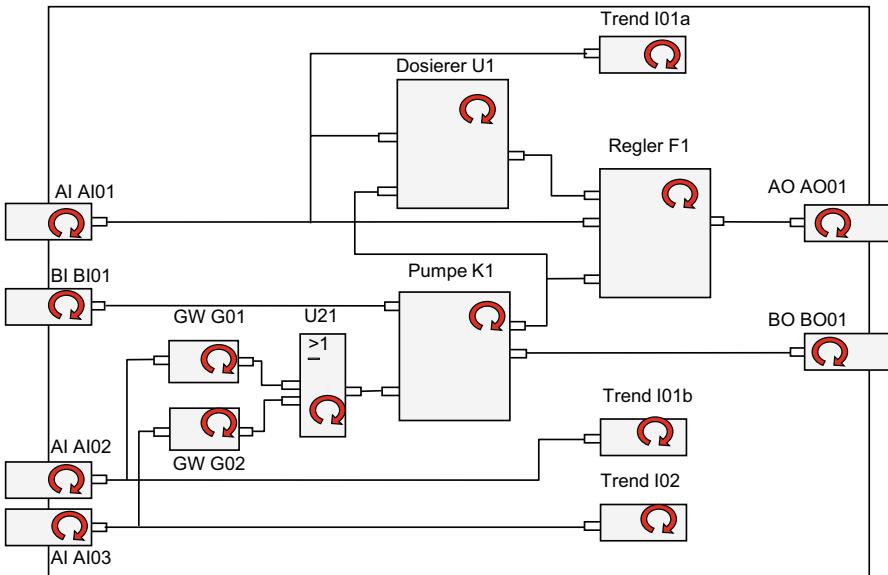


Abb. 6.1 Klassisches Funktionsbausteinsystem mit rein signalorientierter Kommunikation

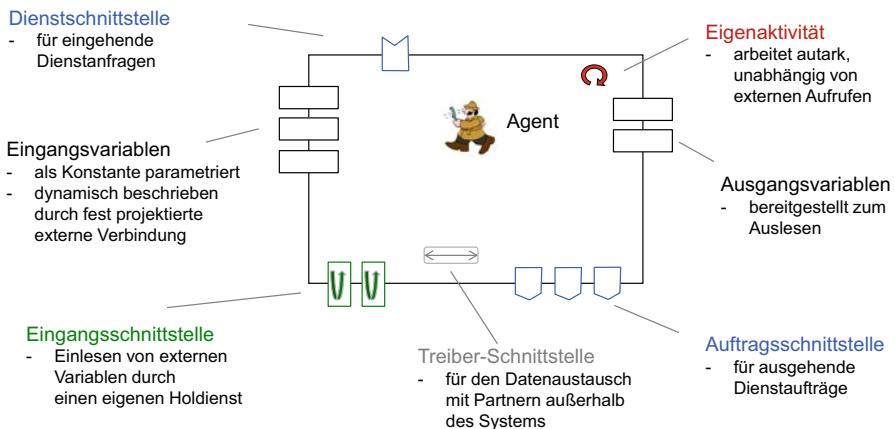


Abb. 6.2 Literaturmodell eines technischen Agenten in der Anlagenautomation

orientierung der Funktionsbausteinkommunikation. In Abb. 6.1 ist ein Beispiel für ein Funktionsbausteinsystem dargestellt.

Erweitert man die Kommunikationsmöglichkeiten eines Funktionsbausteins wie in Abb. 6.2 dargestellt, erhält man ein einfaches Literaturmodell für einen technischen Agenten auf Funktionsbausteinbasis.

Die Eingangs- und Ausgangsvariablen des klassischen Bausteinkonzepts wurden 1:1 übernommen. Neu hinzugekommen sind:

- die Auftragsschnittstelle,
- die Dienstschnittstelle und
- die Eingangsschnittstelle.

Die neu hinzugekommenen Schnittstellen öffnen die Bausteintechnik für eine dienstorientierte Architektur [7, 8]. Die Art der definierten Dienste prägt das Agentenkonzept. Die Dienste sind die Grundlage der agentenspezifischen Interaktion. Die in Abb. 6.2 definierten Schnittstellen lassen sich wie folgt charakterisieren.

6.2.2 Dienstschnittstelle

Über die Dienstschnittstelle erhält ein technischer Agent Aufträge. Ein Agent kann mehrere Dienstschnittstellen besitzen (z. B. eine für die operativen Aufträge, eine für Verwaltungsaufträge usw.). Welche Dienstschnittstellen ein Agent besitzt und welche Dienste er darüber anbietet ist typspezifisch. Der Aufbau und die Interaktion eines Dienstaufrufs ist jedoch ein Charakteristikum des Agentensystems. Ein Dienstauftrag besteht aus

- der Adresse des Dienstempfängers
- der Adresse des Dienstaufrufenden
- der Bezeichnung des Dienstes
- eventuell Dienste spezifischen Parametern und
- einer Reihe von definierten Verwaltungsparametern (QoS, ...).

Die Interaktion erfolgt als einfaches Senden ohne Antwort. Auch die Ablehnung eines Aufrufs wird nicht an den Aufrufenden zurückgesendet. Der Dienstempfänger führt jedoch eine über einen Ausgang lesbare Liste der letzten eingegangenen Aufträge und ihres Status.

Ein Empfänger kann, je nach Typ und Aufgabenstellung, eingehende Aufträge puffern oder nicht.

Der Verzicht auf eine Antwort mag erstaunen, er ermöglicht jedoch die einfache autonome Ablauforganisation im Klienten. Er vermeidet komplizierte Konsistenzsicherungs- und Synchronisationsfunktionen. Der Klient kann blockierungsfrei weiter arbeiten und fragt erst dann, wenn er dies aus technologischen Gründen benötigt, den technologischen Status des Dienstempfängers ab.

Diese Vorgehensweise erlaubt es auch im Falle von Störungen, die Störung extern zu beheben, ohne dass dies zu Schwierigkeiten in den auftraggebenden Einheiten führt. Dazu ein Beispiel: Eine Führungseinheit gibt einer Ventileinheit den Befehl zum Öffnen. Danach gibt sie eine Reihe weiterer Befehle an andere Einheiten aus. Im nächsten Schritt muss sichergestellt werden, dass das Ventil geöffnet ist. Deshalb fragt die Führungseinheit über eine Eingangsschnittstelle die Rückmeldung des Ventils ab. Ist nach einer bestimmten Zeit die Rückmeldung „offen“ nicht da, dann geht die Führungseinheit in Störung. Im Prinzip ist es der Führungseinheit dabei egal, ob sie durch ihren Befehl das Ventil geöffnet hat oder der Anlagenfah-

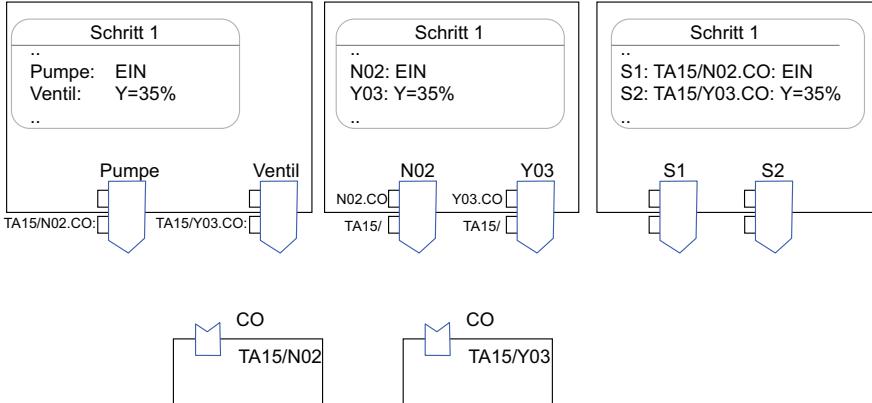


Abb. 6.3 Parametrierung von Auftragsschnittstellen

rer durch händischen Eingriff. Für sie ist nur der erreichte Zustand des Ventils von Bedeutung.

Bei bestimmten Nachrichten ist es erforderlich, sich den Empfang explizit bestätigen zu lassen, z. B. bei der gesicherten Übertragung einer Meldung. In einem solchen Fall sieht das Konzept vor, dass der Endempfänger explizit eine eigene Empfangs-Nachricht direkt an den ursprünglichen Absender schickt. Alle Zwischenheiten können dabei übergangen werden.

6.2.3 Auftragsschnittstelle

Über die Auftragsschnittstelle kann ein Agent Dienstaufrufe an einen Dienstempfänger im Netzwerk absetzen. Auftragsschnittstellen sind eingeschachtelte Bausteine und können selbst außerhalb der Kapselung parametriert werden. So kann z. B. die Adresse des Dienstempfängers im einzelnen Anwendungsfall statisch oder dynamisch vollständig ersetzt, teilweise ergänzt oder gar nicht beeinflusst werden. Dies erlaubt sowohl die Instantiierung von Agententypen mit Alias-Dienstempfängern als auch die direkte Adressierung von Empfängerinstanzen.

In Abb. 6.3 sind die verschiedenen Möglichkeiten dargestellt. In allen drei Fällen wird ein Einschaltauftrag an die Pumpe N02 und ein Positionsaufruf an das Ventil Y03 gesendet.

Im ersten Fall werden im Agenten die Alias-Namen „Pumpe“ und „Ventil“ verwendet. Hier wird die tatsächliche Adresse am externen Eingang des Auftragskonnektors parametriert. Dieser Fall tritt z. B. auf, wenn die Instanz eines typisierten Agenten an eine externe Umgebung angepasst werden muss. Im zweiten Fall trifft der Agent auf einen standardisierten Umgebungstyp. Hier sind die lokalen Namen bekannt und können im Agenten verwaltet werden. Extern ist dann nur noch der

Name der Umgebungsinstanz (z. B. der Name der Teilanlage) zu parametrieren. Im dritten Fall ist der Agent nicht typisiert, sondern intern vollständig für die spezielle Umgebung projektiert. Hier folgt der Aufruf direkt mit dem Instanznamen des Diensterbringers. Die Ausgangsschnittstellen sind frei zuordnenbar.

Die in Abb. 6.3 dargestellte Realisierung ist nur beispielhaft zu sehen. Wichtig ist die Kombinierbarkeit von interner lokaler Adressierung mit einer Adressierung außerhalb der Kapsel. Parametrierbarkeit der globalen Literatur.

6.2.4 Eingangsschnittstelle

Ein weiteres wesentliches Element des Agentensystems ist die Eingangsschnittstelle. Diese Schnittstelle ist eine Dienstschnittstelle (Get-Dienst) zur direkten Abfrage von Bausteinausgängen im Netz. Mit ihr kann jeder Bausteinausgang im gesamten Netzwerk flexibel abgefragt werden. Für die Eingangsschnittstelle gibt es keine Möglichkeit ein blockierendes Verhalten der internen Bausteinlogik (nicht der Ausführungslogik) zu umgehen. In dezentralen Systemen kann es etwas dauern, bis die Antwort auf eine Abfrage eintrifft. Im Allgemeinen wird die Bausteinlogik die Abfrage in einem Zyklus anstoßen und dann in den Folgezyklen prüfen, ob eine aktuelle Antwort eingetroffen ist.

6.3 Agenten der Anlagenautomatisierung

Ein Agentensystem besteht aus einer Vielzahl technischer Agenten, die miteinander interagieren und so ihre Aufgaben lösen. Im Umfeld der Anlagenautomatisierung lassen sich die beteiligten technischen Agenten nach ihren Aufgaben und Eigenschaften wie in Abb. 6.4 dargestellt typisieren.

Die Idee, die Automatisierungsfunktionen der Leittechnik als technische Agenten zu modellieren, entspringt den Anforderungen an mehr Flexibilität bei gleichzeitig sicherer und verlässlicher Handhabung. Damit sich das Ganze zu einem Agentensystem formt, müssen auch die Komponenten, an deren Flexibilität und Zielorientierung nur geringe Ansprüche gestellt werden, als Agenten modelliert werden. Für das Konzept spielt es keine Rolle, dass diese Agenten nur einen geringen Handlungsspielraum besitzen. Wichtig ist, dass sie sich in die Interaktionsplattform integrieren lassen und sich den flexiblen Agenten als systemkonforme Partner darstellen.

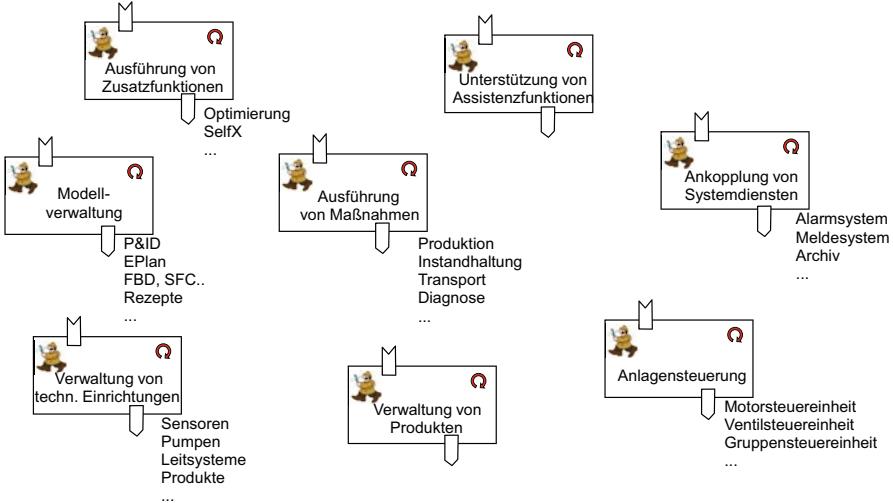


Abb. 6.4 Typisierung der technischen Agenten

6.3.1 Agenten für die Anlagensteuerung

Anlagennahe Steuer- und Regelfunktionen bilden die installierte Funktionalität einer Anlage ab und können prozessunabhängig, wie in Abb. 6.6 dargestellt, als Einzel- und Gruppensteuereinheiten realisiert werden. Diese Steuereinheiten können mit einer Auftragsschnittstelle versehen als technische Agenten angesehen werden. Natürlich wird man einem Motorsteueragenten auch weiterhin nur einen eng begrenzten Handlungsspielraum einräumen, er wird jedoch ansonsten zu einem vollwertigen Agenten: autonom, gekapselt, persistent und mit seiner Auftragschnittstelle eingebunden in die Interaktionsplattform. Anlagennahe Steueragenten werden fest projektiert und bleiben für die Lebenszeit der Anlage persistent installiert. Sie sind, unabhängig davon, ob sie gerade einen Auftrag bearbeiten oder nicht, ständig aktiv und überwachen die ihnen zugeordnete technische Einheit. Für komplexe technische Aufgabenstellungen können sie mit einem eigenen Analysemodul zur Situationserkennung und Reaktionsgenerierung ausgestattet werden [4]. In vielen Fällen werden diese Agenten fest an die unterlagerte Signalebene angekoppelt. Sie lassen sich jedoch auch hierarchisch aufeinander aufbauen. In diesem Fall kommunizieren die überlagerten Steueragenten mit den unterlagerten Steueragenten über das Agentensystem.

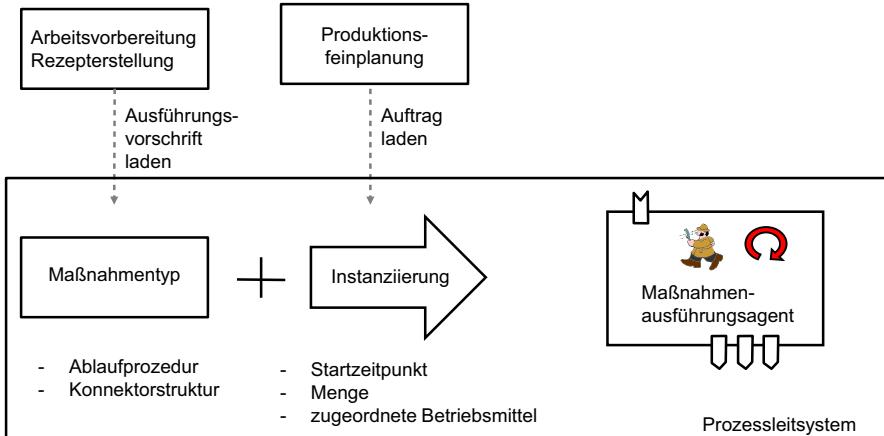


Abb. 6.5 Agenten zur Ausführung von Maßnahmen

6.3.2 Agenten zur operativen Ausführung von Maßnahmen

Unter einer Maßnahme versteht man die operative Durchführung eines technischen Prozesses. Der Begriff ist allgemein auf jeden technischen Prozess anwendbar. Es gibt Produktionsmaßnahmen, Diagnosemaßnahmen, Instandsetzungsmassnahmen, Engineeringmaßnahmen, Baumaßnahmen usw. Die Abwicklung einer Maßnahme erfolgt nach einer vorgegebenen Prozedur die durch eine allgemeine Ausführungs-vorschrift oder ein Rezept festgelegt ist. Die operative Ausführung muss durch eine aktive Steuereinheit erfolgen.

Es bietet sich an, die Steuereinheit, wie in Abb. 6.5 dargestellt, durch einen Agenten zu realisieren. Gerade im Falle der Ausführung von Maßnahmen haben Agenten entscheidende Vorteile. Hier werden die zugeordneten Anlagensteuer-agenten erst durch die Disposition festgelegt. Die Kommunikation kann nicht fest projektiert werden, sondern muss flexibel und dynamisch erfolgen. Das Agentensystem stellt dazu eine geeignete Plattform einschließlich der erforderlichen Sicherungs- und Belegungsmechanismen systemseitig zur Verfügung. Ein Maßnahmen-Ausführungsagent ist nur temporär existent. Er wird dynamisch initiiert, führt seine Aufgabe durch und wird dann wieder gelöscht. In Abb. 6.6 wird das Zusammenspiel zwischen einem Maßnahmen-Agenten und den Anlagenautomatisierungsagenten verdeutlicht.

Maßnahmen können selbst andere Maßnahmen initiieren und zur Lösung ihrer Aufgabe nutzen. Die durch die Handhabung der Maßnahmen erwachsenden Anforderungen an Flexibilität und dynamische Zuordenbarkeit bei gleichzeitiger Sicherstellung der korrekten Funktionalität, war eine der wesentlichen Triebfedern bei der Konzeption agentenorientierter Modelle in der Anlagenautomation.

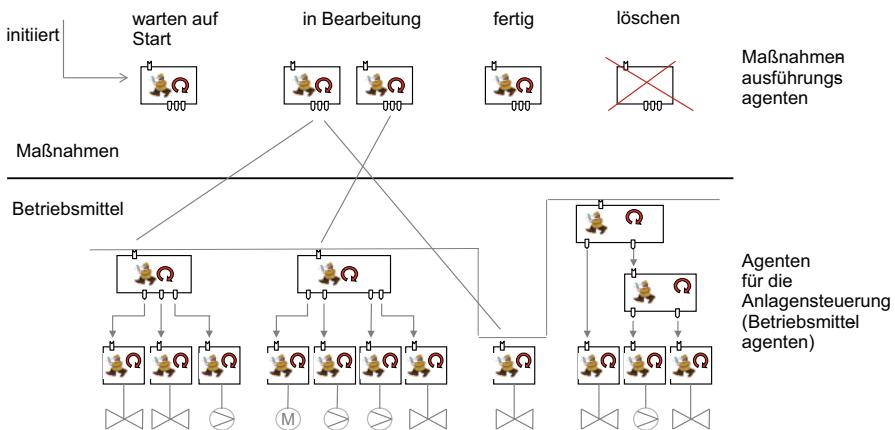


Abb. 6.6 Der Lebenslauf eines Maßnahmen-Ausführungsagenten

6.3.3 Agenten zur Ankopplung der Systemdienste

Systemdienste wie Zustandsarchive, Meldearchive, Merkmaldatenbanken, das Meldesystem, das Alarmsystem usw. sind Querschnittsdienste, die in allen Ebenen und Funktionsbereichen der operativen Prozess- und Betriebsführung benötigt werden. Heute sind diese Querschnittsfunktionen für verschiedene Anwendungen getrennt und heterogen installiert. Ziel muss es sein, für alle Funktionsebenen und Anwendungen einen einheitlichen, sicheren und effizienten Zugang zu schaffen. Dies kann durch technische Agenten als Repräsentanten der Systemdienste erfolgen. Diese Repräsentanz-Agenten bieten die Funktionalität der Systemdienste in Form von einfachen einzelnen Diensten den anderen Anwendungen im Netz an. Es ist jedoch auch möglich die Verwaltung der Systemdienste selbst agentenorientiert zu realisieren. Die Verwaltungssagenten können die verschiedenen Teilsysteme eines Systemdiensts optimal auf die Ressourcen verteilen und die Aufträge nach Auslastung und QoS-Anforderungen der Anwendungen den einzelnen Diensterbringern zuordnen [10]. Hier eröffnet das agentenorientierte Modell ein weites Feld zur Optimierung und zur Nutzung der SelfX-Technologien.

6.3.4 Agenten zur Verwaltung von technischen Einrichtungen

Technische Einrichtungen wie Pumpen, Sensoren, Kabelkanäle, Rohrleitungen usw. sind Assets des Betriebs. In einem agentenorientierten Konzept kann jeder technischen Einrichtung, die als Entität individuell verwaltet wird, ein Agent zugeordnet werden, der den Zustand der technischen Einrichtung aktiv überwacht, eventuelle Probleme erkennt und den Zugriff auf die Zustands- und Verwaltungsinformationen

für die Anwendungen einheitlich organisiert. Bei intelligenten technischen Einrichtungen, wie z. B. Feldgeräten und Automatisierungskomponenten, kann der Agent direkt in der Komponente implementiert werden oder zumindest auf die aktuellen Zustände der Einrichtung zugreifen.

6.3.5 Agenten zur Verwaltung von Produkten

Die Herstellung und Umwandlung von Produkten ist das zentrale Ziel eines technischen Prozesses. Zur einheitlichen Darstellung der Produkte und ihrer Lebenszyklen werden technische Agenten implementiert. Jedem Produkt (Entität) wird ein eigener Agent zugeordnet. Dieser verfolgt das Produkt über dessen gesamten innerbetrieblichen Lebenszyklus, sammelt alle Lebenszyklusdaten, stellt den Anwendungen einen einheitlichen lokalisierten Zugriff auf die Produktinformationen zur Verfügung und überwacht den Produktprozess.

6.3.6 Agenten zur Verwaltung von Modellen

Zur Lösung anspruchsvoller Aufgabenstellungen z. B. bei der Optimierung, Adaption, Diagnose, Selbstheilung, Selbstkonfiguration ist der formale Zugriff auf verschiedene Metadaten erforderlich. Ziel ist es daher, Metadaten, wie z. B. das P&I-Diagramm, die PLT-Stellenpläne, die Elektropläne, die Prozessbeschreibungen, die Ausführungsvorschriften aber auch die Struktur der Prozessführungsfunctionalität und des Agentensystems selbst, in Modellform explizit zu beschreiben, um auf diese Modelldaten während der Laufzeit automatisiert zugreifen zu können. Hier bietet es sich an, Modellverwaltungsagenten zu entwickeln, mit denen auf die Modelle in Form von Dienstaufrufen zugegriffen werden kann.

In Abb. 6.7 sind die Aufgaben eines Modellverwaltungsagenten skizziert. Er unterstützt mit seinen Diensten die Erkundung der verwalteten Modelle, er organisiert die Ausführung von Modelltransformationen, er stellt Verwaltungsdienstprogramme zur Verfügung und kann bei aktiven Modellen (Funktionsbausteinsystem) auch die Modellausführung organisieren und verwalten [9]. Mit diesem Agenten ist es Anwendungen möglich, über eine standardisierte Schnittstelle der Agentenplattform mit Modellen zu interagieren.

6.3.7 Agenten zur Verwaltung von Zusatzfunktionen

Ein besonders interessantes Anwendungsfeld für das Agentenkonzept ist die Modellierung von Optimierungsfunktionen und überlagerten Verwaltungsfunktionen als eigenständigen, separaten Add-On Funktionen. Damit lassen sich kybernetische

Modelltransformations-Interface

- create object
- delete object
- create link
- delete link
- rename object
- set parameter

Modellerkundungs-Interface

- show childs
- show links
- get parameter
- ..

Dienstprogramme-Interface

- save
- load
- ...

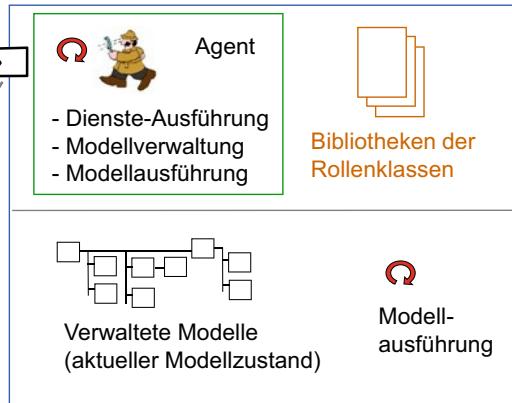


Abb. 6.7 Agent zur Modellverwaltung

Konzepte, wie z. B. das *autonomic computing* [9] auch im leittechnischen Umfeld realisieren. Hier zeigt sich der Vorteil der flexiblen, aber eindeutig definierten und explizit sichtbaren Kommunikation besonders deutlich.

In Abb. 6.8 wird diese Eigenschaft am Beispiel einer Adoptionsfunktion erläutert. Im Fall A ist die Adoptionsfunktion in den Regler integriert. Ihre Wirkung auf den Regelalgorithmus ist nicht mehr sichtbar. Dies ist in vielen Fällen erwünscht. In anderen Fällen, und in der zur Vorsicht neigenden Anlagenautomation treten diese häufig auf, will man jedoch die Kontrolle behalten und sehen können, wie der Adoptionsmechanismus auf den Regler einwirkt. Wird die Adoptionsfunktion, wie in Fall B dargestellt, als eigener Agent realisiert, bleibt die Einwirkung immer explizit sichtbar. Sie kann im Ernstfall jederzeit unterbrochen und ausgesetzt oder von einer anderen Einheit (z. B. dem Anlagenfahrer) übernommen werden. Auf diese Weise öffnet sich für die Betriebe eine elegante Möglichkeit, neue Zusatzfunktionen im laufenden Betrieb nachzurüsten und kontrolliert auszuprobieren.

6.3.8 Agenten zur Unterstützung von Assistenzsystemen

Assistenzsysteme unterstützen den Benutzer bei der Durchführung seiner Aufgaben. Ein wesentliches Element des Assistenzsystems ist die intensive Interaktion

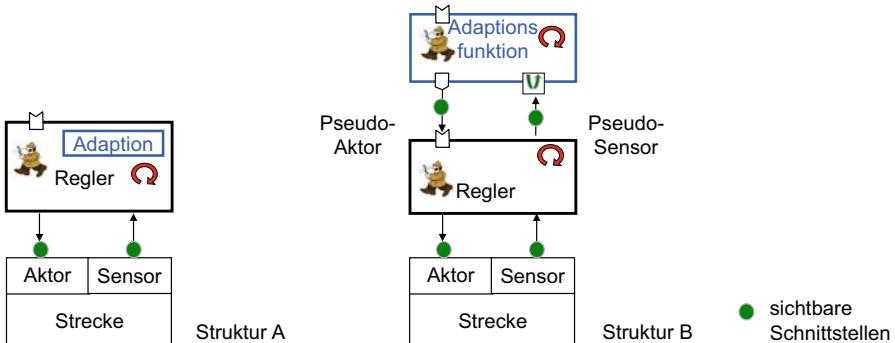


Abb. 6.8 Realisierung einer Adoptionsfunktion als eigenständiger Agent

Assistenzsysteme:	Agentensysteme:
<u>Benutzer-Orientierung</u> <ul style="list-style-type: none"> • Benutzer steht im Mittelpunkt • Benutzer behält immer die vollständige Kontrolle <u>Anwendungs-Orientierung</u> <ul style="list-style-type: none"> • Entwicklung ist auf die einzelne Anwendungsfunktion ausgerichtet • Kooperation der Assistenzfunktionen steht im Hintergrund 	<u>Autonomie-Orientierung</u> <ul style="list-style-type: none"> • Autonomie der Agenten steht im Mittelpunkt • Agenten übernehmen teilweise die Kontrolle von den Benutzern • Tendenz zur Vollautomatisierung <u>System-Orientierung</u> <ul style="list-style-type: none"> • Entwicklung ist auf eine integrierte Systemplattform ausgerichtet • Funktionalität ergibt sich aus der Kooperation der Agenten

Abb. 6.9 Gegenüberstellung von Agenten- und Assistenzsystemen

mit dem Benutzer. Ein Assistenzsystem hat weder den Voll-Automatik noch den reinen Handbetrieb als Ziel, sondern eine Halb-Automatik, bei der der Benutzer eng mit teilautomatisierten Führungsfunktionen zusammenarbeitet. Assistenzsysteme haben damit einen etwas anderen Fokus als Agenten. In Abb. 6.9 sind beide Konzepte einander gegenübergestellt.

Es zeigt sich allerdings, dass Assistenzsysteme wesentlich einfacher und eleganter aufgebaut werden können, wenn sie sich auf ein darunter liegendes Agentensystem abstützen. In diesem Fall nutzt das Assistenzsystem selbst die Rolle eines technischen Agenten, um mit dem unterlagerten Agentensystem zu interagieren. Das Agentenkonzept unterstützt das Assistenzsystem in zwei Bereichen:

1. Es ermöglicht einen einfachen, flexiblen und projektfreien Zugriff auf die implementierte Funktionalität eines unterlagerten Agentensystems über deren Dienstschnittstellen
2. Assistenzsysteme sind nicht statisch, sondern werden während ihres Betriebs verbessert und vervollständigt, dazu müssen sie ständig dynamisch erweitert und verändert werden. Zu diesem Zweck bietet sich eine Nutzung des Agentenkonzepts für die interne Realisierung der Assistenzfunktionalität an.

Ein weiterer interessanter Aspekt ist die Nutzung von Assistenzsystemen zur Verwaltung der Agentenplattform und Pflege des Agentensystems. So ist es z. B. leicht vorstellbar, dass die Implementierung von neuen Zusatzfunktionen oder der Austausch der alten Version einer Komponente durch eine neue Version durch ein Assistenzsystem unterstützt werden kann. Durch das Agentensystem sind solche Änderungen ja flexibel möglich.

6.4 Zusammenfassung

Die klassische Anlagenautomatisierung bietet mit Ihren Modellen und Sprachen einen bewährten Rahmen zur Implementierung von effizienter, fehlerarmer und pflegbarer Softwarefunktionalität. Als Reaktion auf die neu hinzukommenden Herausforderungen ist es nicht sinnvoll, diese bewährten Modelle und Sprachen zu verwerfen und völlig neu zu beginnen, vielmehr erscheint es zielführend, auf den bestehenden Modellen und Sprachen aufzubauen und diese weiterzuentwickeln. Dabei steht insbesondere die Integration von flexibel zu handhabenden Zusatzfunktionen im Blickpunkt des Interesses. Hier bietet die Agentenorientierung einen interessanten Lösungsansatz. Mit ihr lassen sich eine Reihe von aktuellen Innovationshemmnissen überwinden. Das hier vorgestellte, speziell auf die Anforderungen der Anlagenautomatisierung abgestimmte Konzept bietet eine Reihe von Vorteilen:

- *Flexibilität*
Neue Komponenten können jederzeit projektierungsfrei und kontrolliert hinzugefügt werden. (z. B. temporäre Maßnahmen)
- *Transparenz*
Die Kommunikation zwischen den Agenten erfolgt auf der Grundlage von expliziten Nachrichten und kann jederzeit mitprotokolliert, diagnostiziert oder gefiltert werden.
- *Robustheit*
Die spezifizierten Nachrichtendienste sind einfach und netzwerkfähig. Das Konzept setzt auf entkoppelte technologische Rückmeldungen und vermeidet komplizierte Protokolle.
- *Sicherheit*
Das Konzept sieht für jeden Dienstauftrag am Diensteingang eine Kontrolle der Berechtigung des Auftraggebers und eine Überprüfung der Korrektheit und Ausführbarkeit des Auftrags durch.
- *Skalierbare Virtualisierung*
Das Konzept erlaubt es, sowohl mit rein funktionalen Adressen als auch mit Netzwerkadressen zu arbeiten. Auftragsempfangsadressen sind außerhalb der Kapselung sicht- und dynamisch parametrierbar.
- *Anwendungsorientierung*
Die Agentenplattform definiert bewusst keine ausgeprägten Dienste und erzwingt auch keine Möglichkeit zur Diensterkundung. Es wird davon ausgegangen, dass die Partner sich und die zu nutzenden Dienste explizit kennen. Es ist

jedoch vorgesehen, dass von jedem Agenten der Typ und die von ihm unterstützten Rollentypen abgefragt werden können.

Das hier vorgestellte Konzept definiert keine eigene Implementierung einer Agentenplattform. Es stützt sich vielmehr auf das als allgemein verfügbar angesehene Funktionsbausteinsystem (nach IEC61131) und auf vorhandene leittechnischen Kommunikationsstrukturen (nach IEC62541).

Die Agentenorientierung steht in der Anlagenautomatisierung erst am Anfang. Alle technischen, praktischen und methodischen Kriterien sprechen jedoch für einen Einsatz des Agentenkonzepts. Ein gewisses Hemmnis ist die Begrifflichkeit. Sie wird gerne mit umfassenden, methodisch und softwaretechnisch zwar ausgereiften, für die Anlagenautomatisierung jedoch nicht geeigneten Agentenplattformen assoziiert. In der Anlagenautomatisierung ist der Handlungsspielraum oft bewusst eng bemessen. Automatisierungsfunktionen, die sich ihre Aufgaben selbst suchen oder eigene Lösungswege gehen, sind hier typischerweise nicht erwünscht. Hier muss die Automatisierungstechnik ihre Anforderungen deutlich formulieren und ein für sie geeignetes Literaturmodell entwickeln. Das hier vorgeschlagene Konzept bietet dafür eine Ausgangsbasis.

Normen

- [VDI/VDE2653-1] VDI/VDE 2653-1: *Agentensysteme in der Automatisierungs-technik*. Blatt 1: *Grundlagen*. VDI/VDE-Richtlinien. VDI, Düsseldorf 2009.
- [VDI/VDE2653-2] VDI/VDE 2653-2: *Agentensysteme in der Automatisierungs-technik*. Blatt 2: *Entwicklung*. VDI/VDE-Richtlinien. VDI, Düsseldorf 2012.
- [IEC62541] IEC 62541-1: *OPC Unified Architecture*. Publication IEC 62541 Ed 1.0, 2008.
- [IEC61131] IEC 61131-3: Programmable Controllers. Part 3: Programming lan-guages. Publication IEC 61131 Ed 3.2012.

Literatur

- [1] Hewitt, C.: Viewing control structures as patterns of passing message. Art. Intell. **8**(3), 323–364 (1977)
- [2] Maes, P. (Hrsg.): Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back. The MIT press, London (1991)
- [3] Wooldridge, M., Jennings, N. (Hrsg.): Intelligent Agents. Lecture Notes in Artificial Intelligence. Springer Verlag, Heidelberg (1995)
- [4] Epple, U.: Agentensysteme in der Leittechnik. atp Automatisierungstechnische Praxis **42**(8), 42–51 (2000)
- [5] Wagner, T., Göhner, P., De Urbano, A.P.: Softwareagenten – Einführung und Überblick über ei-ne alternative Art der Softwareentwicklung. Teil I: Agentenorientierte Softwareentwicklung. atp – Automatisierungstechnische Praxis **45**(10) (2003)

- [6] De Urbano, A.P., Wagner, T., Göhner, P.: Softwareagenten – Einführung und Überblick über eine alternative Art der Softwareentwicklung. Teil II: Agentensysteme in der Automatisierungstechnik: Modellierung eines Anwendungsbeispiels. *atp – Automatisierungstechnische Praxis* **45**(11) (2003)
- [7] Jammes, F., Smit, H.: Service-oriented paradigms in industrial automation. *IEEE Transactions on Industrial Informatics* **1**, 62–70 (2005)
- [8] Epple, U.: Increasing flexibility and functionality in industrial process control. International Multi-Conference on Systems, Signals & Devices: summary proceedings, SSD12. March 20–23, Chemnitz, Germany (2012)
- [9] Horn, P.: Autonomic computing: IBM's Perspective on the State of Information Technology (2001). <http://www.ibm.com/autonomic/pdfs/autonomiccomputing.pdf>
- [10] Mersch, H., Epple, U.: Requirements on distribution management for service-oriented automation systems. 16th IEEE International Conference on Emerging Technologies and Factory Automation: ETFA'2011. IEEE Toulouse, France, Piscataway, NJ (2011)

Teil III

Agenten im operativen Einsatz

Kapitel 7

Einsatz von Agentensystemen in der Intralogistik

Wilfried Kugler und Dirk Gehlich

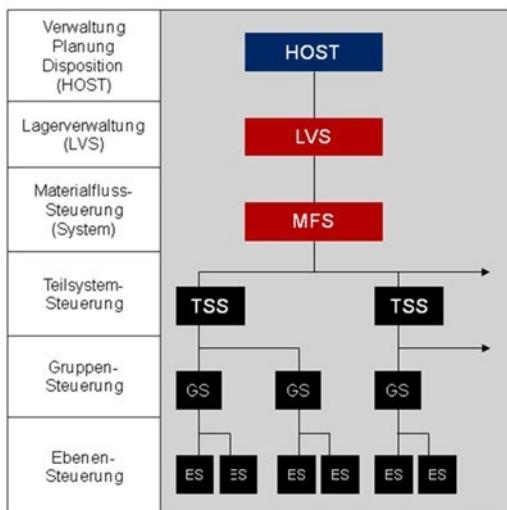
Zusammenfassung Der Beitrag verschafft zunächst einen Überblick über die Entwicklung der Automatisierungstechnik in der Intralogistik und geht auf die Schwachstellen heutiger Automatisierungskonzepte ein. Diesen Schwachstellen soll mit Agentensystemen begegnet werden. Zudem zeigt der Beitrag, wie künftige Materialflusssysteme aus autonomen, intelligenten und kooperierenden Einheiten (Agenten) aufgebaut werden können. Die Ideen und Konzepte des *Internet der Dinge* werden auf die Intralogistik angewendet. Mechatronische Module werden gebildet, und Transporteinheiten werden mit Intelligenz in Form von RFID-Chips ausgestattet. Auf dieser Basis wird ein großes Kommissioniersystem mit den komplexen Anforderungen aus der logistischen Praxis modelliert und auf einer Agentenplattform in einer Testanlage umgesetzt.

7.1 Einleitung

Die Intralogistik umfasst alle manuellen und automatisierten Systeme sowie die Steuerungen und Software, die zur Durchführung des innerbetrieblichen Materialflusses in Produktion und Distribution erforderlich sind. Automatisierte Intralogistik besteht aus mechatronischen Fördertechnikkomponenten wie Regalbediengeräte (RBG), Stetigförderer oder Shuttlesysteme sowie der zur Steuerung der Geschäftsprozesse erforderlichen Informationstechnik. Deren Kernkomponente ist heute ein leistungsfähiges Warehouse Management System (WMS). Die automatisierte Intralogistik kam in den 1970-er Jahren noch als Einzellösung bei Großunternehmen zur Anwendung. Heute ist sie skalierbar und wird nahezu in allen Branchen und Firmengrößen eingesetzt. Große hochautomatisierte Logistikzentren sind in der Lage, mehr als 250.000 Kundenaufträge am Tag zu kommissionieren und an die Kunden zu versenden [1].

Dr.-Ing. Wilfried Kugler (✉), Dipl.-Ing. (BA) Dirk Gehlich
viastore systems GmbH, Magirusstr. 13, 70469 Stuttgart, Deutschland
e-mail: w.kugler@viastore.com

Systemebenen nach VDI 3962 (1996): 6 Ebenen



Ausgeführt (2005): 4 Ebenen

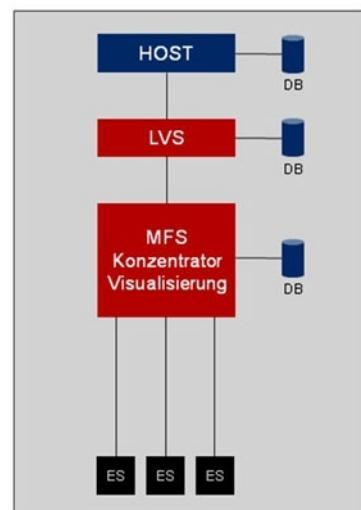


Abb. 1a

Abb. 1b

Abb. 7.1 Entwicklung der Automatisierungsarchitektur in der Intralogistik

7.2 Ausgangssituation

Automatisierte Intralogistikanlagen werden heute als verteilte Automatisierungssysteme mit fester Funktionsaufteilung der Steuerungslogik und mit zentraler Planungs- und Entscheidungsintelligenz gebaut. Eine Aufteilung der Funktionalitäten erfolgte 1995 in der VDI 3962 (s. Abb. 7.1a). Aufgrund der rasanten Weiterentwicklung der zur Verfügung stehenden Rechenleistung hat sich in den Folgejahren die Zahl der Rechner-/Steuerungsebenen reduziert. (s. Abb. 7.1b). Außerdem ermöglicht der konsequente Einsatz von Bussystemen den Wechsel von streng hierarchischen hin zu verteilten Systemen mit horizontaler Kommunikation.

Die Schwachstellen der heute ausgeführten Automatisierungskonzepte in komplexen Anlagen sind:

- Die Abläufe sind in ihrer Vielfalt nicht vollständig planbar. Dies führt dazu, dass erst zur Laufzeit des Systems Zustände eintreten, die ggf. zu einer reduzierten Leistung oder im schlechtesten Fall zum Stillstand führen.
- Große komplexe Programme mit vielen Abhängigkeiten
- Hoher Aufwand für Test, Inbetriebnahme und Hochlauf
- Rechner und Netzwerke werden bei Hochleistungssystemen häufig im Grenzbereich betrieben. Dadurch kommt es zu Reaktionszeiten und damit zu Performance-Einbußen im Materialfluss.
- Die zentrale Komponente Materialflusssteuerung (MFS) ist verfügbarkeits- und performancekritisch.

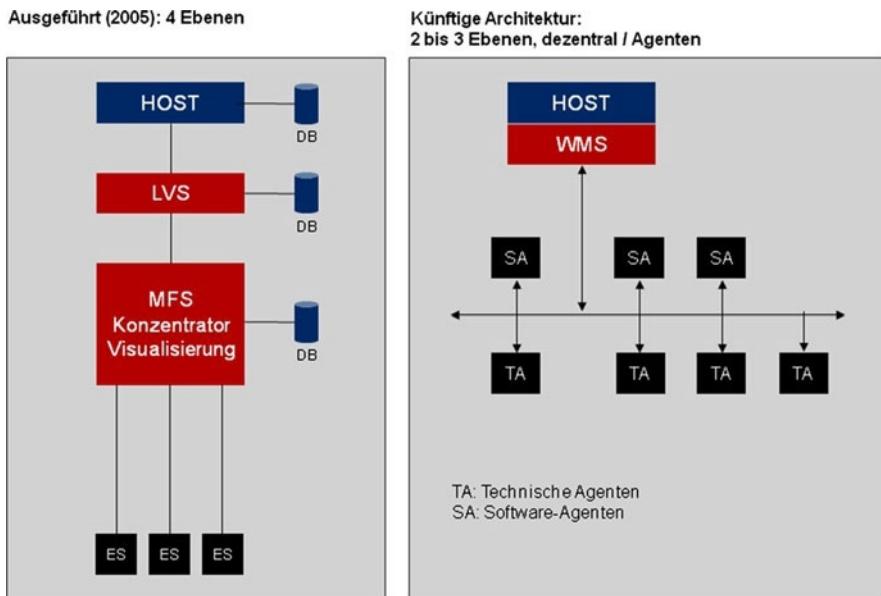


Abb. 7.2 Künftige Architektur von Intralogistiksystemen

- Erweiterungen und Veränderungen der Systeme sind oft mit hohem Aufwand und mit Betriebsrisiken verbunden.

Diese Schwachstellen führten dazu, dass im Rahmen des BMBF-Projekte *Internet der Dinge, Wandelbare Echtzeit-Logistiksysteme auf Basis Intelligenter Agenten für den produktionsnahen Bereich*, in den Jahren 2006 bis 2010 unter der Teilnahme verschiedener Universitäten und Industrieunternehmen, intensiv der Einsatz von Agentensystemen für den Bau von Intralogistiksystemen untersucht wurde [2].

7.3 Konzept

Das im *Internet der Dinge in der Intralogistik* verfolgte Konzept löst die Trennung zwischen Steuerungsebene und der zentralen MFS-Instanz auf. Das System ist ein hierarchieloses Materialflusssystem aus autonomen, intelligenten und kooperierenden Einheiten (Agenten). Es wird zwischen technischen und Software-Agenten unterschieden (s. Abb. 7.2).

Die **Technischen-Agenten (TA)** – im *Internet der Dinge* auch **Modul** genannt – sind sinnvoll gegliederte mechatronische Einheiten, die z. B. nach den Kriterien gemäß VDI/VDE 2653 (Agentensysteme in der Automatisierungstechnik) gebildet werden. Diese bestehen aus Mechanik, Elektrotechnik und der implantierten Agentenlogik in Form von Software. Beispiele für TA können eine Fördertechnikweiche, ein Regalbediengerät oder ein Shuttle-Fahrzeug sein.



Abb. 7.3 Intelligenter Behälter mit RFID-Schreib-/Leseeinheit in der Fördertechnik

Die **Software-Agenten (SA)** übernehmen rein logische oder dispositive Aufgaben und benötigen lediglich einen Rechnerknoten, ggf. mit Agentenplattform.

In einem so automatisierten Intralogistiksystem werden **Transporteinheiten (TE)** bewegt. TE sind Ladungsträger wie Behälter oder Paletten. Diese sind in heute ausgeführten Systemen mit einem Barcode versehen, der als Schlüssel für die übergeordnete Informationsverarbeitung eine systemweit eindeutige Behälternummer enthält. Im *Internet der Dinge* wird der Barcode durch einen RFID-Chip ersetzt. Der Vorteil liegt darin, dass auf einem RFID-Tag auch zusätzliche auftrags- und steuerungsrelevante Daten dynamisch gespeichert und zur Laufzeit geändert werden können. Dies kann dazu dienen, Daten über das Produkt, aber auch Weginformationen auf dem Fördergut zu speichern und damit für die Verarbeitung in einer lokalen Steuerung sofort verfügbar zu haben. Diese Art der Datenverwaltung wird als *Data-on-Tag* bezeichnet (s. Abb. 7.3).

Basierend auf diesen Konzepten wird im Folgenden ein komplexes Kommissionsiersystem modelliert.

7.4 Das agentengesteuerte Kommissioniersystem

Um möglichst viele Ablaufszenarien einer Materialflussteuerung abzubilden und typische Abläufe in einem Kommissioniersystem darzustellen, wird im Folgenden eine Literaturanlage der viastore systems GmbH vorgestellt und modelliert.

Das System erstreckt sich über fünf Ebenen und dient als Puffer zwischen Produktion und Versand. Es ist in spezielle Lagerbereiche unterteilt, die durch ein Materialflusssystem verbunden sind (s. Abb. 7.4).

Die TE werden auf der Fördertechnik, bestehend aus Rollen-, Band- und Kettenförderern und den Fördertechnikplätzen (*ConveyLoc*), durch das Lager transportiert. Zu Beginn jedes Versandauftrags, der mehrere Kartons umfasst kann, wird der Karton beim Kartonaufrichter mit der Kommissioniertransporteinheit (KTE) gemeinsam eingescannt und somit logisch für die Transportdauer mit ihr verknüpft. Die KTE wird zunächst, je nach Auftragspositionen, zum A-Lager und/oder zum B/C-Lager befördert.

Im A-Lager befinden sich Waren, die häufig nachgefragt werden. Dieses Lager verfügt über 29 Kommissionierbahnhöfe, die einzeln von den KTE angefahren werden können. An den Bahnhöfen werden die Artikel vom Bedienpersonal in der angezeigten Menge aus dem Lagerbehälter entnommen und in die KTE gelegt. Die Bahnhöfe werden direkt von den RBG mit Nachschub aus dem Lager versorgt.

Im B-Lager befinden sich seltener nachgefragte Waren (s. Abb. 7.5). In diesem Bereich wird die KTE zum Kommissionierbahnhof befördert, während zeitgleich die benötigte Ware bereitgestellt wird. Die Ware wird in TE aus dem Lager zum Kommissionierbahnhof befördert. Um sicherzustellen, dass diese am Kommissionierbahnhof zeitgleich ankommen, werden die TE auftragsbezogen aus einem automatischen Kleinteilelager (AKL) entnommen und zur Zwischenspeicherung zu einem Turmspeicher befördert. Bei der Ankunft der KTE am Kommissionierbahnhof wird dem Bediener die Ware auftragsbezogen aus dem Turmspeicher bereitgestellt.

Sind noch Auftragspositionen im C-Lager vorhanden, wird die KTE anschließend in das C-Lager transportiert. Die C-Kommissionierung betrifft selten benötigte Waren, die meist auf Paletten gelagert werden.

Anschließend werden Aufträge, die mehrere Kartons umfassen, vor dem Warenausgang zusammengeführt. Diese Aufgabe übernimmt eine Sortieranlage (s. Abb. 7.6), bestehend aus zwei RBG und einem Kreislauf (Loop). Die KTE eines Auftrags verlassen den Kreislauf in Richtung der Universalpackplätze. Die KTE,

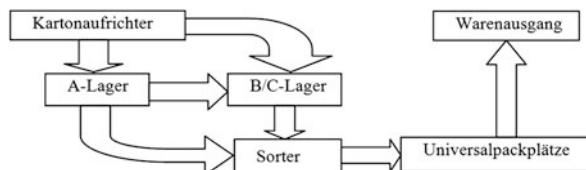


Abb. 7.4 Lagerbereiche der Literaturanlage (viastore systems GmbH)

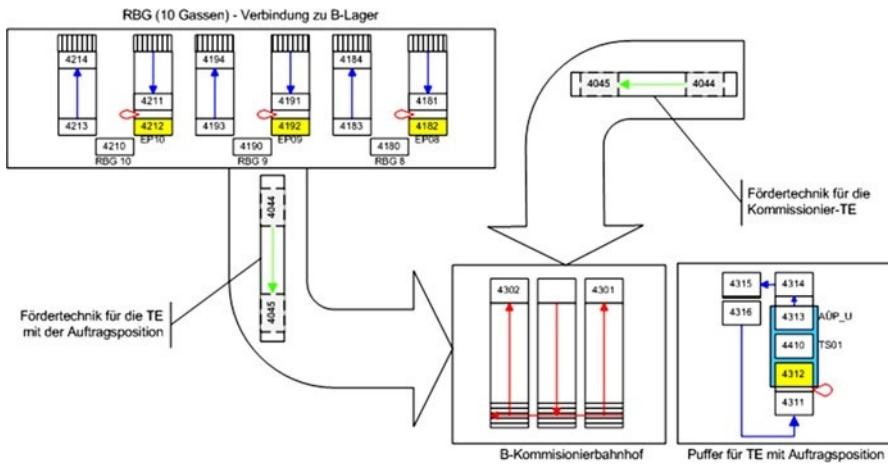


Abb. 7.5 B-Kommissionierung

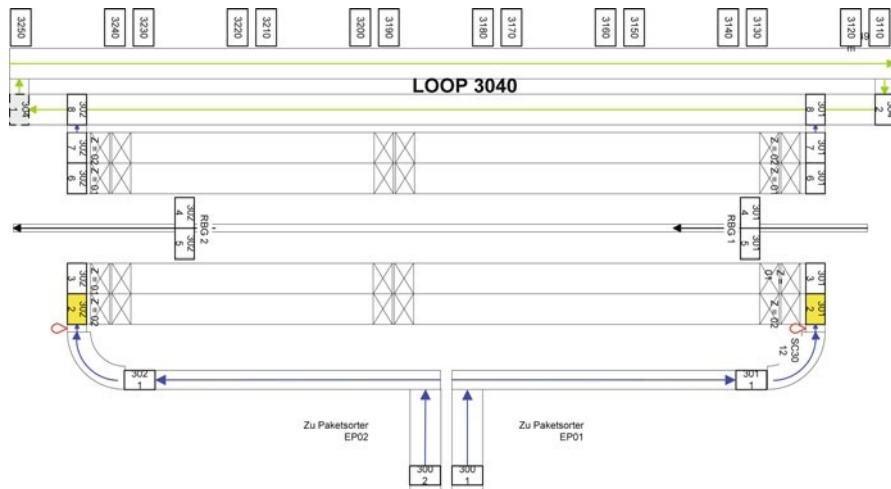


Abb. 7.6 Sortieranlage

die nicht zu dem aktuellen Auftrag gehören, verbleiben so lange im Kreislauf, bis der aktuelle Auftrag abgeschlossen ist und die Bearbeitung des entsprechenden Auftrags begonnen werden kann.

7.4.1 Aufbau des RFID-Systems

Um die steuerungsrelevanten Informationen des MFS verarbeiten zu können, müssen an allen Entscheidungsplätzen der Fördertechnik sowie an den Einlagerplätzen

Tab. 7.1 Gewählte Speicheraufteilung des RFID-Tags

		Speicherbereich		
Feste Daten	Steuerungsdaten	Nutzdaten		
UID	Ziel	Route	Artikelnummer	Gewicht
				Menge

und ggf. auf dem Lastaufnahmemittel (LAM) des Regalbediengeräts RFID-Lese-/Schreibstationen angebracht sein. An diesen Stellen kann das MFS den nächsten anzufahrenden Platz bestimmen und die TE in die ermittelte Richtung weiterleiten. Dazu benötigt es die in dem RFID-Tag gespeicherten Routeninformation und das der TE zugeordnete Ziel.

Die zu schreibende Datenmenge ist abhängig vom Ort des Beschreibens. Ist ein steuerungsrelevanter Datenaustausch an jeder Entscheidungsstelle notwendig (z. B. durch Auswertung und ggf. Korrektur von Routeninformationen), ist die maximal mögliche Datenmenge abhängig von der Fördergeschwindigkeit und der Feldbreite. Diese ist bei Hochleistungsanlagen stark eingeschränkt und beläuft sich auf wenige Kilobyte. An Verwaltungsplätzen wie Kommissionierplätzen, Aufsetzpunkten etc. können große Datenmengen geschrieben werden, da sich die Behälter an diesen Plätzen im Ruhezustand befinden. Informationen für das WMS und für das ERP-System können somit auf dem RFID-Tag gespeichert und verwaltet werden. Das ermöglicht auch eine standortübergreifende Verwaltung von Transportgütern. Der gesamte zur Verfügung stehende Speicherbereich eines RFID-Tags setzt sich somit wie folgt zusammen (Tab. 7.1):

Die Behälter werden bei der Literaturanlage auf Rollen transportiert und fahren über eine circa 400 mm breite Fördertechnikstrecke. Die Fördergeschwindigkeit beträgt im Normalfall 1 m/s. Verschiedene Materialien haben unterschiedliche Einflüsse auf das RFID-Feld und sind somit entscheidend für die Auswahl einer geeigneten Frequenz. In der Regel bestehen Fördertechnikmodule und Rollen aus einer Stahlkonstruktion. Wegen der Anforderungen für hohe Übertragungsgeschwindigkeiten sowie Verträglichkeit in industrieller Umgebung, empfiehlt sich der Einsatz der HF-Technologie mit einer Frequenz von 13,56 MHz. Der heutige Stand der Technik bietet Geräte, die auf Metall abgestimmt sind und dadurch dämpfungs-unempfindlicher sind.

7.4.2 Konzept einer agentenbasierten Steuerung

Ein dezentrales Steuerungskonzept für den Materialflussbereich setzt eine Modularisierung der Elemente des Lagers voraus. Bei einem Agentensystem werden diese Module durch Modulagenten repräsentiert (hier **ConvLoc-Agent** genannt).

Auch die TE und deren Aufgaben sind im System als **TE-Agenten** aktiv. Ein TE-Agent existiert für die gesamte Transportdauer der TE, vom Aufsetzen auf die

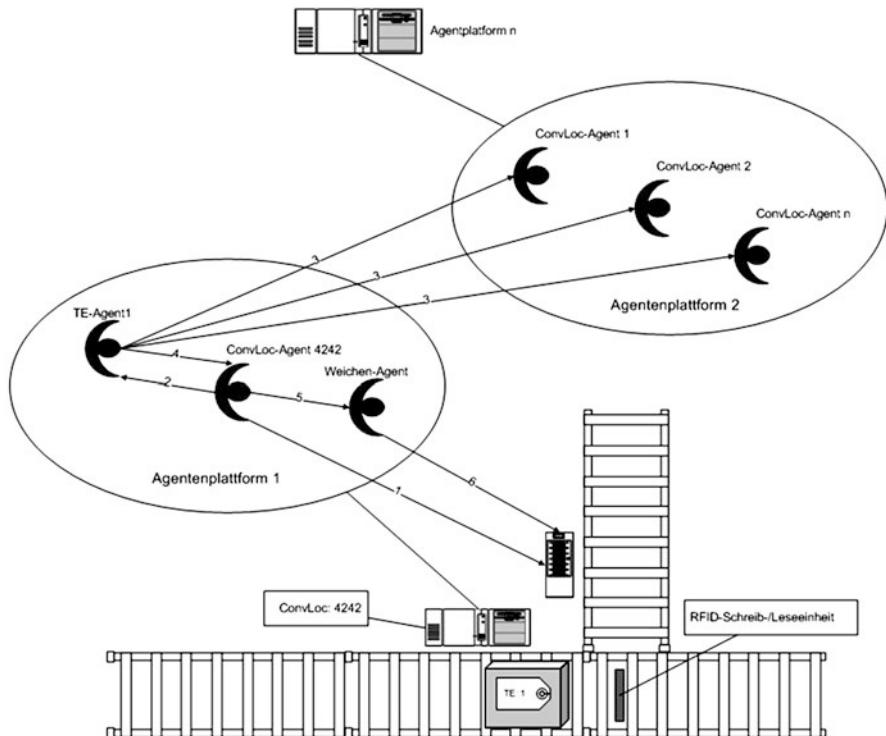


Abb. 7.7 Grundkonzept des Agentensystems

Fördertechnik bis zur Einlagerung. Der Agent selbst kann sich dabei an einem beliebigen Ort innerhalb des Netzwerks befinden.

Abbildung 7.7 verdeutlicht die Verarbeitung einer TE an einem Identifikationspunkt im bestehenden Grundkonzept.

Die exakte globale Adresse des Agenten befindet sich in einem RFID-Chip der TE und wird vom *ConvLoc*-Agenten 4242 ausgelesen (Schritt 1). Ein *ConvLoc*-Agent repräsentiert einen bestimmten Platz (hier einen Identifikationspunkt) im Lager. Dabei kann ein *ConvLoc*-Agent mit einem weiteren speziellen Agenten an diesem Platz verbunden sein. In diesem Fall ist das ein Weichenagent, der die Aufgabe hat, die Weiche anzusteuern und direkt mit der Fördertechnik zu kommunizieren. Die Verbindung zweier Fördertechnikplätze stellt eine Strecke dar. Diese wird von einem der beiden *ConvLoc*-Agenten verwaltet und wird nicht gesondert über einen Streckenagenten repräsentiert. Zusätzlich sind im System weitere Agenten vorhanden, die spezielle Aufgaben übernehmen. Hierzu zählen beispielsweise *Loop*-Agenten, *Einlager*-Agenten und *Verschiebewagen*-Agenten. Diese lassen sich einem *ConvLoc*-Agenten als eine übergeordnete Instanz zuordnen.

Der *ConvLoc*-Agent nimmt nun mit Hilfe der ausgelesenen globalen Adresse Kontakt zu dem zur TE gehörenden Agenten auf und teilt ihm die aktuelle Platz-

nummer mit (Schritt 2). Der zur TE gehörende Agent ermittelt im ersten Schritt alle möglichen Routen zum Ziel der TE, das er bei der Auftragserstellung gespeichert hat. Daraufhin kontaktiert er alle beteiligten *ConvLoc*-Agenten und holt Informationen über die zugehörigen Streckenabschnitte ein (Aktivität, Auslastung etc.) und passt dementsprechend seine Routingtabelle an (Schritt 3). Nachdem die Dijkstra-Berechnung¹ durchgeführt worden ist, teilt er dem *ConvLoc*-Agenten, auf dessen Platz die TE aktuell steht, den nächsten anzufahrenden Platz mit (Schritt 4). Der *ConvLoc*-Agent wiederum gibt diese Information an den Weichenagenten weiter, der den Auftrag zur Weichensetzung erteilt (Schritt 5 und 6).

Neben der Identifizierung von TE entstehen im Lager weitere Steuerungsaufgaben wie die Auftragsreihenfolgebildung, die Loopsteuerung und die Gruppenzielverfolgung. Konzeptionelle Lösungen mit dem Agentenansatz werden im Folgenden für zwei dieser Aufgaben genauer betrachtet.

7.4.3 Platz-zu-Platz-Steuerung

Die Platz-zu-Platz-Steuerung setzt die Grundaufgabe der Materialflussteuerung um. Hierbei sollen die *TE*-Agenten an Fördertechnikkreuzungen entscheiden, in welche Richtung sie weiterfahren, um ihr Ziel zu erreichen. Bisher wurden diese Entscheidungen offline getroffen und statisch in Routing-Tabellen abgelegt. Mit Hilfe des Wegfindungsalgorithmus nach Dijkstra sollen diese Entscheidungen der Anlagendynamik angepasst werden. Entscheidend ist hier der Informationsaustausch zwischen dem *TE*-Agenten und den *ConvLoc*-Agenten, um aktuelle Lasteigenschaften beteiligter Strecken zu erhalten. Abbildung 7.8 stellt einen Ablauf der Kommunikation zwischen den Agenten in Form eines Sequenzdiagramms dar.

In diesem Szenario fährt die TE, repräsentiert in der Steuerung durch *TE-Agent1*, von Platz 4001 nach 4002. Der Platz 4001 erhält von der Maschinensteuerung eine Ankunftsmeldung inklusive der eindeutigen ID, die durch ein an dem Platz angeschlossenes RFID-System ausgelesen wird. Diese Information wird dem Platz-Agenten mitgeteilt. Anhand dieser Informationen kann der Platz 4001 dem *TE*-Agenten seine aktuelle Position mitteilen (*setActConvLoc()*).

Nachdem der *TE-Agent1* die aktuelle Position der TE mitgeteilt bekommen hat, berechnet er den optimalen Weg zum Ziel von dieser Position aus. Da hierfür die Lastinformationen aller beteiligten Strecken erforderlich sind, sollten im Vorfeld Nachrichten an die *ConvLocs* verschickt werden (*getTrackState()*).

Bis dato steht nicht fest, welche Strecken beteiligt sein können. Deshalb müsste eine Nachricht an alle *ConvLoc*-Agenten geschickt werden. Bei einer Lagergröße von 180 Plätzen sind das 360 Nachrichten für eine einzelne TE (180 Anfragen +

¹ Der Dijkstra-Algorithmus ist ein informierter Suchalgorithmus [3]. Er wird zur Routenplanung eingesetzt, um den optimalen Weg zwischen zwei Orten unter Berücksichtigung ihrer Streckeneigenschaften zu bestimmen.

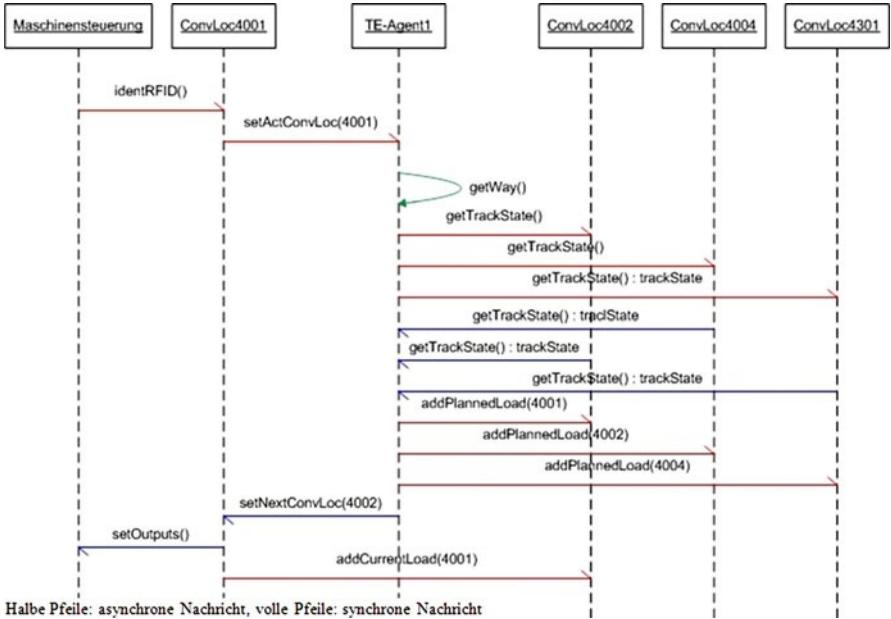


Abb. 7.8 Sequenzdiagramm Platz-zu-Platz-Steuerung

180 Antworten). Bei einer Dichte von 100 parallel arbeitenden *TE-Agenten* steigt die Zahl der verschickten Nachrichten auf 36.000 im gleichen Zeitspektrum. Diese Masse an Informationen ist schwer zu verarbeiten. Zum einen entsteht ein hoher Netzwerkverkehr, zum anderen muss jeder *ConvLoc-Agent* ständig eine Vielzahl an Nachrichten verarbeiten und sie zeitgemäß beantworten können. Unter Berücksichtigung der Art der Nachrichtenverarbeitung² und ihrer Quasi-Parallelität kann hier ein Engpass entstehen. Eine Optimierung dieser Strategie besteht darin, zunächst den Weg auf Basis rein statischer Informationen zu berechnen. Zu diesem Zweck kennt jeder *ConvLoc-Agent* die gesamte Topologie und die zu den Strecken gehörigen statischen Eigenschaften. Diese beinhalten die Fördergeschwindigkeit, die Streckenlänge und die damit verbundene maximale Streckenkapazität. Anschließend werden nur relevante *ConvLocs* auf die dynamischen Eigenschaften der Strecken befragt.

In dem obigen Szenario wird dieser Sachverhalt durch Nachrichten an die *ConvLocs* 4002, 4004 und 4301 deutlich. Nur diese *ConvLocs* liegen auf dem aus statischer Sicht optimalen Weg. Sind alle Streckenabschnitte aktiv und liegt die Auslastung unterhalb der vorhandenen Streckenkapazität, kann dieser berechnete Weg befahren werden. Andernfalls werden die Kosten für die nicht befahrbaren Abschnitte hoch gesetzt und der Weg erneut berechnet. Aufgrund der Tatsache, dass

² Dieses Szenario wurde mit JADE-Framework umgesetzt, das zur Verarbeitung von Nachrichten innerhalb eines Agenten das Round-Robin-Verfahren verwendet [4].

Tab. 7.2 Auftragsnummern

Auftragsnummer	Beschreibung
00	Eilauftrag (unmittelbare Auslagerung)
01	Auslagerauftrag mit einem Auslagerbefehl
10–99	Auslagerauftrag mit mehreren Auslagerbefehlen

erneute Routenberechnungen nur dann durchgeführt werden müssen, wenn Streckenabschnitte voll oder defekt sind, verringert sich die Anzahl der zu sendenden Nachrichten. Dabei werden nur die Nachrichten benötigt, die den *TE*-Agenten über den Zustand der noch zu befahrenden Abschnitte vom aktuellen Platz zum Ziel informieren, im genannten Szenario also 3 statt 180.

Damit die *ConvLocs* ihre Streckeninformationen aktualisieren können, werden die geplanten Streckentransporte des *TE*-Agenten als Auslastungsinformation an die *ConvLoc*-Agenten gesendet (*addPlannedLoad(ConvLoc)*), wobei der Parameter *ConvLoc* dem Platz entspricht, von dem eine TE zu dem geplanten Platz fährt.

Nun kann dem *ConvLoc* 4001 der Fahrbefehl zum Platz 4002 mitgeteilt werden (*setNextConvLoc()*). Dabei ist der *ConvLoc* selbst dafür verantwortlich, wie die Einheit von Platz zu Platz fährt. Es ist also nicht entscheidend, ob der *TE*-Agent einem einfachen Förderer oder einem speziellen Platz wie einem Verschiebewagen den Auftrag erteilt. Die interne Verarbeitung ist dem Platz selbst überlassen.

Da die TE nun die Strecke von 4001 nach 4002 aktiv nutzen möchte, muss dies dem *ConvLoc*-Agenten 4002 mitgeteilt werden, damit dieser die geplante Auslastung mit der aktuellen Auslastung synchronisieren kann. Des weiteren kann zu jeder Zeit festgestellt werden, auf welchem Streckenabschnitt sich eine bestimmte TE befindet, da die momentane und geplante Auslastung der Strecken inklusive der TE-Identifikationsnummer in jedem *ConvLoc*-Agenten hinterlegt ist.

7.4.4 Steuerung der Auftragsreihenfolge

Gehören einem Auftrag mehrere Fahrbefehle an, müssen sie als Gruppenauftrag identifiziert und entsprechend auf der Fördertechnik zusammengeführt werden. Zu diesem Zweck existiert eine Auftragsnummer (ANR). In dem Szenario für die Literaturanlage wird eine zweistellige Auftragsnummer verwendet. Sie beginnt im Normalfall bei 10 und läuft bis 99 (siehe Tab. 7.2). Danach wird die laufende Nummer zurückgesetzt und beginnt von vorn. Aufträge mit der Nummer 01 sind Kleinaufträge und umfassen nur einen Auslagerbefehl. Die Auftragsnummer 00 legt einen Eilauftrag fest, der höchste Priorität genießt.

Bevor nun die Aufträge an das RBG weitergeleitet werden, müssen die *TE*-Agenten untereinander aushandeln, welcher Auftrag als nächster gefahren werden darf. Der Ablauf dieses Kommunikationsakts ist im Sequenzdiagramm in Abb. 7.9 dargestellt.

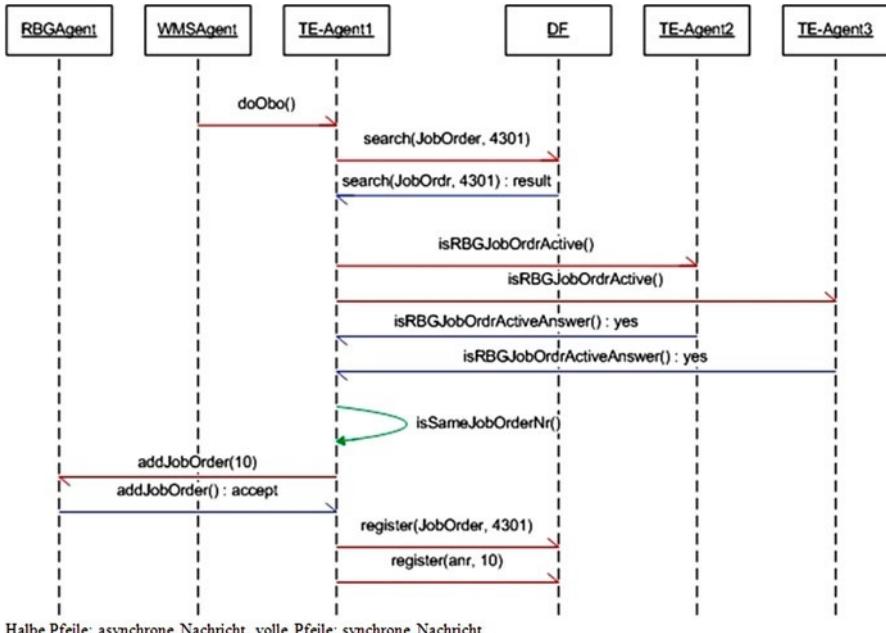


Abb. 7.9 Sequenzdiagramm Auftragsreihenfolge

Der WMS-Agent bekommt vom Lagerverwaltungssystem einen Auslagerauftrag in Form eines Telegramms inklusive einer Auftrags- und ggf. Sortiernummer. Aufgrund dieses Telegramms erteilt der WMS-Agent dem entsprechenden TE-Agenten den Auftrag auszulagern und zum Ziel zu fahren. Der TE-Agent erfragt vom zuständigen *Directory Facilitator* (DF) alle TE-Agenten, die durch den Service mit dem Servicetyp *JobOrder* und dem Servicenamen *ConvLocXY* registriert sind. *ConvLocXY* entspricht dabei einem Zielort im System. Dieser ist meistens ein Kommissionierbahnhof. Das Ergebnis wird in Form von *Agent IDentifications* (AID) zurückgegeben. Diese sind die eindeutigen Identifikationsnamen der jeweiligen TE-Agenten. Nun muss der TE-Agent alle im Ergebnis enthaltenen TE-Agenten über den Status des Auftrags ausfragen. Ein Auftrag gilt so lange als aktiv, bis alle TE dieses Auftrags den *Point-of-no-return* erreicht haben. Dieser Punkt entspricht einer Entscheidungsstelle in der Anlage, nach der sich die TE auf dem Weg zum Ziel nicht mehr über andere Routen überholen und somit die Auftragsreihenfolge unterbrechen können. Ist der Auftrag aktiv, darf die TE ausgelagert werden.

Im Demo-Szenario bekommt der TE-Agent1 als Ergebnis die AID von TE-Agent2 und TE-Agent3 zurückgeliefert. Diese werden nun über den Auftragsstatus befragt (`isRBGJobOrderActive()`), und die Antworten werden ausgewertet. Dem RBG-Agenten wird nun der Auftrag über die Auslagerung mit der entsprechenden Auftragsnummer angeboten. Er fügt den Auftrag seiner Auftragsliste hinzu und sendet eine positive Antwort zurück. Damit andere TE-Agenten über die anstehen-

de Auslagerung Bescheid wissen, registriert der *TE-Agent1* einen neuen Service mit dem Servicetyp *JobOrder*, dem Zielplatz 4301 und der Auftragsnummer (z. B. ANR=10).

In diesem Szenario ist die Dezentralität des Materialflusssystems besonders gut zu erkennen. Während in einer konventionellen Steuerung eine zentrale Entscheidungsstelle nötig ist, handeln hier die Agenten untereinander aus, welche TE ausgelagert werden darf. Das Hauptaugenmerk liegt dabei auf der Informationsbeschaffung für die Entscheidung.

7.4.5 Integration der Steuerung

Für die Umsetzung dieses dezentralen Konzepts ist eine Verteilung der Steuerungshardware notwendig. Für Module werden netzwerkfähige Embedded-Controller mit digitalen I/O eingesetzt. Wegen der hohen Echtzeitanforderungen wird eine Zwei-Schicht-Architektur eingesetzt. Dabei spielt eine geeignete Laufzeitumgebung für die verwendete Software eine entscheidende Rolle. Immer häufiger findet man im Bereich verteilter Anwendungen und Steuerungen den Einsatz von Linux-Betriebssystemen mit einer Echtzeiterweiterung.

Abbildung 7.10 zeigt eine beispielhafte Integration des entwickelten Schichtenmodells auf der Grundlage von RTAI-Linux. Die Steuerungseinheit ist über Ethernet im Netzwerk mit anderen Modulen verbunden, um die Agenten-Kommunikation zu ermöglichen. Der Informationsaustausch zwischen den Agenten findet über den von der Agentenplattform bereitgestellten Nachrichtentransportdienst statt. Die Kommunikation mit benachbarten Systemen erfolgt über eine TCP/IP-Schnittstelle. Die Laufzeitumgebung für die Agenten ist im *User-Space* als niedrig priorisierte Linux-Task zu finden. Innerhalb der Agentenplattform fungieren Software-Agenten als feste Bestandteile jeder dezentralen Steuerungseinheit. Dabei ist es zunächst unerheblich, ob auf einem Steuerungsrechner nur ein Modul-Agent zur Ansteuerung des entsprechenden Fördermoduls läuft oder ob die Hardware für die Installation mehrerer Modul-Agenten und somit für die Verwaltung mehrerer Fördermodule genutzt wird.

Eine Verbindung zwischen der Agentensteuerung und der Maschinensteuerung wird über eine Interprozesskommunikation hergestellt. Wichtig ist dabei nicht, wie die Daten, sondern welche Daten über die Schnittstelle ausgetauscht werden. Die Maschinensteuerung nutzt die Vorteile der RTAI-Erweiterung als Echtzeit-Task und erfüllt somit die Echtzeitanforderungen. Der Steuerungsablauf sollte im Idealfall für jede Entität identisch sein, jedoch stößt man hierbei auf folgende Herausforderung: Reale Anlagen bestehen meist aus einer Vielzahl unterschiedlicher Förderelemente von verschiedenen Herstellern mit verschiedenen mechanischen Komponenten und Antriebstechniken. Jedes Förderelement wird auf unterschiedliche Art und Weise angesteuert. Um dem entgegenzuwirken, muss es eine Möglichkeit geben, die Ansteuerung der Elemente dem eigentlichen Steuerungsablauf hinzuzufügen. Dies kann über Bibliotheken oder Hardwaretreiber bzw. I/O-Treiber erfolgen, die bereits

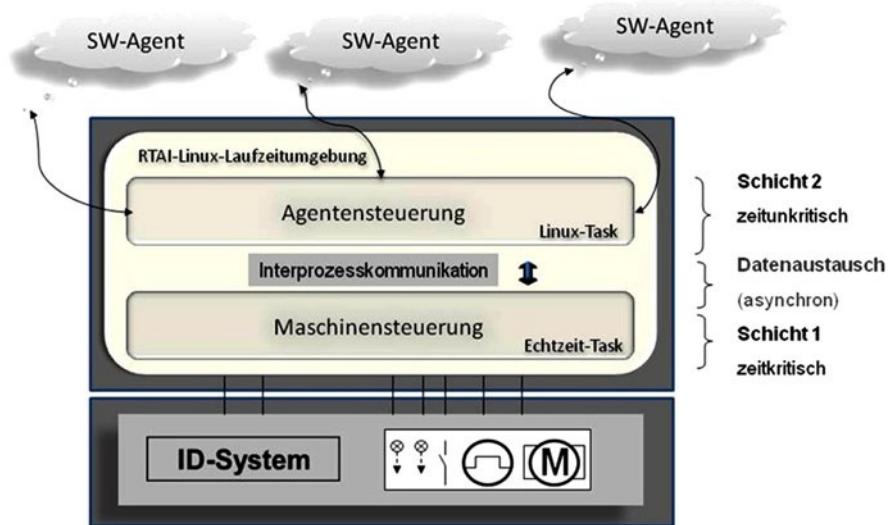


Abb. 7.10 Integrationsbeispiel für einen technischen Agenten

vom Hersteller bereitgestellt werden oder frei programmierbar sind. Zusätzlich ist für die Weiterleitung der TE eine Schnittstelle zum RFID-System notwendig, um die TE zu identifizieren. Die RFID-Einheit und die I/O-Gruppe können sich sowohl innerhalb als auch außerhalb des Moduls befinden.

7.5 Zusammenfassung

Moderne Kommissionierlager zeichnen sich durch den stetig steigenden Grad der Automatisierung aus. Auch bei kleinen bis mittleren Unternehmen kommen immer häufiger vollautomatisierte Kommissioniersysteme zum Einsatz. Gleichzeitig werden diese Systeme komplexer und müssen flexibler gegenüber aktuellen Marktanforderungen werden.

Abhilfe verspricht ein neues Steuerungsparadigma, das im Verbundprojekt des BMBF *Internet der Dinge in der Intralogistik* erarbeitet wurde. Dieses bringt die Vorteile der RFID-Technologie mit einem dezentralen Steuerungsansatz zusammen. Für die Realisierung der Materialflussteuerung wird in diesem Konzept der Einsatz von Softwareagenten. Weitere Ausführungen und Arbeiten der Projektpartner sind in [2] zu finden.

Zur Überprüfung der Anwendbarkeit dieses Steuerungsparadigmas für die Domäne Kommissionierung besteht eine Literaturanlage, die eine Reihe typischer Materialflüsse in einem automatischen Kommissioniersystem integriert. Die anstehen-

den Steuerungsaufgaben wurden in einem Betriebsszenario abgebildet und sind sowohl konzeptionell als auch teilweise in einem Versuchsaufbau umgesetzt.

Eine Herausforderung beim Entwurf der agentenbasierten Steuerung bestand in der Modularisierung der technischen Anlage sowie der Identifizierung einzelner Steuerungsagenten. Dabei mussten die lokal benötigten Informationen und Funktionen erkannt und innerhalb eines Agenten gekapselt werden. Eine weitere Herausforderung bezieht sich auf die Abbildung der Steuerungslogik in Form von Interagentenkommunikation.

Die Umsetzung der Steuerung nach dem Prinzip des *Internet der Dinge* ermöglicht folgende Aussagen über den Reifegrad und die Einsetzbarkeit der verwendeten Technologien.

- RFID-Technik in Kommissionier- und Förderanlagen weist einige Verbesserungspotenziale auf, insbesondere in Bezug auf die Lese- und Schreibgeschwindigkeit. Die Zuverlässigkeit des Lesens hängt unmittelbar von der eingesetzten Technik (Frequenz, Antennentyp, Transpondertyp usw.) und der Installation (Antennenausrichtung, Einstellung der Feldstärke usw.) ab.
- Die Kosten für die vielen dezentralen Steuerungsrechner sind zurzeit noch relativ hoch. Ein sinnvoller Ansatz dabei ist es, die Steuerung mehrerer technischer Module auf einem Rechnerknoten zu vereinen.
- Die Erfüllung der Echtzeitanforderung gilt weiterhin als eine Herausforderung. Abhilfe schafft die Kapselung zeitkritischer Funktionen eines Moduls in einer echtzeitfähigen Maschinensteuerung. In einer Steuerung auf PC-Basis ist dies allerdings nur mit speziellen Werkzeugen (z. B. Echtzeit-Betriebssysteme oder Soft-SPS) zu bewältigen.
- Der Reifegrad der Agentenplattformen in Bezug auf die Implementierung von Agentensystemen mit einer großen Anzahl von Steuerungsagenten wurde bisher nicht ausreichend untersucht, um die Grenzen bestehender Werkzeuge zu erkennen.

Trotz einiger noch offener Fragestellungen lässt sich die im Beitrag vorgestellte Automatisierungsstruktur sehr gut bewerten:

Mit dem Einsatz der RFID-Technik werden die Materialflüsse transparenter. Die lokal vorhandene Information (Data-on-Tag) ermöglicht lokale Entscheidungen und damit eine dezentrale Anlagensteuerung.

Die Agententechnik bietet einen guten konzeptionellen Ansatz für die Dekomposition eines komplexen Steuerungsproblems in einzelne Teilprobleme und trägt somit zur Komplexitätsreduktion bei. Technische Module und Transporteinheiten mit jeweils ähnlichen Funktionalitäten bekommen dieselbe Steuerungslogik, die im Agentenkonzept gekapselt ist. Somit wird die Wiederverwendbarkeit der Softwareblöcke (Modulagenten) garantiert. Ein Modulagent, seine Rechnerplattform und der dazugehörige Fördertechnikabschnitt bilden im Idealfall ein mechatronisches Modul, das bei Bedarf leicht ausgetauscht werden kann.

Literatur

- [1] logistik journal: Kluges Konzept, S. 52–55 (2012)
- [2] Günther, W., ten Hompel, M. (Hrsg.): Internet der Dinge in der Intralogistik. Springer, Berlin (2010)
- [3] Dijkstra, E.W.: A note on two problems in connexion with graphs. In: Numerische Mathematik, S. 269–271 (1959)
- [4] Java Agent Development Framework (JADE). <http://jade.tilab.com/>

Kapitel 8

Von Softwareagenten zu Cyber-Physical Systems: Technologien und Anwendungen

Thorsten Schöler, Christian Ego, Johannes Leimer und Rico Lieback

Zusammenfassung Anwendungen für Softwareagenten sind mannigfaltig. In diesem Kapitel werden einige beispielhafte industrielle Anwendungen beschrieben, gemeinsame Anforderungen daraus abgeleitet und die Weiterentwicklung der gemeinsamen Softwarearchitektur in Richtung Cyber-Physical Systems beschrieben. Eine moderne objekt-funktionale Plattform für Cyber-Physical Systems SMAS wird vorgestellt. Die Dienstorientierung, die Integration von wissensbasierten Komponenten, wie z. B. Regelmaschinen und Complex Event Processing, in SMAS sowie die graphische Benutzeroberfläche werden vorgestellt. Die Leistungsfähigkeit der Plattform wird kurz vorgestellt und anhand einer ersten Anwendung im Bereich Energiemanagement wird die enge Kopplung der Software mit industriellen Prozessen vorgestellt.

8.1 Einleitung

Heutige industrielle Anlagen besitzen eine Komplexität, welche nur noch mittels moderner IT-Technologien beherrschbar ist. Moderne Ansätze im Umgang mit der Komplexität finden sich in Forschungsaktivitäten wie z. B. dem Exzellenzcluster CoTeSys¹ zur Erforschung von Kognition in technischen Systemen, dem Internet der Dinge² sowie im Bereich Cyber-Physical Systems³ (CPS).

Die aktuelle Herausforderung liegt nicht unbedingt in der Erforschung neuer Technologien, sondern vielmehr in der geschickten Integration bestehender Tech-

¹ <http://cotesys.org> (zuletzt abgerufen am 5. April 2012).

² http://de.wikipedia.org/wiki/Internet_der_Dinge (zuletzt abgerufen am 5. April 2012).

³ <http://www.acatech.de/cps> (zuletzt abgerufen am 5. April 2012).

Thorsten Schöler (✉), Christian Ego, Johannes Leimer, Rico Lieback
Hochschule Augsburg, Friedberger Straße 2a, 86161 Augsburg, Deutschland
e-mail: thorsten.schoeler@hs-augsburg.de, christian.ego@hs-augsburg.de,
johannes.leimer@hs-augsburg.de, rico.lieback@hs-augsburg.de

nologien. Die Informatik hat in der Vergangenheit viele Antworten auf diese Fragestellung gefunden. Beispielsweise ermöglichen Softwareagentensysteme eine lose Kopplung heterogener Systeme und ermöglichen somit eine Integration dieser Systeme auf verschiedenen Abstraktionsebenen. Unter Einbindung moderner mobiler und eingebetteter Systeme und der Möglichkeit des Durchgriffs auf industrielle Systeme und Prozesse entstehen so neue zukunftsweisende Systeme und Lösungen.

Bevor eine Softwarearchitektur für ein solches System vorgestellt werden kann, soll zuerst die Ausgangssituation anhand bestehender Anwendungen beschrieben werden.

8.2 Ausgangssituation

Der aktuelle Stand softwareagentenbasierter, komplexer Systeme soll durch bestehende Industrieprojekte im folgenden exemplarisch an einer Projektauswahl dargestellt werden.

8.2.1 Dezentralisierung des Energiemarktes

Durch die Umstrukturierung der Energieversorgung in Richtung nachhaltiger und regenerativer Energien, der Energiewende entstehen neue Herausforderungen für die IT-Systeme zur Verteilung und Koordination des Energiemarktes⁴.

Es müssen mehr und mehr dezentrale Energieerzeuger in ökonomisch sinnvoller Weise integriert werden. Eine Koordination zwischen Energieerzeugern und -verbrauchern muss auf höherem Niveau erfolgen als in der Vergangenheit. Zusätzlich wird diese Integration dadurch erschwert, dass diese Kleinsterzeiger bei der Konzipierung des bestehenden Energienetzes nicht berücksichtigt wurden. Weiterhin muss die Effizienz bei Energieerzeugung und -verteilung erhöht werden, der Energieverbrauch muss ebenfalls optimiert werden.

Ansätze für die Realisierung einer IT-Infrastruktur beinhalten marktisierte Verfahren zur dezentralen sowie lokalen Koordination von Energieerzeugung und dem Verbrauch von Energie. Intelligente Softwareagenten sind eine geeignete Möglichkeit eine solche IT-Infrastruktur umzusetzen.

⁴ Siehe auch E-DeMA-Projekt unter <http://www.e-dema.de/de/projekt.html> (zuletzt abgerufen am 13. April 2012).

8.2.2 Dezentrale Koordination und Steuerung von dynamischen Logistikketten

Im EU-Forschungsprojekt ILIPT wurden wegweisende Technologien für neue Fertigungskonzepte der Automobilindustrie erforscht. Speziell wurden neue Produktions- und Logistikkonzepte erarbeitet, welche eine bestandslose Build-To-Order-Produktion ermöglichen sollen. Diese Konzepte wurde durch ein ebenfalls entwickeltes dezentrales, agentenbasiertes IT-System für das Produktionsnetzwerk evaluiert.

Wichtiges Merkmal der erarbeiteten Agentensystems war eine verteilte, auf Verhandlungen basierende Auftragsbearbeitung. Durch Softwareagenten, welche dezentral an den Verhandlungen teilnahmen wurden unter anderem die sensiblen Firmeninformationen (z. B. Verhandlungsspielräume) vor den anderen Teilnehmern gekapselt. Für die Interaktion der einzelnen IT-Komponenten bildete eine logikbasierte Beschreibung (Ontologie) die Grundlage [1].

8.2.3 Logik-basierte Fehlererkennung und -analyse

In der Wartung komplexer technischer Systeme geht die Tendenz weg von fixen Wartungsintervallen hin zu flexibler, zustandsabhängiger Wartung [2]. Hier müssen verschiedene Systeminformationen und Messgrößen der Sensorik zu einem widerspruchsfreien Verständnis des aktuellen Systemzustandes integriert werden. Aus dem aktuellen Systemzustand müssen Symptome und Fehlerbilder abgeleitet werden. Unterstützend können hier formale logik-basierte Modelle eingesetzt werden. Moderne wissensbasierte Verfahren können auf Inkonsistenzen hinweisen und das Modell durch abgeleitetes Wissen erweitern. Aus einem solchen logischen Modell können Fehlerursachen und konkrete Fehlerfälle mit mathematischer Sicherheit abgeleitet werden.

Eine konkrete Umsetzung eines intelligenten Fehlererkennungsagenten mit wissensbasierten Verfahren zur Fehlererkennung beschreibt [3]. Hier wird die Qualitätsüberwachung von Werkstücken in der kognitiven Fabrik mit Hilfe einer wissensbasierten Softwarekomponente (Regelmaschine in einem Softwareagenten) umgesetzt. Es wird regelbasiert die Einhaltung definierter Qualitätsstandards überwacht und bei Verletzung der Standards entsprechende Überarbeitungsschritte für das Werkstück vorgesehen.

Weitere Anwendungen für ein logik-basiertes Anwendungs- oder Domänenmodell, wie in den letzten Beispielen als Grundlage verwendet, liegen in der Unterstützung von Anwendern komplexer Computerprogramme wie z. B. CAD-Anwendungen. Hier kann ein Softwareagent beispielsweise aus der aktuellen Anwendung der Software auf nachfolgende Arbeitsschritte des Benutzers schließen oder auch wertvolle Hinweise bei der Konstruktion und Auslegung von mechanischen Bauteilen liefern. Nicht verwunderlich ist, dass dieser Ansatz der wissensbasierten Konstruk-

tion (Knowledge-based engineering) auch im Software-Engineering Anwendung findet [4].

8.2.4 *Digitales Produktgedächtnis*

In [5] wird ein softwareagentenbasierter Ansatz zur Umsetzung eines digitalen Produktgedächtnisses beschrieben. Die beschriebene eingebettete Software sieht Komponenten zur dynamischen Einbindung und Abstraktion von Sensoren vor und bietet einen generischen Zugriff auf die abgelegten Produktinformationen über den gesamten Lebenszyklus des Produktes hinweg. Hierzu wurde ein semantisches Modell erarbeitet welches Informationen integriert und einen standardisierten Zugriff darauf ermöglicht. Über eine eingebettete Regelmaschine ist es möglich Anfragen und Auswertungen auf diesem semantischen Datenmodell durchzuführen. Dies ermöglicht dem eingebetteten Agenten intelligente Entscheidungen zu treffen. Anwendungen für das semantische digitale Produktgedächtnis finden sich in der dezentralen Produktionsleittechnik, in der Wartung sowie der Qualitätssicherung.

8.2.5 *Systemmanagement für Industrieanlagen und Medizineräte*

Bei der Überwachung von größeren Industrieanlagen sowie hochwertigen medizinischen Geräten spielt der Fernzugriff über das Internet eine immer größere Rolle [6]. Zum Einsatz kommen Softwareagenten die auf der Industrieanlage oder dem zu überwachenden medizinischen Geräten verteilt werden. Die verteilten Softwareagenten erstellen ein Anlagen- oder Geräteabbild mit aktuellen Betriebsdaten (Asset Management). Auf diesem logischen Modell können anschließend wissensbasiert Fehlerfälle erkannt und proaktiv eine Behebung dieser Fälle angestoßen werden (Event Management). Für den Fall, dass Anlagen oder Geräte mit neuer Software ausgestattet werden müssen, kann diese über Software-Management-Agenten auf die Anlagen und Geräte verteilt werden.

Zur Überwachung von medizinischen Geräten, wie z. B. Computertomographen, wurde das Softwareagentensystem SMACS umgesetzt [7, 8].

Wie in Abb. 8.1 gezeigt, werden auf den zu überwachenden Computertomographen SMACS-Softwareagenten verteilt. Die Softwareagenten analysieren Log-Dateien und Systeminformationen und stellen daraus ein logisches Systemmodell zusammen. Mittels Verfahren des Datenstrommanagement (auch Complex Event Processing, CEP genannt) ist der Softwareagent in der Lage proaktiv Fehlerfälle zu erkennen und über eine Weiterleitung an das IT-Backend zur Behebung dieser beizutragen. Ebenfalls trägt dieser dezentrale Ansatz dazu bei, die Ereignismengen, welche im IT-Backend zur Fehlererkennung verarbeitet werden müssen signifikant

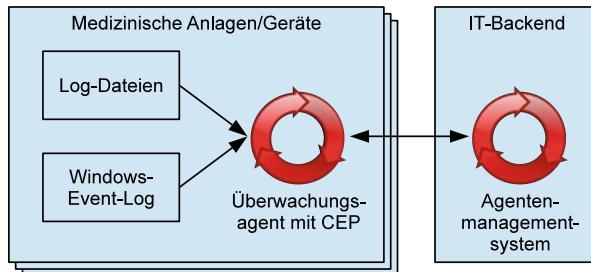


Abb. 8.1 Softwareagent zur Überwachung von technischen Geräten und Anlagen

zu reduzieren. Somit wird wertvolle Bandbreite im Netzwerk sowie Rechenzeit im IT-Backend eingespart.

Ähnliche Szenarien zum Einsatz von Überwachungsagenten finden sich auch in industriellen Anlagen. Hier kann ein logischen Anlagenmodell über den Durchgriff auf SCADA-Systeme (wie beispielsweise OPC) zusammengestellt und ausgewertet werden. Ebenso können Gebäudeautomatisierungssysteme als Datenquelle für Überwachungsagenten dienen. In einem Building Information Model (BIM) können diese Daten aggregiert und mittels Datenstrommanagementsystemen überwacht und ausgewertet werden [9, 10].

8.2.6 Zusammenfassung

Zusammengefasst lässt sich sagen, dass in den gezeigten Anwendungen immer wieder ähnliche Anforderungen an die integrierte Softwareplattform gestellt werden. Unter anderem sind dies Konzepte für autonomes Handeln, die Ankopplung an vorhandener IT-Systeme, verlässliche und sichere Kommunikation zwischen den Systemkomponenten (möglichst unter Echtzeitanforderungen), Modularisierbarkeit (Plug&Play-Fähigkeiten) sowie eine schnelle Umsetzbarkeit von zu erstellenden Anwendungen und Lösungen.

Genau diesen Herausforderungen stellen sich die Cyber-Physical Systems. Wie unter [11] definiert, handelt es sich bei Cyber-Physical Systems um stark in die physischen Prozessen integrierte IT-Systeme. Speziell, eingebettete Systeme wie typische Sensorik- und Steuerungsrechner werden zunehmend untereinander vernetzt und übernehmen intelligente Verarbeitungs- und Steuerungsaufgaben bereits auf sehr prozessnahen Ebenen. Spezielle Herausforderungen für Cyber-Physical Systems liegen in der verlässlichen und sicheren Umsetzung autonomer, dezentrale Algorithmen auf diesen immer leistungsstärkeren eingebetteten und auch mobilen Systemen. Ebenso steht die vertikale Integration von Geschäftsprozessen mit der nun verteilten auf niedrigen Systemschichten angeordneten intelligenten Steuerungs- und Überwachungssoftware.

Im nächsten Abschnitt sollen diese Herausforderungen aufgegriffen und exemplarisch aufgezeigt werden, wie diese Anforderungen durch eine zeitgemäße Softwarearchitektur für Cyber-Physical Systems umgesetzt werden können.

8.3 Konzept, Lösung

Aus den gegebenen Komplexität moderner IT-Systeme sowie der dargestellte Anwendungen, kann eine Softwarearchitektur, welche die angesprochenen Anforderungen erfüllen möchte, aus bereits bestehenden Systemkomponenten effizient zusammengesetzt werden. Ähnliche Schlussfolgerungen können aus dem Markt für Desktop-Betriebssysteme sowie für mobile Plattformen gezogen werden. Hier findet aktuell auch eine Konsolidierung auf unixoide Betriebssysteme (meist Open Source) statt. Mit Ausnahme von Windows (welches im Kern auf VMS basiert), haben die großen Betriebssysteme Linux, Mac OS X, iOS und Android einen unixoiden Kern, welcher in der Open-Source-Gemeinschaft entwickelt und weiter gepflegt wird.

Ähnliche Entwicklungen sind bei Middlewares und auch bei Plattformen für Softwareagenten zu erkennen. Viele Geschäftsanwendungen basieren auf anwendungsneutraler Middleware wie z. B. dem JBoss Application Server⁵, einer robusten und verlässlichen Plattform für Geschäftsanwendungen. Bei den frei verfügbaren Softwareplattformen für Agenten konsolidieren sich viele Entwicklungen um die von TI-Labs entwickelte Plattform JADE⁶.

Auch bei modernen Programmiersprachen findet eine gemeinschaftsgetriebene Entwicklung statt. Als Beispiel können die Entwicklungen um das OpenJDK⁷ sowie um die relativ neue objekt-funktionale Programmiersprache Scala⁸ gesehen werden.

Für eine Middleware für moderne autonome Industriesoftware kann aus diesen Entwicklungen gelernt werden und ein entsprechendes Lösungskonzept abgeleitet werden. Eine moderne Plattform sollte also auf gemeinschaftsgetriebene Komponenten aufsetzen und diese anwendungsspezifisch erweitern. Eine solche Integration und Erweiterung von bestehenden Komponenten zu einer integrierten Architektur für die Umsetzung von Cyber-Physical Systems soll in den folgenden Abschnitten vorgestellt werden.

8.3.1 Einordnung der Technologien

Ausgehend von den bereits beschriebenen Softwarekomponenten wie Betriebssystem und Middleware für verteilte Softwareagenten kann eine beispielhafte Softwa-

⁵ <http://www.jboss.org/jbossas> (zuletzt abgerufen am 5. April 2012).

⁶ <http://jade.tilab.com/> (zuletzt abgerufen am 5. April 2012).

⁷ <http://openjdk.java.net/> (zuletzt abgerufen am 5. April 2012).

⁸ <http://www.scala-lang.org/> (zuletzt abgerufen am 5. April 2012).

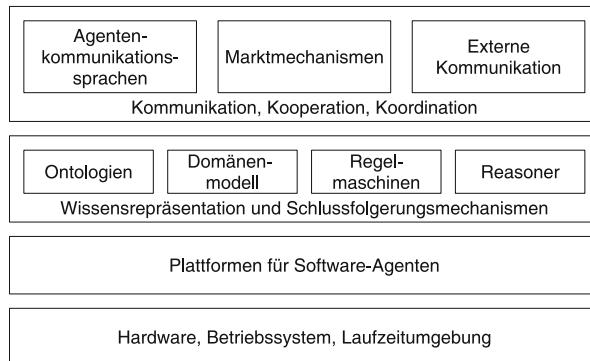


Abb. 8.2 Technologien für Softwareagenten

rearchitektur aus den in Abb. 8.2 gezeigten Softwaretechnologien und entsprechenden Softwarekomponenten bestehen.

Basierend auf der Hardware, dem Betriebssystem und einer Laufzeitumgebung (wie z. B. der virtuellen Maschine von Java) findet man eine Vielzahl von Plattformen für die Umsetzung von Softwareagenten. Beispiele für agentenorientierte Softwareentwicklung für industrielle Anwendungen sowie dafür verwendbare Softwareplattformen finden sich in der Richtlinienreihe VDI/VDE 2653. Exemplarisch für eine Softwareagentenplattform steht hier JADE, welche eine große Akzeptanz im Forschungsumfeld und nicht zuletzt auch in professionellen Anwendungen hat.

Darauf aufbauend werden moderne Softwareagenten mit wissensbasierten Verfahren zur logischen Repräsentation und für logisches Schlussfolgern ausgestattet. Hier finden logische Verfahren wie z. B. Beschreibungslogiken in Form von maschinenverarbeitbaren Ontologien Anwendung. Ziel ist eine semantische Beschreibung der Konzepte sowie der konkreten Umgebung in der ein Softwareagent agiert, das sogenannte Domänenmodell des Softwareagenten. Mit Hilfe von Verfahren der logischen Schlussfolgerung wie z. B. Regelmaschinen und DL-Reasonern kann der Agent neues Wissen aus dem logischen Modell ableiten und mit diesem Wissen sein Verhalten an die jeweiligen Umstände anpassen.

Aufbauend auf einer wissensbasierten Schicht des Softwareagenten kann dieser mit seiner Umgebung kommunizieren. Traditionell erfolgt die Kommunikation von Softwareagenten untereinander mit Hilfe von standardisierten Agentenkommunikationssprachen wie beispielsweise FIPA ACL. Ein Überblick über Agent Communication Languages (ACLs) findet sich unter [12]. Basierend auf der Kommunikation von Softwareagenten untereinander können höhere Kooperations- und Koordinationsverfahren wie z. B. Verhandlungen sowie marktbaserte Verfahren, beispielsweise das weit verbreitete Contract-Net-Protokoll sowie aufwändigeren Auktionsverfahren umgesetzt werden.

Eine Besonderheit der Cyber-Physical System ist, neben der Umsetzung von Agenten auf mobilen und eingebetteten Systemen, die starke Einbindung von Sensorik und Aktorik. Diese Einbindung wird i. A. durch abstrahierte Schnittstellen zur

externen Kommunikation umgesetzt. Als allgemeinverständliches Beispiel für eine solche Schnittstelle kann z. B. die Location-Provider-Schnittstelle [13] im Android-Betriebssystem für mobile Geräte gesehen werden. Diese Schnittstelle abstrahiert verschiedene Lokalisierungsanbieter (GPS, WLAN-Triangulierung, usw.) unter einer einheitlichen API. Cyber-Physical Systems benötigen eine starke Anbindung an Prozesse und die Umgebung und somit müssen verstärkt solche Schnittstellen zur externen Kommunikation angeboten werden. Momentan wird beispielsweise an der Integration von industriellen SCADA-Systemen via OPC sowie an der Integration von Energiemanagementsystemen via Modbus gearbeitet.

Eine bespielhafte, moderne Umgebung zur Umsetzung von Cyber-Physical Systems soll in den folgenden Abschnitten vorgestellt werden.

8.3.2 *Objekt-funktionale Plattform für Cyber-Physical Systems*

Im folgenden wird die moderne objekt-funktionale Plattform für Cyber-Physical Systems SMAS sowie deren graphische Oberfläche vorgestellt.

SMAS-Plattform

Wie bereits erwähnt, haben Cyber-Physical Systems im Vergleich zu klassischen Agentensystemen einige besondere Anforderungen und Bedürfnisse. Beispielsweise werden Cyber-Physical Systems möglichst verteilt und dezentral aufgebaut, während die einzelnen Nodes bzw. Agenten des Gesamtsystems möglichst nah bei ihren jeweiligen Einsatzorten (den industriellen Prozessen) angesiedelt werden.

Diese Verteilung auf viele verschiedene physikalische Computersysteme erfordert wiederum, dass die einzelnen Komponenten möglichst einfach, schnell und zuverlässig miteinander kommunizieren können. Neben der Verteilung und der Kommunikation ist es für eine Plattform zudem wichtig, möglichst flexibel und offen für Erweiterungen und Anpassungen seitens der Systementwickler zu sein. Diese Offenheit und Flexibilität manifestiert sich beispielsweise durch die Integration von Techniken zur Wissensrepräsentation, Schlussfolgerungsverfahren oder ähnlicher verwandter Technologien aus dem Bereich der Maschinenintelligenz.

Um die verschiedenen Anforderungen der Cyber-Physical Systems an eine Plattform zu erfüllen, wurde an der Hochschule Augsburg eine objekt-funktionale Plattform entworfen und implementiert [14]. Ergebnis dieser Arbeit ist SMAS (Scala Multi Agent System), eine Scala-basierte Plattform für Cyber-Physical Systems, die einzelne Agenten in Form von verteilten Akteuren⁹ umsetzt. SMAS realisiert alle für Softwareagentensysteme wichtigen Services und Konzepte und ist dabei für die

⁹ Scala und auch das Akka-Framework unterstützen verteilte, kommunizierende Akteure. Hintergrund zu Akteuren u. a. unter http://en.wikipedia.org/wiki/Actor_model (zuletzt abgerufen am 11. Juni 2012).

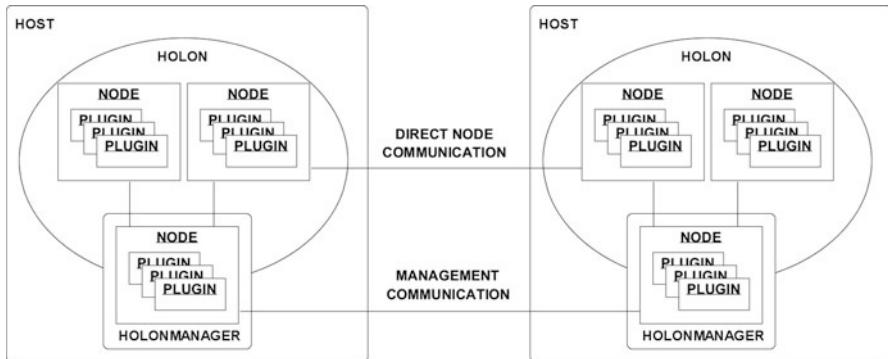


Abb. 8.3 Architektur der CPS Plattform SMAS

Verwendung als Grundlage für CPS optimiert. Diese Optimierung wird beispielsweise durch die dezentrale Realisierung aller wichtigen Plattformdienste erreicht.

Konkret bedeutet das, dass sowohl Kommunikation als auch Verzeichnisdienste (Yellow- und Whitepages) ohne zentrale Strukturen oder Dienste auskommen. Dies ermöglicht die Entwicklung von Systemen, bei denen alle Komponenten gleichmäßig verteilt sind und dennoch ohne großen Aufwand schnell und unkompliziert miteinander kommunizieren können um ihre Aufgabe zu erfüllen.

Abgesehen von der verteilten Realisierung der Dienste und Kommunikation, besitzt SMAS eine offene Architektur, sodass es ohne weiteres möglich ist, weitere Plattformdienste einzuführen, Zugriff auf verschiedenste Hardware oder externe Systeme zu ermöglichen oder erweiterte Technologien, wie beispielsweise Techniken zur Wissensrepräsentation, zu realisierten die es einem Gesamtsystem ermöglichen, seine jeweiligen Aufgaben noch effizienter und effektiver zu erfüllen.

Um diese offene Architektur zu erreichen, wurde bei der Entwicklung der Plattform zum einen ein modulares Pluginsystem, angelehnt an Cougaar [15], umgesetzt, dass die einfache Komposition von Agenten ermöglicht und zum anderen wurden die Plattformdienste selbst auf Basis verschiedener Standardkomponenten (Plugins) realisiert. Die lose Kopplung der einzelnen Plattformkomponenten und ihre Umsetzung als Standardkomponenten des Systems selbst, erhöhen die Erweiterbarkeit und die Wartbarkeit von SMAS erheblich.

Wie in Abb. 8.3 zu sehen ist, können auf Rechnerknoten (Hosts) einzelne Agenten (Nodes) platziert werden. Ein Node besteht aus mehreren Plugins. Nodes untereinander können als Holone zusammengefasst werden. Zwischen den Nodes und deren Plugins findet eine direkte Punkt-zu-Punkt-Kommunikation statt. Jedes Holon ist durch einen Holonmanager im System repräsentiert. Die Holonmanager kommunizieren gleichfalls miteinander.

Eine starke Reduzierung des Implementierungsaufwands für einzelne Agenten, sowie eine gute Skalierbarkeit der Plattform konnte durch den Einsatz der Programmiersprache Scala erreicht werden. Scala ist nicht nur eine moderne, objektfunktionale Sprache, sondern läuft darüber hinaus auf der weit verbreiteten Java Vir-

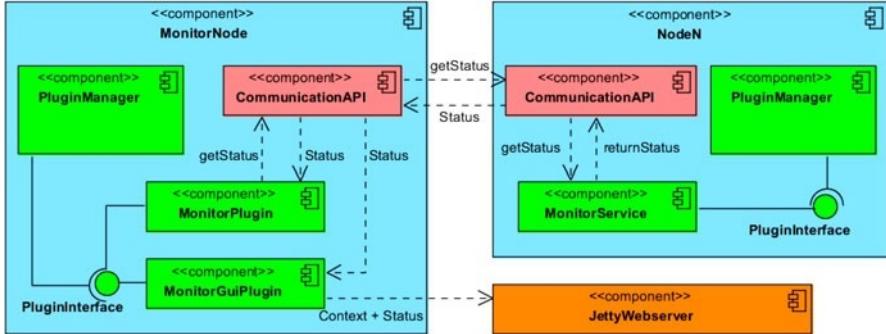


Abb. 8.4 GUI-Plugin eines MonitorNodes

tuellen Maschine (JVM). Diese etablierte und anerkannt leistungsfähige Softwareumgebung, die zudem auf vielen Betriebssystemen, mobilen Plattformen und eingebetteten Systemen verfügbar ist, erleichtert die Planung, Umsetzung und Verbreitung zukünftiger, SMAS-basierter Cyber-Physical Systems. Die vielfältigen Möglichkeiten von Scala zur leichten, sicheren und schnellen Entwicklung von nebenläufiger Software wirken sich außerdem positiv auf SMAS, seine Systemkomponenten sowie SMAS-basierte Systeme aus.

Benutzeroberfläche

Um dem Charakter eines verteilten Cyber-Physical Systems auch bei der Benutzeroberfläche am besten entsprechen zu können, ist eine webbasierte Benutzeroberfläche, welche von einem Node bereitgestellt wird, die bestmögliche Variante. Im Gegensatz zu klassischen Benutzeroberflächen, die nur auf einem lokalen Client verwendet werden können (wie bei JADE), kann auf eine webbasierte Benutzeroberfläche¹⁰ von überall zugegriffen werden und sie hält überdies das System schlank.

Für die objekt-funktionale Plattform SMAS wurde an der Hochschule Augsburg ein Konzept entwickelt, welches auf dem leichtgewichtigen Jetty-Webserver¹¹ basiert [16].

Abbildung 8.4 zeigt dieses Konzept anhand eines Monitor-Nodes, der den Status anderer Agenten abruft und diesen über eine Weboberfläche ausgibt.

In dem vorliegenden Beispiel überwacht ein MonitorNode mit Hilfe eines MonitorPlugin, das für die Statusabfrage zuständig ist, eine beliebige Anzahl von Nodes. Jeder dieser Nodes hält ein MonitorService welcher für die Aggregation und Bereitstellung der Daten der jeweiligen Node zuständig ist und diese bei

¹⁰ Cougaar implementiert ebenfalls eine Web-Oberfläche, allerdings relativ schwergewichtig mittels Tomcat und Servlets.

¹¹ Jetty-Webserver <http://www.eclipse.org/jetty/> (zuletzt abgerufen am 11. Mai 2012).

Anfrage als Status an den `MonitorNode` sendet. Die empfangene Statusnachricht wird dann, entsprechend der Vorgehensweise von Scala Multiagent System an die relevanten Plugins delegiert.

Grundsätzlich kann jedes Plugin die Statusnachricht verarbeiten, allerdings bietet sich die Implementation eines dedizierten `GuiPlugin`s zur Verarbeitung der Daten an. Das `MonitorGuiPlugin` verarbeitet die Statusnachrichten und über gibt den vorbereiteten Context der Weboberfläche an den Jetty Webserver.

Dieses Benutzeroberflächenkonzept wurde als Managementoberfläche für Scala Multi Agent System entworfen und umgesetzt. Die Umsetzung orientiert sich stark an den bewährten Vorbildern der JADE RMA-GUI sowie der webbasierten GUI von Cougaar. Für die zeitgemäße Präsentation als moderne Weboberfläche, wurde als UI-Framework Twitter Bootstrap gewählt¹².

Die GUI besteht im Wesentlichen aus drei Komponenten:

Nodemangement: Das Nodemangement (siehe Abb. 8.5) ist die Hauptansicht der Managementoberfläche. Auf der linken Seite befindet sich, hierarchisch gruppiert, die Auswahl der gesamten Hosts, Holone und Nodes des Systems. Auf der rechten Seite befindet sich die Detailansicht eines einzelnen Nodes. Neben den generellen Informationen über den Node, wie Name Host, Port und ID, sind im Hauptbereich auch die Informationen, die durch das Monitoring-Plugin gesammelt wurden zu sehen. Außerdem kann durch Start-, Pause- und Stop-Button Einfluss auf den Lebenszyklus der einzelnen Nodes genommen werden.

Systemoverview: Die Systemoverview-Sicht bietet eine Gesamtansicht des Systems deren Nodes, Holone und Hosts in deren Relation zueinander.

Einstellungen: Unter Einstellungen kann der Anwender allgemeine Einstellungen des Agentensystems beeinflussen.

Neben einer Web-GUI ist auch eine Android-GUI in Entwicklung. Android als mobile Open-Source-Plattform hat durchaus auch das Potential im Industriemfeld als ernstzunehmende Plattform für beispielsweise SCADA-Anwendungen eingesetzt zu werden. Scala Multi Agent System kann, basierend auf Scala, einer Sprache die zu Java-Bytecode kompiliert, auf Android portiert werden. Ein Konzept sowie eine erste Portierung für eine Android-GUI sind unter [16] beschrieben. Damit ist es möglich, einen spezialisierten GUI-Node auf einem Android-Gerät umzusetzen, der als vollwertiges Element im Multiagentsystem in einer nativen Android-App eine Oberfläche für ein Industriesystem bietet.

¹² Twitter Bootstrap <http://twitter.github.com/bootstrap/> (zuletzt abgerufen am 11. Mai 2012).

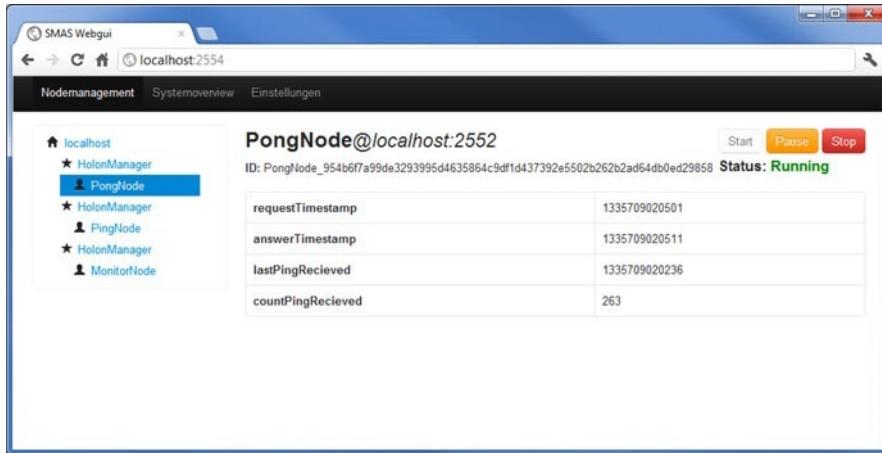


Abb. 8.5 Managementoberfläche Nodemanagement

8.3.3 Integration wissensbasierter Komponenten

Wie bereits am Anfang von Abschn. 8.3 erwähnt, ist die Einbindung wissensbasierter Standardkomponenten ein wichtiges Kriterium für eine zeitgemäße Softwareplattform für Cyber-Physical Systems. Neben DL-Reasonern und sematischen Domänenmodellen ausgedrückt als Beschreibungslogik in OWL DL, spielen regelbasierte Verfahren zur Schlussfolgerung nach wie vor eine große Rolle. Dies bestätigt auch die Verfügbarkeit der modernen Regelmaschine DROOLS im JBoss Application Server. Moderne Regelmaschinen sind zudem in der Lage auch temporale Abhängigkeiten im Domänenmodell zu verarbeiten.

Das Potential von Regelmaschinen und Datenstrommanagementansätzen für Softwareagenten soll in den folgenden Abschnitten kurz erläutert werden.

Regelmaschinen

Ausgehend von klassischen Expertensystemen und Regelmaschinen wie CLIPS [17] sind viele Anwendungen für regelbasiertes Schlussfolgern in industriellen Anwendungen entstanden (siehe auch [3] im Abschn. 8.2).

Regelmaschinen sind eine ressourcenschonende Möglichkeit logisches Schlussfolgern auch auf industriellen Kleinstrechnern (z. B. eingebetteten Systemen) umzusetzen wie das Beispiel des semantischen Produktgedächtnisses unter [18] zeigt. In diesem Ansatz wird eine Fakten- und Regelbasis auf einem eingebetteten Kleinstrechner umgesetzt. Der Kleinstrechner stellt semantische Speicher- und Auswertungsfunktionen zur Verfügung, umgesetzt mit Hilfe einer schlanken Regelmaschi-

ne auf typischen eingebetteten Systemen wie z. B. dem GnuBLIN¹³ oder der gumstix Computer-on-Module-Plattform¹⁴.

Eine aktuelle Herausforderung für regelbasierte Systeme ist der Übergang hin zu ereignisgetriebenen Systemen, welche seit je her in der Industrie üblich sind. Speicherprogrammierbare Steuerungen (SPS) arbeiten traditionell ereignisgetrieben. Sensorereignisse im Eingangsabbild werden durch logische Verknüpfungen zu einem Ausgangsabbild verarbeitet. Der Herausforderung der Ereignisorientierung hat sich die Regelmaschine JBoss DROOLS bereits erfolgreich angenommen. In der Drools Rule Language (DRL) wurden Konstrukte aufgenommen, welche eine ereignis- und datenstromorientierte aber immer noch regelbasierte Verarbeitung erlauben. DROOLS Fusion schlägt die Brücke zwischen traditionellen Regelmaschinen und Datenstrommanagementsystemen bzw. der Complex-Event-Processing-Technologie.

Complex Event Processing

Complex Event Processing oder Datenstrommanagementsysteme sind in der Lage große Ereignismengen effizient zu verarbeiten und über die Korrelation von Einzelereignissen, höheres Wissen zeitnah und kontinuierlich aus den einzelnen Datenströmen zu erkennen. Ursprünglich wurden CEP-Systeme u. a. für den automatisierten Börsenhandel eingesetzt. Die ereignisgetriebene Sicht auf technische Systeme eröffnet neue Möglichkeiten in der Überwachung und Steuerung industrieller Systeme.

Als Beispiel für eine neuartige Architektur zur Überwachung kann ein ereignisgetriebenes SCADA-System für die Intralogistik angeführt werden. Als Ereignisquelle dienen hier bestehende Logistik-IT-Anwendungen und industrielle Steuerungsgeräte der Intralogistik. Die hier generierten Ereignisse werden von einer CEP-Komponente analysiert, gruppiert und ausgewertet. Über deklarativ vorgegebenen Regeln in einer Event Processing Language (EPL) werden aus der Vielzahl von Ereignissen ein Echtzeitabbild des Logistikprozesses erstellt. Die von der CEP-Komponente erkannten Situationen und Fehlerfällen werden auf einer auf Web-Technologien basierten Visualisierungsschicht dargestellt [19]. Ebenso findet die CEP-Technologie Anwendung bei RFID-basierten Logistikprozessen. Hier können die RFID-Ereignisse ebenfalls geeignet bereinigt, korrielt und visualisiert werden.

¹³ Siehe <http://www.gnublin.org> (zuletzt abgerufen am 25. Juni 2012).

¹⁴ Mehr über die Gumstix-Plattform unter <http://www.gumstix.com/> (zuletzt abgerufen am 25. Juni 2012).

Integration in SMAS

Die Regelmaschinen- sowie die CEP-Technologie beweisen eindrucksvoll ihre Leistungsfähigkeit in industriellen Anwendungen. Deswegen ermöglicht die SMAS-Plattform für Cyber-Physical Systems die Einbindung dieser Technologien über die Plug-In-Schnittstelle. Ebenfalls können hierüber OWL-DL-Reasoner-Komponenten eingebettet werden um logisches Schlussfolgern zu ermöglichen.

8.4 Evaluierung, Erfahrung mit der Benutzung

Im folgenden wird eine Leistungsevaluation von SMAS sowie eine der ersten Anwendungen der SMAS-Plattform in einer industriellen Anwendung beschrieben. Momentan wird SMAS in weiteren industriellen Anwendungen u. a. einer dezentralen Steuerung und Optimierung für Fertigungsstraßen analog zu [20] erprobt.

8.4.1 Leistungsevaluation

Wie bereits in Abschn. 8.3.2 erwähnt, ist die Kommunikation zwischen den Komponenten eines Gesamtsystems ein essentieller Bestandteil eines CPS. Aus diesem Grund, werde im folgenden Testaufbauten zur Leistungsevaluation von SMAS vorgestellt und die Ergebnisse dieser Lasttests interpretiert.

Die Aufbauten dieser Tests nutzen eine erweiterte Form des *Ping-Pong*-Beispiels. Ziel dieses Beispiels ist es, dass zwei Nodes einfache Nachrichten austauschen bzw. auf die empfangenen Nachrichten reagieren. Der Arbeitsaufwand bei der Verarbeitung beschränkt sich dabei auf Beantwortung der Nachrichten und ist damit nahezu vernachlässigbar. Im Vergleich zu diesem Aufbau, mit jeweils einem *Ping*- und *Pong*-Node, dienen zwei Aufbauten mit varierbarer Anzahl an Nodes als konkrete Testfälle.

Im ersten Testaufbau wurde überprüft, wie viele Nachrichten ein einzelner *Pong*-Node verarbeiten kann, bevor die Anzahl an Nachrichten zu groß wird. Aus Tabelle 8.1 kann entnommen werden, dass eine größere Menge an *Ping*-Nodes die Anzahl der Nachrichten, welche pro Sekunde verschickt und verarbeitet werden, bis zu einem gewissen Punkt deutlich steigert. Ist dieser Punkt bei etwa vier *Ping*-Nodes zu einem *Pong*-Node erreicht, flacht die Entwicklung ab. Diese Daten zeigen, dass neben Netzwerklatenz und Kommunikationsunterbau auch das Verhältnis der einzelnen Nodes zueinander beachtet werden muss, um optimale Ergebnisse erzielen zu können.

Sehr ähnlich verhält es sich im zweiten Aufbau, bei dem zusätzlich die Anzahl der *Pong*-Nodes variiert wurde. Hierbei lässt sich feststellen, dass ein Verhältnis von vier zu eins auch für größere Aufbauten des *Ping-Pong*-Beispiels optimal ist, während die Leistung für andere Verhältnisse schlechter ausfällt (siehe Tabelle 8.2).

Tab. 8.1 Testergebnisse Leistungstest SMAS Kommunikation Testaufbau 1

Anzahl Ping-Nodes	Anzahl Pong-Nodes	Anzahl Nachrichten/Sekunde	Anzahl Nodes	Verhältnis (Ping/Pong)
1	1	200	2	1/1
2	1	500	3	2/1
4	1	3000	5	4/1
8	1	4550	9	8/1
16	1	5450	17	16/1

Tab. 8.2 Testergebnisse Leistungstest SMAS Kommunikation Testaufbau 2

Anzahl Ping-Nodes	Anzahl Pong-Nodes	Anzahl Nachrichten/Sekunde	Anzahl Nodes	Verhältnis (Ping/Pong)
16	2	3500	18	8/1
16	4	10.000	20	4/1
16	8	12.000	24	2/1
16	16	10.000	32	1/1
32	16	12.000	48	1/1
64	16	12.000	80	4/1

Zudem wird deutlich, dass ab einer bestimmten Gesamtmenge von Nodes (hier etwa 48) die Anzahl an übertragenen Nachrichten nicht weiter steigt. Dies ist mit großer Wahrscheinlichkeit auf den Kommunikationsunterbau zurückzuführen, welcher in der aktuellen Konfiguration nicht in der Lage ist, noch mehr Nachrichten in einer einzelnen Instanz zu verarbeiten.

Zusammengefasst lässt sich sagen, dass SMAS dank seiner dezentralen Kommunikationsarchitektur in der Lage ist auch große Mengen von Nachrichten ohne nennenswerte Leistungseinbrüche zu verarbeiten. Zwar kommt der Kommunikationsunterbau ab einem gewissen Punkt an seine Grenzen, diese Beschränkung kann jedoch durch den Einsatz mehrerer Instanzen bzw. physikalischer Geräte umgangen werden. Zudem dürfte die Verarbeitung der einzelnen Nachrichten bei komplexeren Systemen deutlich länger benötigen, als bei dem hier verwendeten *Ping-Pong*-Beispiel, sodass die Kommunikation im Allgemeinen weniger ausgelastet wird.

Neben dem Nachrichtendurchsatz der einzelnen Testaufbauten wurde zusätzlich der durchschnittliche Speicherverbrauch eines vollständig initialisierten und damit empfangsbereiten Ping-Pong-Nodes ermittelt. Den empfangsbereiten Zustand erreicht ein SMAS-Node, wenn die Zustandsübergänge der Plugins und des Nodes selbst abgeschlossen sind und der Kommunikationsunterbau vollständig geladen wurde. Nachdem ein Node den empfangsbereiten Zustand erreicht hat, benötigt er gemäß der Messung mit der *getFreeMemory*-Methode im Mittel 500 KiB Hauptspeicher.

Die Menge an nötigem Speicher wird zwar in der Realität mit der Komplexität der Nodes wachsen, dennoch ermöglicht der ermittelte Wert, für einen einfachen Node, einen Vergleich mit anderen Systemen oder Nodes mit größerer Funktionalität. Bei dieser Messung wurde der Ressourcenverbrauch der Java-Laufzeitumgebung nicht betrachtet.

8.4.2 Typische Einsatzmöglichkeiten

Die SMAS-Plattform ist eine konsequente Weiterentwicklung der Konzepte, welche sich bereits bei der Überwachung von medizinischen und industriellen Geräten (siehe Abschn. 8.2.5) bewährt haben. Der Plugin-Mechanismus ermöglicht die Integration verschiedener wissensbasierter Komponenten zur Umsetzung der Agentenintelligenz. Weiterhin wurde die Kommunikation der Agenten sicherer und verlässlicher gestaltet.

SMAS kann ebenfalls als Integrationsplattform für Anwendungsfälle in der digitalen Fabrik (siehe Abschnitt 8.2.3) verwendet werden. Beispielsweise kann ein SMAS-Agent nicht nur die angesprochene logik-basierte Fehlererkennung umsetzen, sondern kann, ganz im Stile eines Cyber-Physical Systems, auf prozessnaher, eingebetteter Hardware umgesetzt werden, wie es auch im Anwendungsfall des semantisches Produktgedächtnis bereits vorgestellt wurde.

Auch die Anforderungen einer zunehmende Dezentralisierung des Energiemarktes (Smart Grid) können durch die SMAS-Konzepte umgesetzt werden. Einen ersten Einblick gibt der Anwendungsfall im nächsten Abschnitt.

8.4.3 Anwendung im Energiemanagement

In den vergangenen Jahren ist durch staatliche Subventionen die Anzahl an Kleinst-erzeugern stetig gestiegen, wodurch sowohl auf organisatorischer als auch auf physikalischer Ebene Probleme auftreten. Um diesen zu entgegnen, können Cyber-Physical Systems einen Beitrag zur Automatisierung von Entscheidungen oder Nach-Justierungen bezüglich des Stromnetzes leisten, wenn entsprechende Informationen als Entscheidungsgrundlage vorhanden sind [21]. Im Zuge dieser Wandlung hin zu intelligenten Stromnetzen wird von einem *Smart Grid* gesprochen.

Für eine erste Evaluierung in diesem Gebiet wurde SMAS als Basis eines Cyber-Physical Systems zur automatischen Erfassung und Verarbeitung von Messwerten verwendet. Dabei kamen zahlreiche Funktionen von SMAS, wie beispielsweise das dezentrale Dienste-Verzeichnis (*Yellow-Pages*) oder die transparente Kommunikation zum Einsatz. Die Plugin-Architektur von SMAS unterstützte hingegen eine einfache Implementierung von eigenen Funktionen und Diensten.

Die Architektur der Energiemanagement-Evaluierung aus Abb. 8.6 teilt sich aufgrund der verteilten Struktur von SMAS in mehrere Dienste auf:

Semantic Matcher Service: Innerhalb des Semantic Matcher Dienstes ist ein Objektmodell für die Speicherung von Metadaten angesiedelt, welches als Erweiterung für die allgemeinen Yellow-Pages angesehen werden kann. Anhand von semantischen Suchanfragen werden Geräte über ihren Standort und deren Fähigkeiten aus dem Modell ausgelesen und anschließend mittels SMAS an den entsprechenden Empfänger weitergeleitet. Diese Aufgabe übernimmt das SemanticMatcherPlugin mit Hilfe des SemanticModels in einer SemanticMatcher Node.

Device Registration: Außerdem ist das SemanticMatcherPlugin für die Aufnahme neuer Geräte in das System verantwortlich und bietet dazu den Device-Registration-Dienst an. Somit können Nodes mit Hardware-Zugriff ihre Fähigkeiten dem Rest des Systems bereitstellen und die Geräte in eine zentrale Registrierung eintragen.

Modbus Device Initializer: Für eine abstrakte Beschreibung von Modbus-Geräten wurde eine XML-basierte Spezifikation entworfen und beispielhaft für zwei Geräte realisiert. Diese werden über den ModbusDeviceInitializer eingelesen und anschließend an die Device Registration übermittelt, um sie systemweit zu integrieren.

Hardware Prototype: Als Basis für die Evaluierung wurde ein Prototyp aus Mess- und Steuergeräten hergestellt und auf eine Verwendung mit SMAS vorbereitet (siehe Abb. 8.7). Dazu integrieren alle Geräte eine Modbus-TCP Schnittstelle, wodurch sie über bestehende Netzwerk-Infrastrukturen mit SMAS-Nodes kommunizieren können. Die Geräte-Beschreibungen, welche über den ModbusDeviceInitializer veröffentlicht werden, sind auf diese Geräte angepasst und sind nach abgeschlossener Registrierung einsatzbereit.

Modbus Service: Der Modbus-Service ist ein zustandsloser Dienst, welcher erhaltene Befehle auf Modbus-Pakete umsetzt und an die entsprechenden Geräte weitergeleitet. Er erstellt dazu für jede empfangene Nachricht eine neue Verbindung her und schließt sie nach Vollendung des Prozesses wieder. Im konkreten

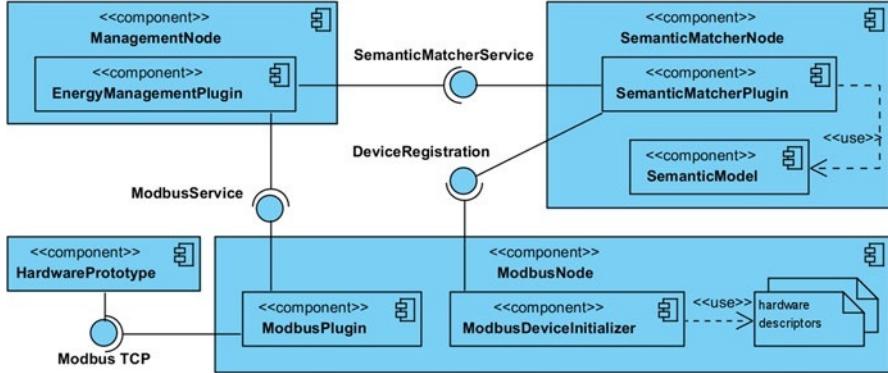


Abb. 8.6 Architektur der Energiemanagement-Evaluierung

Fall der Evaluierung ist der Modbus-Service das Bindeglied zwischen der SMAS-Plattform und der Hardware in Form von Modbus-Geräten.

EnergyManagementPlugin: Die Steuerung und Überwachung des *Smart Grids* wird exemplarisch von dieser Komponente repräsentiert und beinhaltet die dafür notwendige Intelligenz. Hierzu werden alle bereits angesprochenen Dienste und Funktionen innerhalb eines SMAS-Plugins angesprochen und verwendet, um Verwaltungsaufgaben im Smart Grid zu erfüllen. Unter Anderem gehören dazu die Überwachung des Stromverbrauchs von dem angeschlossenen Verbraucher und das Trennen des Stromkreises bei Überschreiten eines festgelegten Schwellwertes.

Durch die oben erläuterten Komponenten wird SMAS dahingehend erweitert, dass das EnergyManagementPlugin die notwendigen Geräte über den Semantic-Matcher-Service abrufen kann und daraufhin direkt per Peer-to-Peer Kommunikation mit dem Modbus-Service interagiert.

Während der Evaluierung wurde ein haushaltstypischer, elektronischer Verbraucher verwendet, dessen aktueller Stromverbrauch sich weitgehend stufenlos verändert hat. Als konkrete Ausprägung eines Anwendungsfalls wurde ein Schwellwert von 50 Watt festgelegt und bei dessen Überschreitung für fünf Sekunden der Stromkreis unterbrochen. Anschließend wurde die Musikwiedergabe von der Anlage in Standardlautstärke fortgesetzt und lag damit wieder unter des Grenzwerts. Die empfangenen Messdaten werden fortlaufend in einem Diagramm dargestellt und ermöglichen eine zusätzliche visuelle Überprüfung der Werte.

Mittels der beschriebenen Softwarearchitektur wurde ein System umgesetzt, welches die Überwachung und Steuerung von realen Stromverbrauchern ermöglicht und somit die Evaluierung von SMAS als Grundlage für ein einfaches

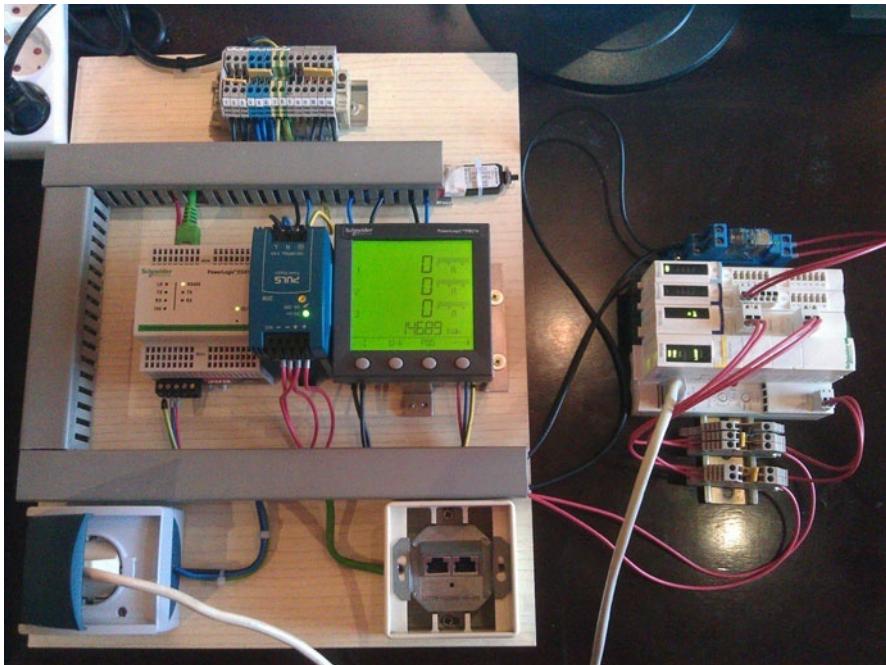


Abb. 8.7 Hardware-Prototyp für die Energiemanagement-Evaluierung [22]

Cyber-Physical System ermöglicht. Durch den einfachen und nachvollziehbaren Anwendungsfall konnte gezeigt werden, dass SMAS in der Lage ist, als Plattform für Cyber-Physical Systems verwendet zu werden und eine dezentrale Struktur bereits durch die Konzepte von SMAS realisiert werden kann.

8.5 Zusammenfassung

Ausgehend von existierenden industriellen Anwendungen von Softwareagenten wurde eine generische Softwarearchitektur beschrieben, welche auch auf mobilen und eingebetteten Systemen eingesetzt werden kann und weiterhin durch den Durchgriff auf industrielle Prozesse zu einer Plattform für Cyber-Physical Systems weiterentwickelt wurde. Die auf bestehenden Erfahrungen basierende SMAS-Plattform stellt eine moderne und leicht erweiterbare Softwarearchitektur zur Verfügung, welche es ermöglicht schnell und einfach sichere verteilte Systeme umzusetzen. Das Plugin-Konzept ermöglicht die Integration von weiteren Komponenten wie z. B. einer CEP-Engine zur Ereigniskorrelation in industriellen Prozessen. Auch eine semantische Beschreibung von Aktoren und Sensoren ist Bestandteil der SMAS-Plattform. Die Architektur und Benutzeroberfläche der

Plattform wurden kurz vorgestellt und eine erste Anwendung im Energiemanagement beschrieben.

Die Erfahrungen mit SMAS in Anwendungen und auch der Ausbildung zeigen das große Potential der quelloffenen objekt-funktionalen Plattform. Die SMAS-Plattform ist als Open-Source-Plattform für Cyber-Physical Systems entwickelt worden und stellt für viele weitere Anwendungen eine wertvolle Grundlage dar.

Aktuell entsteht eine Softwarearchitektur für die RFID-basierte Erfassung, Korrelation und Visualisierung von Prozessen der Intralogistik. Weitere Anwendungen entstehen im Smart-Meter- und Smart-Grid-Umfeld. SMAS stellt durch seine Diensteorientierung, seine klare Aufteilung von Funktionalitäten unter dezentralen Einheiten sowie der leichten Integrierbarkeit von wissensbasierten Komponenten eine geeignete Plattform.

Literatur

- [1] Parry, G., Graves, A.: *Build to order: the road to the 5-day car*. Springer, Berlin Heidelberg (2008)
- [2] Liu, L., Cartes, D.A., Quiroga, J.: Modeling and simulation for condition based maintenance: a case study in navy ship application. Proc. of the 2007 summer computer simulation conference, SCSC, S. 244–249. Society for Computer Simulation International, San Diego, CA, USA (2007). <http://dl.acm.org/citation.cfm?id=1357910.1357950>
- [3] Ostgatthe, M., Zäh, F., Grimmert, P., Schöler, T.: Wissensbasiertes Störungsmanagement in Produktionsabläufen. *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* **106**(11), 838–843 (2011)
- [4] Wongthongham, P., Chang, E., Dillon, T.S.: Ontology-based multi-agent system to multi-site software development. In: Proceedings of the 2004 workshop on Quantitative techniques for software agile process, QUTE-SWAP '04, S. 66–75. ACM, New York, NY, USA (2004). doi: 10.1145/1151433.1151443. <http://doi.acm.org/10.1145/1151433.1151443>
- [5] Seitz, C., Schöler, T., Neidig, J.: An Agent-based Sensor Middleware for generating and interpreting Digital Product Memories. In: Adaptive Agents and Multi-Agents Systems (2009)
- [6] Holm, T., Wiener, P., Horn, S.: Using Service Challenge-based Evaluations for the systematic innovation of proactive remote services – An industry case study. In: Emerging Technologies and Factory Automation, S. 1–4 (2010). doi: 10.1109/ETFA.2010.5641219
- [7] Groetsch, R.: Entwicklung einer Ereignissprachenübersetzung und eines Management-Backends zur Überwachung und Verwaltung von verteilten Korrelationsagenten auf medizinischen Geräten. Bachelorarbeit, Hochschule Augsburg (2011)
- [8] Noetzel, C.: Konzeption, Implementierung und vergleichende Performanceanalyse eines CEP-basierten Überwachungsagenten für medizinische Geräte. Bachelorarbeit, Hochschule Augsburg (2011)
- [9] Weiss, A.: Integration von Ereignisverarbeitungs- und OPC-Infrastrukturen. Bachelorarbeit, Hochschule Augsburg (2011)
- [10] Hell, S.: Regelbasierte Überwachung von Kenngrößen zum Gebäudemanagement mit IBM Websphere ILOG JRules. Bachelorarbeit Hochschule Augsburg (2011)
- [11] Lee, E.A.: Cyber physical systems: Design challenges. Object-Oriented Real-Time Distributed Computing. IEEE International Symposium on **0**, 363–369 (2008). <http://doi.ieeecomputersociety.org/10.1109/ISORC.2008.25>

- [12] Kone, M.T., Shimazu, A., Nakajima, T.: The state of the art in agent communication languages. *Knowledge and Information Systems* **2**, 259–284 (2000). <http://dx.doi.org/10.1007/PL00013712>
- [13] Hashimi, S.Y., Komatineni, S.: Pro Android. Apress, New York (2009)
- [14] Lieback, R.: Objekt-funktionale Plattform für Cyber-Physical Systems. Bachelorarbeit, Hochschule Augsburg (2012)
- [15] B B N Technologies: Cougaar Architecture Document. Change (December) (2004)
- [16] Ego, C.: Anwendung eines Cyber-Physical Systems für Android – Mobile Monitoring und Steuerungsapplikation in einer Home Automation Umgebungen. Bachelorarbeit, Hochschule Augsburg (2012)
- [17] Giarratano, J., Riley, G.: Expert systems: principles and programming. PWS-Kent series in computer science. PWS-KENT Pub. Co. (1989). <http://books.google.de/books?id=qwEvAQAAIAAJ>
- [18] Seitz, C., Lamparter, S., Schoeler, T., Pirker, M.: Embedded rule-based reasoning for digital product memories. AAAI Spring Symposium: Embedded Reasoning. AAAI (2010)
- [19] Spanrunft, T.: Ereignisgesteuerte Echtzeitvisualisierung für Intralogistik. Bachelorarbeit, Hochschule Augsburg (2012)
- [20] Kuehnle, H.: Distributed Manufacturing. Springer, Berlin Heidelberg (2010)
- [21] Khan, U.: Impact of distributed generation on electrical power network. Wroclav University of Technology, Wroclav, Poland (2008)
- [22] Leimer, J.: Anwendung eines Cyber-Physical Systems für intelligentes Energiemanagement. Bachelorarbeit, Hochschule Augsburg (2012)

Kapitel 9

Freie Fahrt – Softwareagenten reduzieren Schadstoffausstoß

Christian Dannegger

Zusammenfassung Dieses Kapitel behandelt ein Projekt, das dazu diente, den Nutzen einer neuartigen Ampel- und Verkehrssteuerung auf Basis von Softwareagenten zu untersuchen und die konkreten Bausteine zum Einsatz eines Komplettsystems in mittelgroßen Städten zu erarbeiten. Die Ergebnisse sind überaus motivierend, eine solche Steuerung zum Einsatz zu bringen. Obwohl – oder gerade weil – der Agentenansatz hier sehr einfach ist, zeigen alle Simulationsergebnisse wie erfolgreich dezentrale Steuerungsansätze komplexe Szenarien dynamisch optimieren können. Nach der Einführung in die Problematik und der wesentlichen Anforderungen folgt die Beschreibung des Gesamtsystems. Nach einem kurzen Abriss des agentenbasierten Lösungsansatzes gehen wir dann etwas ausführlicher auf die Test-Szenarien und deren Auswertung ein.

9.1 Einleitung – Zuviel Lärm, zu viel Schadstoffe, zu viel Stillstand

Die meisten Autofahrer haben das schon oft erlebt: ganz alleine an einer Kreuzung stehen und auf Grün warten. Dann, wenn man endlich Grün hat, biegt man ab und steht schon wieder vor der nächsten roten Ampel. All diese unnötigen Wartezeiten sind nicht nur frustrierend für alle Verkehrsteilnehmer. Darüber hinaus belasten sie unsere Natur mit Schmutz und Lärm, denn der Schadstoffausstoß steigt gleichzeitig mit unserer Frustration.

In der Vergangenheit wurden viele Anstrengungen unternommen, den Verkehrsfluss zu optimieren – meist in größeren Städten. Herkömmlicherweise wird der Verkehrsfluss zunächst analysiert und durch große Computersysteme simuliert, um dann daraus Schlüsse für eine optimierte Verkehrsflusstrategie zu ziehen. Auch

Christian Dannegger (✉)
Kandelweg 14, 78628 Rottweil, Deutschland
e-mail: cd@3a-solutions.com

wenn dabei intelligente Computersysteme zum Einsatz kommen, werden die Kernentscheidungen „offline“ getroffen und dann statisch umgesetzt und eingeführt. Diese bleiben dann so lange aktiv, bis die nächste Optimierungs runde gestartet wird.

Zentrale Systeme sind zu unflexibel

Ohne Zweifel reduziert dieser traditionelle Ansatz die Wartezeiten, aber durch die hohen Vorinvestitionen und die langen Reaktionszeiten bei Veränderungen der Verkehrssituation ist diese Lösung für kleine und mittlere Gemeinden wenig geeignet. Ebenso wenig erfüllt der traditionelle Ansatz den wachsenden Bedarf an selbstständig adaptiven Systemen, die durch permanente Umgebungsanpassung den Schadstoffausstoß weiter substantiell senken.

Das zentrale und „festverdrahtete“ Design herkömmlicher Verkehrssteuerungs-Software birgt natürliche Beschränkungen im Umgang mit der Unvorhersehbarkeit der realen Welt. Deshalb ist es erforderlich für die Entwicklung einer Steuerungssoftware einen neuen Ansatz zu wählen, der es ermöglicht, mit sich dynamisch verändernden Konfigurationen möglichst ohne manuellen Eingriff umzugehen.

Flexibilität und Adaptivität sind gefragt

Eine solche Steuerung zeichnet sich durch absolute Flexibilität und Adaptivität aus und treibt die Automatisierung und Optimierung einer modernen Verkehrssteuerung weiter voran – inmitten ständig steigender Komplexität und Dynamik.

Eine Innovation in diesem Bereich ist deshalb ein entscheidendes Differenzierungsmerkmal für Hersteller und Betreiber von modularen Ampelsteuerungen.

Dieses Kapitel beschreibt die Anforderungen an eine zukunftsfähige Ampelsteuerung bzw. Verkehrsflussteuerung, diskutiert die Ampelsteuerungsprinzipien, zeigt Lösungsansätze auf Basis von autonomen Agentensystemen auf und legt abschließend viel Wert auf die Simulation und die Analyse derselben im Hinblick auf Wartezeiten, Durchsatz und Schadstoff-Emission.

Beschränkung auf aktuelle Technologie

Wichtig bei diesem Projekt war der Einsatz aktuell verfügbarer Technologien. So wurde z. B. bewusst auf die Betrachtung der Möglichkeit verzichtet, direkt zwischen Kreuzungs-Steuerung und Fahrzeug zu kommunizieren, denn dies ist noch „Zukunfts musik“. Diese derzeit in Entwicklung befindlichen Technologien auf Basis von WiFi, Bluetooth oder Ähnliche bereiten natürlich den Boden für weit ausgefelierte Steuerungen. So könnte dann das Fahrzeug der Ampelsteuerung z. B. mitteilen, wo es (bzw. der Fahrer) hin möchte, wenn ein Navigationssystem ebenfalls im Einsatz ist. Oder umgekehrt kann die Ampelsteuerung dem Fahrzeug umfassende

Routenempfehlungen auf Grund der aktuellen Verkehrssituation geben. Dies setzt aber voraus, dass alle Fahrzeuge damit ausgerüstet werden.

„Kooperative Kreuzungen“ senken Wartezeit und Emission

Dieses Projekt zielte aber auf direkt umsetzbaren Nutzen ohne weitere technologische Voraussetzungen. Daher sollte es genügen, den Verkehrsfluss über verbesserte Sensorik zu erfassen und ihn über die bestehenden Signalanlagen zu beeinflussen. Einzig die adaptive Steuerung, d. h. die „Grün-Entscheidungen“ durch kooperative Abstimmung zwischen den einzelnen Kreuzungen genügt, um den Fahrzeugdurchsatz zu erhöhen, aber – noch wichtiger – die Wartezeiten und den Schadstoffausstoß signifikant zu senken.

9.2 Ausgangssituation/Problemstellung

Die Ausgangssituation soll anhand des Beispiels in Abb. 9.1 (Kreuzungsszenario) beschrieben werden.

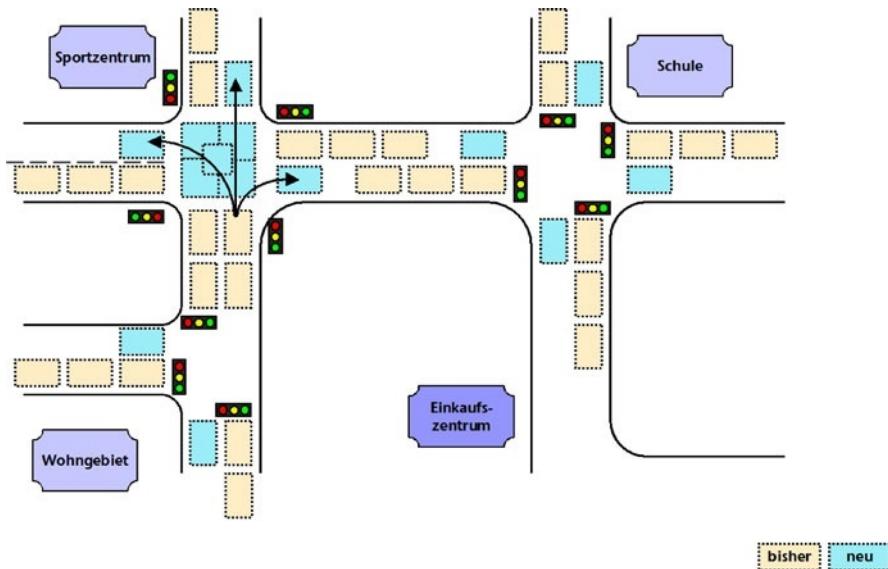


Abb. 9.1 Kreuzungsszenario

Dynamische Frequentierung

Dieses Szenario zeigt drei Kreuzungen mit Ampelsteuerung. Wesentlich sind aber die vier Lokationen, die je nach Tageszeit unterschiedlich frequentiert werden. So wird eine Schule normalerweise morgens und mittags angefahren, wenn Lehrer, Schüler und Eltern zur Schule hin oder von der Schule weg fahren. Das Wohngebiet ist davon ebenso betroffen, wird aber zusätzlich stärker nach Feierabend frequentiert. Dann kommt auch das Sportzentrum ins Spiel. Für ein Einkaufszentrum kann kein allgemeines Muster beschrieben werden, außer einer leichten Erhöhung vor dem Mittagessen und vor dem Abend.

Diese unterschiedlichen Frequentierungen spielen eine zentrale Rolle in diesem Projekt, da gerade die unterschiedlich starken Verkehrsströme über mehrere Kreuzungsknoten und Straßen hinweg optimal fließen sollen. Ziel ist die automatische Erkennung der Verkehrsströme und eine entsprechend autonome Abstimmung zwischen den Kreuzungen, so dass der Autofahrer entlang seiner üblichen Route möglichst eine grüne Welle bekommt. Die grüne Welle soll sich je nach Uhrzeit und Gesamtsituation am Hauptverkehrsstrom ausrichten – vollautomatisch.

Induktionsschleifen und andere Sensoren

Bis heute üblich sind zwei Varianten der Ampel-Steuerung. Entweder ist eine zyklische Steuerung mit festen Phasen und Zeiten im Einsatz. Oder die Ampeln werden durch Induktionsschleifen bedarfsabhängig geschaltet. Dabei erkennt der Induktionssensor 3–10 Fahrzeuge je, nachdem wie weit entfernt von der Kreuzung die Induktionsschleife verbaut wurde. Manche Kreuzungen sind auch schon mit Kameratechnik ausgestattet. Diese erkennen aber normalerweise nur bis zu 3 Fahrzeuge – bei LKWs weniger.

Abbiegeverhalten erkennen

Die maximal möglichen Sensorfelder bei heutiger Technik sind in der Abbildung als hellgraue Kästchen mit gepunktetem Rand dargestellt. Damit wird deutlich, dass nur der wartende Verkehr erfasst werden kann, nicht aber deren Abbiegeverhalten. Genau diese Erweiterung ist ebenfalls Ziel dieses Projektes. Die dunkelgrauen Kästchen zeigen die weiteren Sensorfelder, die durch Kameras erfasst und durch Bilderkennungsalgorithmen erkennbar sind. Damit kann ein System zum einen Statistiken über das Abbiegeverhalten aufbauen und zum anderen als vorgelagerter Sensor für die nächste Kreuzung in Abbiegerichtung dienen. Dadurch ist die Folgekreuzung frühestmöglich über den kommenden Verkehr informiert und kann entsprechend besser reagieren.

Anforderungen an eine zukunftsähige Steuerung

Das Anforderungsprofil an eine solche neue Generation Steuerungssoftware beinhaltet eine Reihe fortgeschritten, einzigartiger Merkmale. Diese dienen dazu, die Komplexität bei Entwicklung, Betrieb und Wartung von Ampelanlagen und Verkehrssteuerungen zu minimieren, ohne dadurch Freiheitsgrade in den Anwendungsszenarien zu verlieren – selbst wenn diese zum Zeitpunkt der Installation noch nicht alle absehbar sind.

Dynamische Durchsatzoptimierung

Die zentrale Anforderung an eine moderne Verkehrssteuerung ist eine kreuzungsübergreifende Optimierungsmöglichkeit, die vorausschauend das Gesamtbild der Verkehrssituation berücksichtigt. Dabei ist nicht nur die Gesamtwartezzeit aller Verkehrsteilnehmer zu minimieren, sondern das Vermeiden von unnötigen Stopps und Beschleunigungsmanövern zielt darüber hinaus auf die Senkung des Schadstoffausstoßes ab. Das Ziel des Einzelnen ist das Ziel des Gesamtsystems: Weniger stehen – weniger warten – weniger CO₂.

Automatische Umgebungsadaption

Ein adaptives System stellt sich automatisch auf eine sich verändernde Umgebung ein. Dazu gehört neben dem Erkennen einer neuen autonomen Kreuzungssteuerung in der Nachbarschaft auch das Einbeziehen von Umwelteinflüssen wie Temperatur oder Niederschlag. Bei glatten Straßen macht ggf. eine andere Ampeltaktung Sinn.

Robuste Störungsresistenz

Für die Minimierung des Wartungsaufwands ist gefordert, dass eine moderne Steuerung Störungen und Ausfälle sofort erkennt, angemessen beurteilt und behebt oder umgeht. Bei z. B. komplettem oder teilweisem Ausfall der Kommunikation optimiert ein modernes System weiter auf Basis der beschränkt zur Verfügung stehenden Daten. Die Optimierung verschlechtert sich zwar etwas, aber das System fällt nicht komplett aus.

Bestehende Struktur nutzen

Aus Kostengründen ist die bestehende Infrastruktur wie Steuerungsanlagen, aber vor allem Sensoren und Kommunikationsmöglichkeiten einzubinden und zu nutzen.

Kostengünstige Installation

Eine Installation und Inbetriebnahme ohne größere Bau- und Erdarbeiten wirken sich sehr günstig auf das Projektbudget aus. Nach Montage und Ausrichtung der Kameratasoren und der Verbindung des Kreuzungssystems mit dem Steuerrechner, konfiguriert sich das System nahezu selbst, insbesondere lokalisiert es sich selbst (optional) und stellt Verbindung mit den Nachbarsystemen her.

Einfache Fernwartung

Und nicht zuletzt ist eine solch fortgeschrittene Lösung mit einem modernen Benutzerinterface auszustatten. Dieses erlaubt einfachste Bedienung und Fernwartungsmöglichkeit über das Internet bzw. Intranet.

9.3 Stand der Technik

Bisherige Kameratasoren mit statischer Erkennung

Neben den allseits bekannten Induktionsschleifen sind auch seit einigen Jahren Kameratasore mit Bilderkennung in Betrieb. Diese arbeiten aber mit einer statischen Bild-Erkennung auf Basis fest vordefinierter Sensorfelder. Aufgrund der Blickwinke „sehen“ diese System max. 2–3 Fahrzeuge (PKW) weit.

In den Voruntersuchungen zu diesem Projekt haben sich folgende Probleme gezeigt:

1. Die Schnittstelle auf Basis stehender Fahrzeuge hat sich als nicht ideal bzw. nicht ausreichend herausgestellt
2. Bei stehenden Objekten können nur max. die ersten drei Fahrzeuge erkannt werden, was deutlich zu wenig für ein sinnvolle Steuerung ist.

Neue Kameratasoren mit dynamischer Erkennung

Neue Erkennungsalgorithmen arbeiten mit der Veränderung im Videobild. Sie erkennen die sich bewegenden Teile bzw. Objekte einer Fahrspur und melden die veränderten Teilbereiche eines Bildes über ihre Schnittstelle. Dies hat einen großen Vorteil, da Bewegung wesentlich besser zu erkennen ist als stehende Objekte.

Dennoch bleibt die Aufgabe für die nachgelagerten Systemebenen, aus den Bewegungsmustern auf tatsächliche Fahrzeuge zu schließen. Es besteht das Problem, dass Objekt „verschwindet“ wenn es steht. Vergleichbar mit der Tarnung eines Tiers in der Natur. Deshalb musste ein Modul entwickeln werden, das aus der Sequenz der sich bewegenden Objekte ableiten kann, welche Warteschlangenlängen sich gebildet haben und mit welcher Geschwindigkeit sich die Fahrzeuge nähern.

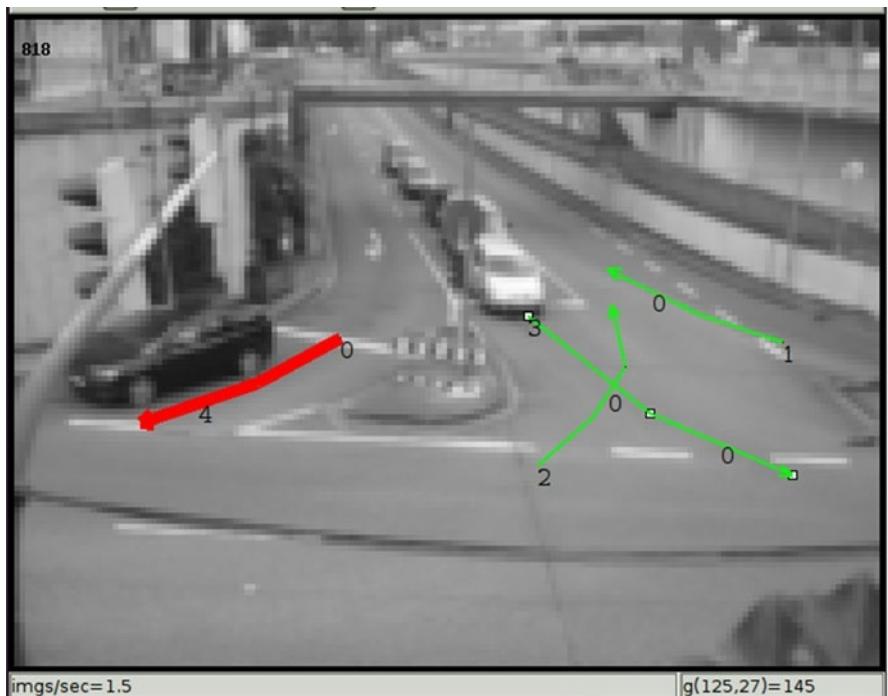


Abb. 9.2 Abbiegeerkennung

Abbiegeerkennung

Abbildung 9.2 zeigt ein Beispiel einer Abbiegeerkennung. Jeder Fahrtmöglichkeit über eine Kreuzung wird eine Abbiege-ID zugeordnet, die eindeutig eine Quellspur mit einer Zielpur verknüpft. Aus der Sicht der Steuerungssoftware stellt sich die Schnittstelle sehr einfach dar: Es werden nur einfache Abbiege-IDs mit Zeitstempel übermittelt.

9.4 Konzept/Lösung

Das Gesamtkonzept der dynamischen Verkehrsflussteuerung besteht aus folgenden Grundelementen bzw. basiert auf diesen Grundprinzipien:

Wartende und herannahende Fahrzeuge werden detektiert, so dass eine bestehende Induktionsschleife nicht mehr notwendig ist bzw. durch zusätzliche Sensordaten ergänzt, wie z. B. Geschwindigkeit.

Das Abbiegeverhalten wird aufgezeichnet und mit einbezogen in die Phasensteuerung einer Kreuzungssampel.

Über das Verkehrsaufkommen werden Statistiken mitgeführt, die für Prognosen über künftige Verkehrsströme herangezogen werden können.

Aufgrund des aktuellen Verkehrs an einer Kreuzung kann diese das erwartete Verkehrsaufkommen für die Folgekreuzungen prognostizieren und es diesen mitteilen. Die „Empfangenden“ Kreuzungen werden über tatsächliche „Abgänge“ informiert und ebenso über prognostizierte „Abgänge“ des erst herannahenden Verkehrs.

Auf Basis physisch detekтирter und erwarteter Fahrzeuge steuert ein Kreuzungsagent die Signale nutzenmaximierend. Zielfunktion ist die Schadstoffreduktion unter Einbeziehung aller Informationen über den aktuellen lokalen Verkehr den Prognosen der Nachbarkreuzungen.

9.4.1 Welches Agentenmodell macht Sinn?

Endlich ist im vorangegangenen Abschnitt das Stichwort „Agent“ gefallen: Kreuzungsagent. Wie kam es aber dazu? In diesem Projekt war klar, dass ein agentenbasiertes System zum Einsatz kommen soll, um die oben beschriebenen Anforderungen erfüllen zu können. Aber nun galt es zu entscheiden, welches der beste Agentenansatz ist bzw. welche Komponenten als Agenten umgesetzt werden sollen.

In jedem Agentensystem analysiert man zunächst, welche Aufträge bzw. Nachfrager es gibt und welche Ressourcen verplant und optimiert werden sollen. Meist „springt“ man auf alles, was sich bewegt. Ein Auto, ein Fahrzeug ist zunächst naheliegend, denn dieses muss warten und erzeugt den Schadstoffausstoß. Noch extremer, könnte man auch jeden Verkehrsteilnehmer, also jede Person als Agent darstellen. Dann hätte ein Fahrzeug schon mehrere Agenten, Fahrradfahrer und Fußgänger wären dann auch erfasst. Ein Bus hätte sehr viele Agenten. Problem hier ist, dass diese derzeit durch keinen Sensor erfassbar sind. Die nächste Überlegung galt jeder Fahrspur, die mehrere Fahrzeuge zusammenfasst, die alle gemeinsam eine bestimmte Fahrtrichtung bzw. Fahrspur nutzen wollen. Dann denkt man schnell an eine Ampel, also ein Kasten mit den drei Farbleuchten grün, gelb und rot. Oder sollte man eine gesamte Kreuzung als Agenten darstellen?

Letztendlich fiel die Wahl im Projekt auf einen **Kreuzungsagenten**, der die Ressource „Kreuzung“ repräsentiert und einen **Fahrspuragenten**, der die Nachfrager (Kunden, Aufträge) darstellt. Die Fahrspuragenten werden durch die neuen Kamerasensoren und bestehenden Induktionsschleifen mit Informationen versorgt. Sie verhandeln dann, als Repräsentanten einer Gruppe von Fahrzeugen, mit der Kreuzung über ihre Grünphase. Der Kreuzungsagent wurde dem Spuragenten vorgezogen, da üblicherweise die Kreuzung nur **genau** eine Ressource darstellt, nämlich die Kreuzungsmitte, über die jede Fahrspur führt.

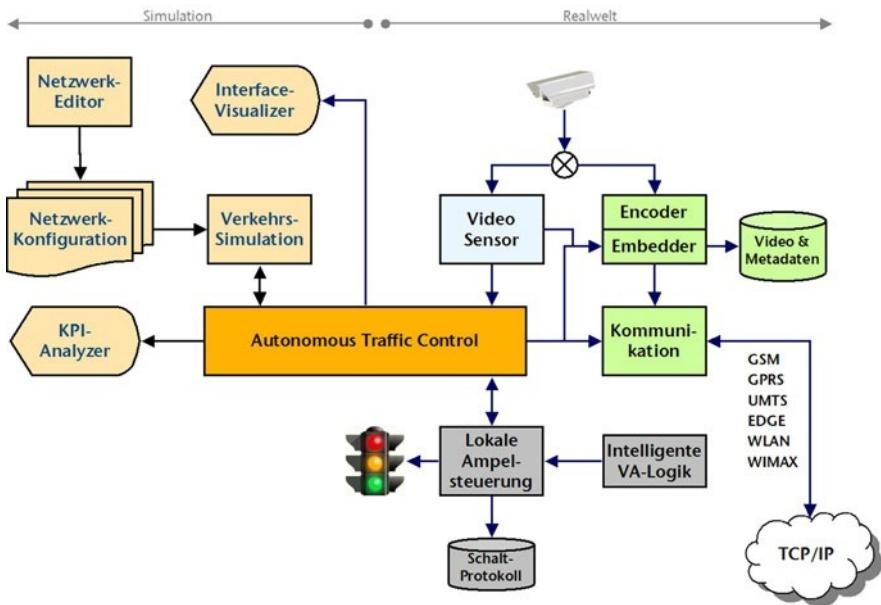


Abb. 9.3 Systemkomponenten

9.4.2 Wie die Agenten arbeiten

Das Herzstück eines Kreuzungsagenten ist seine Nutzenfunktion, die jede Schaltphase bewertet und dabei Parameter mit einbezieht, wie wartende Fahrzeuge, herannahende Fahrzeuge, Wartezeit und Abstand zur Kreuzung. *Anmerkung:* Eine Schaltphase ist der Gesamtzustand aller Leuchten an einer Kreuzung, die i. d. R. dazu führen, dass eine bestimmte Fahrspur freigegeben ist. Bei nicht kreuzendem Verkehr können auch mal zwei Spuren freigegeben sein.

Der Nutzen für jede mögliche Schaltphase an der betrachtenden Kreuzung wird fortlaufend berechnet. Sobald eine Schaltphase die gerade freigegebene Schaltphase um einen gewissen Schwellwert überschreitet, leitet der Kreuzungsagent die Um schaltung ein. Die Nutzenfunktion unterscheidet auch zwischen der Bewertung der aktiven Phase und allen anderen Phasen, um damit u. A. den noch rollenden Verkehr höher zu bewerten und damit deren Anhalten zu verhindern versuchen.

9.4.3 System-Komponenten

Der Aufbau des Gesamtsystems ist in Abb. 9.3 dargestellt. Die agentenbasierte Steuerung in der Mitte Autonomic Traffic Control (ATC) ist entweder eingebunden in eine Simulationsumgebung (links) oder eine Realweltumgebung (rechts).

Wesentlich für die Agentenbetrachtung ist die Steuerung in der Mitte. Dennoch soll hier das Umfeld kurz beschrieben werden, damit deutlich wird, wie der Wechsel zwischen Test, Entwicklung bzw. Simulation zur realen Welt funktioniert.

Die Simulationsumgebung besteht aus einem Editor, der die Erstellung der Simulationsumgebung als Abbild der Realwelt erleichtert. Zur Verkehrssimulation dient ein freies Tool, das wir um Sensor-Simulationen und Aktoren erweitert und mit der ATC verbunden haben. Der Interface-Visualizer zeigt den Verkehrsfluss in allen Details während einer Simulation. Der KPI-Analyser dient schließlich der Auswertung von Simulationsläufen.

Die Realwelt hält mehrere physikalische Komponenten bereit. im unteren Teil dargestellt ist die (herkömmliche) Ampelsteuerung, die Vorschläge für Grün-Phasen entgegen nimmt, sicherstellt, dass keine konkurrierenden Fahrspuren gleichzeitig grün bekommen und das Ganze zum Nachweis auch in einem Schaltprotokoll festhält. Teilweise existiert auch eine verkehrsabhängige (VA)-Logik. Darüber sorgt ein Videosensor und entsprechende Encoder für die Erkennung der Verkehrsströme und deren Aufzeichnung inklusive Metadaten. Die Kommunikationseinheit kann sorgt für die Abstimmung zwischen benachbarten Kreuzungen, d. h. zum Datenaustausch zwischen den entsprechenden Kreuzungsagenten.

9.4.4 In drei Schritten zum Ergebnis

Zur Einrichtung einer Straßenkonfiguration bzw. zur Durchführung einer Simulation sind drei Schritte notwendig, die jeweils durch ein separates Tool unterstützt werden:

EDIT > RUN > ANALYZE

1. EDIT zum schnellen Entwurf eines Straßenmodells mit Szenarien in XML (Abb. 9.4). Ein Straßenmodell ist eingeteilt in Straßennetz (Kreuzungsknoten und Straßenverbindungen) und Kreuzungsdesign (Spuren und erlaubte Abbiegrichtungen). Es gibt auch einfache Rechts-Vor-Links-Knoten, die nicht weiter geregelt werden. Auch Spurergänzungen und Spurreduktionen sind möglich, um z. B. die oft üblichen Zusatzabbiegespuren kurz vor einer Kreuzung abzubilden. Ebenso müssen zur Modellierung sogenannte Einspeise-Knoten definiert werden, von denen der Verkehr ausgesendet werden soll. Für jedes erzeugte (simulierte) Fahrzeug wird festgelegt, zu welchem Knoten es fahren soll.
2. RUN zur Generierung der Simulations-Konfiguration und Durchführung der Simulation. Die Konfiguration umfasst sämtliche Simulationsparameter, wie z. B. Art der Ampelsteuerung, Verkehrsichte, Laufzeit, und viele mehr. Zur Simulation selbst reicht der Server aus, der das Ergebnis in ein Logfile schreibt. Dies macht z. B. Sinn, wenn man sehr viele Szenarien und Parametrisierungen im Batch (über Nacht) laufen lassen möchte und dann offline die Ergebnisse analysiert. Zur Darstellung des Simulationsablaufs lässt sich aber auch die Zuschaltung der Verkehrsvisualisierung möglich, die den Verkehrsfluss, die Fahrzeuge

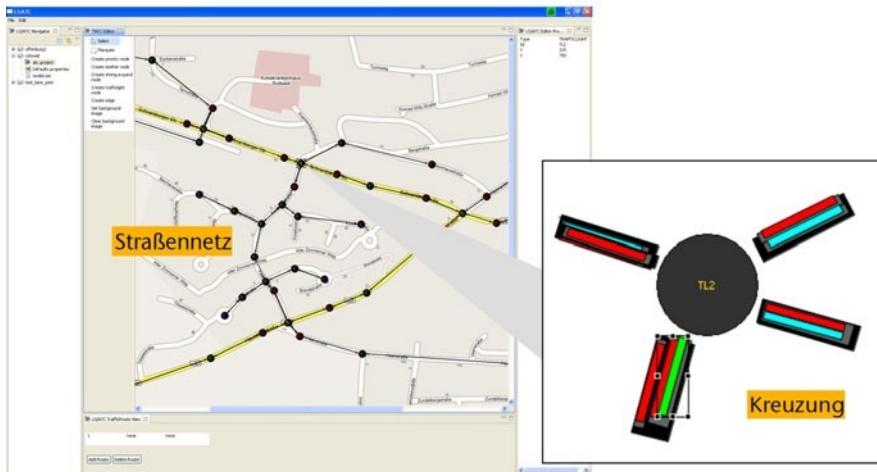


Abb. 9.4 EDIT-Tool

und auch die Sensorzustände sowie Ampelanzeigen in Echtzeit darstellt. Ebenso kann der Echtzeit-Analysemonitor zugeschaltet werden, womit man die Entwicklung der KPIs ebenso in Echtzeit verfolgen kann.

3. ANALYZE zur Auswertung und Analyse der Simulations-Ergebnisse, sowie dem Verhalten der Steuerungen. Die folgenden beiden Abbildungen zeigen die eigentliche Analyse-Darstellung (Abb. 9.5) und die Graphen zur Nutzen-Auswertung (Abb. 9.6).

9.5 Evaluierung/Erfahrung und Nutzen

In den folgenden Abschnitten gehen wir auf den Aufbau der Testumgebung ein und stellen diskutieren unsere Erkenntnisse anhand eines Beispiels aus der realen Welt.

Simulationsumgebung

Zur Verkehrssimulation setzten wir SUMO (Simulation of Urban MObility) ein. Dies ist ein frei zugängliches, kostenloses Verkehrssimulationssystem, das wir um folgende Erweiterungen ergänzt haben:

- Simulation der Kamerasensoren (Induktionsschleifen waren bereits enthalten)
- Aufzeichnung der, für die Analyse benötigten, KPIs
- Kommunikation mit der Agentenplattform bzgl. Sensoren und Aktoren (Ampeln)



Abb. 9.5 ANALYZE-Tool Statistiken und Auswertung

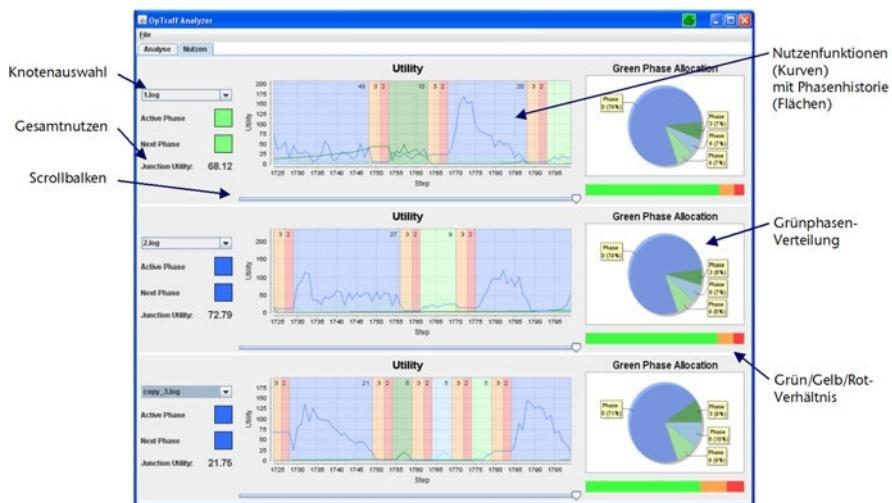


Abb. 9.6 ANALYZE-Tool Nutzen-Darstellungen

Ziele und Zielfunktion (Bewertungskriterien)

Unser einleitend beschriebenes Ziel war es, jeden Verkehrsteilnehmer schnellstmöglich schnell passieren zu lassen, also Anstauungen zu verhindern und damit Wartezeiten zu minimieren. Anrollenden Verkehr sollte frühzeitig erkannt werden und möglichst eine Durchfahrt ohne Anhalten ermöglichen. Um die Unterschie-

de zwischen den verschiedenen Steuerungsprinzipien und Parametereinstellungen festzuhalten, haben wir folgende Bewertungskriterien definiert:

- Anzahl der im Gesamtsystem wartenden Verkehrsteilnehmer
- Mittlere und maximale Wartezeit
- Anzahl Stopps aller Fahrzeuge
- Durchsatz: Anzahl Fahrzeuge, die pro Stunde durch das Verkehrsnetz fließen

Alle vier Kriterien bzw. Messgrößen werden jeweils als aktuelle Zahl (Snapshot) und als absolute Zahl (Summe) betrachtet.

Nach der Bestimmung der Messgrößen galt es aus der Vielzahl von Simulationsmöglichkeiten auszuwählen. Aus der Kombination von verschiedenen Szenarien (Straßenkonfigurationen, Verkehrsichte, Verkehrsfluss) mit unterschiedlichen Steuerungsstrategien ergeben sich unzählige Vergleichsmöglichkeiten.

Betrachtete Szenarien

Zum Test der Steuerungsstrategien haben wir folgende Szenarien definiert, die auch in Abb. 9.7 dargestellt sind:

1. Eine einzelne Kreuzung als einfachsten Fall, die aber durch unterschiedlich viele Fahrspuren auch unterschiedlich komplex sein kann.
2. Zwei Kreuzungen mit gleichen Spuranzahlen
3. Ein Gitternetz aus 16 (4×4) Kreuzungen
4. Eine Hauptverkehrsader mit Nebenstraßen
5. Reale Welt (Beispiel aus einer Stadt)

Betrachtete Steuerungen

Nach den Verkehrs-Szenarien, von denen unendlich viele möglich sind, kann man sich bei den Steuerungen eher begrenzen. Wir haben uns auf zwei herkömmliche und drei neue Steuerungs-Strategien beschränkt.

Bisherig übliche Steuerungslogiken

Cyclic Dies ist die Standard-Ampelsteuerung mit festen Phasen ohne äußeren Einfluss durch Sensoren. Diese Steuerungsart ist im Simulationstool SUMO standardmäßig vorhanden.

Induct Loop Hier handelt es sich um die allseits bekannte verkehrsabhängige Steuerung mit Induktionsschleifen als Sensoren. Im Rahmen des Projektes wurde die entsprechende Logik in SUMO nachgebildet.

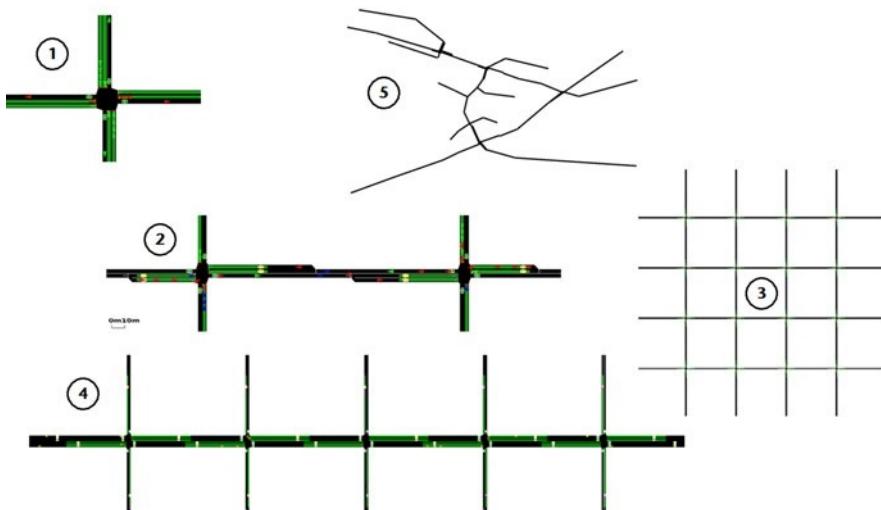


Abb. 9.7 Verkehrsnetz-Szenarien

Neuartige Steuerungslogiken

Single Agent Ein lokal agierender adaptiver Softwareagent pro Kreuzung, der auf Basis einer Nutzenfunktion die Ampelphasen schaltet. Mit Single Agent ist ein autonom agierender Agent gemeint, der nur eine Kreuzung mit den lokal vorhandenen Sensoren steuert, ohne mit anderen Kreuzungen (also deren Softwareagenten) zu kommunizieren.

Cooperative Agent Basierend auf dem Single Agent teilen die Kreuzungsagenten bei dieser Variante den Nachbarkreuzungen den abfahrenden Verkehr mit. Damit hat der empfangende Agent zusätzliche Sensorinformation, er hat sozusagen einen „verlängerten Arm“.

Forecast Agent Diese Variante geht noch eine Stufe weiter und sammelt Daten, um daraus Prognosen ableiten zu können und diese ebenfalls den Nachbarkreuzungen weiterzugeben.

Paarweise Vergleiche

Um die verschiedenen Logiken (logic) gegeneinander „antreten“ zu lassen, wurden Paare gebildet und diese dann in unterschiedlichen Szenarien (network) mit unterschiedlichen Verkehrsdichten (density) in der Simulation im direkten Vergleich gestartet (Abb. 9.8). Die Simulationsumgebung wurde so aufgebaut, dass sie jeweils

Abb. 9.8 Paarweise Vergleiche

<i>logic \ network</i>	single	double	major	grid 4x4	real
cyclic	●				
induct loop	●, ●	●, ●, ●	●, ●, ●	●, ●, ●	●, ●, ●
single agent	●	●, ●	●, ●	●, ●	●, ●
cooperative agent			●, ●, ●	●, ●	●, ●
forecasting agent			●, ●, ●	●, ●	●, ●
cyclic green wave			●		
<i>density</i>	L,M,H	L,M,H	L,M,H	random	L,M,H

2 Szenarien gleichzeitig ablaufen lassen kann, um damit auch visuell möglichst sofort den unterschiedlichen „Erfolg“ der Steuerungslogiken erkennbar zu machen.

Simulationsläufe und deren Ergebnisse

Hier einige im Projekt untersuchte Simulations-Szenarien und eine kurze Erläuterung der Ergebnisse:

- Eine Kreuzung: Festzeit gegen Induktionsschleife mit Verkehrsschüben, um die herkömmlichen Steuerungen im Vergleich zu sehen.
Fazit: Die Anzahl Fahrzeuge im Netzwerk zeigten, dass die Schübe von der Induktions-Logik deutlich besser abgebaut werden als von der Festzeit-Steuerung – was zu erwarten war. Verbesserungen: Kraftstoffverbrauch: 49 %, Wartezeit: 72 %, Durchsatz: 41 %
- Eine Kreuzung: Induktionsschleife gegen Einzel-Agentensteuerung mit Verkehrsschüben, um die übliche adaptive Steuerung als Sparringspartner für die Softwareagenten auszuwerten.
Fazit: Neben dem schnelleren Abbau der Schübe bei der Agentensteuerung ist in den Gesamtkennzahlen auch deutlich eine Verbesserung der anderen KPIs erkennbar. Verbesserungen: Kraftstoffverbrauch: 32 %, Wartezeit: 43 %, Durchsatz: 14 %
- Zwei Kreuzungen: Induktionsschleife gegen kooperative Agentensteuerung mit asymmetrischem Verkehr, um noch eine Stufe der Kooperation dazu zu schalten.
Fazit: Die Anzahl der wartenden Fahrzeuge ist mit der kooperativen Logik durchgehend deutlich geringer. Die Induktions-Logik passt sich zwar an den Verkehr an, doch die beiden Kreuzungen arbeiten in keiner Weise zusammen. Die kooperative Logik koordiniert jedoch die beiden Kreuzungsknoten und kann so dem West-Ost Verkehr eine kleine grüne Welle schaffen. Verbesserungen: Kraftstoffverbrauch: 44 %, Wartezeit: 54 %, Durchsatz: 26 %
- Haupteinfallstraße: Feste grüne Welle gegen kooperative Agentensteuerung mit Hauptverkehr in eine Richtung (Berufsverkehr/Pendler).
Fazit: Die nahe bei einander liegenden Linien aller Kennzahlen zeigen, dass die manuell genau auf den Verkehrsfluss eingestellte Grüne Welle kaum mehr zu

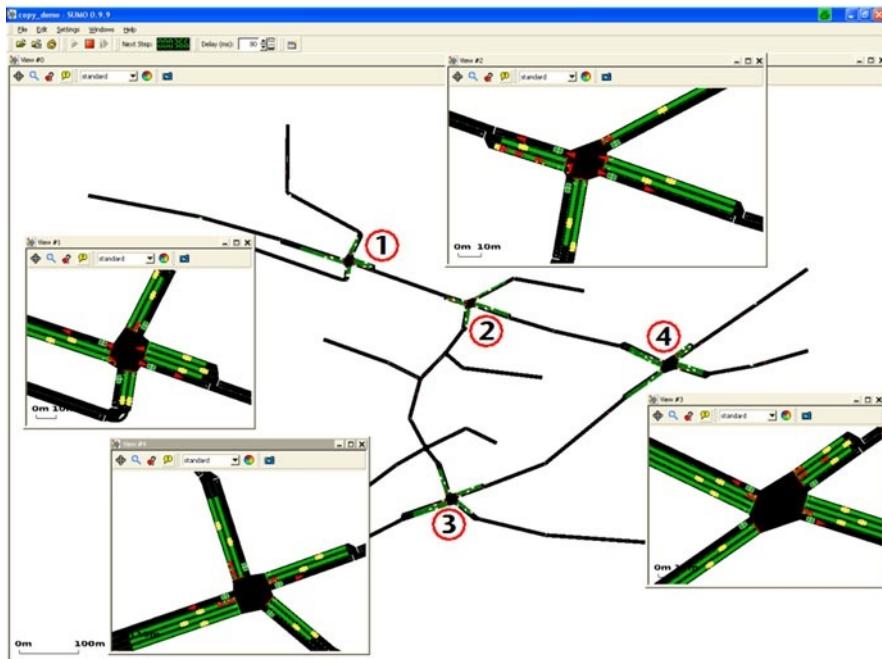


Abb. 9.9 Konfiguriertes Szenario auf Basis eines realen Straßennetzes

verbessern ist. Dennoch sind die Werte der agentenbasierten Steuerung in allen Bereichen geringfügig besser, und das ohne dass diese speziell auf das Szenario eingestellt wurde. Dies liegt im Wesentlichen daran, dass neben der automatisch entstehenden grünen Welle auch adaptiv auf den Verkehr aus den Nebenstraßen reagiert wird. Bei nur leichten Verkehrsflussabweichungen passt sich die Agentensteuerung automatisch an, wo hingegen die manuell optimierte grüne Welle deutlich ineffektiver wird. Verbesserungen: Kraftstoffverbrauch: 9 %, Wartezeit: 15 %, Durchsatz: 4 %

- 4 × 4-Netz: Kooperative Agentensteuerung gegen Induktionsschleife mit zufällig generiertem Verkehr, um der realen Welt etwas näher zu kommen.

Fazit: Gerade in diesem Netz ist eine verkehrsflussabhängige Logik unabkömmlich, doch diese kann noch weiter verbessert werden, wenn zusätzlich eine Koordination zwischen den einzelnen Kreuzungen stattfindet. Die Kennzahlen weisen hohe Schwankungen auf, wobei die agentenbasierte Steuerung zu jeder Zeit besser reagiert. Verbesserungen: Kraftstoffverbrauch: 16 %, Wartezeit: 35 %, Durchsatz: 10 %

- Real-Beispiel (Abb. 9.9): Kooperative Agentensteuerung gegen Induktionschleife mit der Realität nachempfundenen Verkehrsschüben – unterschiedlich zu verschiedenen Tageszeiten.

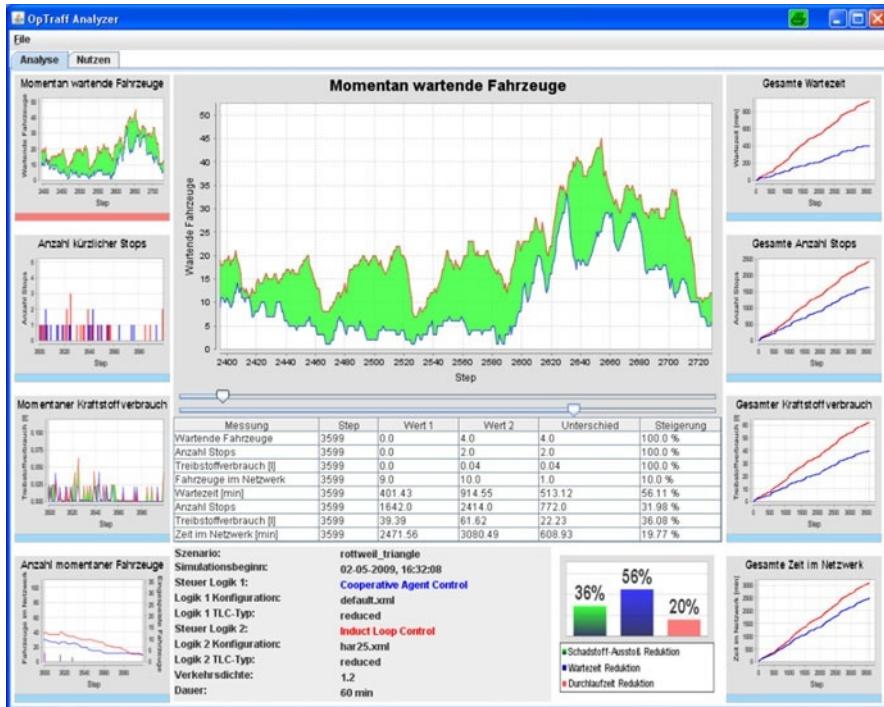


Abb. 9.10 Analyse der Simulation des Realwelt-Beispiels

Fazit (Abb. 9.10): Aus der grünen Differenzfläche (Mitte) kann man deutlich erkennen, dass die agentenbasierte Steuerung durchweg weniger wartende Fahrzeuge hat. Auch in den Graphen der Gesamtkennzahlen (Rechts) ist neben der Wartezeit auch eine deutliche Minderung der Anhaltevorgänge zu erkennen. Daraus resultiert auch die hohe Kraftstoff-Einsparung. Verbesserungen: Kraftstoffverbrauch: 36 %, Wartezeit: 56 %, Durchsatz: 20 %

9.6 Zusammenfassung

Die gemessenen Ergebnisse, wie auch die Ansicht von Praktikern zeigen, dass die neue agentenbasierte Verkehrsflusssteuerung einen gewaltigen Fortschritt bringen kann.

Die adaptiven Steuerungslogiken zeigen zusammengefasst betrachtet folgende Einsparpotenziale:

- Reduktion der Verweildauer im Netz (Erhöhung des Durchsatzes) um 10–20 %
- Reduktion des Kraftstoffverbrauchs um 20–40 %
- Reduktion der Wartezeit um 30–50 %

Darüber hinaus ergeben sich diese weiteren Vorteile:

- Grüne Wellen entstehen automatisch durch die Priorisierung von anrollenden Fahrzeuggruppen.
- Sofortige Anpassung an jede Verkehrsflusssituation ohne manuellen Eingriff.
- Der sehr hohe Planungs- und Implementierungs-Aufwand für jede Änderung an der Ampelsteuerung entfällt.

Kapitel 10

Einsatz von Softwareagenten am Beispiel einer kontinuierlichen, hydraulischen Heizpresse

Andreas Wannagat, Daniel Schütz und Birgit Vogel-Heuser

Zusammenfassung Dieser Beitrag behandelt den Einsatz von Softwareagenten auf der Feldebene zur Steuerung eines Produktionssystems unter Echtzeitbedingungen. Als Fallstudie dient eine kontinuierliche, hydraulische Heizpresse, die neben den hohen zeitlichen Anforderungen eine Reihe interessanter Anforderungen bezüglich des technischen Prozesses stellt, die eine flexible Prozessführung mit Hilfe der Agenten erlaubt. Im Gegensatz zu typischen fertigungstechnischen Anwendungsfällen für Agenten, liegt das Flexibilitätspotenzial weniger in einer flexiblen Materialflusssteuerung oder der Flexibilität seiner modularen Einheiten, sondern in der Adaptivität der Steuerstrategie, um die Produktqualität zu steigern und um Ausfälle zu kompensieren¹

10.1 Einleitung

Bei der kontinuierlichen Hydraulikpresse (Abb. 10.1) handelt es sich um eine Anlage der Holzfaserindustrie, welche ein Holz- und Leimgemisch unter Hitze und hohem Druck zu MDF-, Span- oder OSB-Platten presst [1]. Das Gemisch wird zu Beginn der Presse auf ein Band gestreut und dann zwischen zwei Stahlbändern durch die bis zu 80 m lange Presse geführt. Entlang der Presse sind bis zu

¹ Die Kurzzusammenfassung sowie die folgenden Abschnitte stammen aus der Dissertation „Entwicklung und Evaluation agentenorientierter Automatisierungssysteme zur Erhöhung der Flexibilität und Zuverlässigkeit von Produktionsanlagen“ von Andreas Wannagat, Technische Universität München, 2010.

Andreas Wannagat (✉)

Industry Sector, Siemens AG, Karl-Legien-Straße 190, 53117 Bonn, Deutschland

e-mail: andreas.wannagat@siemens.com

Daniel Schütz, Birgit Vogel-Heuser

Automatisierungstechnik und Informationssysteme, Technische Universität München,

Bolzmannstraße 15, 85748 Garching, Deutschland

e-mail: schuetz@ais.mw.tum.de, vogel-heuser@ais.mw.tum.de

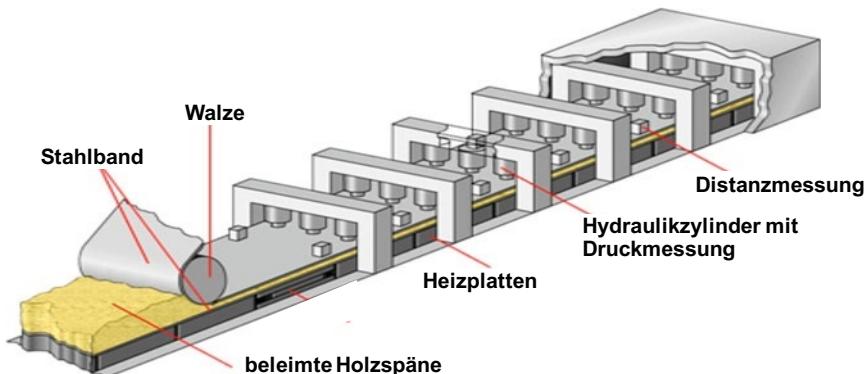


Abb. 10.1 Schematische Zeichnung einer kontinuierlichen Heizpresse

75 Pressrahmen positioniert, welche mit mehreren hydraulischen Druckzylindern in den entsprechenden Zonen der Presse den geforderten Druck einstellen. Jeder der Rahmen innerhalb der Presse ist sowohl mit Drucksensoren und -ventilen als auch mit Distanzsensoren ausgerüstet, die über einen Feldbus mit den Steuerungen verbunden sind.

Die Regelungsaufgabe der Hydraulikzylinder besteht darin, einen Solldruck und eine Solldistanz einzustellen. Jeder Hydraulikzylinder ist mit einem Drucksensor und einem Druckventil ausgerüstet und stellt einen Regelkreis dar. Je nach Typ und Position innerhalb der Presse, sind die mittleren drei der fünf Zylinder eines Rahmens hydraulisch verbunden, so dass die Steuerung praktisch nur drei Regelkreise (rechts, Mitte, links) kontrolliert. Die Dicke der Platte wird über zwei Distanzsensoren erfasst, die lateral positioniert sind. Pro Rahmen sind folglich mindestens fünf analoge Sensoren und drei Hydraulikventile zusammengefasst, welche in einem Intervall von 10 ms gemessen bzw. gestellt werden. Diese Prozessgrößen werden über Buskoppler und Feldbusse an mehrere Steuerungen weitergeleitet. Die Distanzen und Drücke entlang der Presse werden durch ein überlagertes Sicherheitssystem geprüft, um eine maximale Spannung bzw. plastische Verformungen der Heizplatte der Presse zu vermeiden.

Das Ausgangsmaterial für die Herstellung von MDF-Platten sind Holz-Hackschnitzel, welche nach mehreren reinigenden Maßnahmen, gekocht und zerfasert werden. Die Fasern werden anschließend auf etwa 10 % Feuchte getrocknet, mit Leim versehen und auf ein Laufband gestreut. Es entsteht ein Faserteppich, der noch verdichtet wird und das Ausgangsmaterial für die hier betrachtete Presse bildet [2]. In der Presse wird das Material auf die gewünschte Plattenstärke gepresst, wobei es zur Vernetzungsreaktion des Klebstoffs kommt, der für die Qualität des Produkts entscheidend ist und neben den Materialeigenschaften wesentlich von der Temperatur, dem Druck und der Feuchte abhängt. Innerhalb der Presse herrschen Drücke von bis zu 450 N/cm^2 und Temperaturen von etwa 200–240 °C [3]. Das Pressprogramm gibt die Distanzen (Materialdicke), die Temperatur und den Druck an den

entsprechenden Positionen innerhalb der Presse vor. Neben den Vorgaben, die sich auf zeitliche Abfolge der gewünschten Prozesswerte beziehen (längs der Presse), ergeben sich auch Anforderungen quer zur Materialflussrichtung. Der Dampfdruck, welcher durch die hohen Temperaturen und Drücke innerhalb des Materials entsteht, muss seitlich aus dem Material geleitet werden. Dies wird u. a. gewährleistet, indem die äußeren Zylinder mit einem geringeren Druck als die zentralen Zylinder beaufschlagt werden.

Ungeplante Stillstände sind für eine kontinuierliche Heizpresse wirtschaftlich, technologisch und auch sicherheitstechnisch von besonderer Bedeutung. Wirtschaftlich gesehen führt ein Stillstand zu einer Unterbrechung der Produktion und folglich zu einem Produktionsausfall. Darüber hinaus sind die Unterbrechung der Produktion und der anschließende Wiederanlauf mit zusätzlichem Ausschussmaterial verbunden, welches wieder aufbereitet oder entsorgt werden muss. Aus sicherheitstechnischer Sicht können ungeplante Stillstände Risiken für Mensch und Maschine darstellen. Wird die Presse im gefüllten Zustand angehalten, muss das Material aufgrund der hohen Temperaturen abkühlen, bevor die Presse gefahrlos geöffnet werden kann. Andernfalls besteht die Gefahr, dass sich das Material bei Sauerstoffzufuhr entzündet. Um diese Stillstände zu vermeiden, verfügt die Presse über ein Konzept, mit dem der Ausfall einzelner Sensoren durch die Deaktivierung der betroffenen Regelkreise kompensiert werden kann, indem der zuletzt eingestellte Stellwert beibehalten wird. Die Presse wird leergefahren und erst dann gestoppt. Der Einsatz redundanter Sensoren, um die Produktion aufrecht erhalten zu können, wird aufgrund der räumlichen Enge, der extremen Bedingungen innerhalb der Presse, der Vielzahl an Sensoren und der damit verbundenen Kosten in der Praxis nicht verfolgt.

Aufgrund ihrer Bedeutung für die Verfügbarkeit der Anlage, ist eine wesentliche Anforderung an eine flexible Steuerstrategie, den Ausfall von Sensoren zu kompensieren. Für ausgefallene Messwerte könnten dabei aus prozessbedingten Abhängigkeiten Ersatzwerte berechnet und so ein Notbetrieb ermöglicht werden. Die Entscheidung, ob mit einem gegebenenfalls unpräziserem Ersatzwert der Betrieb aufrechterhalten werden kann, ist dabei abhängig von der Anordnung der insgesamt ersetzen Sensorsysteme und der gefahrenen Produkte bzw. Drücke und Distanzen. Weitere Faktoren für die Entscheidung sind die Messgenauigkeit und die Zuverlässigkeit eines virtuellen Sensors in den letzten Stunden sowie die Anzahl der Systeme, denen der Sensor zugeordnet ist. Eine entsprechende flexible Strategie soll daher zur Laufzeit automatisch und situationsabhängig durch die Steuerungssoftware entwickelt werden und insbesondere die Konsequenzen der eigenen Maßnahmen abschätzen und beurteilen können.

10.2 Struktur des Agentensystems

Das vorgeschlagene Agentensystem unterteilt sich grundlegend in Agenten des technischen Systems (Steuerungsagenten und Systemagent), Agenten des techni-

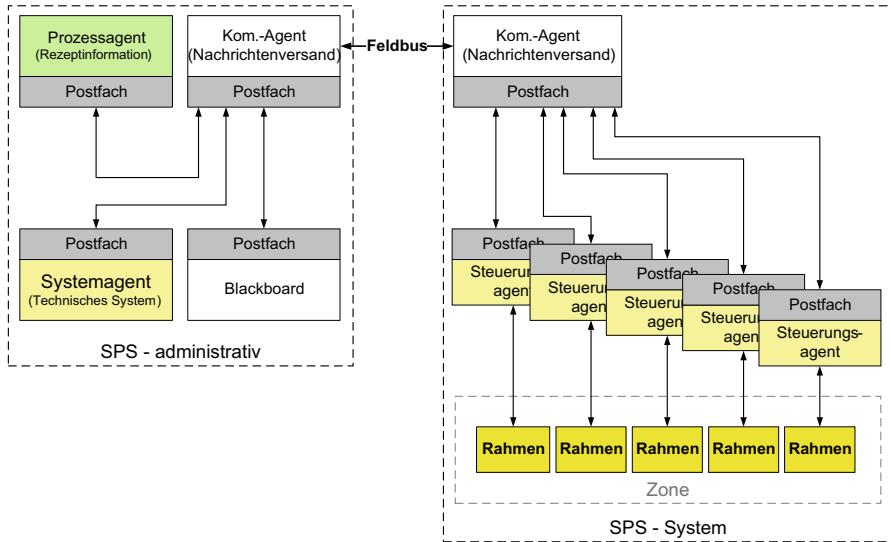


Abb. 10.2 Struktur eines Agentensystems

schen Prozesses (Prozessagent) und Agenten, die für die Kommunikation zwischen verschiedenen Steuerungen verwendet werden (Abb. 10.2).

Die Steuerungsagenten vertreten jeweils einen Pressenrahmen der Presse und setzen die Vorgaben an den Druck, die Materialdicke (Distanz) und die Temperatur lokal um. Der innere Aufbau eines solchen Agenten ist modular und kapselt neben der Steuerungsaufgabe, Diagnoseaufgaben (Fehlererkennung und Behandlung), das Wissen über den Teilprozess (Modell, Regeln, Randbedingungen) und Strategien, wie die lokalen Ziele zu erreichen sind. Die lokalen Vorgaben bezüglich der Sollvorgaben für jeden einzelnen Agenten stellt ein Prozessagent bereit, der die Informationen über den technischen Prozess kapselt und diese zu Beginn auf Anfrage der Steuerungsagenten individuell bereitstellt.

Der Systemagent ist ein Agent des technischen Systems und nur einmal vorhanden. Er hält die Randbedingungen bereit, die über einen einzelnen Rahmen aus der Sicht des technischen Systems hinausgehen. Dazu gehören zum Beispiel der Antrieb der Stahlbänder innerhalb derer das Material durch die Presse transportiert wird, sowie die Regelung der Heizplatten, die sich jeweils über mehrere Rahmen ausdehnen. Gleichzeitig ist der Systemagent die zentrale Anlaufstelle für alle Fehlermeldungen innerhalb des Agentensystems. Der Systemagent verfügt folglich über umfassende Informationen bezüglich des aktuellen Zustands des technischen Systems und der entsprechenden globalen Anforderungen, welche es erlauben den aktuellen Zustand zu bewerten und zu kontrollieren.

Der Agententyp der Kommunikationsagenten übernimmt innerhalb des Agentensystems die Aufgabe, Nachrichten zwischen einzelnen Agenten weiterzuleiten. Die Implementierung dieser Aufgabe in Form eines Kommunikationsagenten ist

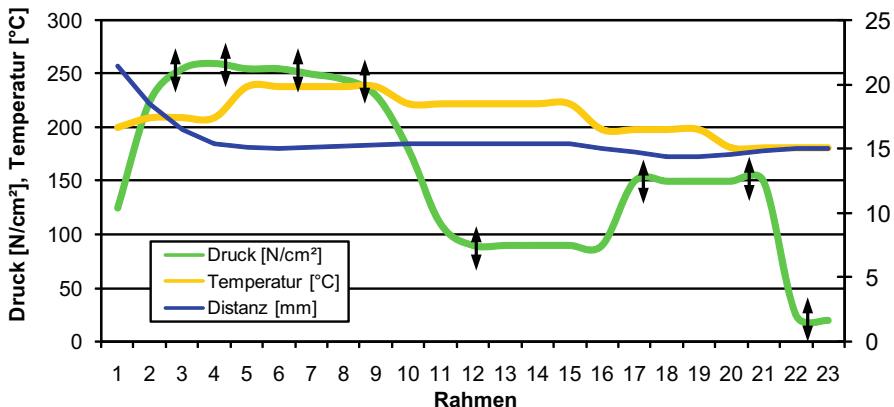


Abb. 10.3 Pressprogramm und Zuordnung der Sollwertvorgaben

spezifisch für die eingesetzte Steuerungsplattform (SPS) und wurde in [4] bereits ausführlich beschrieben. Im Folgenden werden die beiden Agententypen „Prozessagent“ und „Steuerungsagent“ sowie deren Realisierung für den konkreten Anwendungsfall der Hydraulikpresse näher beschrieben.

10.2.1 Prozessagent

Das Pressprogramm, in welchem die technologischen Informationen, wie eine Platte zu fertigen ist, enthalten sind, wird durch den Prozessagenten verwaltet und erlaubt es diesem, die Zuordnung zwischen einer Rahmenposition und den Sollvorgaben in Materialflussrichtung bezüglich der Temperatur, dem Druck und der Distanz vorzunehmen (Abb. 10.3).

Quer zur Materialflussrichtung ergeben sich weitere relativ dazu abhängige Daten, die insbesondere die Druckverteilung innerhalb eines Rahmens betreffen. So müssen zum Beispiel aufgrund des entstehenden Wasserdampfes innerhalb der Platte, die äußeren Drücke geringer sein als die Drücke der mittleren Zylinder, damit der Dampf an den Seiten entweichen kann. Wird an den Rändern der Presse zu stark gepresst, kann der Dampf nicht entweichen und es kommt zu Gaseinschlüssen. Diese führen zu einer unebenen Oberfläche oder im Extremfall zum Aufplatzen der Plattenoberfläche.

Sämtliche Vorgaben, die der Prozessagent vorhält, sind mit Toleranzen hinsichtlich der einzuhaltenden Sollgrößen versehen (in Abb. 10.3 exemplarisch für den Druck durch die senkrechten schwarzen Pfeile gekennzeichnet). Diese Toleranzen stellen den Handlungsspielraum dar, den die Steuerungsagenten der Rahmen für die lokale Regelung nutzen können. Im Allgemeinen gilt die Regel, dass alle Agenten sämtliche Prozesswerte lesen dürfen, Stellgrößen hingegen dürfen nur von dem

Tab. 10.1 Relative Randwertberechnung für Druck des mittleren Zylinders an Rahmen n

Hydraulik- zylinder	Randbe- dingung	Rahmen (n-2)	Rahmen (n-1)	Rahmen (n)	Rahmen (n+1)	Rahmen (n+2)
Zylinder 1	max	24	21	20	19	19
	aktuell	24	20	17	16	15
	min	21	20	16	14	12
Zylinder 2	max	25	22	20	19	18
	aktuell	24	21	18	17	16
	min	21	20	17	14	13
Zylinder 3	max	24	21	20	19	19
	aktuell	24	20	17	16	15
	min	21	20	16	14	13

Agenten geschrieben werden, dem sie fest zugeordnet sind. Der Prozessagent hat folglich die Möglichkeit, sämtliche Prozesswerte hinsichtlich seiner eigenen Vorgaben zu kontrollieren und unerlaubte Prozesszustände rahmenübergreifend zu erkennen.

10.2.2 Steuerungsagenten

Die Steuerungsagenten erhalten sämtliche Vorgaben des technischen Prozesses durch den Prozessagenten. Zusätzlich halten sie sämtliche Randbedingungen des zugeordneten Teils des technischen Systems (ein Pressrahmen) bereit und streben an, die Prozessvorgaben innerhalb der lokalen Randbedingungen zu erfüllen. Zu diesen Randbedingungen gehören absolute Parameterbegrenzungen der einzelnen Bauteile, wie zum Beispiel der maximale Druck oder ein erlaubter Temperaturbereich. Doch auch relative Randbedingungen, wie die mögliche Druck- und Temperaturverteilung innerhalb des Einflussbereichs, oder Differenzen bezüglich der einstellbaren Distanzen sind zu beachten. Letztere sind zum Beispiel durch maximale Verbiegung der Heizplatten begrenzt, die durch entsprechende Differenzen ($> 4 \text{ mm/m}$) in Quer- und Längsrichtung beschädigt werden können. Damit die Steuerungsagenten die Verbiegung der Heizplatte feststellen und entsprechende Vorgaben an ihre lokalen Distanzen berechnen können, haben die Steuerungsagenten eines Rahmens lesenden Zugriff auf die Distanzsensoren der benachbarten Rahmen (Tab. 10.1).

Das Ziel der Agenten zur Laufzeit besteht zunächst darin, die Vorgaben des Prozessagenten mit den lokalen Randbedingungen zu vereinbaren und einen optimalen Arbeitspunkt einzustellen, der möglichst nahe der Vorgabe ist. In dem Fall, dass Prozessvorgaben aufgrund der lokalen Bedingungen nicht einzuhalten sind, wird eine entsprechende Nachricht an den vorgelagerten Agenten versendet, damit ein Kompromiss geschlossen werden kann, der die Einhaltung der Randbedingungen

erlaubt. Kann zum Beispiel eine Distanz aufgrund einer lokalen Druckbeschränkung nicht eingestellt werden, so besteht eine mögliche Lösung der Situation darin, dass der vorgelagerte Rahmen seine lokale Solldistanz im Rahmen der Anforderungen erhöht. Die zu erwartende geringere Distanz am fraglichen Rahmen würde dann das Erreichen der geforderten Solldistanz ermöglichen. Weiterhin ist es auch denkbar, dass einige Agenten keine Vorgaben hinsichtlich der Distanz oder des Drucks erhalten und entsprechende Vorgaben über die Wünsche der benachbarten Agenten einstellen.

Die Wissensbasis für die Steuerungsagenten der Presse wurde in Teilen bereits publiziert [5], weshalb hier nur kurz die wesentlichen Informationen zusammengefasst werden. Die Wissensbasis der Steuerungssagenten enthält die analytischen Abhängigkeiten zwischen den Sensoren im Wirkbereich eines Agenten. Dazu gehören jeweils drei vor- und drei nachgelagerte Distanzsensoren (S), mit denen die Dicke der Matte gemessen werden kann, sowie drei Drucksensoren (P), welche den Druck messen, der auf das Material wirkt. Dem Modell liegt ein einfaches Federmodell zugrunde, wobei die Federkonstante (C) zur Laufzeit aus den Messdaten erlernt wird. Die mathematischen Beziehungen innerhalb des Modells erlauben die Berechnung von virtuellen Sensorwerten, die für die Detektion und die Kompensation von Fehlern genutzt werden können [4].

Die Modellierung der Abhängigkeiten basiert auf einem gerichteten Graphen, wobei die Knoten oval dargestellt sind und einzelne Messstellen repräsentieren (Abb. 10.4). Die Kanten sind gerichtet und beinhalten den mathematischen Zusammenhang zwischen zwei Knoten. Der Pfeil weist dabei von der Messstelle, für welche ein Modellwert errechnet werden kann (Ergebnis), auf die Daten, aus denen der Ersatzwert berechnet wird (Eingangsdaten). Für den Fall, dass mehrere Eingangsdaten für die Berechnung notwendig sind, werden diese in einem Hilfsknoten zusammengeführt, der durch einen kleinen schwarzen Punkt in der Darstellung symbolisiert ist. Jede Kante ist mit einem Qualitätsfaktor (q) versehen, der die Präzision der mathematischen Abbildung ausdrückt. Der fett gedruckte Pfeil in Abb. 10.4 symbolisiert folglich die Berechnung eines virtuellen Sensors für die Messstelle „S2_1“ auf Basis der Daten von Messstelle „S2_2“ und „S2_3“.

Die analytischen Abhängigkeiten basieren auf den Materialeigenschaften, welche durch die Federgleichung angenähert werden, wobei die Federkonstante (D) durch eine Materialkonstante (C) ersetzt wird. Die Differenz (s), um welche die Dicke des Materials während der Pressung an einem Element reduziert wird, entspricht der Stauchung einer Feder

$$F = C \cdot s .$$

Der Druck (p) des Hydraulikzylinders, welcher während des Pressens wirkt, wird entsprechend der Fläche (A) in eine Kraft umgerechnet, die auf das Materialelement wirkt

$$F = p \cdot A .$$

Die dargestellten analytischen Abhängigkeiten (Abb. 10.4) basieren entweder auf einer einfachen Federgleichung, die für die einzelnen Messstellen jeweils um-

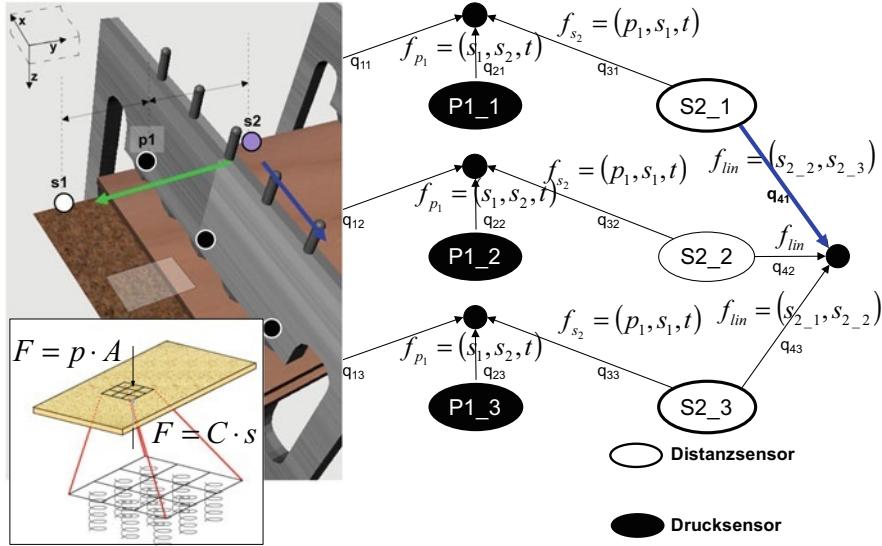


Abb. 10.4 Wissensbasis eines Steuerungsagenten für einen Pressrahmen

gestellt worden ist, oder auf einer linearen Interpolation der Werte benachbarter Messstellen. Für die Therme der Federgleichung ist zu berücksichtigen, dass zusätzlich ein zeitlicher Versatz (t_v) berücksichtigt werden muss, der die Zeit wiedergibt, welche der Transport des Materials von der Position der Distanzmessung bis hin zum Druckpunkt benötigt.

Für die Berechnung der nachgelagerten Distanzsensoren (S_2) wird entsprechend des Materialflusses eine Distanzmessung des eingehenden Materials (S_1) verwendet, die vor $2t_v$ Sekunden und eine Druckmessung (p_1), die vor t_v Sekunden aufgenommen worden ist

$$f_{S_2} = S_1(t - 2t_v) - \frac{p_1(t - t_v)}{C} .$$

In gleicher Weise werden auch die Berechnungen für die vorgelagerten Distanzsensoren vorgenommen

$$f_{S_1}(t - 2t_v) = S_2(t) + \frac{p_1(t - t_v)}{C} .$$

Als Ersatzwert zur Laufzeit kann dieser Zusammenhang nicht eingesetzt werden, da zukünftige Werte für die Berechnung genutzt werden müssten. Für Diagnosezwecke hingegen ist dieser Zusammenhang aber dennoch nützlich. Die Berechnung von Ersatzwerten kann aber durch den vorgelagerten Agenten übernommen werden, da es sich bei den betrachteten Sensoren um dessen nachgelagerte Distanzsensoren handelt. Die Berechnung des aktuell eingestellten Drucks (p_1) erfolgt in gleicher

Weise

$$f_{p_1}(t - t_v) = C \cdot (s_1(t - 2t_v) - s_2).$$

Der letzte verwendete Zusammenhang beruht auf einer linearen Interpolation auf der Basis benachbarter Sensoren, wobei der Druck bzw. die Distanz (einzusetzen für y) gegenüber der Positionen (x) quer zur Materialflussrichtung betrachtet wird

$$f_{lin}(x) = \frac{(y_2 - y_1)}{(x_2 - x_1)}x + y_1.$$

10.3 Realisierung und Umsetzung

10.3.1 Simulation der Presse

Zur Evaluation des Ansatzes wurde ein bestehendes Materialmodell [6] in eine kontinuierlich und in Echtzeit ablauffähige Simulation überführt. Die Simulation der Presse wurde in Matlab realisiert und umfasst das gesamte technische System, nimmt also Stellsignale für die Aktoren entgegen und liefert Messdaten der Sensoren zurück. Eine wesentliche Randbedingung für die Simulation ist, dass diese in Echtzeit ablauffähig ist und trotzdem eine ausreichende Präzision und Komplexität erreicht, um den Agenten hinsichtlich der Steuerungsstrategie genügend Freiräume für die Steuerung der Presse zu liefern.

Die Simulation unterscheidet zwischen der Simulation des technischen Prozesses, also dem Holz-Leim-Gemisch, welches durch die Einwirkung der Presse zu Spanplatten verpresst wird, und dem technischen System, das im Wesentlichen den mechanischen Aufbau der Presse umfasst. Die Simulation ist modular aufgebaut und orientiert sich in der Struktur an der des technischen Systems. Auf der obersten modularen Ebene (Abb. 10.5) sind die drei Zonen der Presse (Einlaufzone, Mittelzone, Auslaufzone) zu erkennen, links davon wird das Holz-Leim-Gemisch entsprechend der initialen Vorgaben erzeugt und in die erste Zone weitergeleitet.

Innerhalb der einzelnen Zonen sind drei bis fünf Pressrahmen enthalten, die jeweils mit den vorgegebenen Werten für den Druck auf die Heizplatte einwirken. Aufgrund der Steifigkeit der Heizplatte ergibt sich ein flächiges Druckprofil, auf dessen Basis sich der tatsächliche Druck für jedes der darunterliegenden Materialien errechnen lässt. Die Temperatur einer Heizplatte wird als konstant angenommen. Die Heizplatte bildet die direkte Schnittstelle zum Material, sowohl der Rollenteppich als auch die Stahlbänder sind in ihrer Wirkung nicht berücksichtigt.

Die Kraft, mit der die hydraulischen Zylinder auf die Heizplatte einwirken, kann über regelbare Proportionalventile beeinflusst werden. Der Simulation liegt dazu ein einfaches Modell eines Hydraulikzylinders zugrunde, mit welchem aus den Druck- und Flächenverhältnissen die Bewegung des Kolbens errechnet wird.

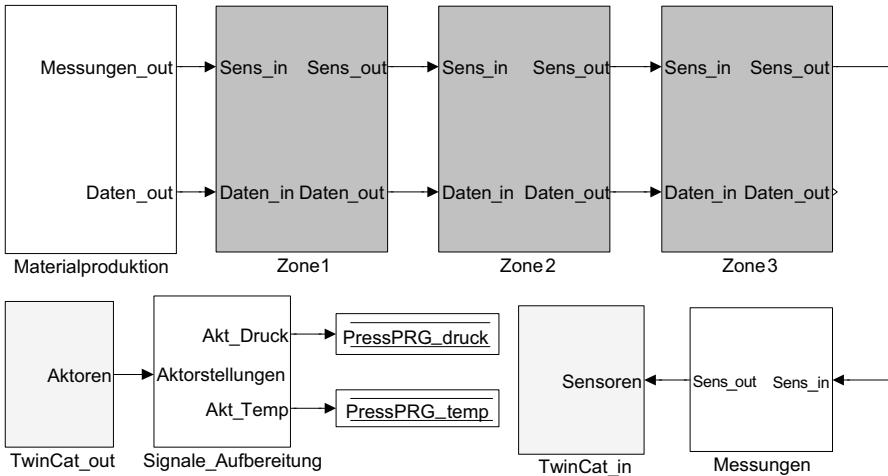


Abb. 10.5 Oberste Ebene des Matlab-Simulationsmodells

Die Simulation der Hydraulikventile und der Kolbenbewegung ist allerdings stark vereinfacht, so dass zwar die herrschenden Drücke berücksichtigt werden und die Kraft auf das Material berechnet wird, die Kolbenposition ergibt sich allerdings im bisherigen Modell aus der Dicke des Materials. Deutlich vereinfacht ist auch das Modell der eingesetzten Ventile. Das zeitliche Verhalten wird mit PT1-Strecken nachgebildet und der Druck hinter dem Ventil wird allein auf Basis der Öffnung und des Drucks vor dem Ventil im Hydrauliksystem beeinflusst. Eine weitere Detaillierung wurde zunächst aus Gründen der Performanz zurückgestellt, zudem hat dies keinen Einfluss auf die Steuerstrategie der Agenten. Neben den Aktoren (Druckventile) sind auch die Druck- und Distanzsensoren berücksichtigt.

Eine besondere Modellierung für die Sensoren gibt es nicht, sie leiten lediglich die entsprechenden Modellparameter an die Steuerung weiter. Die Drucksensoren nehmen dazu den Druck, der auf das Material einwirkt, direkt unterhalb des Zylinders auf. Die Distanzsensoren sind jeweils zwischen den Rahmen angebracht und liefern die dortige Materialdicke zurück. Die Sensoren können im Matlabmodell mit beliebigen Störsignalen beaufschlagt werden oder auch einzeln stillgelegt werden, um so einen Ausfall nachzubilden.

Die Grundlage für die Simulation des technischen Prozesses und insbesondere der Modellierung des Materials, welches die Presse passiert, bilden die Arbeiten von Thoemen, Humphrey, und Haselein [6] sowie Hasener [2], Hanvongjirawat [7] und Zombori [8]. Die Umsetzung wird im Folgenden beschrieben. Die Materialsimulation erstreckt sich in alle drei Dimensionen und umfasst sowohl die Wärme- und Stofftransportvorgänge als auch die Materialverdichtung beim Verpressen des Werkstoffes. Insbesondere die Konvektion des Wasserstoff-Luft-Gemisches, die Gasdiffusion der einzelnen Elemente sowie der Wärmetransport sind berücksichtigt [6]. Das Modell ist an mehreren Stellen deutlich vereinfacht worden. Dies ist im

Wesentlichen der Berechenbarkeit in Echtzeit geschuldet, allerdings besteht auch nicht der Anspruch, eine exakte Qualitätsvorhersage des produzierten Materials geben zu können, sondern viel mehr ein realistisches Verhalten des Materials im Wechselspiel mit dem technischen System und der Steuerung bzw. dem Agentensystem darzustellen.

Das Material wird räumlich nach dem Ansatz der finiten Elemente aufgeteilt. Dabei handelt es sich im Matlab-Modell um eine Serie von Matrizen, die in jedem Berechnungsschritt weiter durch die Presse geführt werden. Die letzte Matrix am Ende der Presse wird gelöscht, die anderen Matrizen rücken nach und der frei gewordene Platz zu Beginn wird durch eine neue Matrix ergänzt, die bei der Produktion des Materials erzeugt wird (Abb. 10.5, oben, links). Die Zahl der Elemente, sowohl innerhalb der Matrix als auch die Zahl der Scheiben, ist beliebig festlegbar ohne das Modell in seiner Struktur verändern zu müssen. Folgende Vereinfachungen sind vorgenommen worden:

Die Matte wird in y -Richtung als symmetrisch angenommen, so dass nur die Elemente von der Deckschicht bis zur Materialmitte berechnet werden, die restlichen Elemente ergeben sich aus einer Spiegelung. Der Materialtransport im Inneren der Matte wird allein auf Basis des Wassertransports (dampfförmig und flüssig) betrachtet und ist nur durch die Druckverhältnisse getrieben. Im Vergleich dazu ist der Materialtransport auf Basis von Diffusion sehr gering [6] und wird ebenso wenig berücksichtigt wie der Materialfluss im Inneren der Matte in Transportrichtung.

In der Simulation werden die Elemente einer Fläche senkrecht zur Flussrichtung in einer gemeinsamen Matrix dargestellt. Sie besteht aus sieben Elementen parallel zur Materialoberfläche (x -Richtung) und drei Elementen senkrecht dazu (y -Richtung). Jeder Rahmen verfügt zudem über zehn dieser Matrizen, die nacheinander an den nachgelagerten Rahmen weitergeleitet werden. Damit ergeben sich für die Berechnung 210 ($7 \times 3 \times 10$) Elemente, die pro Rahmen innerhalb eines Berechnungsschritts berechnet werden müssen.

Das Simulationsmodell wurde anhand von aufgezeichneten Messungen aus einer realen Presse validiert, zudem konnte bezüglich der Materialsimulation auf die Daten von Hasener [2] und Hanvongjirawat [7] zurückgegriffen werden. Die Präzision der Simulation ist zwar für die Evaluation des Agentensystems zweitrangig, trotzdem sind die erreichte Genauigkeit und die Möglichkeit der Echtzeitsimulation sehr vielversprechend, wie ein Experte der Herstellerfirma bestätigte. Entscheidender ist allerdings, dass strategische Fehler oder ein schlechtes Regelverhalten der Agenten zu sichtbaren Konsequenzen führt und dass die Simulation in Echtzeit abläuft, damit das zeitliche Verhalten der Agenten untersucht werden kann.

10.3.2 Implementierung des Agentensystems

Für die Implementierung des Systems wurde die IEC-61131-3 konforme Programmumgebung eingesetzt (siehe Abb. 10.7). Der Aufbau des Steuerungsprogramms trennt die Softwarekomponenten des technischen Systems, des technischen Pro-

zess und des Agentensystems. Die Softwarekomponenten des technischen Systems umfassen die Presse mit dem Bandantrieb, die Heizplatten und ihre Rahmen. Die Rahmen selbst umfassen die Hydraulikzylinder samt der verbauten Sensoren und Aktoren. Alle Komponenten und ihre Varianten sind als Funktionsbausteine (FB) ausgeführt und lassen sich entsprechend ihres Vorkommens mehrfach instanzieren. Die FB der Sensoren und Aktoren stellen die Verbindung zu den Prozessgrößen dar und damit den Zugriff auf die reale Anlage bzw. im vorliegenden Fall auf die Simulation. In den Funktionsbausteinen werden bereits einfache Grundfunktionen realisiert, die später durch die Agenten nutzbar sind. Im Fall der Zylinder ist dies eine Druckregelung, welche die Öffnung des Druckventils entsprechend der Sollvorgabe und der Messgröße (Drucksensor) regelt. Dies kann für jede Zylindervariante separat vorgenommen werden, so dass die Agenten, unabhängig von der vorliegenden Variante, über eine Sollvorgabe einheitlich den Druck einstellen können.

Die Agenten werden ebenfalls als Funktionsbausteine ausgeführt, wobei die Steuerungsagenten aller Rahmen identisch sind. Hinzu kommen ein Prozessagent, und ein Kommunikationsagent für jede eingesetzte Steuerung. Alle Agententypen instanzieren jeweils Standardbausteine, die auch als FB ausgeführt sind, wie ein Postfach. Zusätzlich verfügen die Steuerungsagenten der Rahmen über ein Diagnosemodul, ein Strategiemodul und ein Optionsmodul. Innerhalb des Diagnosemoduls ist eine Matrix hinterlegt, welche das Wissen über die im System verfügbaren Abhängigkeiten enthält und in der Modellierung als Graph abgebildet worden ist (Abb. 10.4).

Ein Optionsmodul der Steuerungsagenten enthält die aktuellen Randbedingungen für die Parameter, die sich im Einflussbereich des jeweiligen Steuerungsagenten befinden. Sämtliche Parameter sind mit entsprechenden Arbeitsbereichen versehen, die sich nach der Herkunft der Beschränkungen (technische System, technischer Prozess) unterscheiden. Abbildung 10.6 zeigt dies für die verschiedenen Randbedingungen bezüglich der Distanz (Ordinate), welche an den jeweiligen Rahmen (Abszisse) gelten. Die dunkelgrüne Linie symbolisiert dabei die Zielvorgabe an die Distanz, die graue Linie den gemessenen Wert für die Dicke (interpoliert). Die hellgrünen Bereiche markieren den Arbeitsbereich, der durch die Randbedingungen des technischen Prozess begrenzt ist und innerhalb dessen eine Spanplatte mit ausreichender Qualität produziert wird. Nach außen hin schließen sich in rötlicher Färbung die Beschränkungen des technischen Systems an, deren Verletzung eine Gefahr für selbiges darstellen. Die Messwerte sind zudem mit einem Qualitätswert versehen, der die Präzision bei der Erfassung des Parameters wiedergibt. Die Randwerte werden innerhalb des Optionsmoduls zur Laufzeit mit den gemessenen Daten verglichen und auf Verletzungen hin überwacht. Gleichzeitig ergibt sich aus den Abständen des gemessenen Arbeitspunktes zusammen mit der aktuell verfügbaren Präzision, der Handlungsspielraum, welcher den Agenten für ihre Aktionen zur Verfügung steht. Dieser kann, wie in der Abb. 10.6 gezeigt, durch einen qualitativ schlechten Sensorwert stark eingeschränkt sein, obwohl sich die Randbedingungen selbst nicht verändert haben. Mit Hilfe des Qualitätswerts sind die Agenten in der

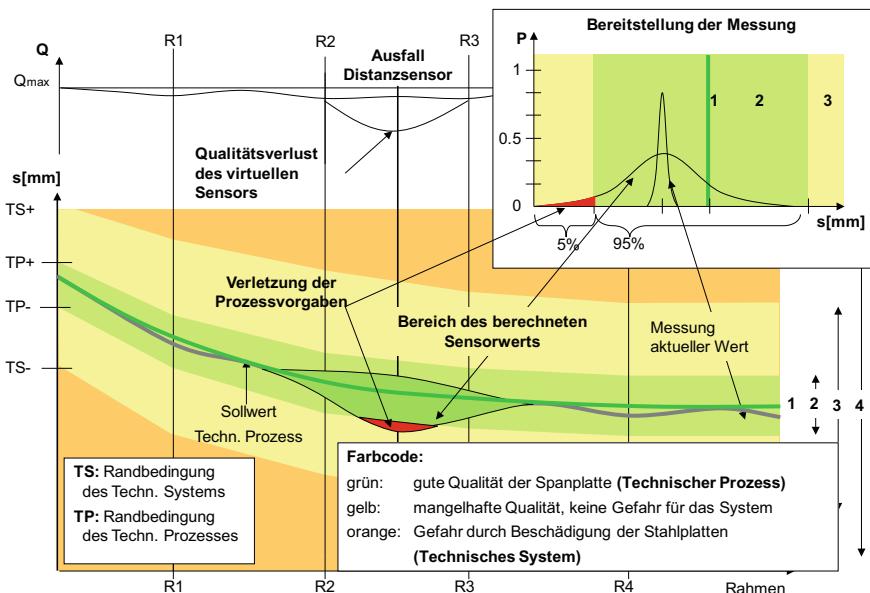


Abb. 10.6 Randbedingungen und Qualitätswerte für die Beurteilung des Betriebs

Lage, die Wahrscheinlichkeit für eine Verletzung der Randbedingung sowohl zu berechnen als auch durch Korrektur des Arbeitspunktes zu beeinflussen.

Das Strategiemodul legt die Interaktion mit anderen Agenten auf der Grundlage der lokalen Gegebenheiten fest. Die Agenten können mit ihren direkten Nachbarn in Verbindung treten, um einen optimalen Arbeitspunkt auszuhandeln. Die Agenten legen während der Verhandlung ihren Verhandlungsspielraum offen und verhalten sich kooperativ. Im Fall der Presse sind die Agenten bemüht ihre lokale Vorgabe durch den Prozessagenten (Druck und Distanz) zu erfüllen und genügend Abstand zu den definierten Randwerten zu halten. Treten sie mit anderen Agenten in Verhandlung, versuchen sie eine ähnliche Belastung auszuhandeln, indem sie ähnlich weit weg vom Optimum bzw. den Randwerten sind. Die Beschränkung auf eine Kommunikation mit den Nachbaragenten ergibt sich zum einen aus der Applikation, so dass eine Verhandlung über mehrere Rahmen hinweg nicht sinnvoll erscheint, und zum anderen reduziert sich die Kommunikationslast. Diese wird auch durch die Verwendung einer offenen Verhandlung reduziert, da für die Lösungsfindung keine iterative Annäherung an die Lösung notwendig ist, sondern diese direkt auf der Basis der vorgelegten Arbeitsbereiche festgelegt wird. Der wesentliche Grund ist aber in Hinsicht auf die Echtzeitanforderungen die definierte zeitliche Länge der Verhandlung, die sich daraus ergibt. Im Strategiemodul ist außerdem festgelegt, wie sich der Agent während des Anlaufens der Presse zu verhalten hat und unter welchen Umständen er mit einem benachbarten Agent interagieren soll. Die Agenten lassen zunächst das Material einlaufen und stellen den vorgegebenen Druck im Rah-

men der Randbedingungen ein, sobald das Material ihre Position erreicht hat. Der Druck wird dabei recht langsam gesteigert, da die Randbedingungen, welche eine zu starke Verbiegung der Heizplatten betreffen, die zulässigen Distanzunterschiede zum Nachbarrahmen beschränken. Erreicht ein Agent seinen Arbeitsbereich bezüglich des Drucks, variiert er diesen innerhalb dieses Arbeitsbereichs, um auch die vorgegebene Distanz zu erreichen. Erreicht er diese, nimmt er Kontakt mit seinem Nachfolger auf, um diesen dabei zu unterstützen dessen Ziele zu erreichen. Der Vorteil dieser Strategie hat sich in der Simulation bestätigt und liegt darin, dass Verhandlungen nur eröffnet werden, wenn sich der Agent in einem stabilen Zustand befindet und nicht etwa umgekehrt, wenn der Agent gerade dann seinen Nachbarn um Hilfe bittet, wenn dieser selbst noch nicht stabilisiert ist.

Mit Bezug auf die angestrebte Steigerung der Verfügbarkeit ist das Agentensystem in der Lage, fehlerhafte Messungen innerhalb eines Steuerungszyklus zu detektieren und zu kompensieren. Auch kombinierte Ausfälle mehrerer Sensoren werden erfasst und müssen nicht explizit durch ein Regelwerk vor der Laufzeit berücksichtigt werden. Die Agenten sind in der Lage, abhängig von der aktuellen Situation auf der Basis des lokalen Wissens und im Rahmen der Vorgaben, eine Lösung zu finden. Die Berücksichtigung der Mess- und Modellqualität erlaubt es zudem, sowohl die Sensitivität der Fehlererkennung daran zu koppeln als auch die zu erwartende Güte des weiteren Produktionsbetriebs mit einem berechneten Ersatzwert anstatt eines realen Sensors abzuschätzen.

Wie in einem klassischen System steuern die Systemagenten den zugeordneten Teilprozess zyklisch unter Einhaltung der Echtzeitanforderungen. Fehlererkennung und -behandlung werden in jedem Zyklus vor der eigentlichen Steuerungsaufgabe durchgeführt und bei Bedarf Fehler kompensiert. Der Austausch von Nachrichten zwischen Agenten nimmt genau einen SPS-Zyklus in Anspruch, dient aber ausschließlich der mittelfristigen Optimierung des Gesamtsystems bezüglich der Vereinbarkeit lokaler Ziele und der Produktqualität.

Für den Informationsaustausch zwischen den Agenten werden Funktionsbaustein-Aktionen verwendet. Mit einem solchen Aufruf kann auch die Übergabe und Rückgabe von Parametern über die Ein- und Ausgangsgrößen des FB nachgestellt werden. Der Vorteil liegt darin, dass bestimmte Aktionen einmalig mit individuellen Parametern von außen aufgerufen werden können. Konkret verfügen die Postfächer, welche innerhalb der Agenten instanziert sind, über Aktionen, welche das Einstellen und das Versenden von Nachrichten erlauben. Wird eine Nachricht durch einen Aktionsaufruf übergeben, wird diese direkt in die Liste des Postfachs mit dem Vermerk „ungelesen“ eingetragen und selbstverwaltende Aufgaben ausgelöst, wie das Auffinden neuer freier Listeneinträge oder die Aktualisierung von Zählervariablen. Ist das Postfach bereits voll, kann dies ebenfalls über eine Statusvariable zurückgemeldet werden. Ebenso verhält es sich mit dem Zugriff des zugeordneten Agenten auf sein Postfach. Er kann über Aktionen Nachrichten zum Versand einstellen oder gelesenen Nachrichten löschen. Das Postfach kümmert sich um den Versand an den Kommunikationsagenten und wiederholt dies auch, wenn der Empfänger nicht erreichbar sein sollte.

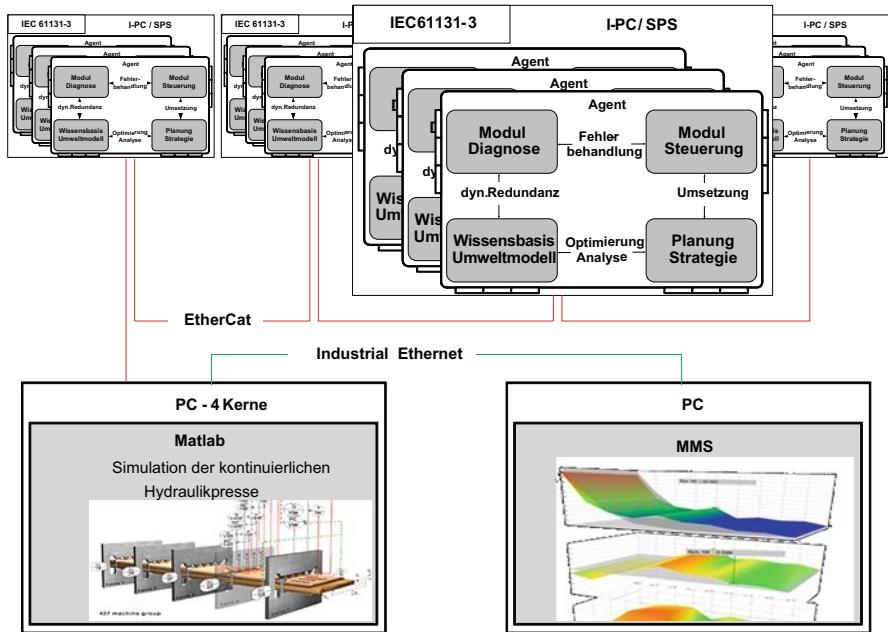


Abb. 10.7 Architektur der Implementierung

10.4 Evaluation durch Messungen im Betrieb

Die Simulation der Presse wird auf einem handelsüblichen PC ausgeführt, der über einen Feldbus direkt mit den eingesetzten Steuerungen verbunden ist (vgl. Abb. 10.7).

Die Messung in Abb. 10.8 zeigt die Reaktion eines Steuerungsagenten auf den Ausfall eines Distanzsensors. In blauer Färbung wird die aktuell gemessene Materialdicke dargestellt, die roten Linien zeigen die Qualität der jeweiligen Sensoren an. Das erste der drei gleichzeitig aufgenommenen Diagramme zeigt die Messungen des realen Sensors, der etwa zum Zeitpunkt $T = 164,3$ s ausfällt und zum Zeitpunkt $T = 174,6$ s wieder in Betrieb genommen wird.

Das zweite Diagramm zeigt den durchgängig verfügbaren virtuellen Sensor, der auf Basis der Dicke des einlaufenden Materials und des aktuellen Drucks, die Dicke des auslaufenden Materials schätzt. Die Qualität der Schätzung ist aufgrund der stabilen Situation der Presse sehr gut und liegt nur knapp unter 1 bzw. 100 %. Im dritten Diagramm sind die Werte aufgezeichnet worden, die der Agent tatsächlich zur Steuerung des Prozesses verwendet. Es ist deutlich zu erkennen, dass der Ausfall des realen Sensors sehr früh erkannt wird und unmittelbar durch den Ersatzwert des virtuellen Sensors kompensiert wird. Die fehlerhafte Messung wird folglich zu keinem Zeitpunkt verwendet. Nach der Wiederinbetriebnahme des realen Sensors erkennt der Agent, dass wieder ein Messwert mit einer höheren Qualität zur Verfügung steht und verwendet diesen unmittelbar.

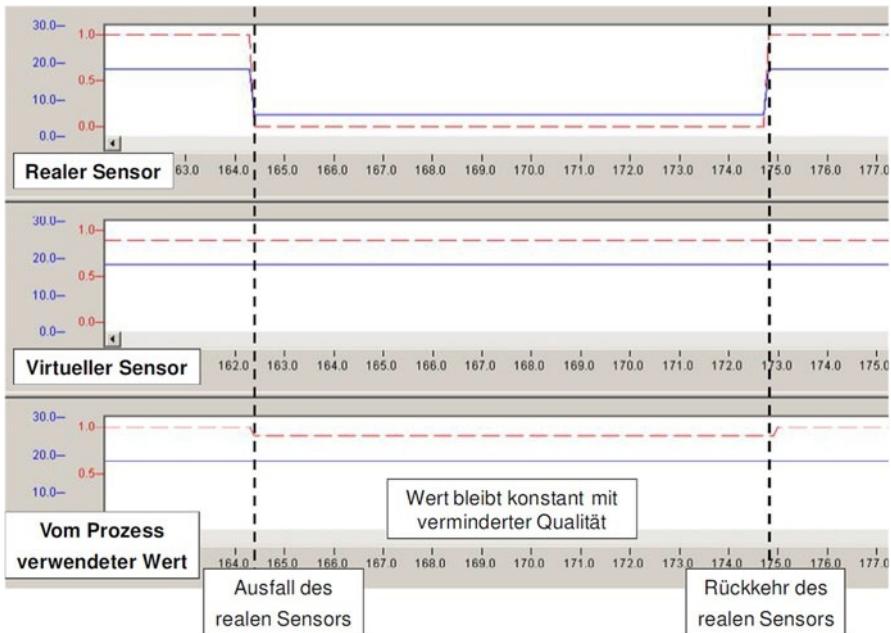


Abb. 10.8 Kommentierte Messung, aufgenommen während eines simulierten Sensorausfalls

10.5 Zusammenfassung und Ausblick

In diesem Beitrag wurde die Implementierung eines Agentensystems für die Steuerung von einer Produktionsanlage vorgestellt, das direkt auf speicherprogrammierbaren Steuerungen nach der Norm IEC 61131-3 lauffähig ist. Die Wissensbasis von Steuerungsagenten und insbesondere das in diesem Beitrag angewendete Redundanzmodell wird durch einen gerichteten Graphen beschrieben, welcher die analytischen Abhängigkeiten zwischen Ressourcen innerhalb des Systems mathematisch beschreibt.

Die entwickelten Agenten sind damit in der Lage den Ausfall von Teilen des technischen Systems und den damit verbundenen Verlust an Funktionen zu erkennen und zu kompensieren. Anhand des Fallbeispiels einer kontinuierlichen Hydraulikpresse wurde dafür die Entwicklung eines Agentensystems und die Wissensbasis der Agenten detailliert vorgestellt. Anhand einer Simulation der Anlage wurde das entworfene Agentensystem, implementiert auf marktüblichen speicherprogrammierbaren Steuerungen und die Fehlerkompensation der Agenten positiv evaluiert.

Der Aufwand für die Erstellung eines solchen Agentensystems konnte durch die Entwicklung von Agentengrundtypen und Standardbausteinen gegenüber einer Neuentwicklung stark reduziert werden. Weiteres Potential zur Beschleunigung der Erstellung eines Agentensystems für die hier betrachtete Anwendungsdomäne liegt in der Entwicklung einer werkzeugunterstützten, durchgängigen Vorgehens-

weise, wie sie in KreaAgentUse [9] verfolgt wird. Auf der Basis eines einheitlichen Beschreibungsmittels wurden, die entwickelten Modelle dorthingehend angepasst, um möglichst automatisch in ein Steuerungsprogramm zu transformieren, welches direkt auf IEC 61131-3-konformen Steuerungen ablauffähig ist. Dieser Vorgang wurde dabei durch die in dieser Arbeit entstandenen Agentengrundtypen unterstützt, welche bereits grundlegende und an die Vorgehensweise angepasste Dienste bereitstellen.

Literatur

- [1] Vogel-Heuser, B.: Automation in wood and paper industry. In: Handbook of Automation. Springer, New York (2008)
- [2] Hasener, J.: Statistische Methoden der industriellen Prozessmodellierung zur Echtzeitqualitätskontrolle am Beispiel einer kontinuierlichen Produktion von Faserplatten. Dissertation. Universität Hamburg (2004)
- [3] Steffen, A., Janssen, A., Kruse, K.: Analyse der Herstellung von MDF mit Hilfe der statistischen Prozessmodellierung. Holz als Roh- und Werkstoff **58**, 419–431 (2001)
- [4] Wannagat, A.: Entwicklung und Evaluation agentenorientierter Automatisierungssysteme zur Erhöhung der Flexibilität und Zuverlässigkeit von Produktionsanlagen. Dissertation. Technische Universität München (2010)
- [5] Wannagat, A., Vogel-Heuser, B.: Increasing flexibility and availability of manufacturing systems – Dynamic reconfiguration of automation software at runtime on sensor faults. Proc. 9th IFAC Workshop on Intelligent Manufacturing Systems (2008)
- [6] Thoemen, H., Haselein, C.R., Humphrey, P.E.: Modeling the physical processes relevant during hot pressing of wood-based composites. Holz als Roh- und Werkstoff **64**, 125–133 (2006)
- [7] Hanvongjirawat, H.: Permeabilität von Holzwerkstoffmatten. Dissertation. Universität Hamburg (2003)
- [8] Zombori, B.G.: Modeling the Transient Effects during the Hot-Pressing of Wood-Based Composites. Dissertation. Virginia Polytechnic Institute and State University (2001)
- [9] Forschungsprojekt „Konzeption, Realisierung und Evaluation einer werkzeugunterstützten Vorgehensweise für die Entwicklung von Agentensystemen in der Automatisierungstechnik unter Berücksichtigung der Usability (KREAagentuse)“, gefördert von der Deutschen Forschungsgemeinschaft (DFG), Förderkennzeichen VO 937/8-1.

Kapitel 11

Einsatz von Agentensystemen zur Optimierung der Logistik in Produktions- und Agrarprozessen

Holger Kremer und Clemens Westerkamp

Zusammenfassung In vielen industriellen Anwendungen werden Entscheidungsprozesse anhand von zentralen Vorgaben aus Enterprise Resource Planning (ERP)-Werkzeugen oder Produktionsplanungssystemen (PPS) durchgeführt. Um Ziele wie eine flexible Fertigung oder schnelle Reaktion auf lokale Änderungen erreichen zu können, müssen Entscheidungen zunehmend dezentral autonom oder teilautonom (mit menschlicher Unterstützung) getroffen werden. Zur Realisierung dieser Anforderungen werden in diesem Beitrag Agenten-Systeme untersucht. Um die entstehende Plattform zur dezentralen Entscheidungsunterstützung universell einsetzen zu können, hat die Hochschule Osnabrück in zwei Projekten Erweiterungen zum weit verbreiteten JADE-Agentensystem entwickelt. Diese sollen die Einsetzbarkeit auf möglichst vielen Hardware-Plattformen und unter erschwerten Kommunikationsbedingungen ermöglichen. Das dabei entstandene KOMOBAR-Agentensystem wird anhand von zwei Beispielen aus der industriellen Produktionslogistik und der Optimierung von Ernteprozessen erprobt.

11.1 Einleitung

Aktuelle Automatisierungssysteme arbeiten bei der Bearbeitung von Produktions- und Logistikaufgaben in der Regel zentralistisch. Es wird erwartet, dass alle Daten der Prozesse zu allen Zeiten zeitnah erfasst und zur Zentrale geleitet werden. Die Steuerung und Optimierung finden dann in der Leitebene auf der obersten Hierarchie eines MES (Manufacturing Execution System) oder LVS (Lagerverwaltungssystem) statt. In der Regel werden entscheidungsunterstützende Systeme bisher nur auf der obersten Ebene eingesetzt. Die acatech-Studie agendaCPS [1] beschreibt die Flexibilisierungsmöglichkeiten durch intelligente Software-Lösungen

Holger Kremer (✉), Clemens Westerkamp
Labor für Softwaretechnik, Hochschule Osnabrück, Albrechtstraße 30,
49076 Osnabrück, Deutschland
e-mail: h.kremer@hs-osnabrueck.de, c.westerkamp@hs-osnabrueck.de

auch auf unterer Ebene mit den Worten „Die vertikale Vernetzung eingebetteter Systeme mit betriebswirtschaftlichen Prozessen durch CPS bietet erhebliche Optimierungspotenziale – vor allem im Bereich der Produktion“.

In der Intralogistik z. B. auf Gabelstaplern oder beim Einsatz von RFID-Lesern kommt hinzu, dass viele Teilnehmer mobil unterwegs sind und mit variierenden drahtlos angebundenen Endgeräten arbeiten, so dass die Kommunikation nicht immer sicher gewährleistet ist. Gleichzeitig könnten viele Produktions- und Transportaufgaben dezentral schneller und ohne Beteiligung der zentralen Systeme gelöst werden. In diesem Beitrag sollen die Vorteile und Einschränkungen beim Einsatz von Software-Agenten zur dezentralen Optimierung von logistischen Prozessen nach ökonomischen und ökologischen Kriterien untersucht werden. Dies geschieht anhand von zwei stark unterschiedlichen Anwendungsbeispielen:

1. Intralogistik-Prozesse in einem Warenausgangslager als Beispiel für Anwendungen der Fabrikautomatisierung. Hintergrund ist das von 2006–2010 bearbeitete BMBF-geförderte Forschungsprojekt LK³S (Leicht konfigurierbare Komponenten kollaborativer Systeme) [1].
2. Logistische Prozesse in der Landwirtschaft am Beispiel der Maisernte einschließlich des Transports zur Verarbeitung. Dies geschieht im Forschungsschwerpunkt KOMOBAR (Entscheidungsstrategien und Kommunikationssstrukturen für Kooperierende Mobile Arbeitsmaschinen), der von der Volkswagenstiftung gefördert wird [2].

11.2 Ausgangssituation/Problemstellung/Stand der Technik

11.2.1 Logistikunterstützung in der Fabrikautomatisierung

In der Fabrikautomatisierung werden seit einiger Zeit sowohl service- als auch agentenorientierte Ansätze zur Optimierung von Produktionsabläufen untersucht. In SOCRADES [3] und SIRENA [4] werden Webservices eingesetzt, um die Kommunikation zwischen den Systemkomponenten innerhalb einer rein dienstorientierten Architektur herzustellen. Das EU-Projekt PAPADIS PrOMiSE [5] verwendet kommerzielle Software-Agenten-Systeme für Standard-PC-Plattformen, um Produktionsprozesse und Supply-Chain Management zu steuern.

11.2.2 Logistikunterstützung im landwirtschaftlichen Bereich

Auch in der Landwirtschaft wurde in den vergangenen Jahren die Optimierung von Geschäftsprozessen mittels unterschiedlicher Ansätze untersucht. Im Forschungsprojekt Robot2Business [6] semi-autonome mobile Maschinen und Prozesse in Geschäftsmodelle integriert. Dies wurde durch die Portierung der

OSGI-Ausführungsumgebung [7] auf Industrie-PCs, also durch einen serviceorientierten Ansatz erreicht. Auch wenn dabei einige Möglichkeiten von SW-Agenten wie einheitliche Ausführumgebung und die Möglichkeit, Java-Code nachzuladen, realisiert werden konnten, ist doch das OSGi-Framework eine sehr mächtige und komplexe Umgebung, deren Einsatz auf mobilen und eingebetteten Systemen mit Schwierigkeiten verbunden ist.

Vom Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI) [8] wird ein größeres Konsortium im BMBF-Projekt iGreen [9] geleitet. Innerhalb des Projekts wird eine Wissens- und Ortsbasiertes Netzwerk entwickelt, um verteilte landwirtschaftliche Produktionsprozesse dezentral zu unterstützen und zu optimieren. Da das Projekt noch in einem frühen Stadium ist, liegen nähere Erkenntnisse noch nicht vor.

11.2.3 Problemstellung Fabrikautomatisierung

Zwei Aspekte wurden in vorhergehenden Projekten zur Fabrikautomatisierung wenig betrachtet.

Bisherige Agentenplattformen basieren meist auf einer Java-Umgebung, wie sie z. B. auf Desktop-PCs und Industrie-PCs gängig ist bzw. leicht realisiert werden kann. In der Praxis sind aber viele Systeme ohne Java im Einsatz. Eine besondere Herausforderung stellen dabei Embedded Systems mit reduzierten Ressourcen bezüglich Prozessor und Speicher dar. Um auch auf diesen Plattformen Software-Agenten nutzen zu können, muss eine vereinfachte Agentenplattform in ANSI-C entwickelt werden, die sich in eine FIPA-ACL-basierende Kommunikation nahtlos einbettet.

Eine weitere Hürde bei der Einführung von Software-Agenten-Systemen in der Fabrikautomatisierung stellen die notwendigen Kenntnisse für ihren Einsatz dar. Während die Entwickler solcher Systeme ihre Plattform und die Einsatzmöglichkeiten der Agenten gut beherrschen, fehlen ihnen die Prozesskenntnisse des Anwendungsbereichs. In diesem sind meist Produktionstechniker, -meister oder -ingenieure mit stark unterschiedlichen Informatikkenntnissen im Einsatz. Für diese anwendungsorientierten Nutzer bietet sich die Bereitstellung von Konfiguratoren an, die nach der Konzeption und Inbetriebnahme eines Agentensystems die fortlaufende Konfiguration und Parametrierung ermöglichen.

11.2.4 Problemstellung Agrarlogistik

Im Gegensatz zur Fabrikautomatisierung ist die Agrarlogistik heute weitgehend dezentralisiert organisiert. Der Forschungsschwerpunkt KOMOBAR verfolgt das Ziel, logistische Prozesse in der Landwirtschaft zu optimieren. Es adressiert die spezifischen Anforderungen der IT-gestützten Steuerung von Supply Chains in ländlichen

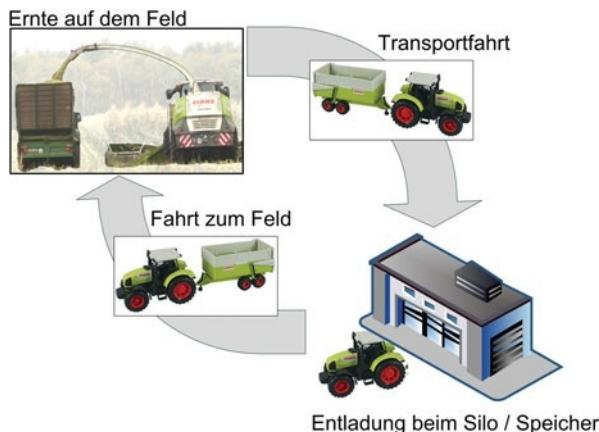


Abb. 11.1 Typisches Maisernte-Szenario

Gebieten. Die dabei zu adressierenden Herausforderungen sollen am Anwendungsszenario der Ernte von Maispflanzen, ausgeführt von einem vom Landwirt beauftragten Lohnunternehmen, veranschaulicht werden. In einer ersten Planungsphase werden die zur Verfügung stehenden Ressourcen ermittelt und die Bearbeitung der Kundenaufträge geplant. Ressourcen stellen Ernte- und Transportmaschinen dar, die von Fahrern bedient werden. Die Kundenaufträge werden in einzelne Produktionsaufträge und diese in Maschinenaufträge aufgeteilt. In der zweiten Phase, der Steuerung, führen die Fahrer die Ernteaufträge aus. Bei der Maisernte wird der geerntete Mais (ohne Zwischenlagerung) direkt in ein parallel zu dem Erntefahrzeug fahrenden Transporter gefüllt (siehe Abb. 11.1).

Wenn die maximale Ladekapazität des Transporters erreicht ist, fährt dieser zu einem entfernten Silo oder Speicher, um die Fracht zu entladen. Verlässt ein Transporter das Erntefahrzeug, wird er umgehend durch einen weiteren Transporter ersetzt. Ist der Abladenvorgang beim Silo abgeschlossen, fährt der nun wieder leere Transporter zurück zum Ernter, um auf den nächsten Füllvorgang zu warten. Ein Großteil der auftragsbezogenen Kommunikation wird momentan telefonisch abgewickelt. Die mobilen Akteure verfügen dabei zunehmend über Geräte mit Internetanschluss und Webbrowsern. Außerdem sind auf den Erntemaschinen und Transportfahrzeugen Embedded Systems zur Steuerung und in Bediengeräten vorhanden. Sie könnten bei geeigneter Erweiterung in ein agentenbasiertes Szenario einbezogen werden, wozu in Abschn. 11.3.2 ein Konzept entwickelt wird.

11.2.5 Stand der Technik zur Agenten-Orientierten Programmierung

Yoah Shoham, der Begründer der Agenten-orientierten Programmierung, beschreibt Software-Agenten als Entitäten, deren Zustand aus mentalen Komponenten, wie Beliefs, Capabilities, Choices und Commitments [10] besteht. Sie haben ein autonomes sowie adaptives Verhalten und führen Aktionen proaktiv aus. Außerdem können Agenten reaktiv auf das Verhalten anderer Agenten reagieren. Die Kommunikation zwischen den Software-Agenten erfolgt durch den Austausch von speziell kodierten Nachrichten.

Die Foundation for Intelligent Physical Agent (FIPA) [11] ist eine IEEE Organisation die sich für Agenten-basierte Technologien und die Interoperabilität mit anderen Standards einsetzt. 2002 veröffentlichte die FIPA eine standardisierte Architektur für ein Agenten-basiertes Kommunikationsprotokoll, die FIPA Agent Communication Language (FIPA-ACL) [12]. Weiterhin spezifizierte die FIPA unter dem Namen Agenten-Plattform (AP) eine Basis, auf der Software-Agenten operieren können. Die Plattform stellt verschiedene zentrale Dienste zur Verfügung.

Nach einer EU-Studie [13] ist die meist genutzte Implementierung der FIPA Standardisierung das Java Agent Development Kit (JADE) [14]. Es ist als Open Source unter der LGPL [15] veröffentlicht. Alle FIPA-spezifischen Komponenten einer Agenten-Plattform, die für das Bereitstellen und Ausführen von Agenten benötigt werden, sind in JADE realisiert. Zuerst wird der Main Container gestartet, in dem das Agenten-Management-System (AMS) Verwaltungsaktionen wie das Registrieren, Starten und Beenden von Agenten oder das Herunterfahren der gesamten Plattform durchgeführt werden. Ein weiterer Dienst im Main Container ist der Directory Facilitator (DF) mit einem Gelbe-Seiten-Dienst für Agenten [14].

Eine Erweiterung für mobile Endgeräte stellt das JADE Lightweight Extensible Agent Plattform (JADE-LEAP) dar. Es lagert die schwergewichtigen Teile, wie die Kommunikation und Organisation der Software-Agenten, auf einen sogenannten Mediator auf dem Server aus.

11.3 Verteilte Systemarchitektur für Fabrik- und Agraranwendungen

Für die besonderen Anforderungen der beiden Einsatzgebiete wurden zwei Architekturen entwickelt, die in diesem Abschnitt beschrieben werden.

11.3.1 Architektur für den Einsatz in der Fabrikautomatisierung

Abbildung 11.2 zeigt die Architektur, mit der der Einsatz eines Software-Agentensystems innerhalb einer vorhandenen Fabrikstruktur realisiert wurde:

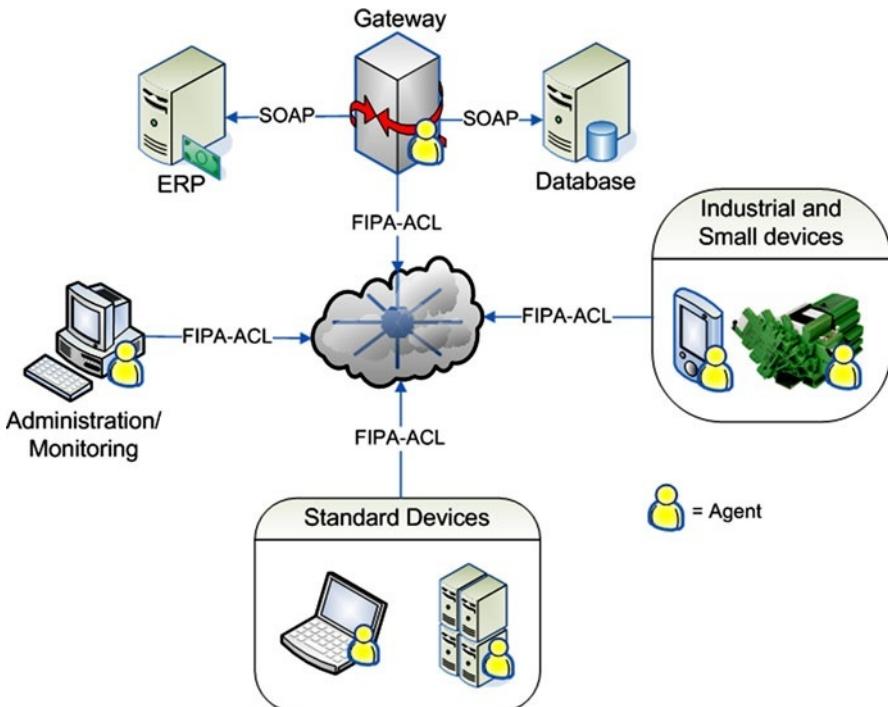


Abb. 11.2 Systemarchitektur für den Einsatz in der Fabrikautomatisierung mit der service-orientierten Koppelung an zentrale IT-Systeme

Im oberen Bereich sind die Systeme der Leitebene zu sehen. Beispielhaft sind ein ERP-System und eine Datenbank dargestellt. Beide werden über Webservices an ein Gateway angebunden. Auf der Software-Agenten-Seite agiert dieses Gateway FIPA-ACL-konform und kann mit den verschiedenen Standorten der verteilten Agentenplattform durchgängig kommunizieren. Als Industrie-Geräte-Plattformen wurden 16Bit Embedded Systeme und PLCs [16] eingesetzt. Es stellte sich heraus, dass der Software-Agenten Ansatz für die beschriebenen Bereiche aufgrund des verteilten Charakters als eine flexible Lösung anzusehen ist.

11.3.2 Architektur für den Einsatz in der Agrarlogistik (KOMOBAR)

In KOMOBAR ist das Planungssystem des Lohnunternehmer-Disponenten die zentrale Instanz, die mit dezentralen Komponenten kommuniziert. Die KOMOBAR-Architektur ist eine Erweiterung der LK³S-Architektur.

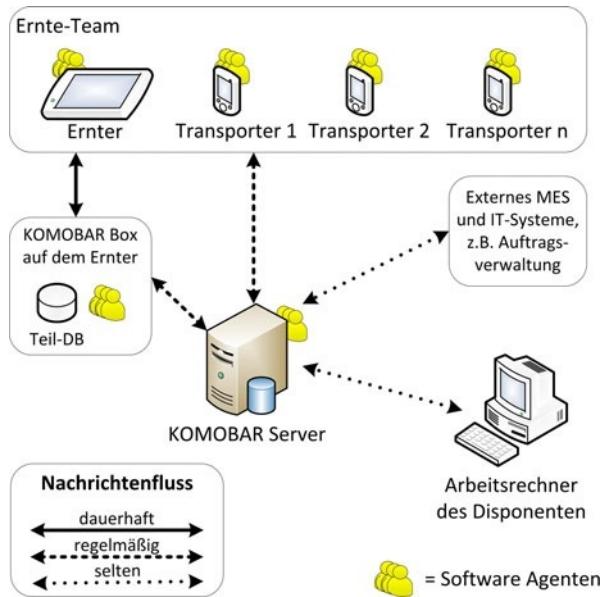


Abb. 11.3 Architekturübersicht für den Einsatz von Software-Agenten in der Agrarlogistik

KOMOBAR Systemansicht

In der Agrarlogistik erstreckt sich die Variation der Plattformen, auf der Software-Agenten ausgeführt werden vom Server über Embedded Systems bis hin zu handelsüblichen Mobilgeräten wie Smartphones und Tablet-Computer [17]. Die Steuerungen und die mobilen Geräte befinden sich während des Erntevorgangs auf den beteiligten Maschinen. In jedem beteiligten Gerät soll der Einsatz von Software-Agenten möglich sein. Die Benutzerführung kann u. a. in einem Webbrowser bereitgestellt werden. Abbildung 11.3 verdeutlicht die Interaktion zwischen allen KOMOBAR-Komponenten.

Der KOMOBAR-Server enthält eine primäre Datenbank und kommuniziert mit den Client-Systemen web-basierend oder über Software-Agenten. Mögliche Clients sind:

- Der Arbeitsrechner des Disponenten, in dem während der Planungsphase die Kunden-, Produktions- und Maschinenaufträge bearbeitet werden. Während der Steuerungsphase wird der Verlauf der Ernte kontinuierlich überwacht.
- Eine KOMOBAR-Box (Embedded PC) als Gateway zu Maschinendaten. Im Falle einer Verbindungsunterbrechung zum KOMOBAR-Server kann die Box als lokaler Server für in der Nähe befindliche Fahrzeuge dienen und eine Teildatenbank bereithalten. Zu Testzwecken wurden auf der Box bereits JADE-Agenten zum Übertragen und Sammeln der GPS-Informationen erfolgreich implementiert.

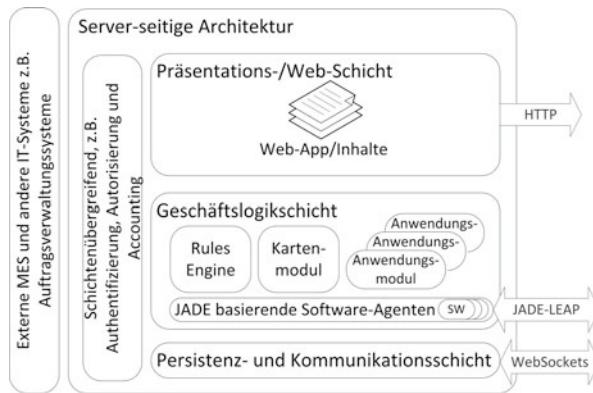


Abb. 11.4 Server-seitige KOMOBAR-Architektur

- Ernte- und Transportfahrer-Clients mit Web- und Agenten-Komponenten (siehe Abschn. 11.4.2).

Server-seitige KOMOBAR-Architektur

Der KOMOBAR-Server führt die Server-seitigen Komponenten aus und ist als gewöhnliche Drei-Schicht-Architektur (siehe Abb. 11.4) entworfen worden:

- Die *Präsentations-/Web-Schicht* stellt die webbasierten Client-Anwendungen zur Verfügung.
- Die *Geschäftslogikschicht* enthält verschiedene Anwendungsmodule. Außerdem ist hier die JADE-basierte Agentenplattform angesiedelt in der die meisten Teile der Geschäftslogik abgebildet sind. Der notwendige Nachrichtenaustausch zwischen Server- und Client-seitigen Software-Agenten wird durch JADE-LEAP zur Verfügung gestellt.
- Die *Persistenz- und Kommunikationsschicht* implementiert Methoden zum Lesen und Schreiben von Daten und verschiedene Webservices. Hierzu gehört u. a. ein WebSocket-Gateway, welches für den auf JADE-LEAP basierenden Kommunikationskanal benötigt wird.

Client-seitige KOMOBAR-Architektur

Die Client-Seite des KOMOBAR-Systems besteht aus einer webbasierten Applikation, die auf Smartphones und Tablet Computern ausgeführt wird. Die Anwendung ist so entworfen, dass sie im Falle eines Verbindungsverlusts zum Server weiterhin ihren Dienst vollständig verrichten kann.

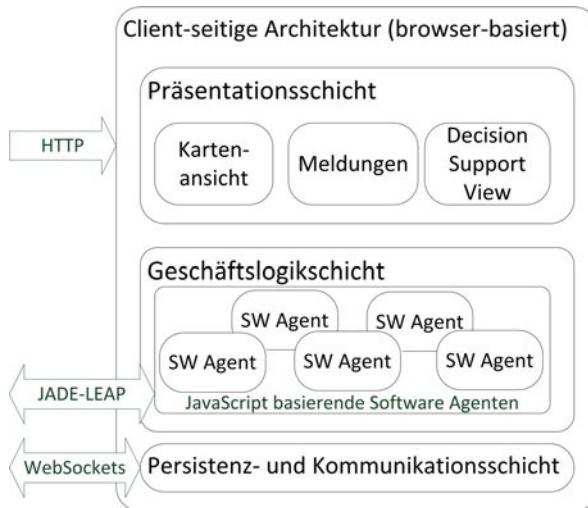


Abb. 11.5 Client-seitige KOMOBAR-Architektur

Die im Webbrowser ausführbaren Clients beziehen die Web-Anwendung vom KOMOBAR-Server oder der KOMOBAR-Box. Die Client-Architektur spiegelt das Drei-Schichten-Modell des Servers wieder (siehe Abb. 11.5):

- Die *Präsentationsschicht* unterstützt in der Benutzeroberfläche den Fahrer visuell bei seinen Entscheidungen während der Auftragsabwicklung.
- Die *Geschäftslogikschicht* beinhaltet Funktionen zur Entscheidungsunterstützung wie GPS-Lokalisierung, Auftragsverwaltung oder Zeitmessung.
- Über die *Persistenz- und Kommunikationsschicht* wird der Zugriff zum Server hergestellt. Hier befindet sich außerdem der Client-seitige Teil des JADE-LEAPs. Darüber hinaus werden hier Daten zur Überbrückung von Offline-Perioden lokal zwischengespeichert.

Kommunikationsarchitektur

Die beschriebene verteilte Plattform benötigt zur Ausführung eine robuste Kommunikationsarchitektur. In der Regel werden die Daten zwischen allen Instanzen über das Mobilfunknetz transferiert. Die für den Auftrag relevanten Daten eines Tages werden morgens beim Lohnunternehmen über WLAN übertragen. Die im Laufe des Tages transferierten Daten beinhalten leichtgewichtige Nachrichten für die Agentenkommunikation. Im Fall eines Verbindungsverlusts zum KOMOBAR-Server kann auf ein Notfall-Kommunikationssystem [18] zurückgegriffen werden. Es basiert auf einem Wireless Ad-hoc-Datentransfer und verwendet Store-Carry-Forward Mechanismen zur Kommunikation. Die Clients besitzen einen Zwischen Speicher für Offline-Perioden und für Historiendaten sowie für zu sendende Nach-

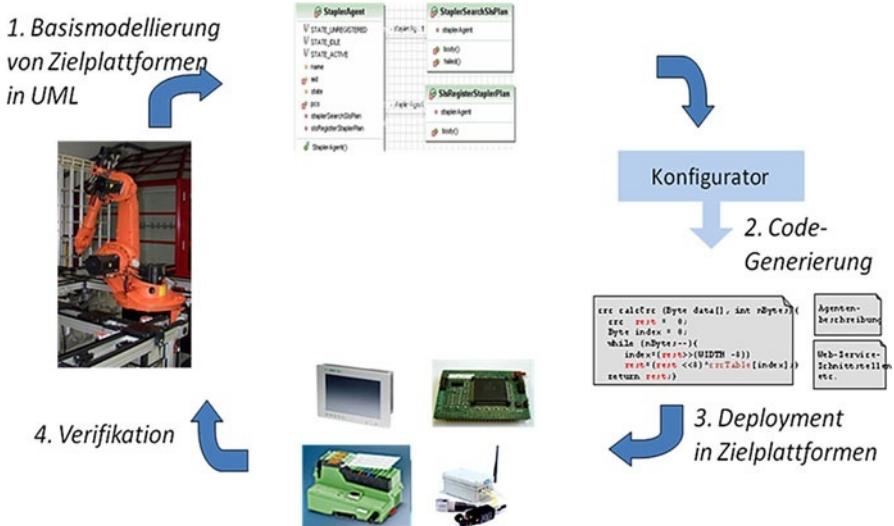


Abb. 11.6 Werkzeuge und Arbeitsablauf für das Meta-Modell (am Beispiel der Fabrikautomatisierung)

richten. Bei gestörter Netzanbindung können Nachrichten per Wireless Ad-hoc an andere Fahrzeuge weitergereicht werden, wenn die lokale Entfernung dies erlaubt. Am Ende des Tages werden die gesammelten Historiendaten in die Datenbank des KOMOBAR-Servers integriert.

11.3.3 Konzeption für Erweiterungen der JADE-Agenten-Plattform

Um den in Abschn. 11.2 beschriebenen Anforderungen zu genügen, wurde die JADE-Plattform um einige Fähigkeiten und Komponenten erweitert.

Workflow zur grafischen Beschreibung des Agentenverhaltens

Um Software-Agenten vereinfacht zum Einsatz zu bringen, wurde ein Meta-Modell entwickelt, das auf einem werkzeuggestützten Workflow (siehe Abb. 11.6) basiert:

Folgende Arbeitsschritte werden für den Einsatz einer Software-Agentenplattform durchlaufen:

1. Ein vorhandenes Software-Agentensystem (hier JADE) wird auf allen direkt unterstützten Zielplattformen in Betrieb genommen. Bei bislang nicht unterstützten Systemen z. B. einem Embedded System werden dessen Software-

- Komponenten auf Basis eines generischen ANSI-C-Agenten traditionell oder in einem UML-Tool (z. B. Rhapsody) erstellt.
2. In einem UML-basierenden Konfigurator, der auch für Nutzer ohne Informatik-Studium geeignet sein soll, wird das konkrete Agentenverhalten modelliert. Dabei können in einer grafischen Oberfläche die Kommunikation- und Entscheidungsabläufe sowie die Berechnung von Entscheidungsmetriken konfiguriert werden
 3. Durch Code-Generierung werden ausführbare Programme für die Zielplattformen erstellt.
 4. Das Verhalten des Agentensystems wird verifiziert und (je nach Änderungsbedarf) das gesamte Agentensystem angepasst. Kleinere Anpassungen werden im Konfigurator vorgenommen.

Um diesen Arbeitsablauf zu ermöglichen, wurde ein UML-basierender Ansatz zur Agentenmodellierung mittels Aktivitätsdiagrammen realisiert, da ähnliche Diagrammarten (Flussdiagramme, Programmablaufpläne) vielen Nicht-Informatikern aus der Steuerungsprogrammierung bekannt sind. Basierend auf der Rich Client Plattform (RCP) des Eclipse-Projektes wurde ein Konfigurator entwickelt, in dem das Verhalten der Agenten grafisch modelliert und parametriert werden kann. Dabei können insbesondere die Kommunikation (zur Abfrage von entscheidungsrelevanten Daten) und die Entscheidungsfindung der Agenten spezifiziert werden, ohne nähere Kenntnisse über Software-Agenten zu haben.

In Abschn. 11.4.1 wird dieses Konzept anhand eines Beispiels aus der Intralogistik erprobt.

11.3.4 Konzept für FIPA konforme, webbasierte Software-Agenten

Eine der Anforderungen, die an die Client-Systeme im KOMOBAR-Projekt gestellt wird, ist die Lauffähigkeit auf aktuellen Smartphones und Tablet-Computern. Hierfür müssen die mobilen Betriebssysteme wie Google Android, Apple iOS oder Microsoft Windows Phone unterstützt werden können. Um eine Plattformunabhängigkeit zu ermöglichen, wird ein neuartiger JavaScript-basierender Ansatz angestrebt. Bisher sind den Autoren keine Implementierungen von Software-Agenten in JavaScript bekannt. Mit der Veröffentlichung von HTML5 wurden mit WebSockets [19] neue JavaScript-Funktionen eingeführt, die Hindernisse bei der Agentenkommunikation überwinden. WebSockets ermöglichen ununterbrochene bidirektionale Kommunikation zwischen Webbrowser und Webserver über das HTTP-Protokoll. Damit ist innerhalb des KOMOBAR-Projekts ein JavaScript basierender FIPA-ACL konformer Adapter zu JADE-LEAP entstanden. Ein WebSocket-Gateway für die Verbindungsverarbeitung beim Main Container sorgt für die Übersetzung der Daten aus den WebSockets in ein JADE-LEAP und FIPA-ACL-konformes Datenformat.

11.4 Ergebnisse

11.4.1 Erprobung in der Fabrikautomatisierung

Der Ansatz der leichten Konfigurierbarkeit eines Software-Agentensystems in der Intralogistik einer Fabrikautomatisierung soll anhand eines Beispielszenarios zur Interaktion von auszulagernden Paletten mit den transportierenden Gabelstaplern erläutert werden. Dazu wird im UML-Konfigurator ein Verhalten (siehe Abb. 11.7) spezifiziert.

Erhalten die Paletten aus einem der zentralen Systeme den Auftrag, sich auszulagern, ermitteln sie zunächst die verfügbaren Stapler und lassen sich Angebote für den Transport unterbreiten. Diese Angebote sind mit wichtigen Parametern wie Kosten, Zeitdauer, Energiebedarf, Risiko etc. versehen.

Damit wird eine Metrik berechnet, die eine Bewertung anhand der aktuell für die gesamte Auslagerung gültigen Optimierungskriterien ermöglicht. Je nach Auf-

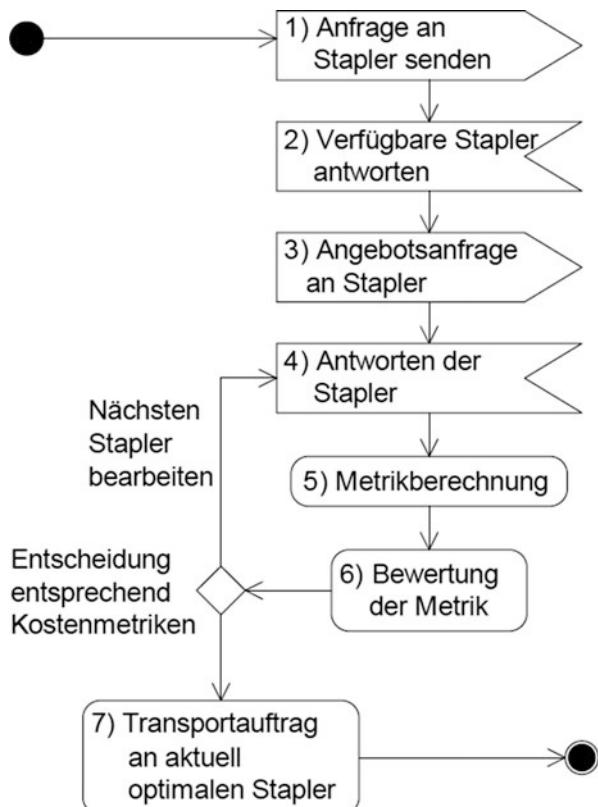


Abb. 11.7 UML-Aktivitätsdiagramm des Verhaltens eines Paletten-Agenten im Konfigurator

tragslage oder Gesamtsituation der Prozesskette unter Einbeziehung von Zulieferern, können vom ERP-System aus Optimierungskriterien vorgegeben werden, die eine Umsteuerung durch das Agentensystem bewirken. Dadurch kann eine optimale Aufteilung der zentral und der dezentral durchzuführenden Anpassungen und Entscheidungen erreicht werden, um z. B. bei lokal auftretenden Störungen eine beschleunigte Reaktion zu erreichen.

Der LK³S-Prototyp wurde als Messedemonstrator auf der SPS IPC Drives und der Hannover Messe Industrie gezeigt. Die Ergebnisse wurden u. a. bei den Projekt-partnern Volkswagen (Emden) und PhoenixContact bei der Weiterentwicklung von Logistiklösungen berücksichtigt.

11.4.2 Erprobung in der Agrarlogistik

Aufbauend auf den Ergebnissen in der Fabrikautomatisierung wurden die in Abschn. 11.3.3 und 11.3.4 beschriebenen Erweiterungen genutzt, um den Einsatz in der Agrarlogistik zu ermöglichen. Dabei kommt eine Kombination aus Web-Anwendungen und Software-Agenten zum Einsatz. Die beiden folgenden Web-Anwendungen wurden zunächst ohne Agentenunterstützung realisiert:

Harvester App Sie wird auf einem Tablet Computer beim Erntefahrer (zukünftig integriert in ein Embedded-Bediengerät) ausgeführt. Es werden die Positionen/Distanzen und der Arbeitsstatus aller Beteiligten des Ernte-Teams auf einer Karte angezeigt. Eine *Decision Support View* zeigt die Reihenfolge der Aufträge für einen Tag. Für den Fall, dass ein unvorhergesehenes Problem auftritt, empfiehlt die App ggf. eine neue Auftragsreihenfolge basierend auf den Ergebnissen des Entscheidungsunterstützungssystems. Weiterhin können die Erntemaschinenfahrer vordefinierte Hinweise (z. B. „Wendemanöver“) an die Fahrer der Transportfahrzeuge über ein Benachrichtigungssystem senden.

Transporter App Diese App wird von den Transportfahrern bedient und auf Smartphones ausgeführt. Sie zeigt ebenfalls alle Beteiligten des Ernteprozesses an und ist mit dem Benachrichtigungssystem verbunden. Im *Decision Support View* wird die Zeit angezeigt, die dem Fahrer bleibt, um wieder beim Erntefahrzeug zu sein. Damit kann der Transportfahrer die Fahrgeschwindigkeit optimieren. Zukünftig sollen bei der Schätzung Ansätze zur Lernfähigkeit angewendet werden.

Agententypen in der Agrarlogistik

Folgende anwendungsspezifische Agententypen kommen auf unterschiedlichen Plattformen zum Einsatz:

Der *Order Agent* verwaltet die am Morgen übertragenen Maschinenaufträge und hält die Bearbeitungsreihenfolge, Kunden- und Felddaten bereit. Dieser Agent verfolgt die Auftragsbearbeitung und berechnet deren voraussichtliche Dauer.

Der *Geo-Positioning Agent* auf allen mobilen Plattformen sendet und empfängt GPS-Daten periodisch von/zu anderen Agenten eines Ernte-Teams. Damit wird ein Update der Kartenansicht mit den jeweiligen Entfernung realisiert.

Der *Notification Agent* implementiert ein Benachrichtigungssystem.

Der *Transporter Agent* ist nur in der Geschäftslogik der Transporter App vorhanden und beherbergt u. a. einen Zustandsautomaten für die Arbeitszustände der Transportmaschine (z. B. Leerfahrt, Transportfahrt, Überladevorgang). Der Transporter Agent ermittelt außerdem die verbliebene Zeit und gefahrene Strecke innerhalb eines Arbeitszustandes. In der ersten Version der Realisierung wird ein Zustandswechsel zunächst manuell durch den Fahrer ausgelöst. Über diesen Wechsel werden die anderen Agenten direkt informiert.

Der *Harvester Agent* unterscheidet sich vom Transporter Agent in der Ermittlung des aktuellen Arbeitszustandes (z. B. Feldfahrt, Überladevorgang) aus Maschinen-daten der KOMOBAR-Box und anderen Kontextdaten.

Der *Persistence Agent* ist für die Speicherung und Übermittlung von Historiendaten verantwortlich. Die relevanten Daten aller Agenten werden in einem lokalen Speicher zwischengelagert. Begegnen sich Transportfahrzeug und Erntefahrzeug, wird der Zwischenspeicher in die Teildatenbank der KOMOBAR-Box übertragen. Am Ende des Erntetages wird die Datenbank der Box mit der Datenbank des Servers synchronisiert. Historiendaten können ortsbezogen für die nächste Ernte sowie für zukünftig geplante lernfähige Optimierung genutzt werden.

Der *Decision Support Agent* berechnet Metriken und macht dem Fahrer Entscheidungsvorschläge. Dabei nutzt er aktuelle Kontextdaten der übrigen Agenten und Auftrags- und Historiendaten. Die Optimierung des Erntevorgangs kann sowohl beim Ernte- als auch beim Transportfahrer erfolgen.

Optimierung beim Transportfahrer

Eine einfache, jedoch weitreichende Entscheidungsunterstützung ist die *Remaining Time View* (siehe Abb. 11.8) der Decision Support View. Es zeigt die verbleibende Zeit, die einem gerade befüllten Transporter bleibt, um für den nächsten Befüllvorgang beim Erntefahrzeug zu sein. Zudem wird die Distanz zur nächsten Arbeitsstation (Erntefahrzeug/Silo) gezeigt.

Die verbleibende Zeit wird u. a. aus den Leistungsdaten des Tages sowie den Historiendaten geschätzt. Mit diesen Informationen hat der Fahrer des Transportfahrzeuges die Möglichkeit, die Fahrtgeschwindigkeit an den Arbeitsfluss der Ernte optimal anzupassen, um z. B. den Energieverbrauch zu reduzieren. Weitere Einflussfaktoren werden momentan auf ihre Nutzbarkeit untersucht.



Abb. 11.8 Remaining Time View der Transporter App

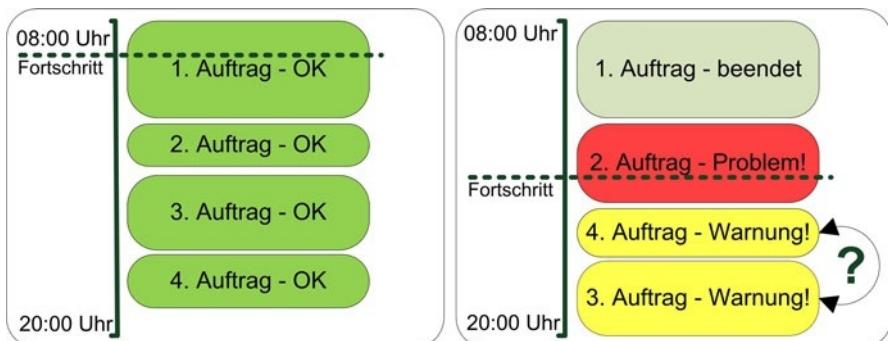


Abb. 11.9 Auftragsliste und Entscheidungsunterstützung beim Erntefahrer – morgens/mittags

Optimierungen beim Erntefahrer

Im Falle eines technischen Problems oder anderen Zwischenfällen ist es wünschenswert, die hieraus resultierende unproduktive Zeit durch den Decision Support Agent zu minimieren. Das Konzept der Benutzeroberfläche der korrespondierenden Decision Support View ist in Abb. 11.9 dargestellt.

Abbildung 11.9 beschreibt die Decision Support View auf dem Erntefahrzeug am Morgen (links) und in der Mittagszeit (rechts). Morgens werden die Aufträge planmäßig ausgeführt. Während des zweiten Auftrages tritt eine unvorhergesehene

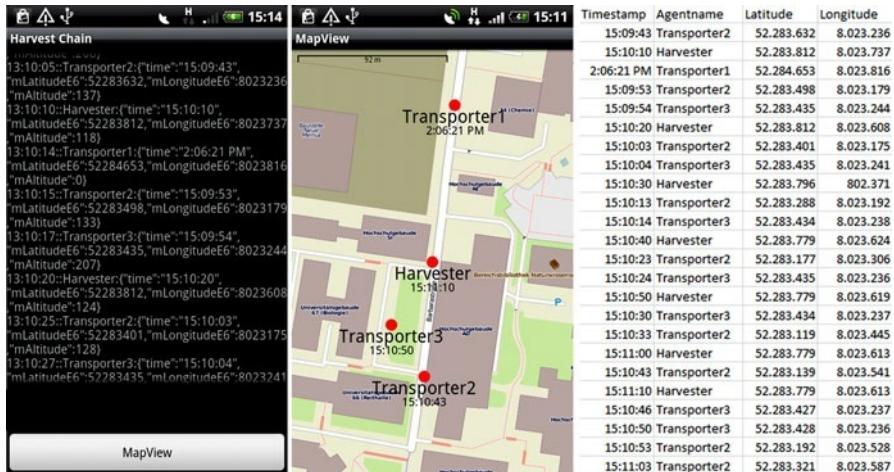


Abb. 11.10 JADE Android Prototyp: Nachrichtenansicht, Kartenansicht, CSV-Datei

Störung auf (z. B. eine durch einen Stein im Mähwerk ausgelöste Sicherheitsabschaltung) und bringt die gesamte Erntekette zum Erliegen. In diesem Fall ist der Fahrer des Erntefahrzeugs aufgefordert, Änderungen an der Reihenfolge der Aufträge vorzunehmen. Der Decision Support Agent ermittelt durch Metrikberechnung und -Evaluation mögliche Auftragsreihenfolgen und deren Auswirkungen. Je nach Auswahl durch den Erntefahrer kann ein optimierter Arbeitsfluss und eine kürzere Erntezeit erreicht werden.

11.4.3 JADEAndroid-Prototyp und erste Testergebnisse

Um die grundsätzliche Verwendbarkeit als Vorstufe für den webbasierten Ansatz im landwirtschaftlichen Bereich zu analysieren, wurde ein Client für das Android-Betriebssystem entwickelt. Damit soll u. a. das prinzipielle Verhalten von Software-Agenten mit mobilen und damit stark wechselnden Netzwerkbedingungen außerhalb eines Intranets untersucht werden. Dazu wurde mithilfe des JadeAndroid Add-Ons [20] ein Geolocation-Agent implementiert. JadeAndroid stellt einen Android-Adapter zur JADE-LEAP-Plattform dar. Er bestimmt die GPS-Position des Geräts, übersetzt diese in eine JSON-kodierte Zeichenkette und versendet sie in einem konfigurierbaren Intervall an alle anderen Agenten (siehe Abb. 11.10, links).

Die eingehenden Nachrichten von anderen Agenten werden auf einer Karte mit einem Zeitstempel angezeigt (siehe Abb. 11.10, Mitte). Die benötigten Kartendaten werden offline von einer lokal gespeicherten OpenStreetMap-Karte bezogen. Für spätere Auswertungen werden die transferierten Mitteilungen einer Datei protokolliert (siehe Abb. 11.10, rechts). Wichtige Voraussetzung für den Betrieb der

JadeAndroid-Agenten ist ein Main Container auf einem stationären PC bzw. Server mit einer festen IP-Adresse.

Der Agent wurde in einem Feldversuch auf mehreren Geräten bezüglich der Empfindlichkeit der Kommunikation in Abhängigkeit von folgenden Parametern untersucht:

- Gegend: Stadt, Land
- Geschwindigkeit: 30, 50 und 70 km/h
- Mobilfunkprovider: o2, Telekom
- Netztyp: GSM/GPRS/EDGE, 3G

Die Aktualisierungsfrequenz der Position betrug zunächst eine Sekunde. Beim Netztyp EDGE und darunter war damit keine Kommunikation möglich. Bei Geräten mit 3G-Unterstützung war eine Kommunikation möglich, aber fehlerhaft. Bei einer Aktualisierungsfrequenz von 10 Sekunden konnten alle Nachrichten mit einer tolerierbaren Latenz (< 5 Sekunden) übertragen werden. Auftretende Funkzellenwechsel hatten dabei keine spürbaren Auswirkungen auf die Übermittlung der Nachrichten. Auch bei Verbindungsverlusten von 30 Sekunden war ein nahtloses Weiterarbeiten möglich. Somit konnte über die gesamte Fahrtstrecke die Positionen der simulierten Ernte-Team Mitglieder echtzeitnah verfolgt werden. Diese Tatsache spricht für die weitere Untersuchung der Software-Agenten Technologie im Bereich der Landwirtschaft.

11.5 Zusammenfassung und Ausblick

Für die beiden Anwendungsgebiete Intralogistik in der Fabrik und Transportlogistik am Beispiel von Ernteprozessen wurden Erweiterungen der Standard-Agentenplattform (JADE) um mehrere Module durchgeführt:

- Unterstützung von Embedded-Systemen
- grafischer Editor für die Definition des Agentenverhaltens
- Integration einer zentralisierten Auftragsverwaltung für Ernteprozesse
- Robuste Kommunikationsarchitektur für den mobilen Einsatz von Software-Agentensystemen im ländlichen Raum
- Einbeziehung der mobil agierenden Teilnehmer des Geschäftsprozesses in einer kontext- und agentenbasierten Umgebung zur Entscheidungsunterstützung für die Nutzung in Standard-Tablet-Computern und Smartphones.

Es zeigte sich, dass durch die beschriebenen Erweiterungen die Akzeptanz und Anwendbarkeit im jeweiligen Umfeld gesteigert werden konnten.

Um die geforderte Plattformunabhängigkeit weiter auszubauen und die Nachladbarkeit von Geschäftslogik und Optimierungsfunktionen zu ermöglichen, wird mittels neuer HTML5-Funktionen eine webbasierende Agentenumgebung entwickelt, deren Prototyp in der Ernteperiode im Herbst 2012 zum Einsatz kommen wird.

Der SW-Agenten-Ansatz hat sich gegenüber traditionellen service-orientierten Ansätzen wie sie in [6, 7] untersucht wurden, als flexibler und plattformunabhängiger erwiesen. In weiterführenden Arbeiten wird vergleichend untersucht, ob einzelne Geräte und Dienste auch mit leichtgewichtigen Ansätzen service-orientierter Architekturen integriert werden können, um die Notwendigkeit von Software-Agenten auf jeder teilnehmenden Plattform umgehen zu können.

Danksagung Diese Arbeit wurde unterstützt vom BMBF, von der Volkswagen Stiftung, den Industriepartnern Volkswagen, PhoenixContact, Beka Engineering, CLAAS, Grimme und der Universität Osnabrück.

Literatur

- [1] Geisberger, E., Broy, M. (Hrs.): agendaCPS – Integrierte Forschungsagenda Cyber-Physical Systems. Bericht der Deutschen Akademie für Technikwissenschaft (2012). <http://www.acatech.de/de/publikationen/materialienbaende/uebersicht/detail/artikel/forschungsfragen-in-produktionsautomatisierung-der-zukunft.html> http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Publikationen/Projektberichte/acatech_STUDIE_agendaCPS_Web_20120312_superfinal.pdf. Zugegriffen: 04.07.2012
- [2] KOMOBAR: KOMOBAR Projektwebsite. <http://www.komobar.de/> (2012). Zugegriffen: 4. Juli 2012
- [3] Cannata, A., Gerosa, M., Taisch, M.: Socrates: A framework for developing intelligent systems in manufacturing. In: IEEM, IEEE International Conference, S. 1904–1908, S. 192–198 (2008)
- [4] Jammes, F., Mensch, A., Smit, H.: Service-oriented device communications using the devices profile for web services. In: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing, S. 1–8. ACM New York (2005)
- [5] Peschke, J., Luder, A., Kuhnle, H.: The pabadis' promise architecture – a new approach for flexible manufacturing systems. ETFA 10th IEEE, **1**, 6, 496 (2005)
- [6] Loeser, Ch. et al.: Context Sensitive Configuration and Execution Distributed Business Processes. Electronic Communications of the EASST, 17 (2009)
- [7] OSGi Alliance (2012). <http://www.osgi.org/>. Zugegriffen: 18. Mai 2012
- [8] DFKI (2012). <http://www.dfgi.de/>. Zugegriffen: 22. April 2012.
- [9] iGreen (2012). <http://www.igreen-projekt.de/>. Zugegriffen: 22. April 2012
- [10] Shoham, Y.: An Overview of Agent-Oriented Programming, S. 271–290. (1990)
- [11] Foundation for Intelligent Physical Agents: Abstract Architecture Specification. (2002).
- [12] Foundation for Intelligent Physical Agents: FIPA ACL Message Structure Specification. (2002)
- [13] Leszczyna, R.: Evaluation of Agent Platforms Version 2.0. (2008)
- [14] Tilab: Java Agent DEvelopment Framework. (2012). <http://jade.tilab.com/>. Zugegriffen: 20. April 2012
- [15] Bellifemine, F., Caire, G., Greenwood, D., NetLibrary, I.: Developing Multi-agent Systems with JADE. (2007)
- [16] Proksch, P., Westerkamp, C., Wuebbelmann, J.: Decision-making embedded devices using FIPA-ACL communication. INDIN 7th IEEE, S. 696–701. (2009)

- [17] Quindt, J., Reetz, E., Kukuck, K., Tönjes R., Westerkamp C.: Agent Based Decision Support System for Optimizing Logistical Processes in Agricultural Production. INDIN 9th IEEE, S. 27–32. (2011)
- [18] Nordemann, F., Tönjes, R.: Transparent and Autonomous Store-carry-forward Communication in Delay Tolerant Networks (DTNs). ICNC, S. 761–765. (2012)
- [19] World Wide Web Consortium: The WebSocket API (2012). <http://dev.w3.org/html5/websockets/>. Zugegriffen: 14. Mai 2012
- [20] Tilab: JADE Tutorial. JADE Programming for Android. (2011)
- [21] Schmidtmann, U. et al.: Leicht konfigurierbare Komponenten kollaborativer Systeme (LK3S). 6. Automatisierungstage 06, S. 192–198. (2008)
- [22] Jänsch, M.: Wenn Maschinen miteinander sprechen. Eilbote **2012**(14), 18–21 (2012)

Kapitel 12

Agentenbasierte Verteilnetzautomatisierung

Sebastian Lehnhoff und Olav Krause

Zusammenfassung Der Beitrag beschreibt ein Verfahren zur integrierten Netz-zustandsbewertung und ggf. Identifikation geeigneter Rekonfigurationsmaßnahmen zur Stabilisierung unzulässiger Betriebszustände. Das Verfahren besitzt eine deterministische Laufzeit und lässt sich so unter Echtzeitanforderungen in zukünftigen agentenbasierten Verteilnetzautomatisierungssystemen einsetzen, in denen auf entsprechende Phänomene oder Störungen autonom und rechtzeitig reagiert werden muss, bevor es zu Ausfällen von Betriebsmitteln und damit großräumigen Störungen kommt. Die Möglichkeit einer SVM-Regression und damit einer funktionalen Darstellung der Constraints zulässiger Betriebsgrenzen im Raum der Knotenleistungen ermöglicht eine effiziente Bewertung und Orientierung von Betriebspunkten hinsichtlich relevanter Grenzen. Das hier kompakt beschriebene Verfahren kommt derzeit in unterschiedlichen Fu E-Projekten sowohl an der University of Queensland als auch am OFFIS – Institut für Informatik zum Einsatz und wird kontinuierlich weiterentwickelt. Eine wichtige Fragestellung ist in diesem Zusammenhang die Entwicklung von Entwurfs- und Analysemethoden für verteilte selbstorganisierende Agentensysteme und die dem System zugrunde liegende IT-Infrastruktur, über die wir in zukünftigen Veröffentlichungen berichten.

Sebastian Lehnhoff (✉)

Institut für Informatik, FuE-Bereich Energie, OFFIS, Escherweg 2,
26121 Oldenburg, Deutschland

e-mail: lehnhoff@offis.de

Olav Krause

School of Information Technology and Electrical Engineering, The University of Queensland,
QLD 4072 Brisbane, Australien
e-mail: o.krause@uq.edu.au

12.1 Das elektrische Energieversorgungssystem

Elektrische Energienetze sind im Wesentlichen in zwei Ebenen gegliedert: vermaschte Höchst- und Hochspannungsnetze (aufgrund ihrer Funktion als Transport- bzw. Übertragungsnetz bezeichnet), die für einen intra- und internationalen Leistungstransport von Großerzeugern zu Lastzentren sorgen, sowie Mittel- und Niederspannungsnetze, die innerhalb eines Lastzentrums Leistung weiter verteilen und Strom kleinerer Kraftwerke aufnehmen (aufgrund ihrer Funktion als Verteilnetze bezeichnet). Hierbei übernehmen die Mittelspannungsnetze die Aufgabe der Leistungsverteilung über mittlere Distanzen, während die Niederspannungsnetze die Feinverteilung bis zu den Endkunden übernehmen. Die Verteilnetze werden weitestgehend passiv betrieben und verfügen neben automatisierten Einrichtungen zu Schutzzwecken heutzutage noch über keine nennenswerte Mess- und Steuerungstechnik. Eine vermehrt dezentrale Stromeinspeisung in den Nieder- und Mittelspannungsnetzen sowie die Steuerung regelbarer Verbraucher für einen gezielten Lastausgleich machen jedoch eine adaptive Mess-, Steuer- und Regelung der Verteilnetze erforderlich.

In bisherigen elektrischen Energieversorgungsmodellen wird eine Vielzahl möglicher Störfälle in messtechnisch nicht erfassten Verteilnetzen durch eine erhöhte Auslegung der Netzkapazität vermieden. Die Auslegung orientiert sich hierbei an möglichen Worst-Case-Szenarien unter Berücksichtigung sog. Gleichzeitigkeitsfaktoren (einer statistischen Beschreibung des aggregierten Lastverhaltens) und geht i. d. R. von einer Einspeisung auf den oberen sowie einem Leistungsbezug auf den unteren Spannungsebenen aus. Hierbei werden dezentrale Einspeisungen bei der Auslegung bislang nur in Ausnahmefällen berücksichtigt. Derart ausgelegte Netze können unter konventionellen Betriebsbedingungen (unter Vernachlässigung von Betriebsmittelausfällen) nicht überlastet werden. Diese Worst-Case-Auslegung von elektrischen Betriebsmitteln stößt bei vermehrter dezentraler Einspeisung auf den unteren Spannungsebenen an ihre Grenzen, da Leistungsflüsse auftreten können, die der konventionellen top-down Versorgungsrichtung entgegenwirken, diese im Extremfall sogar umkehren können und so zu Problemen für konventionelle Schutz- und Leitechnik führen, die zur Gewährleistung der Versorgungssicherheit und Fehlerklärung in unidirektional belasteten Versorgungsnetzen installiert und konfiguriert sind. Bestimmte Anwendungsszenarien, die im Kontext von Smart Grids viel diskutiert werden, wie das Verschieben der Leistungsaufnahme ausgewählter Endgeräte mit zeitflexiblen Betriebszyklen zur Verstetigung von Schwankungen in Versorgungssituationen (Demand-Side Management, Demand Response), führen darüber hinaus zu einer Erhöhung der Gleichzeitigkeit von Lasten und reduzieren den Abstand zu Kapazitätsgrenzen, die bei der Netzplanung zugrunde gelegt wurden [1]. Dies gilt besonders für den Verbleib innerhalb zulässiger Spannungsbänder in strahlenförmigen Verteilnetzen. Mit der zunehmenden Integration von Elektrofahrzeugen wird dieses Problem noch weiter verschärft, da es sich bei der Ladung von Fahrzeugbatterien um eine Anwendung mit hoher Leistungsaufnahme und hoher zu erwartender Gleichzeitigkeit handelt.

12.1.1 Anforderungen an die Verteilnetzautomatisierung

Eine der wesentlichen Unterscheidungen im Bereich elektrischer Energieversorgungsnetze und ihrer Betriebsmittel ist die Unterscheidung zwischen Primär- und Sekundärtechnik. Die Primärtechnik umfasst alle Elemente, die direkt an der Übertragung elektrischer Energie beteiligt sind, während die Sekundärtechnik traditionell Einrichtungen des Messens, Steuerns und Regelns inklusive der benötigten Kommunikationsinfrastruktur umfasst. Zu den traditionellen Funktionen der Sekundärtechnik, die sich im Wesentlichen nach den Aspekten Schutz- und Leittechnik, zur Sicherstellung eines möglichst zuverlässigen, störungsminimalem und technisch-wirtschaftlich optimalen Betrieb gliedern, kommen in zunehmendem Maße Funktionen mit energiewirtschaftlicher Relevanz hinzu. Die Spannweite der von der Sekundärtechnik umzusetzenden Prozesse reicht hier von Schutzfunktionen innerhalb einzelner Schutzgeräte und Schaltanlagen bis hin zur Überwachung intra- und internationaler Höchstspannungsnetze (z. B. die Höchstspannungsnetze des europäischen Verbundnetzes). Innerhalb der, durch die Sekundärtechnik abgebildeten, Funktionen haben sich verschiedene Domänen herausgebildet (z. B. die Schutz- und Leittechnik), die problemspezifisch proprietäre, oft nicht interoperable, Lösungen und Standards hervorgebracht haben. Die steigende Systemkomplexität, die sich aus der zunehmenden Anzahl der Akteure (z. B. dezentrale Energieumwandlungsanlagen, Lastmanagementsysteme und zukünftig Elektromobilität) und der größeren Spannweite möglicher Netzbelaustungszustände ergibt, wird zwangsläufig zu einer Erhöhung der Anzahl und zu einer zunehmenden Vernetzung schutztechnischer und steuerungstechnischer Einrichtungen führen müssen. Wesentliche Gründe sind die Reduktion erreichbarer Prognosegüten auf regionaler und speziell lokaler Ebene sowie die sich aus der Perspektive traditioneller Schutzgeräte zunehmend dynamisch verschiebende Grenzen zwischen Normalbetrieb und Fehlerfall. Ein kontinuierlicher Austausch über den aktuell vorherrschenden Betriebszustand und die daraus abgeleitete Neubestimmung der Grenze zwischen Normalbetrieb und Fehlerfall durch die Schutzgeräte wird unumgänglich, um auch weiterhin eine höchstmögliche Zuverlässigkeit und Selektivität der fehlerlärenden Eingriffe in dieser hochdynamischen Umgebung gewährleisten zu können. Die derzeitigen, zumeist proprietären, domänenbezogenen Lösungen in Form von Hard- und Software stellen hier ein Hindernis dar. Zukünftig benötigt werden, sowohl hardware- als auch softwareseitig, herstellerunabhängige, offene Kommunikationsstandards und Datenmodelle, ohne die sich die absehbar benötigten, koordinierten Verteilnetzautomatisierungsprozesse nicht, oder nur stark verzögert und zu höheren Kosten, abbilden lassen werden.

12.1.2 Anforderungen an zukünftige Leitsysteme

Die Aufgabe eines Leitsystems ist der Betrieb und die Verwaltung einer großen Anzahl dezentraler Akteure und Messstellen und der damit verbundenen Prozesse und Aufgaben im elektrischen Energieversorgungssystem mit dem Ziel eines sicheren und zuverlässigen Betriebs des Gesamtsystems [2]. Bis dato beschränken sich elektrische Netzeleitsysteme im Wesentlichen auf die übersichtliche Darstellung des komplexen zeitlichen und räumlichen Systemzustands, um einen steuernden menschlichen Eingriff zu unterstützen oder überhaupt erst ermöglichen. Wie bereits dargestellt, wird sich die Qualität und Quantität der Steuer-, Regel-, und allgemeiner der Automatisierungsaufgaben und -prozesse im zukünftigen Smart Grid drastisch verändern, um der wachsenden Systemkomplexität zu begegnen.

12.2 Netznutzungskoordination in Verteilnetzen

Als zentrale Komponente eines derartigen Verteilnetzleitsystems beschreibt der vorliegende Beitrag die Netznutzungskoordination (s. Abb. 12.1), eine Abfolge von Algorithmen und Aufgaben, die von zukünftigen verteilten Softwareagenten auszuführen sind, um aus verfügbaren (lokal mess- und ableitbaren) Informationen einen Systemzustand zu berechnen und die verfügbaren Abstände zu relevanten Betriebsgrenzen unter Berücksichtigung verfügbarer (Anlagen- und Nutzer-)Freiheitsgrade effizient auszunutzen. Die Prozesse dieser Netznutzungskoordination sind im Einzelnen:

- Adaptive Zustandsbewertung: Die Bestimmung oder Schätzung von Betriebszuständen ist eine der wichtigsten Informationen in aktiven Smart Grids. Die Güte einer adaptiven Zustandsbewertung wird maßgeblich von der Verfügbarkeit und Qualität von Messdaten bestimmt. Die größten Herausforderungen sind die zu erwartende Ungleichmäßigkeit der Verteilung von Messstellen und -einrichtungen, ihre Heterogenität im Hinblick auf die Präzision und Zusammenstellung der erhobenen Messgrößen sowie die zu erwartenden stark unterschiedlichen Messintervalle. Ein Algorithmus zur Zustandsbewertung in diesem Umfeld muss ein hohes Maß an Flexibilität aufweisen und speziell robust sowie zuverlässig auf variierende Beobachtbarkeit und Verfügbarkeiten von Messgrößen reagieren. Er muss über leistungsfähige statistische Analysetechniken zur Detektion von Fehlinformationen und potentiellen Topologie- und Parameterirrtümern verfügen.
- Bestimmung und Nutzung von Verschiebepotenzial (Arbitration): Durch die Bestimmung zulässiger Betriebsgrenzen für die Betriebsmittel innerhalb eines Netzbereichs, unter Berücksichtigung ihrer sich überlagernden Einflüsse, können Betriebszustände integriert hinsichtlich dieser Grenzen bewertet werden. Entfernung zu Betriebsgrenzen werden automatisch in minimale Korrekturvektoren übersetzt, die der optimalen Aktivierung von verfügbaren Freiheitsgraden innerhalb eines betrachteten Netzbereichs entsprechen.

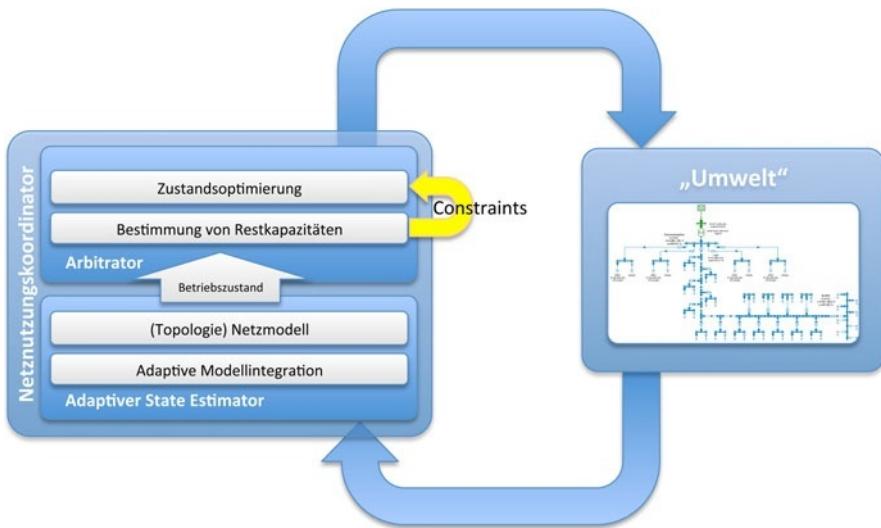


Abb. 12.1 Aufgaben der Netznutzungskoordination innerhalb eines Agenten

12.2.1 Agentenbasierte Netznutzungskoordination

Die Möglichkeit der Koordination dezentraler Energieanwendungen mithilfe von Multiagentensystemen wird derzeit im Smart Grid Umfeld in zahlreichen Arbeiten untersucht. Hierbei kann im Wesentlichen zwischen zentralisierten Ansätzen, wie die energiewirtschaftliche Aggregation von Erzeugungskapazitäten zu virtuellen Kraftwerken [3] oder in Verteilnetzen unter besonderer Berücksichtigung lokaler Netzphänomene [4] und dezentralen/hierarchisch organisierten Arbeiten unterschieden werden, wie die marktplatzorientierte Koordination in [5] oder eine selbstorganisierende agentenbasierte Einsatzplanung für virtuelle Kraftwerke in [6] oder selbstorganisierende Verbünde in [7]. Zunehmend rücken auch die Auswirkungen koordinierender Eingriffe und den damit verbundenen Synchronisationen und Erhöhung der Gleichzeitigkeit auf die Verteilnetze in den Fokus relevanter Arbeiten. In diesem Zusammenhang gibt es eine Vielzahl von Ansätzen, die sich zumeist auf einzelne Aspekte der Auslastung von Verteilnetzen konzentrieren. So wird z. B. in [8] auf die Problemstellung der Spannungshaltung eingegangen. Dies ist allerdings nur ein Aspekt der Begrenzung der Belastbarkeit in Verteilnetzen. Ein weiterer wesentlicher Aspekt sind die Leitungs- und Transformatorauslastungen, also maximale thermische Belastungsströme.

Diese Arbeiten beschränken sich bislang jedoch weitestgehend auf die Detektion der Verletzung von Belastbarkeitsgrenzen. Zielfunktionen zur Identifikation optimaler/minimaler Eingriffe zur Wiederherstellung der Zulässigkeit des Betriebspunktes werden bislang – wenn überhaupt – stark vergröbert betrachtet oder heuristisch gelöst. Für eine präzise Identifikation stabilisierender betrieblicher Rekon-

figurationen muss jedoch zunächst die Menge aller zulässigen Leistungskombinationen – abgegrenzt durch die jeweils betrachteten Belastbarkeitsgrenzen – bekannt sein. Die Bestimmung der Belastbarkeitsgrenzen ist bereits seit einiger Zeit Gegenstand der Forschung. In [9] wurde ein erster Ansatz vorgestellt, mit dem die Belastbarkeitsgrenzen im Raum der Knotenleistungen ermittelt wurden. Dieser wurde durch [10] ergänzt und es konnte gezeigt werden, dass die Menge der zulässigen Leistungskombinationen nicht immer konvex ist. In [11–13] konnten diese Ergebnisse bestätigt werden. Darüber hinaus ist es gelungen, eine Methodik zu entwickeln, mit der die Menge aller zulässigen Leistungskombinationen auf einem Netz mit PQ-Knoten zu bestimmen. Zulässigkeit bedeutet in diesem Zusammenhang, dass keine Leitungs- oder Transformatorüberlastungen auftreten, an allen Knoten des Netzes das Spannungsband eingehalten wird und es nicht zu Spannungs- oder Winkelinstabilitäten kommt.

12.3 Netzzustandsberechnung

Vor dem Hintergrund der Nachteile bestehender Analyse- und Berechnungsverfahren wird an dieser Stelle ein Ansatz vorgestellt, der als Vorlage für eine Bestimmung der Belastbarkeitsgrenzen von Verteilnetzen dienen soll und in zukünftigen dezentralen agentenbasierten Verteilnetzautomatisierungssystemen zum Einsatz kommen kann. Der grundlegende Unterschied zu bereits existierenden Ansätzen besteht darin, die Belastbarkeitsgrenze nicht im Raum der Knotenleistungen zu ertasten, sondern von ihrer Entstehung aus ihre Abbildung in den Raum der Knotenleistungen zu verfolgen und sie dort in Gänze darzustellen. Hierdurch wird nicht nur der Einfluss einzelner Akteure berücksichtigt, sondern der Raum möglichen gemeinsamen Verhaltens durch die Belastbarkeitsgrenzen beschränkt. Als die wesentlichen, die Belastbarkeit begrenzenden Aspekte werden das zulässige Spannungsband und die zulässigen Leitungs- und Transformatorströme berücksichtigt. Die Darstellung der Räume basiert auf linearen Ungleichungen, deren Verwendung wiederum die Konvexität der beschriebenen Mengen voraussetzt.

12.3.1 Bestimmung der Räume zur Spannungshaltung

Das zulässige Spannungsband definiert ein Intervall zulässiger Beträge aller komplexwertigen Knotenspannungen eines Netzes. In der komplexen Ebene einer Knotenspannung bilden die zulässigen Werte einen Ring mit oberer Spannungsgrenze (hier 110 %, oder 1.1 per unit (pu)) und unterer Spannungsgrenze (hier 90 %, oder 0.9 pu) (siehe Abb. 12.2). Alle Spannungen innerhalb des Rings sind nach dem Kriterium der Spannungsgrenzen zulässig.

Wird dieser Ring durch die nichtlinearen Leistungsflussgleichungen in den Raum der komplexwertigen Knotenleistungen abgebildet, wird das im Raum der

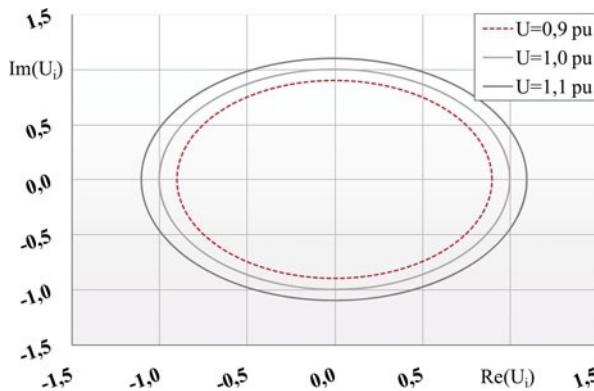


Abb. 12.2 Spannungsring in der komplexen Ebene der Knotenspannungen

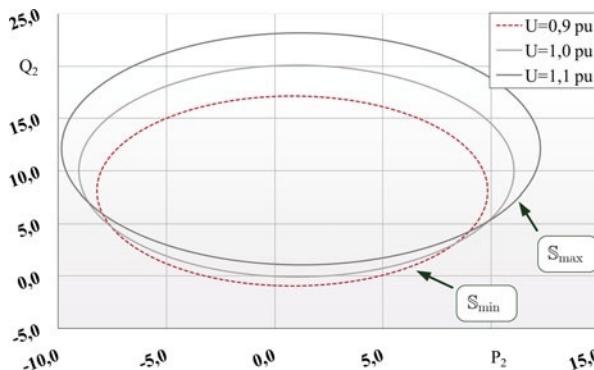


Abb. 12.3 Spannungsring abgebildet in die komplexe Ebene der Knotenleistungen

komplexen Knotenspannungen zusammenhängende Gebiet der zulässigen Spannungen auf zwei disjunkte Mengen abgebildet (siehe Bild 12.3).

Mathematisch sind Leistungskombinationen beider Gebiete zulässig, allerdings ist nur das Gebiet, das den Ursprung enthält, technisch relevant. Da für die gewählte Beschreibung der Teilräume Konvexität vorausgesetzt ist, werden die beiden Räume \mathbb{S}_{\min} und \mathbb{S}_{\max} zur Beschreibung der beiden disjunkten Gebiete wie folgt definiert. Der Teilraum \mathbb{S}_{\min} umfasse alle Leistungskombinationen, die in dem, durch das Bild der unteren Spannungsgrenze umschlossenen, Teilraum liegen, während der Teilraum \mathbb{S}_{\max} alle Leistungskombinationen enthalte, die in dem, durch das Bild der oberen Spannungsgrenze umschlossenen, Teilraum liegen. Somit kann die Menge der technisch relevanten Leistungskombinationen mit Gl. 12.1 beschrieben werden. Ist eine Leistungskombination Element dieser Menge, führt ihre Umsetzung nicht zu einer Verletzung des Spannungsbandes

$$S \in \mathbb{S}_{\min} \setminus S \notin \mathbb{S}_{\max} . \quad (12.1)$$

Das zweite Gebiet, auf welches das Spannungsbandsystem abgebildet wird kann mit Gl. 12.2 beschrieben werden. Es handelt sich zwar um mathematisch gültige Lösungen. Sie sind allerdings technisch nicht relevant, da kein Pfad vom Ursprung (also dem Flat-Start) in dieses Gebiet führt, der keine unzulässigen Werte durchläuft und Trajektorien in dieses Gebiet Grenzen der statischen Stabilität passieren

$$S \in \mathbb{S}_{\max} \bigvee S \notin \mathbb{S}_{\min} . \quad (12.2)$$

Die zuvor dargestellten Ergebnisse beziehen sich auf ein Beispielnetz mit zwei Knoten. Hierbei übernimmt einer der Knoten die Rolle des Referenzknotens, was wiederum der Einspeisung aus einem überlagerten Netz entspricht. Soll die Menge der zulässigen Leistungskombinationen für Netze mit mehr als zwei Knoten bestimmt werden, muss das Spannungsbandsystem als kartesisches Produkt der Mengen der zulässigen Spannungswerte der einzelnen Knoten dargestellt werden. Um später eine Darstellung mithilfe von linearen Ungleichungen zu ermöglichen, werden konvexe Mengen benötigt. Daher werden an dieser Stelle für jeden Knoten $i \neq r$ (r : Referenzknoten) separat die Mengen $\mathbb{S}_{\min,i}$ und $\mathbb{S}_{\max,i}$ bestimmt. Hierzu wird jeweils das kartesische Produkt aller Spannungen minimalen bzw. maximalen Spannungsbetrags des jeweiligen Knotens mit den vollständigen Spannungsbändern der anderen Knoten gebildet und über die Leistungsflussgleichungen (LFG) abgebildet. Auf diese Weise werden für ein Netz mit n Knoten, von denen einer als Referenzknoten fungiert, $2(n-1)$ Mengen gebildet. Für den Referenzknoten r wird hierbei eine konstante Spannung angenommen (vgl. Gl. 12.3 und 12.4)

$$\left(\begin{array}{l} \{|U_r| = U_{ref}\} \times \{|U_i| = U_{\min,i}\} \\ \times \bigcap_{\substack{j=1 \\ j \neq i \\ j \neq r}}^n \{U_{\min,j} \leq |U_j| \leq U_{\max,j}\} \end{array} \right) \xrightarrow{\text{LFG}} \mathbb{S}_{\min,i} \quad (12.3)$$

$$\left(\begin{array}{l} \{|U_r| = U_{ref}\} \times \{|U_i| = U_{\max,i}\} \\ \times \bigcap_{\substack{j=1 \\ j \neq i \\ j \neq r}}^n \{U_{\min,j} \leq |U_j| \leq U_{\max,j}\} \end{array} \right) \xrightarrow{\text{LFG}} \mathbb{S}_{\max,i} . \quad (12.4)$$

Auf Basis der so bestimmten Mengen ist es nun möglich einen beliebigen Vektor von Knotenspannungen S auf seine Zulässigkeit im Hinblick auf Spannungsbandsystemverletzungen zu bewerten. Der Ausdruck 12.1 für ein zweiknotiges Netz geht für größere Netze in Gl. 12.5 über

$$S \in \bigcup_{\substack{i=1 \\ i \neq r}}^n \mathbb{S}_{\min,i} \wedge S \notin \bigcup_{\substack{i=1 \\ i \neq r}}^n \mathbb{S}_{\max,i} . \quad (12.5)$$

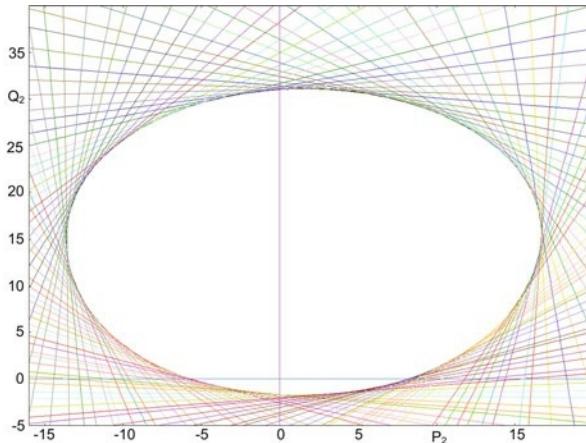


Abb. 12.4 Schnitt durch $\mathbb{S}_{\min,i}$

Abbildung 12.4 stellt exemplarisch einen Schnitt durch einen vier-dimensionalen Ergebnisraum $\mathbb{S}_{\min,i}$ dar (für ein Netz mit 2 Knoten).

12.3.2 Bestimmung der Räume zur Vermeidung zu hoher Leitungs- und Transformatormatrizen

Neben der Einhaltung des Spannungsbandes ist der Verbleib der Strombelastung der Leitungen und Transformatoren der zweite wesentliche kapazitätsbegrenzende Aspekt des Netzbetriebs. Diese Menge lässt sich in der komplexen Ebene eines jeden Leistungsstroms als Scheibe zulässiger komplexer Leistungsströme beschreiben (vgl. Abb. 12.5)

Die zulässigen Leistungsstromkombinationen lassen sich wiederum als kartesisches Produkt auf den jeweiligen Leitungen und Transformatoren zulässigen Ströme darstellen (vgl. Gl. 12.6)

$$\bigcap_{j=1}^n \{|I_j| \leq I_{\max,j}\}. \quad (12.6)$$

Zu beachten ist hier, dass nicht zwangsläufig alle leitungsindividuell zulässigen Leistungsstromkombinationen auch real auftreten können. In einem Netz mit n Knoten und m Leitungen kommen für $m \geq n$ Maschen vor, für die die Kirchhoff'sche Maschenregel gilt. Die Menge aller Leistungsstromkombinationen, die die Kirchhoff'sche Maschenregel erfüllen, entspricht dem Spaltenraum $SR(Y_1)$ der Leitungsadmittanzmatrix Y_1 . Die Schnittmenge von zulässigen und möglichen Leistungsstromkombinationen wird über die Pseudoinverse Leitungsadmittanzmatrix

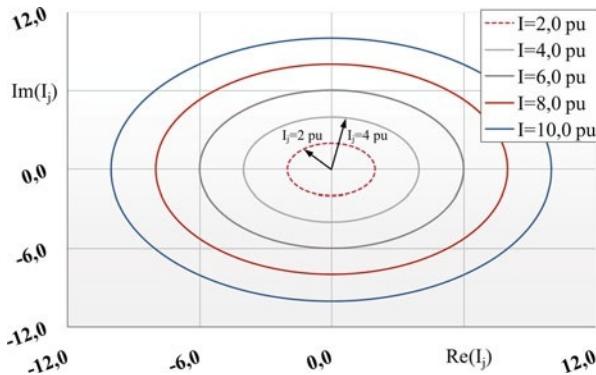


Abb. 12.5 Menge zulässiger Leistungs- bzw. Transformatorströme

Y_l^\dagger in den Raum der komplexen Knotenspannungen abgebildet

$$\left(\left(\bigcap_{j=1}^m \{ |I_j| \leq I_{\max,j} \} \right) \cap SR(Y_l) \right) \xrightarrow{Y_l^\dagger} \mathbb{U}_{I_{\max}} . \quad (12.7)$$

Die Elemente der Menge $\mathbb{U}_{I_{\max}}$ erfüllen nicht zwangsläufig die vorgegebene Knotenspannung des Referenzknotens und müssen dementsprechend angepasst werden. Da die Leitungsadmittanzmatrix immer den Rang $(n-1)$ hat existiert immer ein rechter Nullraum, der vom n -ten rech-ten Singulärvektor aufgespannt wird. Ein beliebiges Vielfaches dieses Singulärvektors kann auf jedes Element von $\mathbb{U}_{I_{\max}}$ addiert werden, ohne dass es zu einer Änderung der entsprechenden Leistungsstromkombination kommt. Diese Eigenschaft kann dazu verwendet werden, um für jedes Element von die Erfüllung der Referenzknotenspannung zu erreichen. Ist dies erfolgt, wird diese korrigierte Menge von Knotenspannungskombinationen mit Hilfe der Leistungsflussgleichungen in den Raum der Knotenleistungen abgebildet. Als Beispiel ist in Abb. 12.6 das Ergebnis für ein zweiknotiges Netz dargestellt.

Eine Kombination von Leistungen, die innerhalb der Vereinigung der Mengen (in Abb. 12.6 durch gestrichelte Linie angedeutet) liegt, führt nicht zu Überschreitungen der zulässigen Maximalströme auf Leitungen oder Transformatoren des jeweiligen Netzes.

12.3.3 Zusammenführen der Teilergebnisse

Werden die Teilergebnisse kombiniert, ergibt sich die Menge aller Betriebspunkte, die weder zu Spannungsbandverletzungen, noch zu Leistungs- oder Transformator-

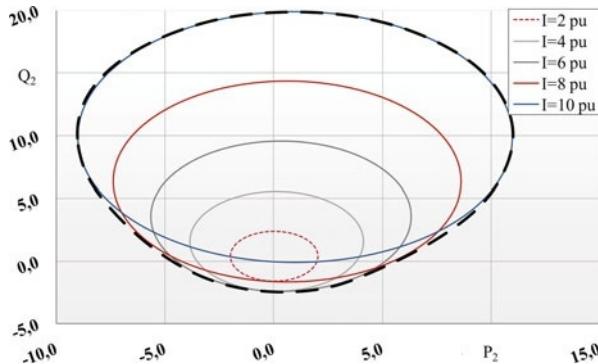


Abb. 12.6 Zulässige Leistungen unterhalb eines maximalen Leistungsstroms

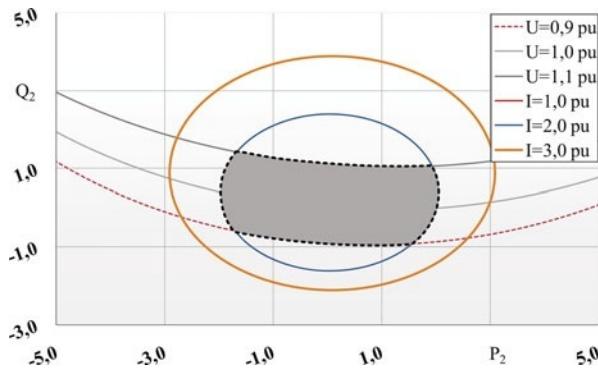


Abb. 12.7 Integrierte Darstellung zulässiger Leistungskombinationen

überlastungen führen. Diese ist in Abb. 12.7 für einen maximalen Leistungsstrom von 2 pu als graue Fläche angedeutet.

Ein Koordinationsverfahren zwischen Verbrauchern und Erzeugern elektrischer Energie kann auf Basis dieser Menge zum einen entscheiden, ob ein angestrebter Betriebspunkt aus Perspektive des Netzbetriebs zulässig ist, zum anderen bei Unzulässigkeit korrektive Maßnahmen erkennen und einleiten. Eine funktionale Abbildung der Betriesgrenzen im Raum der Knotenleistungen ist bislang nicht bekannt. Aus diesem Grund werden die Grenzen zulässiger Knotenspannungen und Leistungsströme in ihren Ursprungsräumen abgetastet und die Mengen einzelner Extrembetriebspunkte wie in Abschn. 12.3.1 und 12.3.2 abgebildet und kombiniert. Das Ergebnis sind Punktwolken, sog. \mathcal{V} (ertex)-Polytope $P_{\mathcal{V}} = \text{conv}(p_1, \dots, p_m)$ in \mathbb{R}^n , die die gesuchten Mengen repräsentieren. Das bislang bei der Netznutzungskoordination (Arbitrator) umgesetzte Verfahren stellt diese für eine Bewertung von Betriebszuständen (hinsichtlich zulässiger Knotenspannungen und Leistungsströme) notwendigen konvexen Teilmengen als sog. \mathcal{H} (alfspace)-Polytope $P_{\mathcal{H}} = \{x \in (\mathbb{R})^n : \vec{n}_0 \cdot x \leq b\}$ dar. Ein \mathcal{H} -Polytop ist der Schnitt einer endlichen Menge von

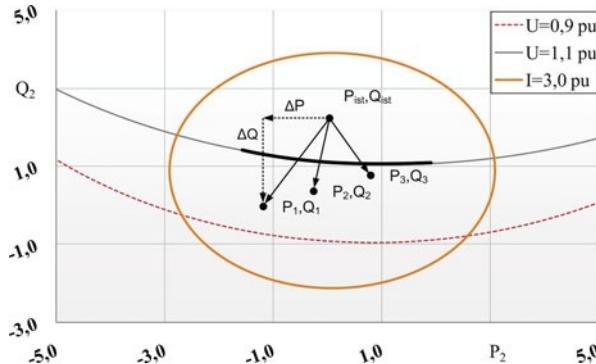


Abb. 12.8 Betriebspunkt außerhalb zulässiger Betriebsgrenzen und identifizierter Korrekturvektor

Halbraumdefinitionen der Form:

$$s \leq \vec{n}_0 \cdot \vec{p} - b . \quad (12.8)$$

Diese Darstellung der Halbraumdefinitionen in Hesse'scher Normalform erlaubt, den Abstand s eines beliebigen Punktes p , gegeben als Ortsvektor \vec{p} , zur Hyperebene durch einfaches Einsetzen in die Normalform zu berechnen. Die Berechnung eines \mathcal{H} -Polytops $P_{\mathcal{H}}$ aus einem \mathcal{V} -Polytop $P_{\mathcal{V}}$ bezeichnet man allgemein als Facetten-Aufzählungsproblem (*Facet Enumeration Problem*). Ist $P_{\mathcal{H}}$ außerdem beschränkt, lässt sich das Problem auf das sog. Konvexe-Hüllen-Problem reduzieren (*Convex Hull Problem*). Die Berechnung einer konvexen Hülle aus einem gegebenen \mathcal{V} -Polytop ist sehr aufwändig. Die Anzahl von Facetten, gegeben als Ungleichungen der Form $\vec{n}_0 \cdot x \leq b$, wächst dabei exponentiell in der Zahl der Dimensionen n und der Punkte des \mathcal{V} -Polytops m . Ein Punkt q liegt im \mathcal{H} -Polytop der zulässigen Betriebspunkte genau dann, wenn er alle Ungleichungen erfüllt, also unter allen Facetten liegt. Die Anzahl der Facetten ist in der Regel sehr viel größer als die Anzahl der Punkte des \mathcal{V} -Polytops, aus denen die Hülle berechnet wurde (ausgenommen innere und linear abhängige Punkte) [14]. Das Problem der Bewertung des betrachteten Punktes q gegen alle Facetten-Ungleichungen lässt sich stark parallelisieren, da die Ungleichungen parallel ausgewertet werden können und ein gemeinsames Ergebnis aller Vergleiche nicht aufwändig aus der Kombination der Teilergebnisse ermittelt werden muss. Verletzt ein Betriebspunkt zulässige Betriebsgrenzen, kann über eine effiziente Verwaltung der Hyperebenen (Facetten) in konstanter Zeit die Halbraumdefinition mit dem geringsten Abstand zum Betriebspunkt identifiziert werden. Eine Korrektur des Betriebspunktes entlang des Normalenvektors dieser Facette $-\vec{n}_0$ um den Abstand zu dieser Hyperebene stellt dann den minimalen Eingriff der Versorgungskonfiguration zur Stabilisierung des Betriebspunktes dar (s. Abb. 12.8).

12.3.4 Klassifizierung der Menge zulässiger Betriebspunkte über Support Vector Machines

Support Vector Machines (SVM) sind ein statistisches Verfahren zur Mustererkennung. Eine SVM ist ein Klassifikator, der eine Menge von Trainingsobjekten $T = \{(x_1, y_1), \dots, (x_N, y_N)\}$, die mit $y_1, \dots, y_N \in \{-1, 1\}$ gelabelt sind (aus denen die Klassenzugehörigkeit, hier ‘‘Zulässigkeit’’, hervor geht), so in Klassen unterteilt, dass um die Klassengrenzen herum ein möglichst breiter Bereich frei von Objekten bleibt. Unter der Annahme, dass die Trainingsobjekte linear durch eine Hyperebene trennbar sind, konstruiert eine SVM eine Hyperebene (oder eine Menge von Hyperebenen) mit dem Normalenvektor β und dem Stützvektor β_0 , die zur Klassifizierung von Datensätzen verwendet werden können. Ein neuer Testvektor x kann dann entsprechend der Seite der Hyperebene, auf der er liegt, klassifiziert werden. Zu diesem Zweck wird die folgende Entscheidungsfunktion definiert:

$$f(x) = x^T \cdot \beta + \beta_0 = \sum_{i=1}^N x_i \cdot \beta_i + \beta_0 . \quad (12.9)$$

Während das Eingangsproblem in einem Raum mit einer bestimmten Anzahl an Dimensionen aufgestellt werden kann, müssen die zu klassifizierenden Mengen nicht unbedingt im gleichen Raum linear durch die Hyperebenen trennbar sein. In diesem Fall wird das Problem in höhere Raumdimensionen überführt, bis eine lineare Trennung vorgenommen werden kann. Das Ergebnis ist eine nicht-lineare (und nicht notwendigerweise zusammenhängende) Grenze im Ursprungsraum. Die Abbildung muss dabei selbst nicht bekannt sein, sondern lediglich das abgebildete Ergebnis des Skalarprodukts der Merkmalsvektoren. Dieses Skalarprodukt zweier Merkmalsvektoren x, x' wird mit einer sog. Kernelfunktion $k(x, x')$ berechnet.

Es existieren eine Reihe von Kernelfunktionen, die unterschiedlich strukturierte Probleme unterschiedlich gut klassifizieren. Das Ergebnis ist wieder eine Hyperebene im höher dimensionalen Raum, die mithilfe der Kernelfunktion beschrieben werden kann. Ein neuer Textvektor x kann dann in seinem (niedrig dimensionalen) Ursprungsraum über eine erweiterte Entscheidungsfunktion klassifiziert werden:

$$f(x) = \sum_{i=1}^N \hat{a}_i y_i k(x, x') + \beta_0 . \quad (12.10)$$

Die Parameter \hat{a}_i und β_0 werden dabei von der SVM automatisch gewählt, um eine möglichst guten Trade-off zwischen Klassifikationsperformanz und Abstand zu den Merkmalsvektoren zu erzielen. Für weitere Details verweisen wir auf [15] und [16]. Die Klassifikationsperformanz wird üblicherweise durch eine Verlustfunktion $L(\hat{y}(x), y(x))$ ausgedrückt, die einen Klassifikationsfehler, der einem Testvektor x anstelle des Labels $y(x)$ einen Label $\hat{y}(x)$ zuweist, eine bestimmte Strafe zuordnet. Eine häufig verwendete Verlustfunktion ist eine 0-1-Strafe, bei der

eine Fehlklassifikation mit einer Strafe von 1 belastet wird, während eine korrekte Klassifikation nicht bestraft wird. Bei dem Training von SVM wird hingegen häufig eine dynamische Strafe verwendet, die proportional zu dem Abstand zur trennenden Hyperebene ist.

Die beiden Seiten der Hyperebene lassen sich so durch das Vorzeichen der Entscheidungsfunktion bestimmen. Existiert mehr als eine Hyperebene, dann wählt die SVM diejenige Hyperebene, die den größten Abstand zu den Trainingspunkten besitzt. Diese Distanzmaximierung führt dazu, dass nur wenige Datenpunkte benötigt werden, um die Hyperebene zu ermitteln. Diese Datenpunkte, die die Merkmalsvektoren des Merkmalsraums darstellen, werden bei SVM als *Support Vectors* bezeichnet.

Dieses Klassifikationsverfahren lässt sich unmittelbar zur Klassifizierung der vorliegenden \mathcal{V} -Polytope anwenden. Erste Experimente hierzu sind äußerst vielversprechend und trennen den Zustandsraum scharf zwischen den gesuchten Mengen. Der Vorteil dieser Vorgehensweise ist unmittelbar ersichtlich: anstelle von $(2n - 1)$ konvexer Mengen in \mathbb{R}^n wird lediglich eine funktional beschriebene Menge in \mathbb{R}^n , beschrieben durch eine große Zahl an Halbraumdefinitionen, benötigt. Abb. 12.9 zeigt die SVM-Darstellung von zulässigen Spannungsbändern in einem Beispiel Netz mit 2 Knoten (vgl. hierzu auch Abb. 12.3).

Für die Verwendung von SVM bei der Zustandsklassifikation ist die Rechenkomplexität sowohl für das Training der SVM als auch für die Bewertung eines Trainingspunktes relevant. Die asymptotische Rechenkomplexität des SVM-Trainings (angegeben in Landau-Notation) ist $O(m^3 + mN + m d N)$, hierbei ist m die Anzahl der Merkmalsvektoren, N die Anzahl der Trainingspunkte und d die Dimensionalität der Trainingsdaten. Der Worst-Case beim Trainieren einer SVM ist der Fall, wenn alle Testvektoren als Merkmalsvektoren zur Beschreibung der Hyperebene im (höher dimensionalen) Merkmalsraum heran gezogen werden, in diesem Fall ($m=N$) lässt sich die Rechenkomplexität mit $O(m^3 + m^2 + m^2 d)$ angeben. Die Bewertung eines Testvektors skaliert linear mit der Anzahl der Merkmalsvektoren in $O(mK)$, dabei ist K die Anzahl der Rechenoperationen, die benötigt wird, um die Kernelfunktion zu berechnen.

Wie bereits diskutiert, sind bei der Verletzung von Betriebsgrenzen Richtungen (minimale Korrekturvektoren) zurück in die Menge zulässiger Betriebspunkte (s. Abb. 12.8) von Interesse. Bei der Verwendung von SVM kann diese Richtung direkt aus Gl. 12.10 bestimmt werden. Der Gradient der Entscheidungsfunktion 12.10 für einen als nicht zulässig klassifizierten Leistungsvektor x entspricht der Richtung auf die trennende Entscheidungsfunktion und daher in Richtung zulässiger Betriebspunkte:

$$\nabla_x f(x) = \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_N} \right)^T. \quad (12.11)$$

Mit Gl. 12.11 lässt sich nicht nur der kürzeste Weg zurück in die Menge zulässiger Betriebspunkte ermitteln, sondern auch eine Länge, die einer Information über die Stärke des notwendigen Eingriffs entspricht. Eine solche Korrektur eines unzulässigen Betriebspunktes lässt sich dann in eine optimale Aktivierung von zeit-

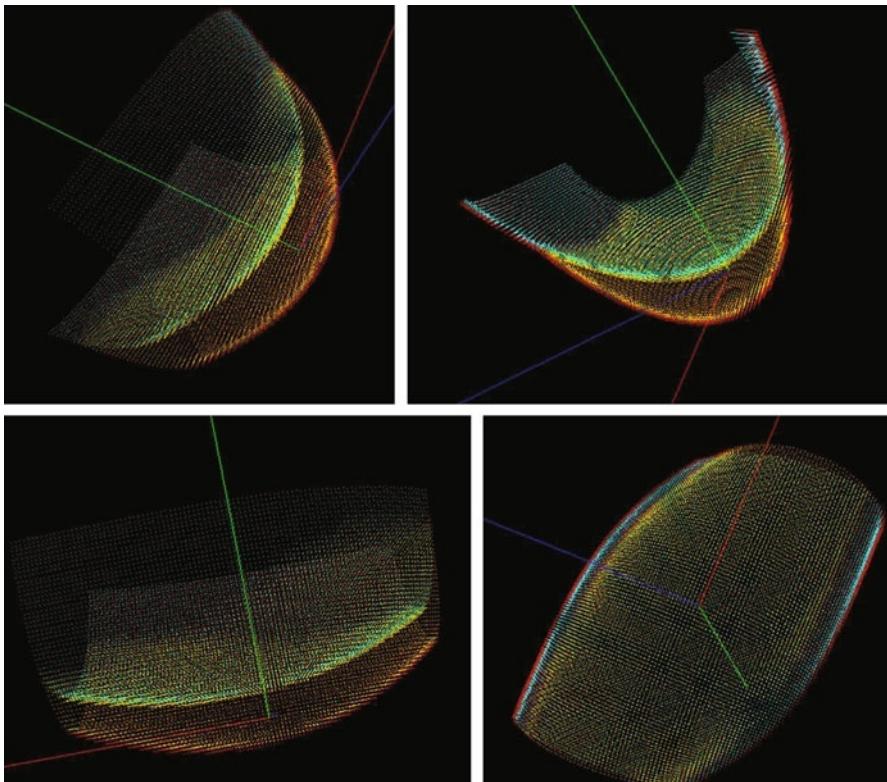


Abb. 12.9 3-dimensionaler Plot der SVM zur Klassifikation zulässiger Spannungsbänder in \mathbb{R}^4

flexiblen/dynamischen Verbrauchen und Erzeugern übersetzen, die in zukünftigen Smart Grids als Freiheitsgrade zu diesem Zweck bereit stehen können. Für eine erweiterte Fallstudie sei auf [17] verwiesen.

12.4 Zusammenfassung

Der Beitrag beschreibt ein Verfahren zur integrierten Netzzustandsbewertung und ggf. Identifikation geeigneter Rekonfigurationsmaßnahmen zur Stabilisierung unzulässiger Betriebszustände. Das Verfahren besitzt eine deterministische Laufzeit und lässt sich so unter Echtzeitanforderungen in zukünftigen agentenbasierten Verteilnetzautomatisierungssystemen einsetzen, in denen auf entsprechende Phänomene oder Störungen autonom und rechtzeitig reagiert werden muss, bevor es zu Ausfällen von Betriebsmitteln und damit großräumigen Störungen kommt. Die Möglichkeit einer SVM-Regression und damit einer funktionalen Darstellung der Constraints zulässiger Betriebsgrenzen im Raum der Knotenleistungen ermöglicht eine

effiziente Bewertung und Orientierung von Betriebspunkten hinsichtlich relevanter Grenzen. Das hier kompakt beschriebene Verfahren kommt derzeit in unterschiedlichen FuE-Projekten sowohl an der University of Queensland als auch am OFFIS - Institut für Informatik zum Einsatz und wird kontinuierlich weiterentwickelt. Eine wichtige Fragestellung ist in diesem Zusammenhang die Entwicklung von Entwurfs- und Analysemethoden für verteilte selbstorganisierende Agentensysteme und die dem System zugrunde liegende IT-Infrastruktur, über die wir in zukünftigen Veröffentlichungen berichten.

Literatur

- [1] Benze, Diedrich, Honecker et al.: Energieinformationsnetze und -systeme – Bestandsaufnahme und Entwicklungstendenzen. Positionspapier der Informationstechnischen Gesellschaft im VDE (ITG) (2010)
- [2] Scheppele, F.C., Handschin, E.J.: State Estimation in Electric Power Systems: A Generalized Approach. Proceedings of the IEEE **62**(7), 972–982 (1974)
- [3] Bitsch, R.: Perspektiven im Energiemanagement bei Stromversorgungsnetzen mit dezentraler Erzeugung. Tagungsband des Kasseler Symposium Energie-Systemtechnik, 178–189 (2000)
- [4] Thoma, M.C.: Optimierte Betriebsführung von Niederspannungsnetzen mit einem hohen Anteil an dezentraler Erzeugung. Dissertation. ETH Zürich (2007)
- [5] Kamphuis, R., Kok, K., Hommelberg, M. et al.: Massive Coordination of Dispersed Generation using Powermatcher based Software Agents. Proc. of the 19th International Conference on Electricity Distribution. Vienna (2007)
- [6] Tröschel, M., Appelrath, H.-J.: Towards Reactive Scheduling for Large-Scale Virtual Power Plants. Proc. of the 7th German Conference on Multiagent System Technologies (MATES'09), S. 141–152. (2009)
- [7] Nieße, A., Lehnhoff, S. et al. Market-Based Self-Organized Provision of Active Power and Ancillary Services: An Agent-Based Approach for Smart Distribution Grids. Proc. of the 2012 IEEE Workshop on Complexity in Engineering. Aachen (2012).
- [8] Wolter, M., Brenner, S., Isermann, T., Hofmann, L.: Application of Adaptive Agents in Decentralized Energy Management Systems for the Purpose of Voltage Stability in Distribution Grids. Proc. of the North American Power Symposium (2009)
- [9] Galiana, P.D., Jarjis, J.: Feasibility Constraints in Power Systems. Proc. of the IEEE Power Engineering Summer Meeting (1978)
- [10] Makarov, Y.V., Zhao, Y.D., Hill, D.J.: On Convexity of Power Flow Feasibility Boundary. IEEE Transactions on Power Systems **23** (2008)
- [11] Krause, O.: Bestimmung des Nutzungspotenzials von Verteilnetzen. Sierke Verlag, Göttingen (2009)
- [12] O. Krause, S. Lehnhoff, C. Rehtanz, E. Handschin, H.F. Wedde. On-line Stable State Determination in Decentralized Power Grid Management. Proc. of the 16th Power Systems Computation Conference (PSCC'08). Glasgow, UK (2008)
- [13] Krause, O., Lehnhoff, S., Rehtanz, C., Handschin, E., Wedde, H.F.: On Feasibility Boundaries of Electrical Power Grids in Steady State. IJEPES **31**(9), 437–444 (2009)
- [14] Lehnhoff, S.: Dezentrales vernetztes Energiemanagement – Ein Ansatz auf Basis eines verteilten adaptiven Realzeit-Multiagentensystems. Vieweg+Teubner, GWV Fachverlage GmbH, Wiesbaden (2010)

- [15] Cristianini, N., Shawe-Taylor, J.: An introduction to Support Vector Machines. Cambridge University Press, Cambridge (2000)
- [16] Schölkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and beyond. MIT Press, Cambridge (Massachusetts) (2002)
- [17] Blank, M., Gerwinn, S., Krause, O., Lehnhoff, S.: Support Vector Machines for an efficient Representation of Voltage Band Constraints. Proc. of the international IEEE Conference on Innovative Smart Grid Technologies (ISGT'11). Manchester, UK (2011)

Kapitel 13

Agentenbasiertes Selbstmanagement von Automatisierungssystemen

Hisham K. Mubarak

Zusammenfassung Die steigende Komplexität moderner Automatisierungsanlagen gefährdet zunehmend die Beherrschbarkeit der Systeme durch das Prozesspersonal und macht eine Unterstützung der technischen Betriebsbetreuung erforderlich. Darunter sind die wertschöpfenden Tätigkeiten des Prozesspersonals zusammengefasst, welche zur Organisation und Aufrechterhaltung des Betriebs einer Automatisierungsanlage erforderlich sind. Um dieses Ziel zu erreichen wird ein generisches agentenorientiertes Konzept für ein Selbstmanagementsystem vorgestellt, welches die effiziente Integration von Selbstmanagementfunktionen in Automatisierungsanlagen erlaubt. Ausgehend von den spezifischen Anforderungen an das Selbstmanagement von Automatisierungsanlagen wird eine grundlegende 3-Schicht-Architektur abgeleitet, welche mithilfe von sechs spezialisierten Agententypen umgesetzt wird. Das Funktionsprinzip des agentenbasierten Selbstmanagementsystems wird am Beispiel einer Personenaufzugssteuerung, die um Selbstheilungsfunktionalität erweitert wird, erläutert.

13.1 Einleitung

Die automatisierungstechnischen Entwicklungen und Innovationen der letzten Jahre zeigen einen deutlichen Trend zur Dezentralisierung von Automatisierungssystemen (vgl. [1]). Zusammen mit der gleichzeitig erfolgenden funktionalen Flexibilisierung von Automatisierungskomponenten führen diese Entwicklungen zu leistungsfähigeren, anpassbareren und wirtschaftlicheren Automatisierungssystemen, die jedoch eine immer höhere Gesamtkomplexität im Betrieb aufweisen und aufgrund der damit einhergehenden Überforderung des Betriebspersonals die Beherrschbarkeit der Systeme gefährden. Gleichzeitig unterliegt der moderne An-

Hisham K. Mubarak (✉)
BASF SE, Carl-Bosch-Str. 38, 67056 Ludwigshafen, Deutschland
e-mail: hisham.mubarak@bASF.com

lagenbetrieb vielfältigen Anforderungen [2]. Neben der eigentlichen Erfüllung des Produktionsziels, also der Herstellung von Produkten und Gütern in gewünschter Qualität und Menge, muss ein sicherer Anlagenbetrieb gewährleistet werden. Zusätzlich muss zur Sicherstellung einer globalen Wettbewerbsfähigkeit eine energieoptimierte und ressourcenschonende Produktionsweise sichergestellt werden. Nichtsdestotrotz müssen Aufwendungen für Instandhaltung sowie die Personalkapazität unter dem Druck des globalen Wettbewerbs reduziert werden. Die Situation wird in Zukunft zusätzlich dadurch erschwert, dass die Ausbildung von Fachkräften und der notwendige Aufbau von Erfahrung nicht mit dem starken Wachstum der Industrialisierung in Schwellenländern mithalten kann [2]. Dieser Trend wird durch den demografischen Wandel in westlichen Industrieländern verstärkt, mit der Folge, dass immer weniger qualifizierte Fachkräfte für die Bewältigung komplexerer Betriebsbetreuungsaufgaben zur Verfügung stehen [3].

Es ist daher unabdingbar das Betriebspersonal wo möglich zu entlasten und es bei der Durchführung technischer Betriebsbetreuungsaufgaben in bestmöglicher Weise zu unterstützen. Die verfügbare menschliche Expertise soll dabei in den wichtigen Funktionen effizient eingesetzt werden, während automatisierbare Funktionen durch ein Unterstützungssystem autonom und ohne Zutun des Betriebspersonals durchgeführt werden. Eine Möglichkeit zur Entlastung des Betriebspersonals und zur Beherrschung komplexer Systeme ist das Prinzip der Selbstorganisation [4]. Im Kontext des Betriebs komplexer Automatisierungssysteme bedeutet dies die autonome Durchführung von technischen Betriebsbetreuungsaufgaben durch Softwareagenten.

Der vorliegende Beitrag stellt eine agentenbasierte Erweiterung klassischer Automatisierungssysteme durch ein **Selbstmanagementsystem** vor, welches die schrittweise Einführung von Selbst-X-Funktionen (automatisierte Funktionen zur Unterstützung der Betriebsbetreuung), vgl. [5], in Automatisierungssysteme erlaubt. Der Beitrag ist wie folgt gegliedert: Abschn. 13.2 beschreibt die grundlegende Idee der Automatisierung von Betriebsbetreuungsaufgaben in Automatisierungssystemen durch Selbstmanagement-Funktionen (SMF), sowie grundlegende Anforderungen an ein Unterstützungssystem für technische Betriebsbetreuungsaufgaben. Abschn. 13.3 motiviert den Einsatz eines Agentensystems für die Entwicklung und autonome Durchführung von SMF. Daraus wird eine generische Architektur für ein Selbstmanagementsystem (SMS) abgeleitet und die eingesetzten Agententypen werden erläutert. Abschn. 13.4 beschreibt ein Anwendungsbeispiel für ein Selbstmanagementsystem. Schließlich werden die wichtigsten Aussagen und Ergebnisse in Abschn. 13.5 zusammengefasst.

13.2 Selbstmanagement zur Unterstützung der technischen Betriebsbetreuung von Automatisierungssystemen

13.2.1 Technische Betriebsbetreuungsaufgaben in Automatisierungssystemen

Ein automatisiertes System besteht aus drei interagierenden Teilsystemen [6]: Dem technischen System, dem Rechner- und Kommunikationssystem sowie dem Prozesspersonal. Das technische System dient der physischen Durchführung eines technischen Prozesses, beispielsweise ein Reaktor in einer verfahrenstechnischen Anlage. Das Rechner- und Kommunikationssystem (Automatisierungssystem) dient der Erfassung, Verarbeitung und Darstellung von Informationen über das Prozessgeschehen, sowie der Ausgabe von Informationen (z. B. Steuersignale) zur zielgerichteten Beeinflussung des Ablaufs des technischen Prozesses im technischen System. Das Prozesspersonal verfolgt das Prozessgeschehen und leitet die Vorgänge in den darunterliegenden Teilsystemen. Ebenfalls greift das Prozesspersonal in Ausnahmesituationen oder bei Störungen in das Prozessgeschehen ein.

Es gilt zu beachten, dass in modernen Automatisierungsanlagen die manuellen Tätigkeiten des Prozesspersonals zum einen der Führung des operativen Betriebs dienen und zum anderen sich mit der technischen Betriebsbetreuung (Inbetriebnahme, Instandhaltung, Optimierung und Sicherung des Automatisierungssystems) befassen. Das Ziel des Selbstmanagements ist die Unterstützung des Prozesspersonals durch Automatisierung der Betriebsbetreuungsaufgaben. Selbstmanagement beschreibt die Fähigkeit eines Automatisierungssystems den eigenen Betrieb ohne Zutun des Menschen zu managen [7–10]. Hierzu gehören insbesondere die Bereitstellung von Funktionen für Selbstkonfiguration, Selbstheilung, Selbstschutz und Selbstsicherung. Betriebsbetreuungsaufgaben lassen sich in die Kategorien Konfiguration, Heilung, Optimierung und Schutz einteilen. Durch ihre Automatisierung entstehen entsprechende Selbst-X-Funktionen [4, 11, 12].

13.2.2 Anforderungen an ein Selbstmanagementsystem in der Automatisierungstechnik

Abhängig von der Größe und der Komplexität einer Automatisierungsanlage kann die Anzahl des Prozesspersonals, welches mit Betriebsbetreuungsaufgaben befasst ist, variieren. Wie in Abb. 13.1 skizziert, findet eine Spezialisierung des Prozesspersonals statt. Aufgaben wie Prozesssicherung, Fehlerdiagnose, Störungsbehandlung, Prozessoptimierung oder Konfiguration von Anlagenelementen erfordern spezifische Sichtweisen, Vorgehensweisen und Expertenwissen zur Aufgabenbearbeitung. Zur Erfüllung dieser Aufgaben interagieren die jeweiligen Personen individuell und auf unterschiedliche Weise mit dem Automatisierungssystem und nutzen auf-

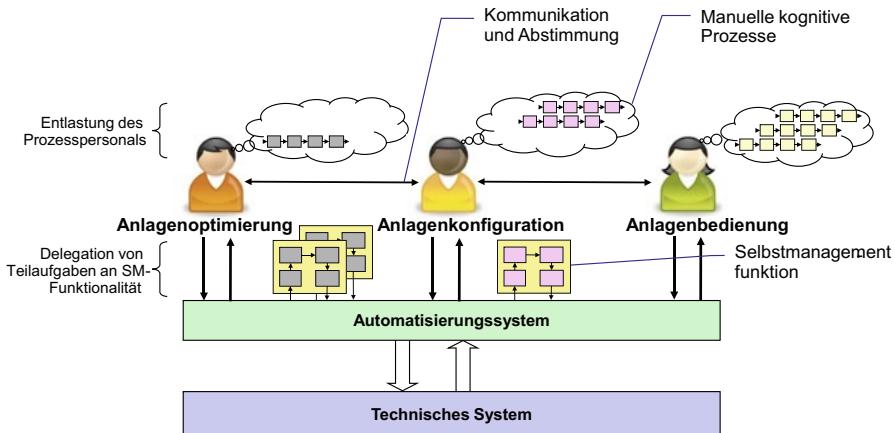


Abb. 13.1 Ansatz der Delegation von manuellen Aufgaben des Prozesspersonals an Selbstmanagementfunktionen des Automatisierungssystems

gabenspezifische Werkzeuge, welche die jeweilige Tätigkeit unterstützen. Bei der Durchführung von Eingriffen in das Automatisierungssystem besteht die Gefahr, dass sich einzelne Handlungen gegenseitig störend beeinflussen oder gar widersprechen und daher zeitlich koordiniert werden müssen. Zu diesem Zweck findet eine Abstimmung zwischen den zuständigen Personen des Prozesspersonals statt. Ist ein gemeinsamer Konsens erreicht, werden die Handlungen im Automatisierungssystem manuell durchgeführt. Grundlage für diese Interaktionen zwischen menschlichen Bedienern und Automatisierungssystem sind dem Menschen eigene kognitive Prozesse. Die Unterstützung des Betriebs von Automatisierungssystemen erfordert daher die Nachbildung und Automatisierung von kognitiven Prozessen durch ein Selbstmanagementsystem. Ein Selbstmanagementsystem ist ein Unterstützungssystem, das Selbstmanagementfunktionen für ein Automatisierungssystem bereitstellt. Im Selbstmanagementsystem werden sämtliche Prozesse zur Automatisierung von technischen Betriebsbetreuungsaufgaben realisiert. Ein Selbstmanagementsystem automatisiert jedoch keine technischen Prozesse im technischen System.

Der Begriff Selbstmanagementfunktion bzw. Selbst-X-Funktion beschreibt allgemein die Selbstmanagementfähigkeiten eines Computersystems. Dabei steht das X als Platzhalter für unterschiedliche Funktionen wie Konfigurations- oder Optimierungsmaßnahmen, die ein Computersystem selbstständig durchführen soll. Durch diese Vorgehensweise soll die mit dem Betrieb des Systems verbundene Komplexität reduziert werden. Insbesondere werden darunter Selbstkonfiguration, Selbstoptimierung, Selbstheilung und Selbstschutz verstanden.

Selbstmanagementfunktionen müssen nach dem Stellvertreterprinzip realisiert werden, da sie sich am menschlichen Vorgehen orientieren und eine Stellvertretung des Prozesspersonals erlauben sollen. Dementsprechend sind autonomes und zielorientiertes Handeln eine wesentliche Voraussetzung für Selbstmanagementfunktionen und müssen vom Entwicklungsparadigma unterstützt werden.

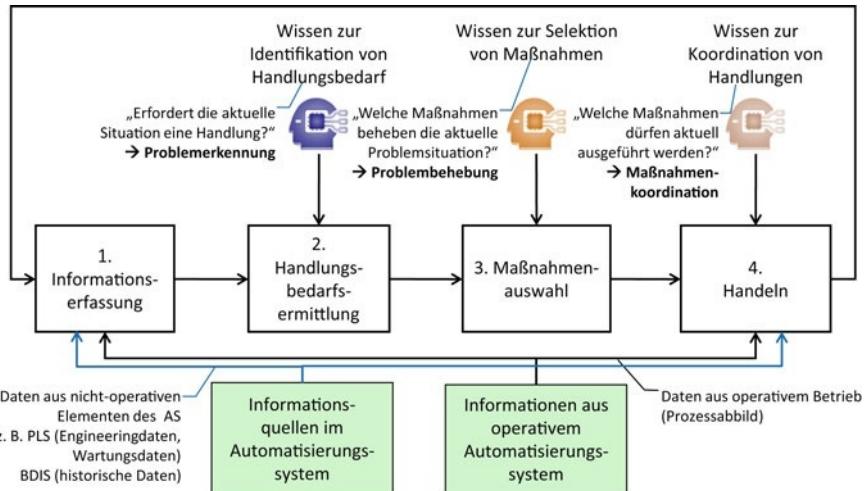


Abb. 13.2 Wissens- und Informationsbedarf für Selbstmanagementfunktionen

Die Entwicklung von Selbstmanagementfunktionen erfordert eine umfangreiche Integration von funktionsspezifischem Wissen, um einerseits Situationen im Automatisierungssystem zu identifizieren, welche einen Handlungsbedarf erfordern, und andererseits, um im Bedarfsfall geeignete Maßnahmen auszuwählen und im Automatisierungssystem umzusetzen. Darüber hinaus ist funktionsübergreifendes Wissen zur Koordination der Handlungsmaßnahmen aus unterschiedlichen Selbstmanagementfunktionen erforderlich, um beispielsweise Konflikte durch sich widersprechende Maßnahmen aufzulösen. In [11, 13, 14] wird ein Modell für Selbstmanagementprozesse beschrieben, welches eine systematische Analyse und Beschreibung von Selbstmanagementfunktionen erlaubt und damit die Grundlage für deren Automatisierung darstellt. Anhand dieses Modells lassen sich die drei genannten Wissenskategorien den einzelnen Prozessschritten zuordnen, in welchen sie benötigt werden (vgl. Abb. 13.2).

Für die automatisierte Durchführung von Selbstmanagementfunktionen ist die Möglichkeit des automatisierten Beobachtens und Eingreifens in das operative Automatisierungssystem erforderlich. Daraus folgt, dass die Bestandteile des Automatisierungssystems, welche Gegenstand der technischen Betriebsbetreuung sind, also bei der Durchführung von technischen Betriebsbetreuungsaufgaben als Informationsquelle oder Stellglied genutzt werden, für einen automatisierten Zugriff (Beobachten und Eingreifen) zugänglich gemacht werden müssen. In Analogie zur Automatisierung von technischen Prozessen werden also für die Automatisierung von Betriebsbetreuungsaufgaben entsprechende „Management-Sensoren“ und „Management-Aktoren“ benötigt. Ein Selbstmanagementsystem muss die Heterogenität der eingesetzten Technologien und die Struktur eines Automatisierungssystems berücksichtigen und eine davon unabhängige Management-Sensorik und -Aktorik bereitstellen.

In [15] wird für die Einführung von Selbstmanagement-Technologien in den betrieblichen Alltag von IT-Systemen ein evolutionärer Prozess gefordert. Die Einführung von Selbstmanagementfunktionen muss schrittweise erfolgen, da das Prozesspersonal sich an die Delegation von einzelnen Betriebsbetreuungsaufgaben gewöhnen muss, bevor weitere eingeführt werden können. Auch kann der tatsächliche Bedarf an Unterstützung und Entlastung des Prozesspersonals erst im betrieblichen Alltag festgestellt werden, so dass ein Selbstmanagementsystem diese schrittweise Evolution des Umfangs an Selbstmanagementfunktionen konzeptionell unterstützen muss.

Im Falle der Delegation von Betriebsbetreuungsaufgaben an ein Selbstmanagementsystem verbleibt weiterhin die Verantwortung beim Prozesspersonal. Daher muss das Prozesspersonal immer in der Lage sein, zu entscheiden, welche Betriebsbetreuungsaufgaben an Selbstmanagementfunktionen delegiert werden und welchen Grad die Automatisierung von delegierten Betriebsbetreuungsaufgaben einnehmen soll. Um eine Akzeptanz von Selbstmanagementfunktionen beim Betriebspersonal zu erlangen ist es essentiell, dass die Handlungen und Aktionen von Selbstmanagementfunktionen zu jeder Zeit nachvollziehbar und transparent sind.

13.3 Agentenbasiertes Selbstmanagementsystem für Automatisierungssysteme

13.3.1 Agentenorientierte Architektur eines Selbstmanagement-Systems

Aus den beschriebenen Anforderungen wird das Konzept eines Selbstmanagementsystems abgeleitet. Aufgrund der geforderten Autonomie bei der Durchführung von Selbstmanagementfunktionalitäten, der notwendigen Unterstützung individueller Sichten auf das Automatisierungssystem und der Orientierung am menschlichen Verhalten eignet sich ein agentenorientierter Ansatz zur Konzeption eines Selbstmanagementsystems [16, 17]. Die dem agentenorientierten Paradigma inhärente strukturelle Flexibilität unterstützt eine schrittweise Erweiterung eines Selbstmanagementsystems im Sinne einer schrittweisen Evolution, welche den Betrieb des AT-Systems begleitet [18].

Grundidee des Ansatzes ist es, basierend auf dem Modell des Selbstmanagementprozesses individuelle Selbstmanagementfunktionen in dedizierte Selbstmanagement-Agenten zu kapseln, welche vollständig autonom handeln und somit den Bediener von der entsprechenden Aufgabe entlasten.

Zu diesem Zweck wird das klassische Prozessautomatisierungssystem um ein viertes Teilsystem, das Selbstmanagement-System, erweitert. Aus den im vorigen Abschnitt beschriebenen Anforderungen lassen sich drei Kernaufgaben eines Selbstmanagementsystems ableiten:

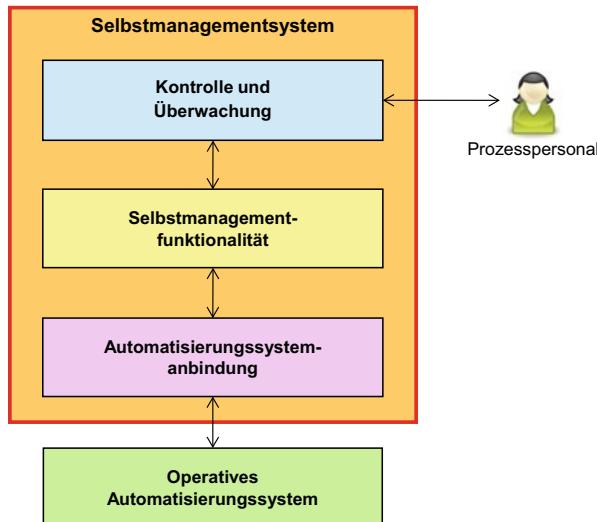


Abb. 13.3 Architektur eines Selbstmanagement-Systems für Automatisierungssysteme

1. Sicherung der Kontrolle und Nachvollziehbarkeit durch das Betriebspersonal
2. Bereitstellung der Selbstmanagement-Funktionalität
3. Technologietransparente Anbindung an das operative Automatisierungssystem

Auf Basis dieser Kernaufgaben wird das Selbstmanagementsystem in drei funktionale Ebenen aufgeteilt. Abbildung 13.3 zeigt das Selbstmanagementsystem, welches sowohl mit dem Bedienpersonal, als auch mit dem darunterliegenden Automatisierungssystem interagiert. Innerhalb der Ebenen des Selbstmanagementsystems interagieren spezialisierte Agenten, um die jeweiligen Aufgaben zu erfüllen. Diese werden im Folgenden beschrieben.

13.3.2 Agenten zur Kontrolle und Überwachung von Selbstmanagement-Funktionalitäten

Die oberste Ebene, Kontrolle und Überwachung, ist der Interaktion mit dem Prozesspersonal gewidmet. Hier gilt es dem Prozesspersonal eine einheitliche Schnittstelle für die Interaktion mit dem Selbstmanagementsystem und den darin vorhandenen Selbstmanagementfunktionen bereitzustellen. Das Prozesspersonal wird über Ereignisse und Entscheidung der Selbstmanagementfunktionen informiert und kann deren Autonomiegrad im Sinne einer *Adjustable Autonomy* festlegen [19].

Abbildung 13.4. zeigt die Agenten der Ebene *Kontrolle und Überwachung*. Hier kommen die Agententypen Systemverwaltungsagent, Berichterstattungsagent und Benutzerinteraktionsagent zum Einsatz.

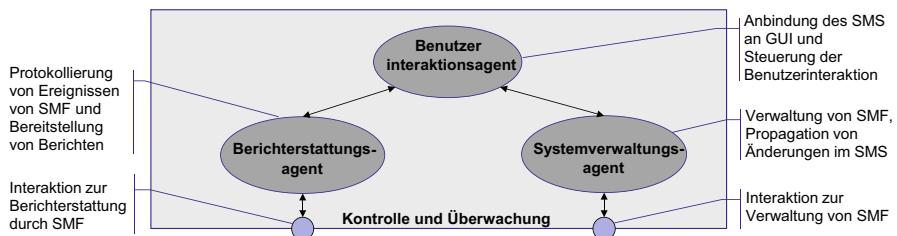


Abb. 13.4 Agenten zur Kontrolle und Überwachung von SMF

Ein *Systemverwaltungsagent* ermittelt dynamisch die in der darunterliegenden Ebene der Selbstmanagement-Funktionalität vorhanden Selbstmanagementfunktionen und ermöglicht deren Konfiguration und Parametrierung. Da das Prozesspersonal immer in der Lage sein muss, zu entscheiden, welche Betriebsbetreuungsaufgaben an Selbstmanagementfunktionen delegiert werden und welchen Grad die Automatisierung von delegierten Betriebsbetreuungsaufgaben einnehmen soll, erlaubt der Systemverwaltungsagent die explizite Delegation von Betriebsbetreuungsaufgaben durch Aktivierung der entsprechenden Selbstmanagementfunktionen. Nach der Aktivierung von Selbstmanagementfunktionen erlaubt der Systemverwaltungsagent den Autonomiegrad festzulegen, in dem der Benutzer festlegen kann, welche Schritte des Selbstmanagementprozesses durch eine Selbstmanagementfunktion autonom durchgeführt werden sollen. So kann ein Anlagen-Operator beispielsweise die Überwachung eines Prozesses und die Auswahl von Maßnahmen an eine Selbstmanagementfunktion delegieren wollen, nicht jedoch die Koordination und Ausführung der Maßnahmen. Wesentlich ist, dass das Betriebspersonal den Autonomiegrad mit Hilfe des Systemverwaltungsagenten explizit festlegen und jederzeit anpassen kann. Hierzu werden die drei Autonomiestufen informierend, anweisend und autonom unterstützt.

Der *Berichterstattungsagent* sichert die Nachvollziehbarkeit von Aktivitäten der Selbstmanagementfunktion. Hierzu interagiert der Berichterstattungsagent dynamisch mit in der darunterliegenden Ebene der Selbstmanagement-Funktionalität vorhandenen Selbstmanagementfunktionen und protokolliert deren Ereignisse und Entscheidungen. Selbstmanagementfunktionen sind gegenüber dem Berichterstattungsagenten berichtspflichtig, so dass das deren Verhalten jederzeit durch das Prozesspersonal nachvollzogen werden kann. Auf diese Weise dokumentiert der Berichterstattungsagent sowohl Handlungen als auch das Unterlassen von Handlungen durch Selbstmanagementfunktionen und kann diese auch rückwirkend transparent darstellen. Damit leistet der Berichterstattungsagent einen elementaren Beitrag zur Akzeptanz von Selbstmanagementfunktionen.

Der *Benutzerinteraktionsagent* stellt die Benutzerschnittstelle eines Selbstmanagementsystems dar. Er verwaltet die Benutzerinteraktion und interagiert hierzu mit dem Systemverwaltungsagent und dem Berichterstattungsagent. Darüber hinaus kann der Benutzerinteraktionsagent zur Anbindung einer GUI an das Selbstmanagementsystem eingesetzt werden.

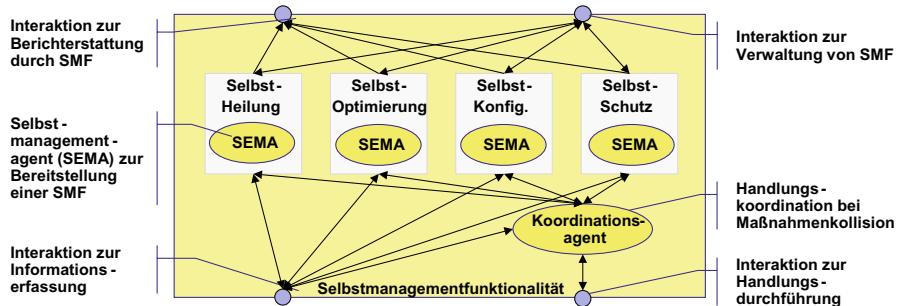


Abb. 13.5 Agenten zur Bereitstellung und Koordination von Selbstmanagementfunktionalität

13.3.3 Agenten zur Bereitstellung von Selbstmanagement-Funktionalitäten

Die mittlere Ebene „Selbstmanagement-Funktionalität“ ist der Kern des Selbstmanagementsystems und dient der Bereitstellung der eigentlichen Selbstmanagement-Funktionalität. Die Ausprägung von Art und Anzahl der hier bereitgestellten Selbstmanagementfunktionen sind abhängig vom jeweiligen Automatisierungssystem und der hierfür gewünschten Unterstützung durch Selbstmanagement. Die autonome Durchführung von Selbstmanagementfunktionen erfordert neben deren eigentlicher Bereitstellung auch die Koordination ihrer Handlung im Automatisierungssystem. Analog zur Abstimmung des Prozesspersonals bei der manuellen Durchführung, muss diese Koordination ebenfalls automatisiert erfolgen können, um einen autonomen Betrieb zu gewährleisten. Abbildung 13.5 zeigt die Ebene „Selbstmanagementfunktionalität“. Hier kommen Selbstmanagementagenten und Koordinationsagenten zum Einsatz.

Ein *Selbstmanagementagent* (SEMA) stellt genau eine SMF bereit. Aufgrund der Heterogenität der Betriebsbetreuungsaufgaben und den dementsprechend heterogenen Lösungsverfahren erlaubt die individuelle Realisierung durch einen Selbstmanagementagent die höchste Flexibilität. Ein Selbstmanagementagent wird spezifisch für eine Betriebsbetreuungsaufgabe eines Automatisierungssystems entwickelt und kapselt das funktionsspezifische Wissen zur Identifikation von Handlungsbedarf und zur Ableitung von Maßnahmen zur Behebung funktionalitätsrelevanter Situationen (vgl. Schritte 2 und 3 in Abb. 13.2). Hierzu interagiert jeder Selbstmanagementagent mit der darunterliegenden Ebene, der Automatisierungssystemanbindung, um die von ihm zur Erfüllung seiner Aufgaben benötigten Informationen aus dem Automatisierungssystem zu ermitteln. Erfasste Informationen werden vom Selbstmanagementagent analysiert und wissensbasiert auf einen Handlungsbedarf ausgewertet. Ist dies zutreffend erfolgt in einem weiteren wissensbasierten Schritt die Ableitung von Maßnahmen zur Ausführung im Automatisierungssystem. Sowohl die Handlungsbedarfsermittlung als auch die Maßnahmenauswahl können durch regelbasierte Systeme zur Formalisierung und Nutzung des

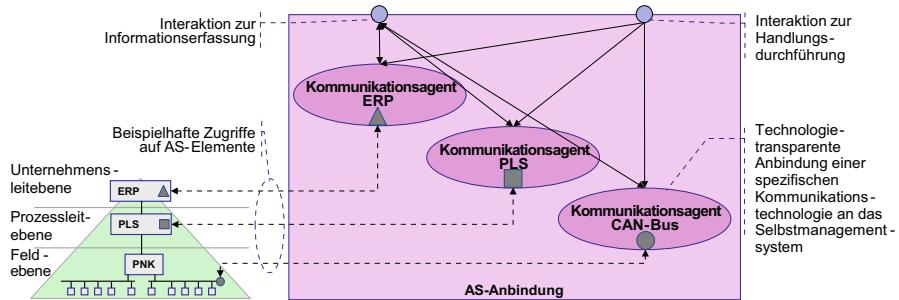


Abb. 13.6 Kommunikationsagenten zur Anbindung an das Automatisierungssystem

entsprechenden Wissens unterstützt werden. Sämtliche Aktivitäten und Entscheidungen eines Selbstmanagementagenten werden umgehend an den Berichterstattungsagenten gemeldet.

Der *Koordinationsagent* spielt eine zentrale Rolle bei der Sicherung der Konsistenz von Handlungsmaßnahmen der im Selbstmanagementsystem befindlichen Selbstmanagement-Agenten. Er verfügt über eine Wissensbasis zur Koordination von Handlungen im Automatisierungssystem. Zur Wahrnehmung dieser Aufgaben werden dem Koordinationsagent Handlungsanfragen von den Selbstmanagementagenten übermittelt. Diese werden vom Koordinationsagent auf Konflikte überprüft und bei Bedarf mit dem aktuellen Zustand des Automatisierungssystems (AS) abgestimmt. Sofern keine Konflikte vorliegen, leitet der Koordinationsagent die Handlungsanfrage an das Automatisierungssystem weiter. Im Konfliktfall erfolgt eine Auflösung mit Hilfe des Koordinationswissens. Das Koordinationswissen ist für ein Automatisierungssystem spezifisch und bildet die Eskalations- und Priorisierungsstrategien des Prozesspersonals ab.

13.3.4 Agenten zur Anbindung des Automatisierungssystems

Die unterste Ebene *Automatisierungssystemanbindung* stellt die für das Selbstmanagement erforderlichen Schnittstellen zum Automatisierungssystem dar. Die Bestandteile des Automatisierungssystems, welche Gegenstand des Selbstmanagements sind, werden für einen automatisierten Zugriff (Beobachten und Eingreifen durch Selbstmanagementfunktionen) zugänglich gemacht. Aufgrund der Heterogenität der Technologien der Bestandteile des Automatisierungssystems muss hier eine Abstraktion von diesen Technologien erfolgen und der einheitliche Zugriff für Selbstmanagementfunktionen ermöglicht werden. Abbildung 13.6 zeigt die Ebene Automatisierungssystemanbindung. Hier kommen technologiespezifische Kommunikationsagenten zum Einsatz.

Ein *Kommunikationsagent* realisiert die Anbindung einer Kommunikationstechnologie an das Selbstmanagementsystem. Zu diesem Zweck verfügt ein Kommu-

nikationsagent über einen generischen Teil zur einheitlichen Interaktion innerhalb des Selbstmanagementsystems und einen technologiespezifischen Teil zur Kommunikation mit den entsprechenden Bestandteilen des Automatisierungssystems. Die Selbstmanagementagenten melden ihren individuellen Informationsbedarf bei Kommunikationsagenten an. Hierbei unterscheidet ein Kommunikationsagent die einmalige Abfrage einer Information, oder die regelmäßige Meldung der Information an einen Selbstmanagementagent. Die regelmäßige Meldung kann von einem Selbstmanagementagent sowohl zyklisch, als auch ereignisgesteuert angefordert werden.

Gleichzeitig dienen Kommunikationsagenten als Schnittstelle zum Zugriff auf das Automatisierungssystem. Vom Koordinationsagent freigegebene Handlungen werden zur Ausführung im Automatisierungssystem an die entsprechenden Kommunikationsagenten übermittelt.

Auf Basis dieses allgemeinen Selbstmanagementsystems kann für beliebige Automatisierungssysteme eine Unterstützung durch Selbstmanagementfunktionen bereitgestellt werden. Hierzu sind die Entwicklung der spezifischen Selbstmanagementfunktionen und die Konfiguration der benötigten Kommunikationsagenten notwendig.

13.4 Anwendungsbeispiel: Selbstheilende Personenaufzugssteuerung

Anhand des Beispiels einer Selbstheilung von Ausfällen bei einem Personenaufzug soll das Funktionsprinzip des Selbstmanagementsystems verdeutlicht werden.

Das in Abb. 13.7 dargestellte System besteht aus zwei Aufzugsschächten mit je einer Kabine. Zur Erfassung der Kabinenposition sind je Stockwerk vier taktile Sensoren erforderlich. Beim Ausfall eines Sensors muss der Betrieb des Aufzugs eingestellt werden, da er nicht mehr von der Steuerung positioniert werden kann. Die Steuerung sowie sämtliche Sensoren und Aktoren sind über einen CAN-Bus vernetzt.

Zur Bereitstellung einer Selbstheilungsfunktion, welche Teilausfälle von Positionssensoren, wie in Abb. 13.7 rechts dargestellt, kompensieren kann, müssen für dieses System fünf Agenten des Selbstmanagementsystems realisiert werden. Neben den Agenten zur Kontrolle und Überwachung ist ein Kommunikationsagent für den Zugriff auf den CAN-Bus notwendig. Er interpretiert ankommende Nachrichten auf dem CAN-Bus und übermittelt diese an einen Selbstheilungsagenten.

Der Selbstheilungsagent verfügt über ein internes Modell der Aufzugsanlage, mit dessen Hilfe er Ausfälle einzelner Sensoren durch Ausbleiben der Positionsmeldung erkennen kann. Ist dies der Fall, stellt der Selbstheilungsagent einen Handlungsbedarf fest (Ausfall kompensieren). Zu dessen Erfüllung legt der Selbstheilungsagent dynamisch fest, wie ein Schätzwert für den oder die ausgefallenen Sensoren auf Basis benachbarter funktionsfähiger Sensoren errechnet werden kann. Hierzu greift er auf eine Regelbasis zurück, welche situationsbasiert ausgewertet

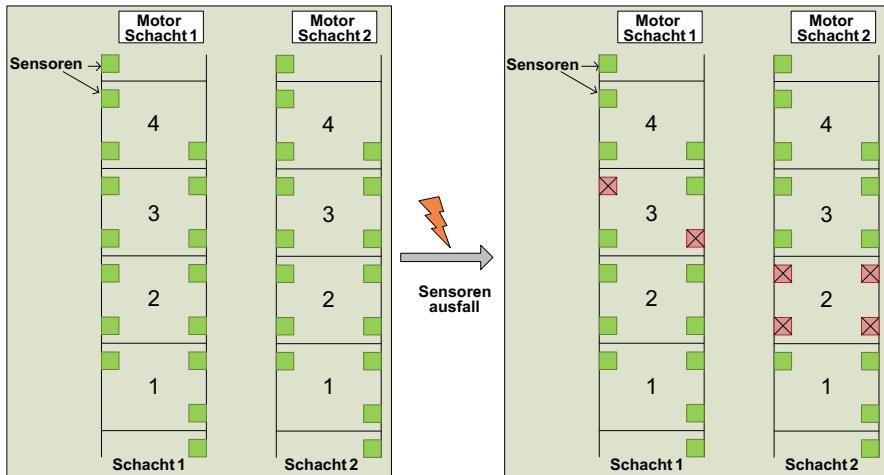


Abb. 13.7 Schematische Darstellung der Sensoren des Personenaufzugs im Normalzustand (*links*) und mit Teilausfall (*rechts*)

wird. Die Ausführung der festgelegten Maßnahme erfolgt durch eine Rekonfigurationsanforderung an die operative Steuerung. Die Anforderung wird mit Hilfe des Kommunikationsagenten an das operative Automatisierungssystem übermittelt. Da in dem Beispiel lediglich eine Selbstmanagementfunktion realisiert wurde, wird auf einen Koordinationsagenten verzichtet.

13.5 Zusammenfassung

Die Zunahme der Komplexität moderner AT-Systeme macht eine Unterstützung bei Betriebsbetreuungsaufgaben erforderlich. Hierzu wurde ein allgemeines, agentenbasiertes Konzept vorgestellt, welches die einfache Integration von Selbstmanagementfunktionen parallel zum Betrieb des Automatisierungssystems erlaubt. Der Agentenorientierte Ansatz weist dabei vielseitige Vorteile für das Selbstmanagement von Automatisierungssystemen auf.

Softwareagenten zeichnen sich durch Autonomie und Zielorientierung aus. Beide Eigenschaften gemeinsam tragen dazu bei, dass eine agentenorientierte Lösung den Anforderungen an eine aktive Unterstützung des Prozesspersonals gerecht werden kann. Die Zielorientierung erlaubt das komplexe Problem der gleichzeitigen Unterstützung vielfacher Aufgaben der technischen Betriebsbetreuung in dedizierte Entitäten zu zerlegen, wobei jede Entität auf die Erfüllung ihrer spezifischen Ziele fokussiert ist. Die Autonomie einzelner Agenten erlaubt ihnen für ihren jeweiligen Aufgabenbereich Entscheidungen zu treffen und Handlungen auszuführen, die ihrem jeweiligen Ziel dienlich sind. Die Komplexität der Aufgabenstellung lässt

sich in einem agentenorientierten Ansatz beherrschen, indem das Problem systematisch in Teil- und Subprobleme zerlegt wird. Die Bearbeitung der überschaubaren Teilprobleme wird von spezialisierten Agenten übernommen, die simultan die Erfüllung ihrer jeweiligen Ziele verfolgen. Durch gezielte Interaktion der einzelnen Agenten kann sichergestellt werden, dass die resultierenden Handlungen koordiniert und aufeinander abgestimmt sind. Der menschliche Bediener muss dann nicht mehr aktiv eingreifen oder explizit Funktionen aufrufen sondern kann die Bearbeitung einzelner oder mehrerer Aufgaben an ein Agentensystem delegieren. Agenten agieren als Stellvertreter des Prozesspersonals.

Eine große Herausforderung bei der automatisierten Unterstützung der technischen Betriebsbetreuung ist der Zugriff auf verteilte Informationen im Automatisierungssystem über unterschiedliche Leitebenen hinweg. Das Problem, viele verteilte Kommunikationsschnittstellen zu überwachen kann durch die Übertragung dieser Aufgaben an dedizierte Agenten gelöst werden. Dadurch wird gewährleistet, dass die Überwachung der Schnittstellen parallel erfolgt und die Erfassung relevanter Daten und Ereignisse ohne Zeitverlust erfolgt. Die für diese Aufgabe verantwortlichen Agenten stellen sicher, dass über geeignete Interaktionsmechanismen alle betroffenen Agenten im Agentensystem über Änderungen informiert werden. Die Fähigkeit zur Interaktion trägt dazu bei, dass die Aufgabe der Anbindung an Kommunikationsschnittstellen nur einmal gelöst werden muss. Die Anwendung von Agenten erlaubt die Anbindung auch physikalisch verteilter Datenquellen und Kommunikationsschnittstellen, da innerhalb eines Agentensystems das Prinzip der Lokationstransparenz herrscht. Bei Bedarf können Agenten eines Agentensystems räumlich verteilt werden, um direkten Zugriff auf Kommunikationsschnittstellen zu erlangen.

Agentensysteme zeichnen sich durch Offenheit aus. Diese Eigenschaft erlaubt es zu jedem Zeitpunkt die Systemfunktionalität durch die dynamische Integration zusätzlicher Agenten anzupassen und zu Erweitern. Auf diese Weise ist es mit geringem Aufwand möglich ein agentenorientiertes Unterstützungssystem an die Veränderung des Unterstützungsbedarfs anzupassen.

Wird das Agentensystem zur Unterstützung der technischen Betriebsbetreuung so auf Agenten abgebildet, dass die Struktur des Agentensystems mit realen und konzeptionellen Strukturen des Unterstützungsproblems korrespondiert, ist die evolutionäre Anpassung des Agentensystems an Veränderungen mit geringem Aufwand möglich. Eine Veränderung kann dann direkt auf zusätzliche Agenten abgebildet werden oder bestehende Agenten werden entsprechend konfiguriert. Dabei ist die Eigenschaft der Kapselung von Agenten von Vorteil. Die Veränderung eines Unterstützungsziels wird zur Laufzeit durch Benachrichtigung des verantwortlichen Agenten umgesetzt.

Agenten können zur Sicherung der Akzeptanz autonomer Unterstützungssysteme durch das Prozesspersonal beitragen, indem die Transparenz und somit die Nachvollziehbarkeit für das Prozesspersonal erhöht wird. Der aktuelle Status einer Unterstützungsfunktion kann durch direkte Kommunikation mit dem zuständigen Agenten ermittelt werden. Gleichzeitig wird das Prozesspersonal durch eine dynamisch anpassbare Konfiguration von Unterstützungszielen nur in dem Umfang

durch das Agentensystem unterstützt, der aktuell gefordert ist. Das Verhalten und die Strategien von Agenten zur Unterstützung der technischen Betriebsbetreuung kann am menschlichen Verhalten orientiert werden, sodass durch Agenten ausgelöste Aktionen mit dem Verhalten des Prozesspersonals konsistent sind. Die flexible Anpassbarkeit der Autonomie auf Basis von Benutzervorgaben (dynamische Anpassung von Zielen), trägt letztendlich dazu bei, dass die Bedarfsangemessenheit durch ein Agentensystem berücksichtigt wird.

Das vorgestellte Konzept wurde auf Basis der Agentenplattform JADE [20] realisiert und an unterschiedlichen Automatisierungssystemen erfolgreich erprobt.

Literatur

- [1] Wahlster, W., Raffler, H.: Studie 2008 des Feldafinger Kreises „Forschen für die Internet-Gesellschaft: Trends, Technologien, Anwendungen“ (2008). http://www.feldafingerkreis.de/Feldafinger-Kreis_Studie_2008.pdf. Zugegriffen: 15.09.2009
- [2] Gote, M., Neumann, J., Bauer, M., Horch, A.: Trends in Operations und Plant Asset Management – ein Diskussionbeitrag. atp **50**(10), 48–54 (2008)
- [3] Gote, M.: Plant Asset Management – betriebliche Wirklichkeit. atp Automatisierungstechnische Praxis **49**(2), 28–33 (2007)
- [4] Jelasity, M., Babaoglu, O., Laddaga, R.: Self-Management through Self-Organization. IEEE Intelligent Systems **21**(2), 8–9 (2006)
- [5] Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer **36**(1), 41–50 (2003)
- [6] Lauber, R., Göhner, P.: Prozessautomatisierung I, 3. vollst. überarb. Aufl. Springer, Berlin, Heidelberg, New York (1999)
- [7] Heylighen, F.: Self-organization, emergence and the architecture of complexity. Proc. of the 1st European Conference on System Science (1989)
- [8] Herrmann, K., Mühl, G., Geihs, K.: Self-Management – Potentiale, Probleme, Perspektiven. PIK – Praxis der Informationsverarbeitung und Kommunikation **27**(2), 74–79 (2004)
- [9] Herrmann, K., Mühl, G., Geihs, K.: Self-Management: The Solution to Complexity or Just Another Problem? IEEE distributed systems online **6** (2005)
- [10] Heylighen, F., Gershenson, C.: The Meaning of Self-organization in Computing. IEEE Intelligent Systems **18**(4), 72–75 (2003)
- [11] Mubarak, H., Göhner, P.: An Agent-Oriented Approach for Self-Management of Industrial Automation Systems. 8th IEEE Int. Conf. on Industrial Informatics **2010**, 717–722 (2010)
- [12] Mühl, G., Werner, M., Jaeger, M., Herrmann, K., Parzyegla, H.: On the Definitions of Self-Managing and Self-Organizing Systems. In: Braun, T., Carle, G., Stiller, B. (Hrsg.) KIVS 2007, Kommunikation in Verteilten Systemen. VDI Verlag, Düsseldorf (2007)
- [13] Mubarak, H.: Unterstützung der Leitebene durch Selbstmanagement-Funktionalität. In: GMA-Kongress 2007 Automation im gesamten Lebenszyklus. VDI-Bericht, Bd. 1980. VDI-Verlag, Düsseldorf (2007)
- [14] Mubarak, H., Göhner, P.: Einsatz von Agenten für das Selbstmanagement von Automatisierungssystemen. In: Schumann, M., Kolbe, L.M., Breitner, M.H., Frerichs, A. (Hrsg.) Multikonferenz Wirtschaftsinformatik 2010, S. 167–168. Göttingen (2010)

- [15] Miller, B.A.: The autonomic computing edge: The path to level 5, full autonomic maturity (2005). <http://www.ibm.com/developerworks/autonomic/library/ac-edge8>. Zugriffen: 15.09.2009
- [16] Parunak, H.V.D.: Practical and industrial applications of agent-based systems. Environmental Research Institute of Michigan (ERIM) (1998)
- [17] Mubarak, H.: Developing Flexible Software Using Agent-Oriented Software Engineering. IEEE Software **25**(5), 12–15 (2008)
- [18] Jennings, N.R., Wooldridge, M.: Agent-Oriented Software Engineering. Artificial Intelligence **117**, 277–296 (2000)
- [19] Maheswaran, R.T., Tambe, M., Varakantham, P., Myers, K.: Adjustable Autonomy Challenges in Personal Assistant Agents: A Position Paper. In: Nickles, M., Rovatsos, M., Weiss, G. (Hrsg.) Agents and Computational Autonomy – Potential, Risks, Solutions. Springer, Berlin, Heidelberg, New York (2003)
- [20] JADE (Java Agent DEvelopment Framework). <http://jade.tilab.com/>

Teil IV

Agentenbasierte Assistenzsysteme

Kapitel 14

Ready for Take-Off – Softwareagenten disponieren Flugcharter-Verkehr

Christian Dannegger

Zusammenfassung Dieses Kapitel beschreibt die Entwicklung eines agentenbasierten Dispositioessystems für den Flugverkehr. Ein junges Softwareunternehmen, gegründet von erfahrenen Piloten, hat es sich zur Aufgabe gemacht, die mangelfhafte Software-Unterstützung der Disponenten von Flugcharter-Unternehmen zu beenden. Mehrere Disponenten, die gleichzeitig an einem gemeinsamen Dispositioessplan arbeiten, durch autonome Agenten zu unterstützen, fordert nicht nur eine gründliche Analyse der Prozesse rund um die Flugplanung und Flugdurchführung. Wichtiger noch ist der Kernherausforderung der mehrfachen Ressourcenoptimierung von über 100 Flugzeugen und mehreren 100 Piloten bei Beachtung des europäischen Luftfahrtrechts. Die Problemstellung, Herangehensweise und Ergebnis dieses Projektes sind in den folgenden Abschnitten beschrieben.

14.1 Einleitung

Die Luftfahrtbranche wird vom normalen Bürger als hochtechnologisch wahrgenommen. Immer größere, moderne, leisere und schadstoffärmere Passagierjets zeigen den riesigen Fortschritt deutlich auf. Im Hintergrund allerdings, in Bereichen, mit denen der normale Fluggast nicht in Berührung kommt, folgt die Branche unverändert den Regeln und Prozessen aus den Anfängen der Luftfahrt.

Immer noch sind die meisten Abläufe papiergetrieben und verlangen – selbstverständlich im Namen der Sicherheit – mehrfache manuelle Kontrollen und papierhafte Unterlagen über jeden Prozessschritt. Aber seit Jahrzehnten traut sich niemand, die neuen technologischen Möglichkeiten zu nutzen. Deshalb werden z. B. immer noch Flugkarten auf Papier mitgeführt, die nicht nur schwer zu tragen sind, sondern auch einem zu langen und – natürlich – manuellen Änderungszyklus unterliegen.

Christian Dannegger (✉)
Kandelweg 14, 78628 Rottweil, Deutschland
e-mail: cd@3a-solutions.com

Einige innovative Piloten nutzen dennoch tragbare Tablet-PCs oder vergleichbare Computer. Aber dann aus eigenem Antrieb – quasi privat – und zusätzlich zum prall gefüllten Pilotenkoffer.

In diesem Umfeld sind auch die Disponenten meist bei der papierbezogenen Arbeitsweise „hängen geblieben“. Auch wenn einige Softwaresysteme zur Verfügung stehen, sind doch die Prozesse noch „die alten“ und damit die Denkweise und Vorgehensweise bei Disposition, Flugvorbereitung, Flugdurchführung und abschließender Protokollierung (Logbuch des Flugzeuges und der Piloten). Die Vorschriften verlangen z. B. weiterhin, die Piloten vor dem Abflug mit einem kompletten Satz an Flugunterlagen auf Papier zu versorgen.

Im Kern konzentrierte sich dieses Projektes darauf, die Disposition und Optimierung für Flugcharter und Business Jets zu unterstützen. Herkömmliche Systeme bieten zwar eine Datenhaltung der Plandaten, aber helfen keineswegs bei der eigentlichen Aufgabe, im dynamischen Umfeld des Fluggeschehens fortlaufend optimierend den Dispositionsplan zu überprüfen und anzupassen.

Genau hier lag die Motivation für das flexible Softwaresystem FL3XX (sprich: [fleks] wie flexibel), entwickelt von einem gleichnamigen Startup-Unternehmen aus Österreich. Neben der automatischen Disposition bestehen das Ziel und der Ehrgeiz, den gesamten Luftfahrt-Betrieb durch revolutionär schlanke und auf den Punkt dem Bedarf angepasste Prozesse in ein neues Zeitalter zu überführen. In diesem Kapitel konzentrieren wir uns aber auf die agentenbasierte Optimierung von FL3XX und dessen Umsetzung und Nutzen für ein Flugcharter-Unternehmen.

Das Softwaresystem bietet einen One-Stop-Shop für Business Jet Airlines. Der webbasierte Service (SaaS: Software as a Service) deckt alle Bereiche einer typischen Charterflug-Gesellschaft ab, von der Angebotserstellung mit Preisberechnung über die Flugzeug- und Crew-Planung bis hin zur Wartung und dem Qualifikations-Management der Piloten. Dabei erhalten die Disponenten bei ihrer Dispositionsaufgabe fortlaufende Unterstützung durch die im Hintergrund agierenden Softwareagenten. Ähnlich einem Navigationssystem erarbeitet FL3XX neue Vorschläge, sobald ein neu eingetroffenes Ereignis neue Fakten und damit neue Randbedingungen schafft.

Mehr Flexibilität in Flugplanung und Flugbetrieb heißt die Devise. Erst durch ein agiles Softwaresystem sind auch agile Prozesse möglich, denn jede Entscheidung muss ein immenses, mehrere dicke Ordner füllendes, europäisches Regelwerk zum Flugbetrieb (EU-Ops: [1]) berücksichtigen. Die folgenden Abschnitte gehen nochmal auf die konkrete Ausgangssituation mit Business Case ein, um nach dem Lösungsansatz und dem Agentenmodell den Nutzen durch das System darzustellen.

14.2 Ausgangssituation/Problemstellung/Stand der Technik

Die Luftfahrtbranche teilt sich im Wesentlichen in zwei große Bereiche: den Linienflugverkehr mit festen Flugplänen, den der normale Urlaubs- und Geschäftspas-

sagier kennt, und den Privat- bzw. Charter-Flugverkehr (General Aviation) mit individuellen nach Bedarf durchgeführten Flügen. Dieser Bericht bezieht sich auf den zweiten Bereich, den Charter-Flugverkehr.

Üblicherweise mietet dabei der Charter-Kunde ein Flugzeug mit Kapitän und Co-Pilot (auch First Officer genannt) mit der dazugehörigen Dienstleistung zur kompletten Durchführung eines oder mehrerer Flüge. Bei diesem Geschäftsmodell „gehört“ das Flugzeug während des Fluges exklusiv nur dem einen Charter-Kunden – egal, ob er alleine in einem achtsitzigen Flugzeug sitzt oder ob er mit weiteren Passagieren eine vierzitzige Maschine voll auslastet. Privatsphäre und Unabhängigkeit von Anderen werden in diesem Fall sehr groß geschrieben.

Neben diesem üblichen Geschäftsmodell entwickeln einige Unternehmen seit geraumer Zeit ein neues Angebot: Das Air Taxi. Unternehmen, die dieses Modell anbieten, sind in der Air Taxi Association organisiert [2]. Genauer gesagt handelt es sich dabei um eine Art Sammeltaxi, bei dem sich mehrere Passagiere ein Taxi, und hier eben ein Flugzeug, für eine Strecke teilen. Deshalb nennt man dieses Modell auch „Per-Seat-On-Demand“, man bucht also nicht das ganze Flugzeug, sondern nur einen Sitzplatz („per seat“ wie bei Linienflügen), aber man gibt seinen gewünschten Zeitrahmen („on demand“) an, in dem man fliegen will.

Die Kombination aus beiden üblichen Modellen „per seat“ des Linienflugs und „on demand“ des Business Charter ergibt eine interessante Synergie aus Flexibilität und Kostenreduktion (durch Ressourcen-Teilung). Auf der anderen Seite erhöht sich die Planungskomplexität immens, denn es ist nicht nur ein Auftrag einem Flug zuzuordnen, sondern die Disposition muss auch die kostengünstigste Kombination aus allen gebuchten Flugtickets finden – und das natürlich auch im weiterhin dynamischen Umfeld.

Hohe Fixkosten und hohe Dynamik

Zusätzlich zur immer weiter steigenden Dynamik kämpft die Business-Jet-Branche mit sehr hohen Fixkosten (für die Flugzeuge und deren Unterhalt) bei gleichzeitig geringen Margen. Hinzu kommt die fehlende Systemintegration, so dass die Disponenten mit einer Vielzahl von Systemen umgehen und mit ebenso vielen Systembrüchen leben müssen. Zentraler Punkt aber ist die fehlende systematische Optimierung der teuren Ressourcen, die in den Köpfen weniger erfahrener Disponenten stattfindet.

In den allermeisten Fällen wird das Geschäft manuell betrieben und basiert, insbesondere in der Disposition, auf rein menschlichen Entscheidungen ohne Unterstützung durch eine adäquate Planungs- und Optimierungssoftware. Plantafeln in Form von Whiteboards oder Magnettafeln sind noch häufig im Einsatz (Abb. 14.1).

Mancherorts wird ein Tabellenkalkulationsprogramm als zentrales Planungsinstrument hinzugezogen oder, was schon die Spitze des aktuellen Technologieeinsatzes bedeutet, der Einsatz einer speziellen Branchensoftware für den Individualflugverkehr. Diese Systeme unterstützen typischerweise in der Stammdatenpflege, Kundendatenverwaltung, Flugabwicklung, Crew-Planung, Wartungs-Planung,



Abb. 14.1 Plantafel

Dokumenten-Management, sogar bis hin zur Disposition. Letztere allerdings nur in einer manuellen Version, was letztendlich dem elektronischen Ersatz von gelben Haftnotizen entspricht.

Diese bestehenden Arbeitsweisen machen Entscheidungen, Flugplananpassungen und insbesondere Flugkalkulationen sehr aufwändig und träge. So hat eine Untersuchung ergeben, dass zur Erstellung eines Angebots in herkömmlichen Systemen weiter über 100 Klicks notwendig sind, wobei dies mit intelligenter und automatisch disponierender Software in maximal zehn Klicks möglich ist.

Um konkurrenzfähig zu bleiben, muss jede Kundenanfrage für einen Flug individuell durchkalkuliert werden. Dabei ist die aktuelle Flugplanung mit allen Details zu berücksichtigen, insbesondere die notwendigen Positionierungsflüge. Dies sind die Leerflüge, die notwendig sind, um einen Passagier am gewünschten Ort abzuholen – vergleichbar mit der Leerfahrt eines Taxis.

Neben der reinen Angebotserstellung geht es um Angebotserstellung, Verkauf, Disposition, Durchführung, Compliance Management, Reports und Forecasts. All diese Funktionen umgesetzt als Mensch-Maschine-Interaktion sollen mit einer, im Hintergrund arbeitenden, autonomen Ressourcen-Optimierung kombiniert werden. Dabei ist immer genau zu klären, wann der Mensch und wann die Maschine das Sagen hat, also wer gerade disponiert.

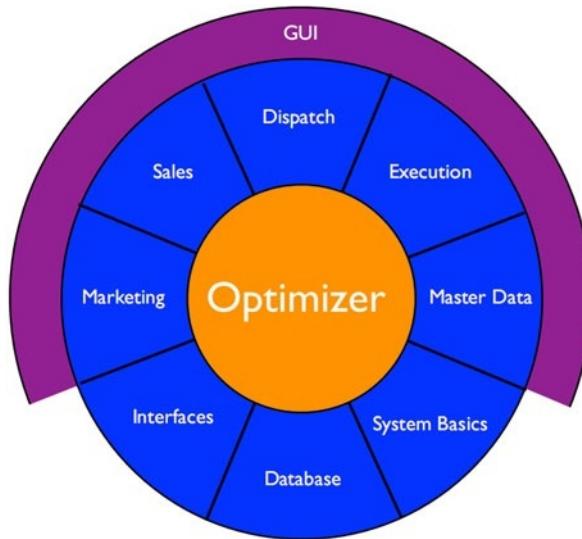


Abb. 14.2 Funktionale Module, die den Optimierungskern umgeben

Dabei gelten die übergeordneten Ziele:

- Reduktion der Flugkosten durch Vermeidung von Leerflügen
- Reduktion des CO₂-Ausstoßes durch weniger Flugstunden
- Reduktion von manueller Arbeit durch schlanke systemunterstützte Prozesse
- Verbesserte Ressourcenauslastung durch intelligente Routenplanung
- Erhöhte Sicherheit durch strikte systemüberwachte Compliance Checks
- Verbesserte Transparenz durch vollständige Datenhaltung und Verfügbarkeit für alle Beteiligten
- Skalierbarkeit des Geschäfts durch hochgradige Automatisierung der Abläufe

14.3 Konzept/Lösung

Das System besteht im Kern aus der autonom arbeitenden dynamischen Ressourcen-Optimierung – dem Optimizer (Abb. 14.2). Dieser ist umgeben von Interaktionsmodulen, die durch ein Graphisches User Interface (GUI) den Benutzern die Funktionen des Systems aus folgenden Bereichen zur Verfügung stellen: Marketing, Vertrieb, Disposition, Ausführung und Stammdaten. Das Ganze stützt sich dabei auf eine Zahl von grundlegenden Funktionen zu Schnittstellen, Datenbanken und allgemeine Systemgrundfunktionen (Logging, etc.).

Optimierungskern

Auf den Optimizer als Kern der ganzen Lösung wollen wir nun hier etwas tiefer eingehen.

Als grundlegend zu optimierende bzw. einzuplanende Einheit nimmt man in der Luftfahrt natürlich das Ticket. Ein Ticket ist als Auftrag eines Passagiers für einen Flug von Flughafen A nach Flughafen B in einem gewünschten Zeitraum (frühester Abflug, späteste Ankunft) zu sehen. Zusätzliche Einschränkungen durch spezielle Kundenwünsche (z. B. ein bestimmter Flugzeugtyp oder Interieur) sind nicht weiter relevant für die Optimierung und werden hier nicht weiter behandelt.

Je nach Geschäftsmodell (Flugzeugcharter oder Per-Seat-On-Demand) sind die Tickets zu Flügen zu kombinieren, dessen Parameter sich aus den Rahmendaten aller beteiligten Tickets ergeben, so dass sämtliche Ticket-Vorgaben erfüllt werden. Dies bezieht sich hauptsächlich auf das gemeinsame Flugintervall und natürlich die Flughäfen, die übereinstimmen müssen. Damit gilt ein Flug für den Optimizer quasi als Stellvertreter für „seine“ Tickets und stellt damit die Bedarfsseite für das Optimierungsproblem dar.

Um diesen Bedarf zu bedienen (den Flug durchführen zu können), benötigt jeder Flug drei Ressourcen: Ein Flugzeug und die Crew, bestehend aus zwei Piloten: Den Commander und den First Officer. Die Crew, die bei kleinen Business Jets aus den erwähnten zwei Piloten besteht, umfasst bei größeren Flugzeugen oder bei Interkontinentalflügen weitere Piloten und auch eine Cabin Crew.

Agentenmodell

Es geht also um eine Multiressourcenoptimierung, wobei jede Ressource seinen eigenen Zeitplan (schedule) hat und diese möglichst optimal mit dem Zeitfenster des Fluges in Übereinstimmung gebracht werden müssen. Für dieses System war von vorneherein klar, dass diese komplexe Aufgabe in nahezu Echtzeit nur durch einen Agentenansatz lösbar ist. Mit der bisher beschriebenen Problemanalyse ist die Definition eines passenden Agentenmodells auch nicht weiter kompliziert. Alle Tickets, Flüge, Flugzeuge und Piloten wurden jeweils als Agenten implementiert (Abb. 14.3) die miteinander bilateral verhandeln, bis keine weitere Verbesserung für einen Flug mehr möglich ist.

Neben den reinen Optimierungsagenten decken weitere Agenten die Bereiche Events & Notification ab, sowie Schnittstellen unter anderem zum Flugzeugtracking-System *Spidertracks* und zur Flugbörsse *Avinode*. Weitere Hilfs-Agenten bedienen das GUI und managen, vor allem instanziieren, die Ressourcen und Auftragsagenten.

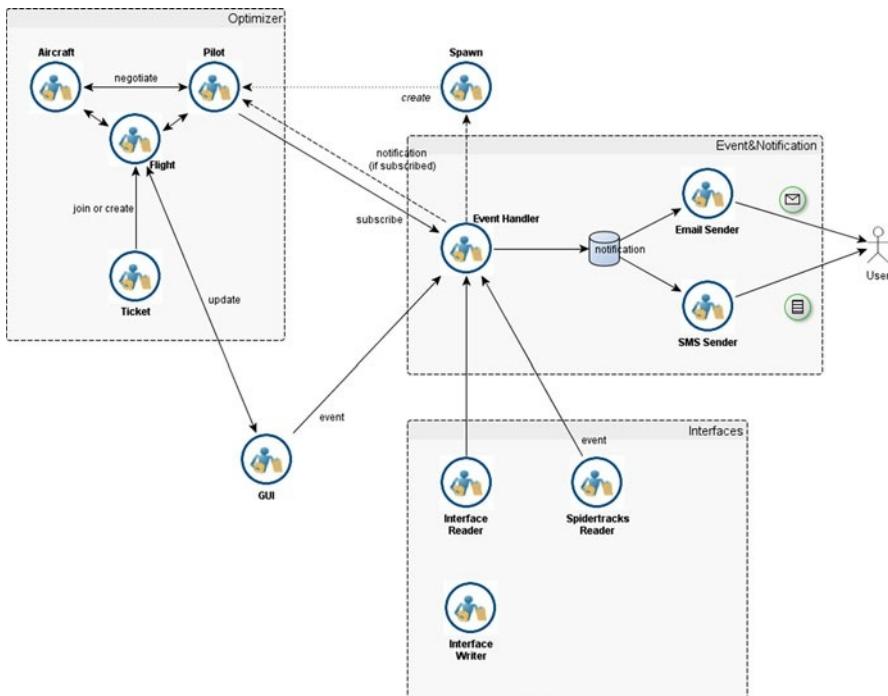


Abb. 14.3 Agentenmodell

Bilaterale Verhandlungen

Das Verfahren stellt sich grob wie folgt dar: Es beginnt mit der verbindlichen Buchung eines Tickets. Dieses Ticket, genauer gesagt der Ticketagent, der durch die Buchung erstellt wurde, versucht einen passenden Flug zu finden, um diesem beizutreten (quasi „zuzusteigen“). Gibt es keinen passenden Flug, so wird ein neuer Flug angelegt (und ein Flugagent gestartet) und diesem beigetreten.

Nun beginnt der Flug zunächst ein Flugzeug zu finden, was über eine Art Ausschreibung basierend auf einer internen „Währung“ erfolgt (Contract Net [3]). Sobald ein Flugzeug zu einem bestimmten Zeitpunkt verfügbar ist, beginnt die zweite Phase der Ausschreibung. Nun werden zur gefundenen Flugzeit noch Piloten gesucht – mit prinzipiell dem gleichen Verfahren. Der Flugzeug-Agent kombiniert die Piloten-Angebote mit dem Eigenen und bietet dem nachfragenden Flug einen Gesamtpreis für den Flug mit sich (dem Flugzeug) an. Die Auswahl-Logik des Flug-Agenten ist sehr einfach: Der Günstigste wird gewählt.

Die eigentliche Komplexität liegt im Flugzeug-Agenten, der nicht nur mehrere Piloten finden muss, sondern sich auch um einen eventuell notwendigen Positionierungsflug kümmert. Das Flugzeug selbst „weiß“ ja, wo es zum Bedarfszeitpunkt steht bzw. stehen wird und muss dann in seinem Angebot den Positionierungsflug

automatisch erstellen und dafür ebenfalls Piloten finden. Alles zusammen fließt in das Angebot mit ein.

Damit aber nicht genug. Es gibt nicht nur den Positionierungsflug vor dem gewünschten Passagierflug, sondern auch eventuell einen Positionierungsflug danach – zum Abflug-Flughafen des folgenden Passagierfluges. Alle preisrelevanten Bestandteile fließen mit ein.

Ein wesentlicher Teil der Agentenkommunikation ist als Auszug im Interaktionsdiagramm in Abb. 14.4 dargestellt. Es beginnt oben links mit dem Versuch, einen passenden Flug zu finden, bevor ein neuer Flugagent gestartet wird. Der zentrale Flugagent „flight“ ist links und rechts umgeben vom Pilotenagenten „pilot“, wobei ein Flug mindestens zwei Piloten benötigt, und dem Flugzeugagenten „aircraft“. Die weiteren Agenten „other flight“, „other pilot“, „other aircraft“ dienen dazu, die Kommunikation mit den vor einem Optimierungsschritt relevanten Ressourcen und Flügen darzustellen. Die Hauptschleife selbst besteht aus den drei Teilen:

1. Günstigstes Angebot bzw. Flugzeug/Piloten-Kombination finden
2. Information über Zuschlag an die neuen Ressourcen
3. Freigabe der bisherigen Belegung bei den alten Ressourcen

Weitere Optimierungsprinzipien

Bestätigte Flüge werden weniger umgeplant

Ein Flug befindet sich während seiner Existenz im System in verschiedenen Zuständen. Einige davon sind relevant für die Flexibilität einen Flug umplanen zu dürfen. Ein neuer Flug hat zunächst keine Einschränkungen. Sobald ein Flug als verbindliche Option angeboten oder konkret bestätigt wird, kann der Flug nicht mehr beliebig verschoben werden. In gewissem Rahmen von einigen Minuten ist das noch möglich, denn für den Passagier ändert sich die Anreisezeit zum Flughafen nicht. Sobald ein Flug Slots von der Flughafenkontrolle zugewiesen bekommen hat, darf dieser Flug maximal innerhalb der Slot-Grenzen verschoben werden. Wurde ein Flugplan bestätigt, gelten ähnlich einschränkende Regeln. Und es gibt noch viele mehr davon. Der Optimizer berücksichtigt diese Flexibilitätsregeln bei jedem Optimierungsschritt.

Manuelle Disposition jederzeit möglich

Ein System bekommt nur dann die notwendige Benutzer-Akzeptanz, wenn es durch den Benutzer vollständig kontrolliert werden kann. Der Disponent muss in allen Belangen das letzte Wort haben können. Das heißt jeder Flug und jede Dispositionentscheidung muss der Disponent manuell übersteuern können. Dieses Zusammenspiel zwischen automatisch arbeitenden Optimierungsagenten und manuell arbeitenden Disponenten erfolgt völlig transparent und intuitiv. Jeder Flug kann auf

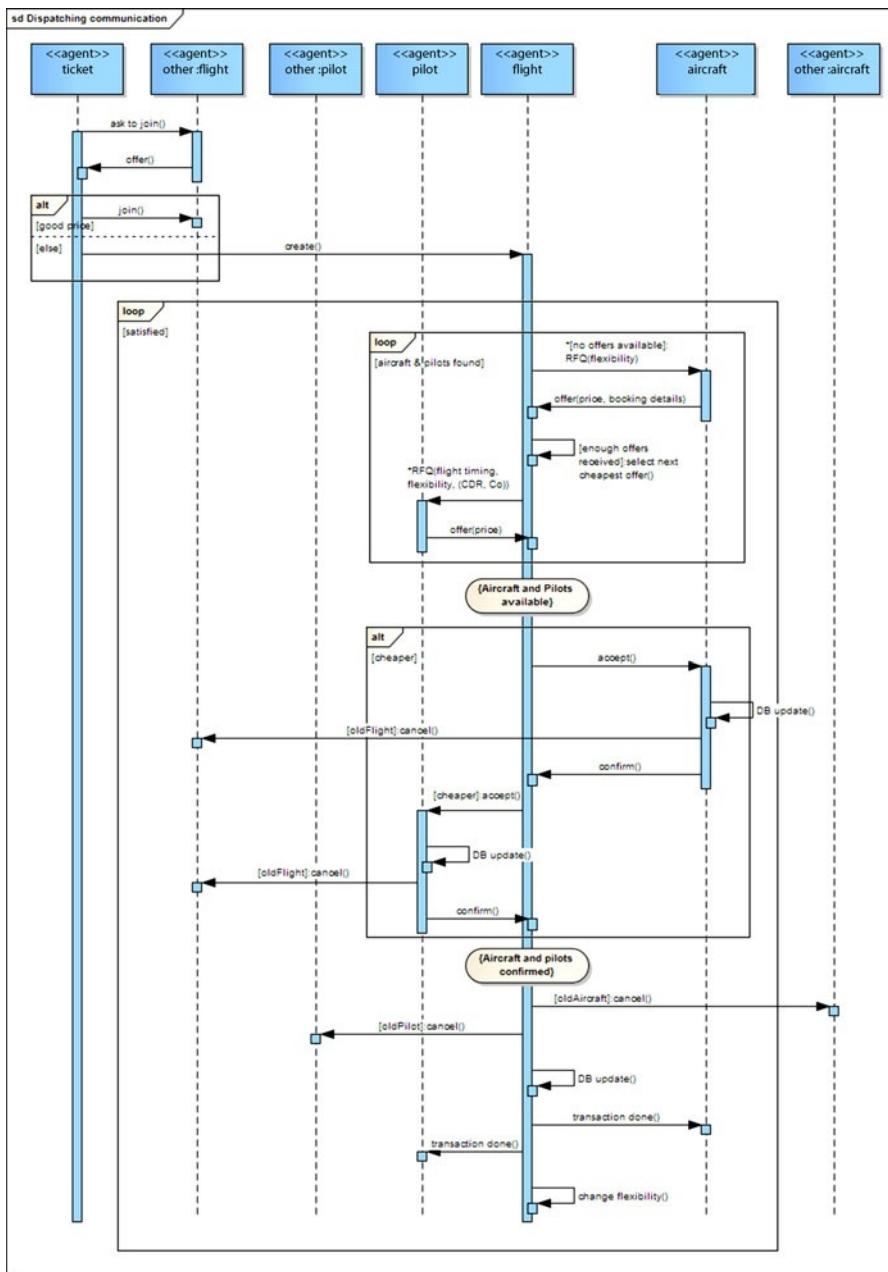


Abb. 14.4 Kern der Interaktion zwischen den Dispositionsgesagten

	OE-FCB C510 4s	OE-FFB C510 4s	OE-FHK C510 4s	OE-FID C510 4s	OE-FHA C510 4s	C
19						
20						
21						
22						
23						
00						
01						
02						
03	LOWL 550C	PEI MTR				
04		LOWL 548C	CTR RRO			
05	LIEO		LSZH		LOWL 540C SKO MFU	
06	549C	PEI MTR TBA MALE	547C	CTR RRO TBA MALE	539C SKO MFU EDDM TBA MALE	
07			LIEO		541C SKO MFU EDOW TBA MALE	
08	LFPG				543C SKO MFU ECLL TBA MALE	
09					545C SKO MFU EDDT TBA MALE	
10						
11						
12						
13						
14						
15						

GAC 549C

LIEO OLBIA	Departure Thu 09 0600	Slot n/a	Blocks Off	Takeoff	HDLG
LFPG City not known	Arrival Thu 09 0817	Slot n/a	Blocks on	Landing	HDLG
Remaining Duty Time 5h15					FPL
Remaining Duty Time 5h15					FUEL
CDR Philipp Eichel					
FO Maria Traugott TBA MALE					

Abb. 14.5 Dispo-Board mit Flügen und Piloten, sowie Flugdetails

dem Dispo-Board durch einfaches Drag&Drop verschoben werden, wodurch es ein Kennzeichen „manuell“ erhält und somit nicht mehr von den Agenten verändert wird.

Dispo-Board

Das Dispo-Board (Abb. 14.5) ist eine flexible und dynamische Kalenderansicht aller Flüge mit einer vertikalen Zeitachse und den Flugzeugen in horizontaler Anordnung. Die Ansicht ist reduziert auf die wesentlichsten Daten je Flug, wobei ein Popup-Window bei Bedarf weitere Details sofort anzeigt. Als Zusammenfassung aller statusrelevanten Teile erscheint ein Flug in der Farbe rot (Handlung notwendig), gelb (beobachten) oder grün (alles OK). Positionierungsflüge sind gestreift dargestellt.

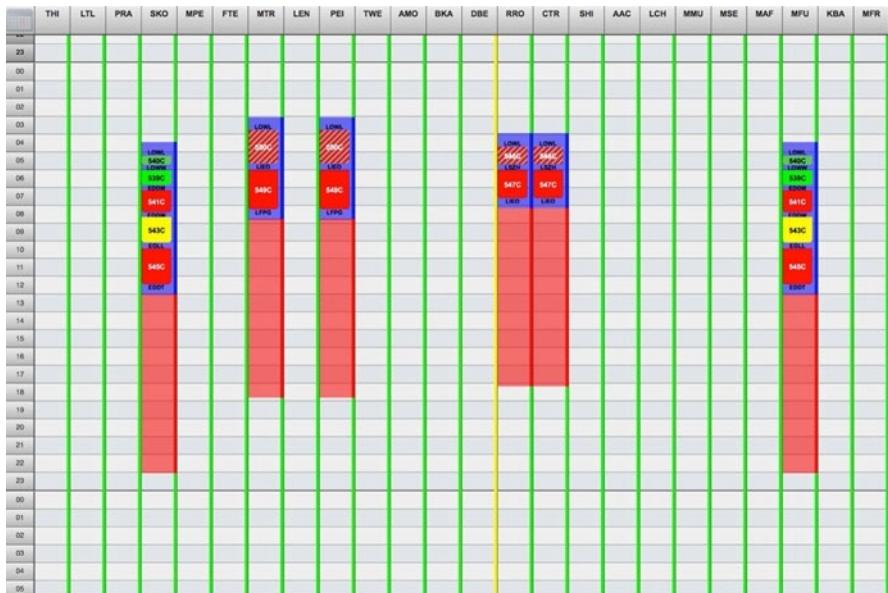


Abb. 14.6 Crew Roster mit den Flügen je Pilot sowie den Arbeitszeiten und den obligatorischen Pausenzeiten

Crew Roster

Das Crew Roster (Abb. 14.6) zeigt die gleichen Flüge wie das Dispo-Board aber mit Anordnung per Pilot. Die Hintergrundfarbe differenziert Arbeitszeiten von Pausenzeiten. Der Rand zeigt an, ob ein Pilot verfügbar ist.

14.4 Evaluierung/Erfahrung und Nutzen

FL3XX befindet sich zum Zeitpunkt der Berichterstellung im ersten Pilotbetrieb. Das Agentensystem hat im Testbetrieb gezeigt, dass die Optimierungsaufgabe jeweils im Bereich von einer Sekunde gelöst werden kann.

Den ersten großen Nutzen haben die Verkaufsmitarbeiter des Pilotanwenders sofort festgestellt: Eine Angebotserstellung dauert durch die schnelle Einplanung und die voll integrierte und automatische Angebotsversendung statt ca. 30 Minuten nur noch 3 Minuten und statt mehrerer 100 Klicks nur noch 10 bis 15 Klicks – je nach Passagieranzahl und Sonderwünsche. Die Zeitsparnis ist enorm.

Für die Disponenten ergibt sich ein ähnlich signifikanter Vorteil, denn auch ihre Aufgabe, alles um den Flug herum zu organisieren, wird ebenfalls durch automatische Service-Buchung bei den Flughäfen viel einfacher.

Beide Nutzen-Blöcke werden aber noch um das eigentliche Ziel der Optimierung ergänzt: Leerflüge werden vermieden und alle Ressourcen werden maximal ausgenutzt, ohne die Luftfahrt-Regularien zu verletzen.

Die ersten Nutzer bzw. Pilotanwender des Systems waren ungläubig erstaunt, dass eine derart komplexe Optimierung unverzüglich in so kurzer Zeit möglich ist. Der Nutzen wurde als sehr hoch bewertet und dem Einsatz wird entgegen gefiebert.

Literatur

- [1] http://en.wikipedia.org/wiki/EU_OPS
- [2] <http://www.atxa.com/>
- [3] http://de.wikipedia.org/wiki/Contract_Net

Kapitel 15

Agentenbasierte Koordination und Überwachung des Designs mechatronischer Systeme

Holger Seemüller, Mohammad Chami und Holger Voos

Zusammenfassung Der Entwicklungsprozess mechatronischer Produkte ist geprägt durch deren interdisziplinären Charakter und steigende Komplexität. Es gilt, diese Komplexität durch methodische Ansätze beherrschbar zu machen und, unter Berücksichtigung gegenseitiger Abhängigkeiten, die beteiligten Entwickler effizient zu koordinieren. Modellbasierte Ansätze zur Modellierung von System und Entwicklungsaktivitäten haben bereits erste Möglichkeiten hervorgebracht, den Entwicklungsprozess in diesem Sinne zu unterstützen. Dieser Beitrag stellt einen agentenbasierten Ansatz zur IT-unterstützten Koordination und Überwachung des Designs mechatronischer Produkte vor. Dabei werden Resultate modellbasierter Ansätze als Wissensbasen verwendet, um Informationen über System und Prozess zu erhalten und gegenseitige Abhängigkeiten zu überwachen. Der Beitrag stellt das Konzept sowie die Agentenarchitektur vor und zeigt dessen Anwendbarkeit anhand eines Anwendungsbeispiels.

15.1 Einleitung

Mechatronische Systeme stellen heute einen ständig wachsenden, enormen weltweiten Markt dar. Die Integration von Lösungsprinzipien aus unterschiedlichen Ingenieursdisziplinen ermöglicht eine Erweiterung der Funktionsvielfalt bei gleichzeitiger Reduktion der Kosten und erfüllt damit wichtige Anforderungen des heutigen Marktes an die Industrie [1]. Besonders die Nutzung von Synergieeffekten zwischen den involvierten Disziplinen erweist sich als starker Innovationsfaktor für

Holger Seemüller (✉), Mohammad Chami
Hochschule Ravensburg-Weingarten, 88241 Weingarten, Deutschland
e-mail: seemueller@hs-weingarten.de, chami@hs-weingarten.de

Holger Voos
Faculty of Science, Technology and Communication, Automatic Control Research Group,
Universität Luxemburg, 6, rue Richard Coudenhove-Kalergi, 1359 Luxemburg, Luxemburg
e-mail: voos@uni.lu

die Entwicklung neuer Lösungskonzepte und Funktionen. Die entstehenden mechatronischen Systeme und Produkte prägen heute Alltag und Industrie, wie zum Beispiel im Automobilbau, der Luftfahrt oder auch im Consumer-Bereich.

Die Mechatronik stellt dabei besondere Anforderungen an den Entwicklungsprozess. Aufgrund des vernetzten Zusammenspiels verschiedener Wissensdomänen sind mechatronische Systeme durch eine hohe Komplexität gekennzeichnet und Ingenieure unterschiedlicher Disziplinen müssen in oftmals verteilten Entwicklungsumgebungen zusammenarbeiten. Die notwendige Kooperation vieler beteiligter Personen sowie die Erfüllung von komplexen, domänenübergreifenden Anforderungen führen zu einem erhöhten Bedarf an Koordinationsprozessen und Mechanismen für den Austausch von Wissen und Informationen [2]. Im Rahmen dieser Zusammenarbeit müssen unterschiedliche Erfahrungen, Vorgehensweisen, Modelle, Notationen und Termini aus zahlreichen Fachrichtungen zusammengeführt und Daten innerhalb einer heterogenen Entwicklungsinfrastruktur ausgetauscht werden. Dieser Austausch benötigt im Vergleich zur traditionellen Produktentwicklung erweiterte methodische und tool-unterstützte Ansätze, um den Entwicklungsprozess zu optimieren.

Eine interdisziplinäre Modellierung des mechatronischen Systems aus einer ganzheitlichen Sicht bietet vielversprechende Möglichkeiten zur Beherrschung der Komplexität. Methodisch unterstützt, ermöglicht die Systemmodellierung eine kollaborative Entwicklung des Produkts unter gleichberechtigter Berücksichtigung disziplinspezifischer Aspekte von den ersten Phasen des Entwicklungsprozesses an. Zahlreiche Notationen und Methoden sind das Resultat dieses Ansatzes, der immer noch Gegenstand der Forschung ist [3, 4]. Diese Notationen sollen den beteiligten Entwicklern eine gemeinsame Sprache bieten, auf deren Basis das System gemeinschaftlich entwickelt und modelliert werden kann.

Weiterhin haben sich Prozessmodelle, wie zum Beispiel das V-Modell der VDI-Richtlinie 2206 [5], etabliert, die dem Entwicklungsprozess einen konzeptionellen Rahmen geben und bei der Koordination der beteiligten Disziplinen unterstützen sollen. Allerdings bewegen sich viele Prozessmodelle auf einer sehr abstrakten Ebene und bieten kaum Unterstützung bei der Planung konkreter Arbeitsabläufe. Weiterhin mangelt es an Toolunterstützung, um Projektbeteiligte durch den Entwicklungsprozess zu führen und diesen zu koordinieren.

Agentensysteme besitzen Eigenschaften, die sich für die Unterstützung von Entwicklungsprozessen eignen. Diese Systeme zeigten in bisherigen Einsatzszenarien besondere Vorteile bei verteilten Aufgaben mit dynamischen Strukturen und Zusammenhängen, wie zum Beispiel bei der Informationssuche, bei verteilten Simulationen oder im Bereich der Logistikplanung [6]. Diese Eigenschaften können auch bei mechatronischen Entwicklungsprozessen beobachtet werden: In der heutigen, globalisierten Welt befinden sich Entwicklungsteams während des Entwicklungsprozesses oft verteilt an verschiedenen Standorten und müssen über größere Entfernung hinweg zusammenarbeiten. Die Koordination dieser Teams ist hier auf Grund von Änderungen von Anforderungen oder Randbedingungen sehr starken Unsicherheiten und vielfachen Einflüssen unterworfen. Diese hohe Dynamik

erschwert eine feste Planung des Prozesses im Vorfeld und erfordert flexible Anpassungsmöglichkeiten an den aktuellen Kontext.

Auf gleiche Weise kann auch das zu entwickelnde mechatronische System unter dem Gesichtspunkt der Verteiltheit betrachtet werden. Dabei wird das System als Menge interagierender Komponenten beschrieben.

Mit jeder dieser Komponenten sind disziplinspezifische Entwicklungs- und Designaktivitäten innerhalb des Entwicklungsprozesses verbunden. Diese Schritte werden von festgelegten Rollen ausgeführt und erzeugen disziplinspezifische Modelldaten. Dabei werden bereits vorhandene Informationen über die jeweiligen Komponenten und ihr Zusammenspiel mit dem System verwendet um diese in daraus abgeleiteten Designaktivitäten zu verfeinern bzw. neue Informationen zu erzeugen. Diese neuen Informationen liefern eine verfeinerte Struktur des Systems, welche wiederum zu neuen Aktivitäten führt. Hierdurch entsteht eine iterative Vorgehensweise bei der System- und Prozessplanung.

Diese bidirektonalen Wechselwirkungen zwischen Komponenten und Aktivitäten ermöglichen es, die Entwicklung eines mechatronischen Systems in seiner Gesamtheit als ein System interagierender Komponenten, Aktivitäten, Rollen und Modelldaten darzustellen. Diese Sichtweise weist auf einen nutzbringenden Einsatz eines Agentensystems für die Unterstützung mechatronischer Entwicklungsprozesse hin.

Dieser Beitrag stellt ein agentenbasiertes Framework vor, das den Koordinationsprozess innerhalb eines mechatronischen Entwicklungsprozesses proaktiv unterstützen und überwachen soll. Das Framework betrachtet den Entwicklungsprozess mit den beteiligten Entwicklern und dem zu entwickelnden Produkt als verteiltes, interagierendes System und repräsentiert entsprechende Rollen als Agenten. Für die Koordination der Agenten werden Informationen aus dem Systemmodell und einer Beschreibung der Entwicklungsaktivitäten verwendet. Abschnitt 15.2 zeigt den aktuellen Stand der Forschung im Bereich der Planung und Modellierung mechatronischer Entwicklungsprozesse sowie der interdisziplinären Systemmodellierung. Abschnitt 15.3 beschreibt das Konzept und die Architektur eines agentenbasierten Engineerings mechatronischer Systeme. Nach einem Anwendungsbeispiel in Abschn. 15.4 wird der Ansatz abschließend in Abschn. 15.5 zusammengefasst und reflektiert.

15.2 Ausgangssituation

Die Unterstützung und Optimierung mechatronischer Entwicklungsprozesse ist Gegenstand zahlreicher Projekte in Forschung und Industrie. Dieser Abschnitt gibt einen kurzen Überblick über den aktuellen Stand der Technik im Bereich der Planung und Modellierung mechatronischer Produktentwicklungsprozesse sowie der interdisziplinären Systemmodellierung.

15.2.1 Modellierung von Entwicklungsprozessen

Entwicklungsprozesse unterscheiden sich von traditionellen Geschäftsprozessen (business processes), die durch etablierte Workflow Engines bereits unterstützt und teilweise automatisiert werden können. Durch fest definierte, stets wiederkehrende Abläufe können diese Workflows formal modelliert und ausgeführt werden. Im Gegensatz dazu weisen Entwicklungsprozesse die Eigenschaft eines festen und wiederholbaren Ablaufes nicht auf. Diese Prozesse sind durch starke Abhängigkeiten von äußeren Einflüssen geprägt, beispielsweise der Änderung von Anforderungen oder sonstigen Randbedingungen. Zusätzlich erschwert die Wechselwirkung zwischen interner Systemstruktur und abgeleiteten Designaktivitäten sowie die oftmals lange Projektlaufzeit eine durchgängige Planung sämtlicher Arbeitsschritte im Vorfeld. Prozessmodelle, wie beispielsweise das V-Modell der VDI-Richtlinie 2206 [5], strukturieren den mechatronischen Entwicklungsprozess durch ein geeignetes Vorgehensmodell. Diese Modelle sind jedoch vergleichsweise abstrakt, wodurch sie einerseits generisch anwendbar sind, andererseits jedoch nur wenig Unterstützung bei der Koordination der alltäglichen Entwicklungsarbeit in der Praxis bieten.

Zahlreiche Forschungsprojekte befassen sich mit dem Management von Entwicklungsprozessen auf unterschiedlichen Abstraktionsebenen. Lindemann [7] untergliedert diese Ebenen in vier Granularitätsstufen zwischen einer sehr detaillierten Mikrologik, welche elementare Handlungsabläufe beschreibt, und einer Makrologik, die Meilensteine des Gesamtprojekts umfasst. Auf abstrakter Ebene beschreiben [8, 9] die Anwendung des V-Modells an konkreten Projekten. Hohenberger et al. [10] beschäftigen sich hingegen auf einer feingranularen Stufe mit den Informationsflüssen und den Zusammenhängen von Informationen zwischen beteiligten Entwicklern und eingesetzten Tools. Die Wahl der Granularitätsstufe hat großen Einfluss auf die Art der Modellierung und Ausführung von Entwicklungsprozessen. Auf diesem Gebiet hat sich bis zum heutigen Zeitpunkt keine standardisierte Notation durchsetzen können. Während in der Industrie Prozesse und Abläufe oftmals in Form einfacher Grafiken oder Gantt-Charts festgehalten werden, hat sich eine formale Modellierung mit Ausführungsmöglichkeiten noch nicht durchsetzen können. Die „Software & Systems Process Engineering Meta-Model Specification“ (SPEM) [11] bietet eine Notation, welche ursprünglich für die Modellierung von Softwareentwicklungsprozessen entworfen wurde, sich jedoch auch auf mechatronische Entwicklungsprozesse anwenden lässt. [2] stellt einen Modellierungsansatz vor, der die starken Abhängigkeiten zwischen Produkt und Prozessschritt auf Metalebene bereits beschreibt und sich damit bei der Modellierung nutzen lässt. Dieser Idee wird auch in dem hier vorliegenden Ansatz aufgegriffen und erweitert.

Agentensysteme haben bereits in früheren Ansätzen Einsatzmöglichkeiten und ihre Eignung für die Ausführung von Geschäftsprozessen gezeigt [12]. Das autonome Verhalten der Agenten zeigt hierbei besondere Vorteile in agilen und dynamischen Umgebungen. Basierend auf diesen Ergebnissen wurden erste Ansätze entwickelt, um mit Hilfe von Agenten Entwicklungsprozesse auszuführen. Ma-

dhusudan [13] stellt einen Ansatz vor, der Agenten in einer Workflow-basierten IT-Infrastruktur einbettet um Produktentwicklungsprozesse zu koordinieren. Braubach et al. [14] verwendet für die Modellierung von Entwicklungsprozessen eine eigene zielorientierte Modellierungssprache, welche für die agentenbasierte Ausführung optimiert wurde. Es konnte gezeigt werden, dass Agentensysteme durch ihre flexiblen Anpassungsmöglichkeiten für die Unterstützung von Entwicklungsprozessen sehr gut geeignet sind.

15.2.2 Systemmodellierung

Die formale Modellierung eines mechatronischen Systems auf ganzheitlicher Ebene sowie die Integration unterschiedlicher disziplinspezifischer Aspekte in einem zentralen Modell ist ein Ansatz für die Beherrschung der zunehmenden Komplexität. Während in früheren dokumentenbasierten Ansätzen Informationen über das System in unstrukturierten Dokumenten oftmals textuell festgehalten wurden, führte die Einführung des Paradigmas des modellbasierten Systems Engineering zu einer zunehmenden Verwendung formaler Modelle. Dadurch sollte nicht nur der Austausch und die Wiederverwendbarkeit von Modellen erleichtert, sondern auch die Kommunikation der Projektbeteiligten verbessert werden [15].

In der Forschung und Industrie wurden deshalb zahlreiche Notationen entwickelt, um ein System ganzheitlich unter Berücksichtigung aller involvierten Ingenieursdisziplinen formal zu modellieren [3, 4]. Unterstützt durch festgelegte Vorgehensweisen, wie zum Beispiel SYSMOD [16], kann das System kollaborativ bereits von den ersten Phasen des Entwicklungsprozesses an entworfen werden. Das Ziel ist hierbei, Aspekte aller Disziplinen gleichberechtigt in das Systemdesign einfließen zu lassen, dadurch Synergieeffekte optimal zu nutzen und spätere, langwierige Iterationen auf Grund von Designänderungen zu vermeiden. Das dabei entwickelte Systemmodell enthält Informationen über die Struktur des Systems, das Verhalten, die Anforderungen sowie die internen Zusammenhänge und Abhängigkeiten.

Dieses Wissen kann auch im weiteren Entwicklungsprozess verwendet werden. Basierend auf der formalen Definition des Systemmodells kann es mit Hilfe geeigneter Modelltransformationen in Simulationsmodelle überführt werden. Dadurch ist es beispielsweise möglich, das dynamische Verhalten des Systems zu analysieren [17]. Andere Ansätze befassen sich mit Möglichkeiten der automatischen Transformation von Informationen aus dem Systemmodell in disziplinspezifische Modelldaten [18, 19].

15.3 Agentenbasiertes Engineering

Die Eigenschaften mechatronischer Entwicklungsprozesse sowie die verteilte Struktur mechatronischer Systeme ermöglichen deren Modellierung als ein dy-

namisches System interagierender Agenten. Basierend auf dieser Sichtweise kann eine Umsetzung des Problems durch einen Agentenansatz abgeleitet werden, die es ermöglicht, bestimmte Unterstützungsauflagen proaktiv und autonom durchführen zu lassen. Dieser Abschnitt stellt die Ziele und konzeptionellen Grundlagen des Ansatzes vor und leitet daraus konkrete Anforderungen an das Framework ab. Im Anschluss wird die Architektur des Agentensystems skizziert und abschließend notwendige Konfigurationsschritte beschrieben.

15.3.1 Ziele

Bei der Entwicklung mechatronischer Systeme müssen Entwickler unterschiedlicher Fachrichtungen, eventuell verteilt an verschiedenen Standorten, ein komplexes System mit einer Vielzahl an internen Abhängigkeiten und Wechselwirkungen erarbeiten. Dabei gilt es, die Entwickler im Sinne des „Concurrent Engineering“ auf möglichst effiziente Art zu koordinieren, um die Entwicklungszeit zu verkürzen und die Konsistenz der disziplinspezifischen Modelle zu gewährleisten. Diese Aufgaben lassen sich durch den Einsatz von Software-Agenten unterstützen, deren Nutzungsmöglichkeiten im Folgenden beschrieben werden.

Der gemeinschaftliche Entwurf eines mechatronischen Systems kann methodisch durch die Verwendung eines ganzheitlichen interdisziplinären Systemmodells unterstützt werden. Innerhalb des mechatronischen Systems bestehen zahlreiche komplexe Abhängigkeiten zwischen disziplinspezifischen Parametern. So erfordert zum Beispiel die Änderung des Raddurchmessers einer mobilen Roboterplattform nicht nur die Anpassung geometrischer Bauteile, sondern auch die Anpassung des kinematischen Modells und der Antriebssteuerung. Weiterhin existieren über Abstraktionsebenen hinweg zusätzliche Abhängigkeiten zwischen Parameterwerten und Anforderungen. Für die Berücksichtigung solcher Abhängigkeiten kann ein ganzheitliches Systemmodell als konsistente Datenbasis verwendet werden. Dieses Modell beinhaltet Informationen und Zusammenhänge unterschiedlicher Disziplinen auf verschiedenen Abstraktionsebenen. Dieses Wissen kann verwendet werden, um die Auswirkung von Parameter- und Designänderungen auf andere Komponenten zu bestimmen und Randbedingungen bzw. Anforderungen zu prüfen. Durch die verteilte Struktur eines mechatronischen Systems kann diese Prüfung von Auswirkungen sowie die Propagierung von Änderungen durch Agenten unterstützt werden. Dabei vertritt jeder Agent jeweils eine Systemkomponente und besitzt das notwendige Wissen über eigene Parameter sowie Wechselwirkungen mit anderen Komponenten bzw. Agenten. Bei einer Änderung können die Agenten miteinander kommunizieren und die übergreifenden Auswirkungen bestimmen.

Modelle von auszuführenden Designaktivitäten helfen bei der Koordination von Entwicklungsprozessen [20]. In der Praxis werden hierfür noch häufig einfache grafische Darstellungen verwendet, um den Ablauf eines Entwicklungsprozesses zu modellieren. Für eine Optimierung der Entwicklungszeit im Sinne des Concurrent Engineering ist die Verwendung von Annahmen und Abschätzungen besonders

in der frühen Phase des Entwicklungsprozesses gängige Praxis, um die Nebenläufigkeit von Aktivitäten zu erhöhen. Hierdurch sollen Abhängigkeiten zwischen einzelnen Prozessschritten überwunden und Wartezeiten verkürzt werden. Ein wesentlicher Nachteil dieses Verfahrens ist jedoch die Propagierung von geänderten Annahmen oder Konkretisierungen an die betroffenen Entwickler. Die Prozesskoordination lässt sich durch die Verwendung eines Agentensystems unterstützen, da diese selbstständig die Planung der Abfolge von Designaktivitäten übernehmen können und, basierend auf diesen zeitlichen Abhängigkeiten, Änderungen weiterreichen können. Im Verlauf des Designprozesses wird von den Entwicklern aus den einzelnen Fachdisziplinen eine Vielzahl unterschiedlicher disziplinspezifischer Modelle durch spezialisierte IT-Tools erstellt, zum Beispiel Schaltpläne, CAD-Modelle oder Modelle des dynamischen Verhaltens. Diese Modelle basieren auf Informationen aus dem interdisziplinären Systemmodell, wodurch Parameter aus diesem in den disziplinspezifischen Modellen enthalten sind. Mit Fortschreiten des Entwicklungsprozesses muss die Konsistenz zwischen Daten des interdisziplinären Systemmodells und der disziplinspezifischen Modelle gewährleistet werden. So müssen Änderungen am übergeordneten Systemmodell an die Werkzeuge zur Bearbeitung der disziplinspezifischen Modelle automatisch weitergegeben und Änderungen an disziplinspezifischen Modellen mit dem übergeordneten Systemmodell abgeglichen werden. Diese Überwachung findet wieder in einer heterogenen verteilten Umgebung statt, wodurch der Einsatz eines Agentensystems zweckmäßig ist.

Zusammengefasst können damit drei Arten von Beziehungen identifiziert werden, welche durch Agenten überwacht und abgeglichen werden können:

1. Abhängigkeiten und Wechselwirkungen innerhalb des mechatronischen Systems zwischen verschiedenen Abstraktionsebenen
2. Chronologische Beziehungen zwischen einzelnen Designaktivitäten
3. Konsistenzbeziehungen zwischen interdisziplinärem Systemmodell und disziplinspezifischen Modelldaten

Zwischen den identifizierten Beziehungen existieren weitere Wechselwirkungen. So bestehen zwischen Designaktivitäten und dem zu entwickelnden Produkt enge Zusammenhänge [2, 21, 22]. Konkret basieren diese Aktivitäten auf dem Aufbau und den Komponenten des Systems. Damit dienen Informationen aus dem Systemmodell als Eingangsdaten für disziplinspezifische Designaktivitäten. Während der Ausführung dieser Aktivitäten werden diese Informationen entweder verfeinert und an das Systemmodell reflektiert oder neue Informationen dienen als Erweiterung für das bestehende Modell. Es entsteht ein iterativer Kreislauf zwischen Systemmodellierung und Aktivitätsplanung. Basierend auf diesem Zusammenhang können nach einer Änderung eines Systemparameters damit nicht nur Wechselwirkungen innerhalb des Systems sondern auch Auswirkungen auf bereits ausgeführte Designaktivitäten ermittelt und propagiert werden. Auf Basis dieses Grundsatzes können getroffene Annahmen für die Ausführung von Aktivitäten nachträglich propagiert werden.

Weiterhin sind Designaktivitäten stets mit einer oder mehreren Rollen assoziiert, welche die Aufgabe ausführen und dabei disziplinspezifische Modelldaten erzeu-

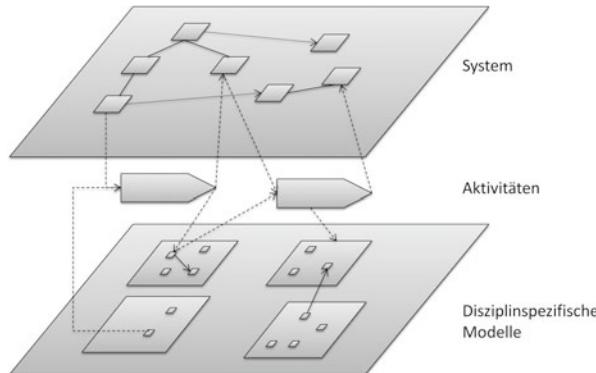


Abb. 15.1 Aktivitäten als Bindeglied zwischen Abstraktionsebenen

gen. Die Aktivitäten verwenden Eingabedaten aus dem Systemmodell und erzeugen neue Ausgabedaten und -informationen. Basierend auf dieser Erkenntnis können Beziehungen zwischen Informationen aus dem Systemmodell, involvierten Rollen und den verteilten Modelldaten erzeugt werden, die für die Konsistenzprüfung verwendet werden können.

Aus diesen Wechselwirkungen ergibt sich ein System aus verteilten Prozessbestandteilen, welche, basierend auf den beschriebenen Beziehungen, miteinander interagieren. Abbildung 15.1 stellt diesen Zusammenhang mit einem 3-Ebenen-Modell dar. Auf der Systemebene wird das Produkt mit den enthaltenen Wechselwirkungen mit Hilfe eines interdisziplinären Systemmodells beschrieben. Die disziplinspezifische Modellebene stellt die Modelldaten dar, welche das Produkt mit den spezialisierten Modelldaten beschreibt. Die Aktivitätsebene repräsentiert in dieser Sichtweise das verbindende Glied zwischen System- und Modellebene, da es einerseits Informationen aus dem Systemmodell liest, verfeinert und erzeugt und gleichzeitig disziplinspezifische Modelldaten erzeugt.

Ein Agentensystem eignet sich in besonderer Weise für die proaktive und selbstständige Verwaltung dieses Systems aus Prozessbestandteilen und Beziehungen. Durch die verteilte Struktur sowie die dynamischen Abläufe und Interaktionen können Rollen, Aktivitäten und das mechatronische System sinnvoll als Agenten abgebildet werden. Basierend auf den drei Ebenen der Konzeptbeschreibung können in einem ersten Entwurfsschritt drei Gruppen von Agenten identifiziert werden:

1. Repräsentationsagenten des mechatronischen Systems
2. Koordinationsagenten für die Planung des Entwicklungsprozesses
3. Repräsentationsagenten der Entwickler aus den Fachdisziplinen und ihrer Modelldaten

Die Agenten erhalten ihr notwendiges Wissen über das System aus einem interdisziplinären und ganzheitlichen Systemmodell sowie einer formalen Beschreibung der Designaktivitäten.

15.3.2 Anforderungsanalyse

Aus dem beschriebenen Konzept lassen sich konkrete Anforderungen an das Framework ableiten. Diese lassen sich in Anforderungen, die sich aus der Entwicklungsmethode ergeben, Anforderungen bzgl. der Infrastruktur der Entwicklungsumgebung, sowie funktionalen Anforderungen an die Unterstützungsaufgaben der Agenten unterteilen.

Methodische Anforderungen

Neuentwicklungen mechatronischer Produkte können aufgrund der hohen Komplexität nicht in einem Schritt sondern in einer iterativen Vorgehensweise erfolgen. Dies bedeutet, dass das Systemmodell ausgehend vom Erstentwurf mit fortschreitender Entwicklungszeit immer weiter detailliert wird. Bei der Planung und Entwicklung eines komplexen Systems wird hier von einer komponentenbasierten Dekomposition des Gesamtsystems ausgegangen [23]. Bei diesem Ansatz wird ein System in Systeme, Subsysteme und mechatronische Komponenten hierarchisch gegliedert. Eine mechatronische Komponente stellt die atomare Einheit dar, welche Informationen unterschiedlicher Disziplinen integriert. Das vorgestellte Konzept eines agentenbasierten Unterstützungssystems soll diese Ansätze einer komponentenbasierten Entwicklung eines Systemmodells unterstützen.

Mit fortschreitender Reife des Systemmodells müssen parallel disziplinspezifische Designaktivitäten geplant und passenden Rollen zugeordnet werden. Diese Aktivitäten verwenden Informationen aus dem Systemmodell sowie die Ergebnisse vorheriger Aktivitäten und erstellen neue Modelldaten. Das Resultat dieser Aktivitäten sind verfeinerte oder neue Informationen, welche in das Systemmodell reflektiert werden können. Damit wird das Modell in der nächsten Iteration erneut detailliert und erhält einen weiteren Reifegrad. Diese neue Version des Systemmodells erzeugt erneut weitere Designaktivitäten. Das Framework soll diese bidirektionale Beziehung zwischen System und Designaktivitäten unterstützen und eine parallele Planung und Modellierung von System und Aktivitäten ermöglichen.

Infrastrukturanforderungen

Die interdisziplinäre Entwicklung mechatronischer Systeme geschieht in einer heterogenen und verteilten IT-Infrastruktur. Experten aus unterschiedlichen Fachrichtungen arbeiten verteilt an verschiedenen Standorten mit ihren etablierten Entwicklungswerkzeugen. Dabei kommt eine Vielzahl an Notationen und Werkzeugen zum Einsatz, welche nur schwer austauschbare Modelldaten erzeugen. Das Agentensystem muss in diese Entwicklungsumgebung integriert werden. Diese Integration beinhaltet die Netzwerkfähigkeit des Frameworks sowie die Möglichkeit, flexibel mit unterschiedlichen Entwicklungswerkzeugen kommunizieren zu können.

Der Ansatz der interdisziplinären Systemmodellierung hat eine Vielzahl unterschiedlicher Notationen hervorgebracht. Teilweise wurde für diese Notationen bewusst keine formale Semantik definiert, um eine generische Anwendung zu ermöglichen. So kann SysML mit Hilfe von Profilen und Stereotypen projektspezifisch angepasst werden [3]. Für eine nahtlose Integration soll das Framework unabhängig von der definierten Semantik mit beliebigen Systemmodellierungssprachen arbeiten können.

Für die Modellierung von Designaktivitäten hat sich bis heute noch kein Standard etablieren können. Während in der Praxis oftmals mit proprietären Beschreibungen gearbeitet wird, haben sich in der Forschung unterschiedliche Notationen entwickelt. Wie bei der Systemmodellierung soll sich das Framework in eine bestehende Infrastruktur integrieren lassen und mit beliebigen modellbasierten Sprachen arbeiten können.

Funktionale Anforderungen an Agentensystem

Basierend auf den Zielen des vorgestellten Konzepts lassen sich konkrete funktionale Anforderungen an das Framework stellen. Das Agentensystem soll die Aktivitäten der Beteiligten an einem Entwicklungsprojekt im Sinne des Concurrent Engineering koordinieren und planen. Als Basis für diese Planung dient ein Modell der Aktivitäten mit einer Beschreibung der benötigten und produzierten Produktinformationen. Basierend auf vorhandenen Ressourcen soll das Framework selbstständig die Aktivitäten an die verantwortlichen Rollen verteilen und die chronologischen Abhängigkeiten speichern. Parallel soll das Framework die Entwickler bei der Abarbeitung der Aktivitäten proaktiv unterstützen und Parameteränderungen prüfen und ggf. an betroffene Entwickler weiterleiten. Diese Propagierung von Änderungen geschieht auf Basis von Wechselwirkungen innerhalb des Systems sowie den chronologischen Abhängigkeiten der bereits durchgeführten Aktivitäten. Für die Benutzerinteraktion mit dem Framework soll eine grafische Benutzeroberfläche angeboten werden.

15.3.3 Architektur des Agentensystems

Die genannten Anforderungen dienen als Grundlage für den Aufbau eines Agentensystems zur Unterstützung mechatronischer Entwicklungsprozesse. Die in der Konzeptbeschreibung identifizierten Gruppen dienen hierfür als Basis und werden entsprechend den Kernelementen des Entwicklungsprozesses verfeinert. Die Kernelemente sind: (1) die einzelnen Komponenten des Systems, deren Schnittstellen sowie deren Abhängigkeiten und Wechselwirkungen, (2) die Anforderungen an das System zusammen mit den Beziehungen zu den Komponenten, (3) Designaktivitäten und (4) involvierte Rollen. Diese Menge an Kernelementen ermöglicht die Repräsentation des aktuellen Kontextes der Entwicklung. Es ist zu beachten, dass

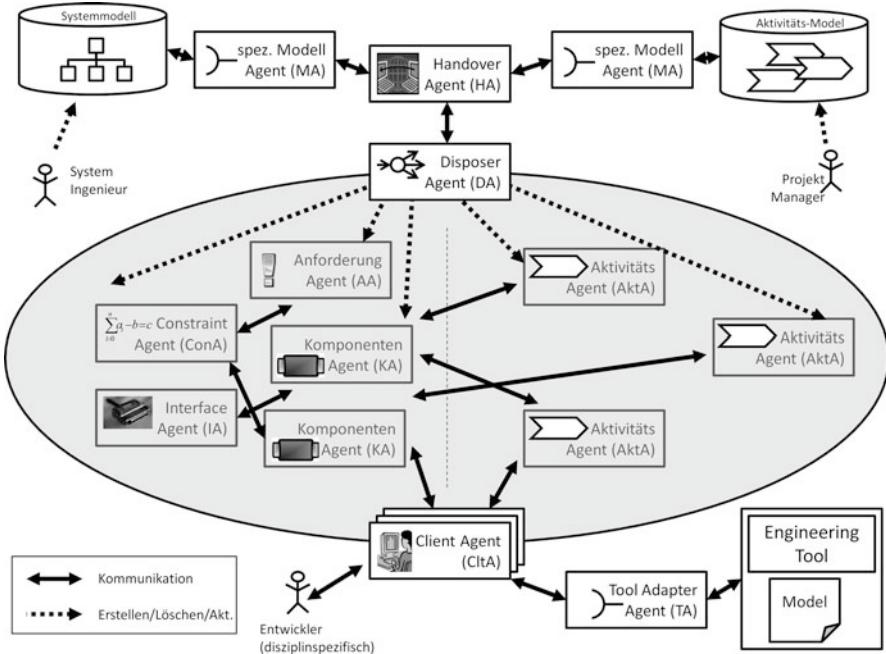


Abb. 15.2 Architektur des Agentensystems

eine iterative Entwicklung zu einer stetigen Änderung des Kontextes und damit zu einer dynamischen Anpassung der Kernelemente führt. Damit ergibt sich je nach Entwicklungsstand eine unterschiedliche Konfiguration für das Agentensystem. Für die Erfüllung der Infrastruktur- sowie methodischen Anforderungen werden zusätzlich spezialisierte Agenten für die Behandlung disziplinspezifischer Modelldaten bzw. System- und Aktivitätsmodell benötigt und im nächsten Abschnitt separat vorgestellt. Aus diesen Überlegungen lassen sich die im Folgenden beschriebenen Agenten ableiten: Komponenten-Agenten (KA), Interface-Agenten (IA), Anforderungs-Agenten (AA), Constraint-Agenten (ConA), Aktivitäts-Agenten (AktA), Client-Agenten (CltA). Die spezialisierten Agenten setzen sich zusammen aus den Modell-Agenten (MA), dem Handover-Agenten (HA), dem Disposer-Agenten (DA), sowie den Tool Adapter-Agenten (TA) (siehe Abb. 15.2 für eine Darstellung der Architektur des Agentensystems).

Ein Komponenten-Agent (KA) repräsentiert ein mechatronisches System, Subsystem oder Komponente innerhalb des Gesamtsystems. Eine solche Komponente vereint Informationen und Aspekte aus unterschiedlichen Fachrichtungen und kann deshalb keiner eindeutigen Disziplin zugeordnet werden [23, 24]. Der Agent besitzt Wissen über die einzelnen Parameter und ihre Werte. Weiterhin kennt er Beziehungen zu seinen Schnittstellen (IAs), zu Anforderungen (AAs) sowie die Wechselwirkungen (ConAs) zu anderen Agenten. Darüber hinaus speichert er alle

Aktivitäten, die Informationen aus seinem Kontext abrufen oder ändern. Ein KA besitzt als primäres Ziel die Erhaltung der Konsistenz seiner Parameter. Erkennt er eine Änderung eines Parameters, so kann er prüfen, ob andere Agenten davon betroffen sind und informiert diese.

Ein Interface-Agent (IA) speichert das Wissen über eine Schnittstelle basierend auf dessen Darstellung durch Parameter. Ein IA dient als Verbindungsglied zwischen zwei KAs, die über eine Schnittstelle verbunden sind. Weiterhin kann auch ein IA Beziehungen zu ConAs aufbauen, um Wechselwirkungen zwischen der Schnittstelle und anderen Elementen des Systems zu propagieren. Das Ziel des IA ist der Erhalt der Konsistenz seiner Parameter sowie die Propagierung von Änderungen.

Anforderungen an das System werden durch Anforderungs-Agenten (AA) repräsentiert. Mit Hilfe von Parametern können funktionale Anforderungen genauer spezifiziert und über ConAs Beziehungen zu KAs aufgebaut werden. Diese Beziehungen können von dem Agenten in zwei Richtungen verwendet werden. Zunächst können Änderungen an den Anforderungen top-down-gerichtet an die betroffenen KAs weitergegeben werden, welche die direkten oder indirekten Auswirkungen prüfen und ggfs. an die CltAs weitergeben können. Umgekehrt ist es möglich, Parameteränderungen innerhalb einer Komponente selbstständig gegenüber spezifizierten Anforderungen zu prüfen. Im Falle einer Verletzung kann der Ursprung der Änderung benachrichtigt werden.

Constraint-Agenten (ConA) stellen das zentrale Verbindungsglied zwischen KAs, IAs und AAs dar. ConAs kennen die Beziehung zwischen Parametern anderer Agenten, welche mit Hilfe mathematischer Ausdrücke spezifiziert werden können. Nach einer Änderung eines Parameters, bspw. einer Komponente, wird der ConA von dem betroffenen Agenten benachrichtigt und dieser kann die Abhängigkeit auswerten. Im Fall einer Beeinflussung eines anderen Agenten kann dieser benachrichtigt werden und die Änderung wird kaskadierend weitergeprüft.

Aktivität-Agenten (AktA) vertreten disziplinspezifische Designaktivitäten. Sie besitzen das Wissen, welche Daten aus dem Systemmodell benötigt werden und welche neuen Informationen die Aktivität liefert. Weiterhin kennt ein AktA die zugeordnete Rolle, die die jeweilige Aktivität ausführen soll, sowie das zur Umsetzung verwendete Design Tool. Basierend auf diesem Wissen ist der Aktivitäts-Agent selbstständig in der Lage, seine Ausführung entsprechend der verfügbaren Informationen zu koordinieren und weiterhin zeitliche Abhängigkeiten zu speichern. Zudem stellt der AktA die Relation zwischen Systemmodell und entsprechenden disziplinspezifischen Modelldaten her.

Ein Client-Agent (CltA) dient als Stellvertreter eines Entwicklers aus einer bestimmten Fachdisziplin. Der Agent bietet dem Benutzer eine grafische Benutzeroberfläche, welche anstehende Aufgaben darstellt und ihm die Möglichkeit zu deren Ausführung bietet. Ein CltA ist damit die Schnittstelle zwischen einem menschlichen Benutzer und dem Agentenframework.

Das Wissen der Agenten ist verteilt in disjunkte Datenmengen, die den entsprechenden Agenten zugeteilt sind. Diese beinhalten lokale, agentenspezifische Informationen eines Agenten (z. B. die Parameter einer Komponente oder die Be-

schreibung einer Anforderung) und speichern die Abhängigkeiten und Beziehungen zu anderen Agenten. Diese Informationen werden zentral aus dem Systemmodell und einer formalen Beschreibung der Aktivitäten gelesen und zur Laufzeit für die Konfiguration der entsprechenden Agenten verwendet.

15.3.4 Laufzeitkonfiguration des Agentensystems

Agenten, die Kernelemente des Entwicklungsprozesses repräsentieren, müssen auf Grund der iterativen Entwicklung stets den aktuellen Kontext des Entwicklungsprozesses wiederspiegeln. Die Bereitstellung der dazu notwendigen Informationen erfolgt dabei durch eine automatische modellbasierte Konfiguration des Agentensystems. Diese dynamische Konfiguration sorgt für den Erhalt der Konsistenz zwischen dem realen Kontext des Entwicklungsprozesses sowie der Agentenrepräsentation. Hier muss einerseits sichergestellt sein, dass jedes Kernelement des Entwicklungsprozesses als Agent vertreten ist. Weiterhin muss das Wissen bereits vorhandener Agenten aktualisiert werden. Üblicherweise wird bei der Neuentwicklung von Systemen zunächst mit abstrakten Modellen gearbeitet, die durch daraus abgeleitete Aktivitäten verfeinert werden. Dadurch kann das Modell der internen Struktur des Systems erweitert und neue Aktivitäten geplant werden. Im Sinne dieser iterativen Entwicklung des mechatronischen Systems muss der Entwicklungskontext stets überwacht und das Agentensystem an den aktuellen Zustand angepasst werden.

Die Ermittlung des aktuellen Zustandes des Entwicklungsprozesses geschieht durch die Überwachung des Systemmodells sowie des Modells der Entwicklungsaktivitäten. Das Systemmodell beinhaltet alle Informationen über das mechatronische System zum jeweils aktuellen Entwicklungsstand. Während in den frühen Phasen des Prozesses das Modell noch sehr abstrakt ist und nur eine grobe Granularität der modellierten Komponenten und Parameter aufweist, wird dieses Modell im Verlauf der Entwicklung mehr und mehr detailliert. Die Komponenten-, Interface-, Constraint- und Anforderungs-Agenten müssen dementsprechend konfiguriert werden. Ähnlich enthält das Modell der Entwicklungsaktivitäten eine formale Darstellung der Aufgaben der Entwickler. Jede dieser Aktivitäten muss durch einen entsprechenden Aktivitäts-Agenten vertreten werden.

Die Realisierung dieser Anforderung geschieht über vier Konfigurations-Agenten: zwei spezialisierten Model-Agenten (MA), dem Handover-Agent (HA) sowie dem Disposer-Agent (DA). Die MAs sind für die Überwachung des Systemmodells sowie des Aktivitätsmodells zuständig. Diese Aufgabe ist von besonderer Bedeutung für die Steuerung des Entwicklungsprozesses, da sich die Modelle mit Verlauf des Prozesses weiterentwickeln und die Konfiguration des Agentensystems an die aktuelle Struktur des Systems sowie an die geplanten Aktivitäten angepasst werden muss. Die Vielzahl an unterschiedlichen Notation sowie deren teils nicht formal definierte Semantik erfordert den Einsatz von spezialisierten Agenten, welche an die Notation sowie die verwendete Semantik der Metamodell-Elemente angepasst sind. Diese Agenten überwachen kontinuierlich die Modelle

und transformieren diese für die weitere Verarbeitung in eine Instanz festgelegter Metamodelle. Diese Modelle nimmt der HA entgegen, der Bezüge zwischen Aktivitäten und den Systemmodellelementen aufbaut und das Gesamtmodell in ein Konfigurationsmodell des Agentensystems transformiert. Dieses Modell dient dem Disposer-Agenten, der den Erhalt der Konsistenz zwischen aktiven Agenten und den Informationen des Agentenkonfigurationsmodells sicherstellt. Diese Transformationskette ermöglicht eine flexible Überwachung und Interpretation des aktuellen Entwicklungszustandes.

Zur Überwachung disziplinspezifischer IT-Tools werden spezialisierte Tool-Adapter-Agenten (TA) eingesetzt, welche die Tools kapseln und einen definierten Zugriff auf die Modelldaten bieten. Dadurch ermöglichen sie eine automatische und proaktive Überwachung der Daten.

15.4 Anwendungsbeispiel

Das beschriebene Konzept eines agentenbasierten Unterstützungssystems für mechatronische Entwicklungsprozesse wurde prototypisch auf Basis der Agentenplattform Jadx [25] realisiert. Metamodelle wurden mittels Eclipse EMF [26] entwickelt und innerhalb des Handover-Agenten implementiert, um die Schnittstelle zwischen dem Framework und den verwendeten Modellierungswerkzeugen formal zu definieren. Die Metamodelle werden in [27] näher erläutert. In einer früheren Arbeit wurde für den praktischen Einsatz eine SysML-basierte Modellierungsmethode entwickelt [28] und ein darauf abgestimmter Systemmodell-Agent implementiert. Dieser analysiert das Modell und transformiert es in eine Instanz des Schnittstellen-Metamodells. Parallel wurden unterschiedliche Prozessmodellierungsansätze zusammengetragen und auf eine Eignung für das Framework hin evaluiert.

Der folgende Abschnitt skizziert die Verwendung des Frameworks an der vereinfacht dargestellten Entwicklung einer mobilen Roboterplattform. Zu Beginn der Entwicklung steht eine Liste von Anforderungen zur Verfügung, die die Funktionalität und Leistung des Roboters beschreiben. Es werden im Vorfeld keine Einschränkungen bzgl. der Realisierung und Umsetzung getroffen.

15.4.1 Iterative Entwicklung der Modelle

Neuentwicklungen mechatronischer Systeme können aufgrund ihrer Komplexität in der Regel nicht vollständig im Vorfeld entworfen und geplant werden. Vielmehr ist es nötig, iterativ Entwurfs- und Simulationsschritte durchzuführen, um geeignete Lösungsprinzipien zu erarbeiten, diese gegeneinander abzuwägen und das System damit zu der nötigen Reife zu führen. Ein erster Grobentwurf wird dadurch verfeinert, bis er den gewünschten Grad an Granularität und Detaillierung besitzt.

Dieses Vorgehen wird durch das vorgestellte Framework unterstützt. Mit Hilfe von SysML wird zu Beginn des Systementwurfs ein initiales Systemmodell erstellt, das die gesammelten Anforderungen an die Plattform enthält. Weiterhin wird die Roboterplattform als Black Box mit wichtigen Parametern modelliert. Der Systemmodell-Agent erkennt die Änderung, liest das Modell selbstständig ein und leitet es an den Handover-Agenten weiter. Da noch kein Aktivitätsmodell vorliegt, wird das Modell für die Weiterverarbeitung transformiert und dem DA übergeben. Dieser verwendet das erhaltene Konfigurationsmodell und erstellt damit eine erste Agentenpopulation. Auf diese Weise werden für jede modellierte Anforderung ein AA, für die abstrakte Systemkomponente ein KA und für eventuell modellierte Abhängigkeiten jeweils ein ConA erstellt. Zusätzlich werden die Wissensbasen der Agenten mit den Informationen aus dem Systemmodell initialisiert.

Für die weitere Entwicklung werden nun erste Designaktivitäten durchgeführt. Die Realisierung einer mobilen Roboterplattform kann durch unterschiedliche Arten von Antrieben umgesetzt werden. So entscheidet die Anzahl und Anordnung der Antriebsräder über das Erreichen der Anforderungen an die Plattform. In einem ersten Designschritt wird deshalb das kinematische und dynamische Verhalten mehrerer Anordnungen modelliert. Dieser Schritt wird in einem Aktivitätsmodell formal definiert. Dabei werden die Tätigkeiten textuell beschrieben und einer Rolle zugewiesen. Weiter müssen die Ein- und Ausgabeinformationen, die die Aktivität aus dem Systemmodell benötigt bzw. liefert, spezifiziert werden. So benötigt der Entwickler für die Erstellung des Modells das Gewicht, die maximale Geschwindigkeit und die maximale Beschleunigung und liefert neue Werte hinsichtlich der Anordnung des Antriebes. Der Aktivitätsmodell-Agent erkennt die neue Aktivität und leitet das Modell an den HA weiter. Dieser stellt nun die Verbindung zwischen Ein- und Ausgabeparametern der Aktivität und den modellierten Parametern des Systemmodells her. Abschließend leitet er das aufbereitete Modell an den DA weiter, der einen AktA erstellt. Dieser Agent prüft nun autonom, ob alle notwendigen Informationen verfügbar sind und weist die Aktivität der entsprechenden Rolle zu. Während der Abarbeitung der Aufgabe durch den Entwickler prüft der KA die Einhaltung der Eingabeparameterwerte und Anforderungen des Systemmodells. Der Entwickler liefert als Ergebnis der Aktivität die mathematischen Modelle der Anordnungen und eine erste Menge von Parametern, die den Aufbau näher beschreiben und eine Realisierung der modellierten Anforderungen erlauben.

Diese Erkenntnisse fließen nun in die Weiterentwicklung des Systemmodells mit ein. Nach Evaluierung der unterschiedlichen Realisierungsmöglichkeiten wird eine Entscheidung für ein bestimmtes Design getroffen und das Systemmodell den neuen Erkenntnissen angepasst. Durch die Anpassung erhält das Modell neue Komponenten, welche durch die gelieferten Mengen von Parametern näher beschrieben werden. Die linke Seite der Abb. 15.3 zeigt den Fortschritt des Systemmodells. Während zu Beginn des Prozesses nur wenige Informationen in dem Modell gespeichert waren, enthält es nun eine Dekomposition der Plattform in die einzelnen Antriebeinheiten, dem Chassis und der Controllerbox. Aus diesen neuen Komponenten können nun weitere Designaktivitäten abgeleitet werden. Dazu werden in dem Aktivitätsmodell neue Aktivitäten hinzugefügt. Elektrotechnik-Ingenieure werden be-

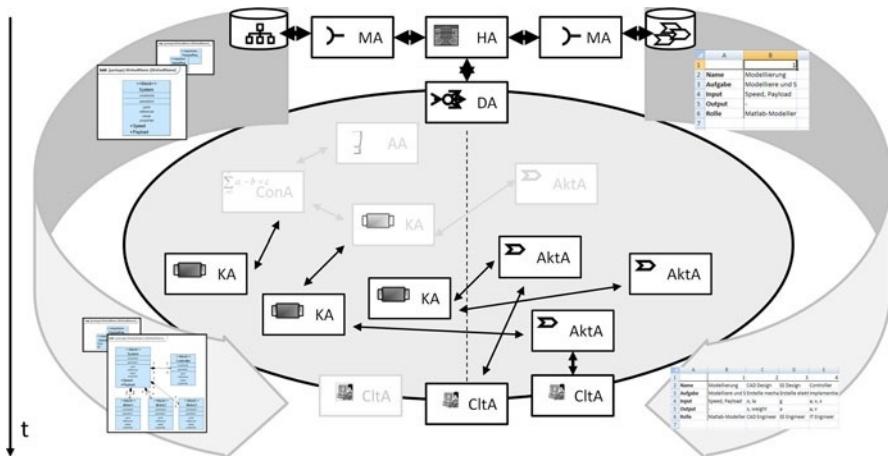


Abb. 15.3 Beispiel einer Laufzeitkonfiguration des Agentensystems

auftragt, nach geeigneten Komponenten für den Antrieb und die Energieversorgung zu suchen. Für die Planung geeigneter Gehäusebauteile werden Maschinenbau-Ingenieure angewiesen, diese mit Hilfe von CAD-Werkzeugen zu modellieren. Zudem muss die Ansteuerung der Motoren von Informatikern programmiert werden.

Der MA der Aktivitäten erkennt die neuen Aktivitäten und leitet die nötigen Informationen über den HA an den DA weiter. Dieser erstellt daraufhin drei neue AktAs, welche aktiv auf die Verfügbarkeit der nötigen Eingabeparameter und der zugewiesenen Rollen warten und abschließend die Aktivität einer Rolle zuweisen. Abbildung 15.3 zeigt die Agentenpopulation nach dieser Iteration. Die in Grau dargestellten Agenten stellen die Repräsentanten aus der ersten Iteration dar. Diese Agenten sind weiterhin aktiv und werden ggfs. mit neuem Wissen aktualisiert. Die in Schwarz dargestellten Agenten sind das Ergebnis des neuen Modells und stellen somit die Konsistenz zwischen Modelldaten und Agentensystem wieder her.

Es ist offensichtlich, dass die Aktivitäten starke Wechselbeziehungen aufweisen und für ihre Ausführung Informationen anderer Entwicklungsteams benötigen. Gleichzeitig haben eigene Designentscheidungen Einfluss auf die Arbeit anderer Aktivitäten. In traditionellen Entwicklungsprozessen wird diese Tätigkeit sequentiell ausgeführt und Entwickler müssen auf die Ergebnisse anderer Arbeitsschritte warten. Eine nachträgliche Änderung der Mechanik ist aus diesem Grund mit starken Verzögerungen verbunden. Durch den Einsatz des Agentenframeworks kann die Arbeit besser parallelisiert werden. KAs überwachen dazu die disziplinspezifischen Parameter und prüfen deren Wechselwirkung mit anderen Aktivitäten und KAs. Nach einer Änderung werden modellierte Randbedingungen wie zum Beispiel Anforderungen geprüft und der neue Wert an betreffende Entwickler automatisch weitergeleitet. Diese können dann weiter mit konsistenten Daten arbeiten. Gleichzeitig speichern die KAs, welche Aktivität zu welchem Zeitpunkt Lese- bzw. Schreibzugriff auf einen bestimmten Parameter hatte.

Auf diese Weise ist es möglich, die Designaktivitäten parallel durchführen zu können. Durch die Kommunikation zwischen den Agenten wird sichergestellt, dass alle Entwickler mit konsistenten und gültigen Mengen von Parametern arbeiten können. Zu Beginn der Aktivität getroffene Annahmen werden somit zeitnah aktualisiert. Weiterhin ermöglicht das Speichern der Lese- und Schreibzugriffe auf Parameter eine nachträgliche Benachrichtigung der betroffenen Entwickler im Fall einer späteren Designänderung des Modells.

15.5 Zusammenfassung

Agentensysteme können durch ihr autonomes und proaktives Verhalten den Designprozess komplexer, mechatronischer Systeme unterstützen. Das vorgestellte Konzept eines agentenbasierten Unterstützungssystems zeigt Möglichkeiten auf, basierend auf Systemmodellinformationen und formal modellierten Aktivitäten den Entwicklungsprozess zu koordinieren. Die Koordination erfolgt durch die proaktive Überwachung von Designparameteränderungen und der Analyse der Auswirkung auf betroffene Entwickler. Zusätzlich wird eine feingranulare Beschreibung der Designaktivitäten verwendet, um Bezüge zwischen disziplinspezifischen Modelldaten und Entwicklern und der Evolution des Systemmodells herzustellen.

Diesen wird dadurch ein transparentes Werkzeug angeboten, welches ihre Änderungen überwacht, anstehende Aufgaben zuteilt und sie über Änderungen benachrichtigt. Das Systemmodell dient in diesem Ansatz als Mittel der Komplexitätsbewältigung beim Systementwurf sowie als zentrale Datenbasis. Dadurch grenzt sich das Framework von Ansätzen ab, die das Systemmodell als Basis für ganzheitliche Simulationen und Analysen verwenden oder mit Hilfe von Modelltransformationen Modellskelette für weiterführende Aktivitäten bereitstellt. Durch die Verwendung spezialisierter Modell-Agenten und der formales Definition des Eingabemodells für das Framework können diese Ansätze jedoch weiterhin verwendet werden.

Durch ihre Einsatzmöglichkeiten in verteilen, heterogenen IT-Landschaften und durch das proaktive, autonome Verhalten der einzelnen Agenten zeigen Agentensysteme in diesem Ansatz vorteilhafte Eigenschaften. Die Aufteilung des zentralen Systemmodells in ein dezentrales Agentensystem definiert hierbei klare Zuständigkeiten und für jeden Agenten den verantwortlichen Kontext. Weiterhin ist es möglich, ohne Architekturänderungen neue spezialisierte Tool-Agenten in das Framework einzupflegen, wodurch alle Infrastrukturanforderungen erfüllt. Die parallele Entwicklung des System- und Aktivitätsmodells unterstützt aktiv eine iterative Modellierung des Systems. Der Disposer-Agent ist in der Lage, Änderungen an Modellversionen zu extrahieren und das Agentensystem konsistent mit der Datenbasis zu halten. Diese Funktionalität erfüllt die methodischen Anforderungen.

Mit Hilfe zusätzlicher Anwendungsbeispiele wird der Ansatz in Zukunft weiter evaluiert und die Kommunikation zwischen den Agenten optimiert werden. Zusätzlich befinden sich weitere spezialisierte Agenten für Entwicklungswerzeuge sowie für eine geeignete Aktivitätsmodellierungssprache in der Entwicklung.

Literatur

- [1] Tomizuka, M.: Mechatronics: from the 20th to 21st century. *Control Engineering Practice* **10**(8), 877–886 (2002)
- [2] Nahm, Y.E., Ishikawa, H.: Integrated Product and Process Modeling for Collaborative Design Environment. *Concurrent Engineering: R&A* **12**(1), 5–23 (2004)
- [3] OMG: OMG Systems Modeling Language (OMG SysML) 1.2 (2010). <http://www.omg.org>
- [4] Frank, U.: Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme. Ph.D. thesis. Universität Paderborn, Heinz Nixdorf Institut, Rechnerintegrierte Produktion (2006)
- [5] VDI, V.D.I.: Entwicklungsmethodik für mechatronische Systeme. VDI Richtlinie, Bd. 2206. Beuth Verlag, Berlin (2004)
- [6] Unland, R., Klusch, M., Calisti, M. (Hrsg.): Software Agent-Based Applications, Platforms and Development Kits. Birkhäuser, Basel (2005)
- [7] Lindemann, U.: Methodische Entwicklung technischer Produkte: Methoden flexibel und situationsgerecht anwenden. Springer, Berlin Heidelberg (2009)
- [8] Rahman, R.B.A., Pulm, U., Stetter, R.: Systematic Mechatronic Design of a Piezo-Electric Brake. 16th International Conference on Engineering Design, ICED'07. Paris, France (2007)
- [9] Ziemiak, P., Stania, M., Stetter, R.: Mechatronics Engineering on the Example of an Innovative Production Vehicle. 17th International Conference on Engineering Design, ICED, Bd. 1, S. 61–72. Stanford, CA, USA (2009)
- [10] Hehenberger, P., Egyed, A., Zeman, K.: Understanding the Relationship of Information in Mechatronic Design Modeling. Eurocast 13th International Conference on Computer Aided Systems Theory. Las Palmas, Spain (2011)
- [11] OMG: Software & Systems Process Engineering Meta-Model Specification 1.2 (2008). <http://www.omg.org/spec/SPEM/2.0>
- [12] Jennings, N.R., Faratin, P., Norman, T.J., O'Brien, P., Odgers, B.: Autonomous Agents for Business Process Management. *Appl. Art. Intell.* **14**(2), 145–189 (2000)
- [13] Madhusudan, T.: An agent-based approach for coordinating product design workflows. *Comp. Ind.* **56**(3), 235–259 (2005)
- [14] Braubach, L., Pokahr, A., Jander, K., Lamersdorf, W., Burmeister, B.: Go4Flex: Goal-oriented Process Modeling. In: M.E. (Hrsg.) Proc. 4th International Symposium on Intelligent Distributed Computing (IDC-2010). Intelligent Distributed Computing IV, SCI, Bd. 315, S. 77–87. Springer-Verlag, Berlin Heidelberg (2010)
- [15] Friedenthal, S., Moore, A., Steiner, R.: A Practical Guide to SysML: The Systems Modeling Language. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2008)
- [16] Weilkiens, T.: Systems Engineering with SysML/UML: Modeling, Analysis, Design. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2008)
- [17] Cao, Y., Liu, Y., Paredis, C.J.: System-level model integration of design and simulation for mechatronic systems based on SysML. *Mechatronics* **21**(6), 1063–1075 (2011)
- [18] Shah, A., Kerzhner, A., Schaefer, D., Paredis, C.: Multi-view modeling to support embedded systems engineering in SysML. In: Engels, G., Lewerentz, C., Schäfer, W., Schürr, A., Westfechtel, B. (Hrsg.) Graph Transformations and Model-Driven Engineering. Lecture Notes in Computer Science, Bd. 5765, S. 580–601. Springer, Berlin (2010)
- [19] Qamar, A., Wikander, J., During, C.: Designing Mechatronic Systems: A Model-Integration Approach. 18th International Conference on Engineering Design, ICED11. Copenhagen, Denmark (2011)

- [20] Fricke, E., Negele, H., Schrepfer, L., Dick, A., Gebhard, B., Haertlein, N.: Modeling of Concurrent Engineering Processes for Integrated Systems Development. Digital Avionics Systems Conference. Bellevue, WA, USA (1998)
- [21] Schwarz, J., Adameck, N., Frank, D., Siebert, R., Haasis, S.: The use of feature-based workflow techniques in automotive product development. 7th International Conference on Concurrent Enterprising. Bremen, Germany (2001)
- [22] Roelofsen, J., Lauer, W., Lindemann, U.: Product model driven development. 14th European Concurrent Engineering Conference. Ghent (2007)
- [23] Stetter, R., Voos, H.: AGENTES – Agent-based Engineering of Mechatronic Products. Proceedings of the 8th International Symposium on Tools and Methods of Competitive Engineering (TMCS) 2010. Ancona, Italy (2010)
- [24] Thramboulidis, K.: Challenges in the development of mechatronic systems: the mechatronic component. ETFA, S. 624–631 (2008)
- [25] Braubach, L., Pokahr, A.: Jadex – BDI Agent System (2012). <http://jadex-agents.informatik.uni-hamburg.de>
- [26] Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework, 2. Aufl. Addison-Wesley, Boston, MA (2009)
- [27] Seemüller, H., Voos, H.: Model-based Product and Process Integration for Enhanced Collaboration during Mechatronic Design Processes. 2nd Workshop on Process-based Approaches for Model-Driven Engineering. Lyngby, Denmark (2012)
- [28] Stetter, R., Seemüller, H., Chami, M., Voos, H.: Interdisciplinary system model for agent-supported mechatronic design. 18th International Conference on Engineering Design, ICED11. Copenhagen, Denmark (2011)

Kapitel 16

Agentenbasierte dynamische Testfallpriorisierung

Christoph Malz

Zusammenfassung Die begrenzte Testzeit und die begrenzte Anzahl an Testressourcen sind eine sehr große Herausforderung, insbesondere beim Systemtest von Automatisierungssoftware. Der Testmanager ist gezwungen, mit hohem Aufwand von Testzyklus zu Testzyklus bzw. gegebenenfalls während eines Testzyklus eine Priorisierung der zur Verfügung stehenden Testfälle durchzuführen. In diesem Beitrag wird ein Konzept vorgestellt, das mithilfe von Softwareagenten diesen manuellen Aufwand reduziert. Softwareagenten werten die im Systemtest zur Verfügung stehenden Informationen aus der Systementwicklung, dem Systemtest und dem Systembetrieb aktiv aus und schlagen konstruktiv eine Testfallpriorisierung vor, die den Fehlerentdeckungsgrad und die Testeffektivität optimiert.

16.1 Einleitung

Software hat in den letzten Jahren in der Automatisierungstechnik eine enorme Verbreitung gefunden. Es gibt kaum noch Geräte, Maschinen oder Anlagen, in denen die Steuerung nicht über Software realisiert wird. Beispielsweise werden im Automobil, vom Motormanagement über die Getriebesteuerung bis zum Bremsassistenten, immer mehr Steuerungsfunktionen von Mikroprozessoren übernommen und durch entsprechende Software umgesetzt. Software trägt somit entscheidend zum Funktionieren der Geräte und Anlagen bei. Neben unbestreitbaren Vorteilen für die Automatisierungstechnik bringt die Verwendung softwareintensiver automatisierter Systeme auch Risiken mit sich. Der hohe betriebswirtschaftliche Schaden, welchen ein Ausfall oder eine Fehlfunktion eines automatisierten Systems verursachen kann, erfordert qualitativ hochwertige Automatisierungssoftware, um die Verlässlichkeit

Christoph Malz (✉)

Institut für Automatisierungs- und Softwaretechnik, Universität Stuttgart, Pfaffenwaldring 47,
70569 Stuttgart, Deutschland
e-mail: christoph.malz@ias.uni-stuttgart.de

des automatisierten Systems nicht durch fehlerhafte Software zu beeinträchtigen. Die Qualität der Software ist somit zum entscheidenden Faktor für den Erfolg von Produkten und Unternehmen geworden. Viele Unternehmen streben eine verbesserte Qualität ihrer Softwaresysteme und ihres Softwareentwicklungsprozesses an. Ein Mittel, dies zu erreichen, ist das systematische Testen der entwickelten Software. Die zunehmende Komplexität von Automatisierungssoftware führt jedoch zu einem wachsenden Testaufwand, welcher mittlerweile einen beträchtlichen Anteil der Gesamtentwicklungskosten automatisierter Systeme ausmacht und im Spannungsverhältnis zu zunehmend kürzeren Produktentwicklungszyklen steht [1]. Testmanager stehen insbesondere beim Systemtest, bei dem das gesamte integrierte System getestet wird, vor dem Dilemma, dass sich die vorhandenen Testfälle schnell zu einigen hundert oder tausend Testfällen, die durchzuführen sind, addieren. Oft gibt es aber nur ein kleines Zeitfenster kurz vor der Auslieferung für die Durchführung der Testfälle des Systemtests. Des Weiteren erfordert der Systemtest von Automatisierungsssoftware die Überprüfung der dynamischen Wechselwirkungen mit dem technischen System. Die entsprechende Testumgebung, d. h. unterschiedliche Hardware, muss als Testressourcen zur Verfügung stehen. Die zur Verfügung stehenden Testressourcen sind jedoch begrenzt [2]. Der Testmanager hat somit in der Regel zu wenig Zeit und zu wenig Testressourcen, um eine hohe Anzahl an Testfällen durchführen zu können. In dieser Situation wird vom Testmanager dennoch erwartet, dass er verlässliche Resultate ab liefert und sicherstellt, dass die Software ausreichend getestet wird. Da ein vollständiger Test nicht möglich ist, ist ganz entscheidend, wie die beschränkte Testzeit eingesetzt wird. Um unter diesen Voraussetzungen zu einem zufriedenstellenden Ergebnis zu kommen, müssen die Testfälle priorisiert werden. Nur so ist es mit angemessenem Aufwand möglich viele Fehler und insbesondere schwerwiegende Fehler nachzuweisen und unnötige Testfälle, die zu keinen neuen Erkenntnissen führen, zu vermeiden.

16.2 Aktuelle Situation und Problematik bei der Priorisierung von Testfällen

Aufgrund der Größe und Komplexität heutiger Softwaresysteme ist die Priorisierung von Testfällen eine sehr komplexe und zeitaufwendige Aufgabe. Der Testmanager muss Informationen über hunderte Softwaremodule oder Teilsysteme des Softwaresystems sowie tausende Testfälle auswerten, um eine begründete Priorisierung durchzuführen. Aufgrund dieser Priorisierung müssen die beschränkten Testressourcen den Testfällen zugeordnet werden. Eine weitere Erschwernis bringen dynamische Ereignisse mit sich, wie die Änderungen des zu testenden Systems, die einen erneuten Test in einem neuen Testzyklus bedingen. Dabei besteht die Schwierigkeit bei komplexen Systemen darin, bei einer Änderung an einer Stelle zu erkennen, an welchen anderen Stellen sich diese Änderung auswirkt und diese Auswirkungen dann entsprechend beim Test zu berücksichtigen [3]. Für den neuen Testzyklus bzw. gegebenenfalls auch während eines Testzyklus müssen die Testfälle

neu priorisiert werden, unter Berücksichtigung der Softwareänderungen sowie weiterer dynamischer Ereignisse und Informationen wie den aktuellen Testergebnissen. Es sind somit umfangreiche Koordinationsprozesse notwendig, um die Testfallpriorisierung für einzelne Testzyklen durchzuführen. Heute eingesetzte Testmanagementwerkzeuge, die eine Testfallpriorisierung unterstützen sollen, basieren auf Datenbanken und bieten Mechanismen zur Erfassung und Verwaltung von Testfällen [4]. Sie überwachen den Status von Testfällen und werten aus, ob, wann, wie oft und mit welchem Resultat ein Testfall ausgeführt wurde. Der Testmanager muss diese Informationen mithilfe seines Erfahrungswissens auswerten, um die Testfälle für einen Testzyklus zu priorisieren. Die heutigen Testmanagementwerkzeuge bieten somit keine konstruktive Unterstützung der Testfallpriorisierung und keine aktive und kontinuierliche Anpassung der Testfallpriorisierung als Reaktion auf dynamische Ereignisse und Informationen. Der Testmanager muss mit hohem Aufwand Testfälle manuell priorisieren. Aufgrund des hohen Zeitdrucks wird deshalb die Testfallpriorisierung oft intuitiv bestimmt, sodass wichtige Testfälle unberücksichtigt bleiben oder es werden alle Testfälle wiederholt, sodass unnötige Testfälle ausgeführt werden und Auslieferungsdeadlines nicht eingehalten werden können. Dies hat zur Folge, dass der Test wenig effektiv ist. Dies bedeutet, dass pro ausgeführten Testfall wenig Fehler und insbesondere wenig schwerwiegende Fehler gefunden werden.

Die Problematik des hohen manuellen Aufwands beim Priorisieren von Testfällen in den Testzyklen des Systemtests, um möglichst viele und insbesondere schwerwiegende Fehler zu finden, bildet den Ausgangspunkt dieses Beitrags. Der in diesem Beitrag beschriebene agentenbasierte Ansatz zur dynamischen Testfallpriorisierung soll eine geeignete Unterstützung für den Testmanager bei der Priorisierung von Testfällen bereitstellen. Er soll den manuellen Aufwand bei der Testfallpriorisierung in aufeinander folgenden Testzyklen reduzieren und den Fehlerentdeckungsgrad und die Effektivität des Tests steigern. Es wird ein Ansatz aufgezeigt, der Testfälle für Testzyklen des Systemtests rechnergestützt priorisiert. Das Ziel dabei ist eine Testfallreihenfolge zu finden, mit der in der zur Verfügung stehenden Testzeit möglichst viele und insbesondere schwerwiegende Fehler gefunden werden. Die Grundidee der rechnergestützten dynamischen Priorisierung mit dem hier genannten Ziel wird im nächsten Abschnitt vorgestellt.

16.3 Grundidee der dynamischen Testfallpriorisierung

Die Grundidee, um das oben genannte Ziel zu erreichen, ist die Erstellung und Umsetzung eines Konzepts zur dynamischen Testfallpriorisierung, das in der Systementwicklung, im Systemtest und im Systembetrieb bereits vorhandene dynamische Informationen zur Testfallpriorisierung aktiv sammelt, diese auswertet und konstruktiv Vorschläge für die Priorisierung vorhandener Testfälle ausgibt (siehe Abb. 16.1). Dabei ist die Testfallpriorisierung dynamisch, d. h., sie wird von Testzyklus zu Testzyklus bzw. gegebenenfalls während eines Testzyklus auf Basis neu-

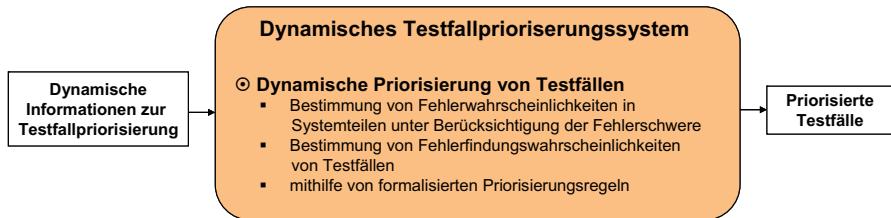


Abb. 16.1 Grundidee der dynamischen Testfallpriorisierung

er Informationen angepasst. Zur Priorisierung der Testfälle bestimmt das dynamische Testfallpriorisierungssystem durch Auswertung der gesammelten Informationen zum einen Fehlerwahrscheinlichkeiten in Systemteilen des zu testenden Systems unter Berücksichtigung der erwarteten Schwere der möglichen Fehler. Zum anderen bestimmt das dynamische Testfallpriorisierungssystem auf Basis der gesammelten Informationen die Fehlerfindungswahrscheinlichkeit der vorhandenen Testfälle in einzelnen Systemteilen. Die Fehlerwahrscheinlichkeiten in Systemteilen sowie die Fehlerfindungswahrscheinlichkeiten der Testfälle werden aus den gesammelten Informationen mithilfe von formalisierten Priorisierungsregeln, die auf dem Erfahrungswissen der Testmanager basieren, abgeleitet. Die im hier vorgestellten Konzept standardmäßig verwendeten Informationen zur Bestimmung der Fehlerwahrscheinlichkeiten, Fehlerfindungswahrscheinlichkeiten und somit zur Priorisierung werden in den nächsten Abschnitten genauer beschrieben. Es handelt sich um Informationen aus der Systementwicklung, dem Systemtest und dem Systembetrieb, die aus theoretischen Untersuchungen und aus Erfahrungswissen von Testmanagern aus der Industrie festgelegt wurden. Diese Informationen sind anpassbar, je nachdem welche Informationen für wichtig erachtet werden, bzw. in einem Projekt vorhanden sind.

16.3.1 *Informationen zur Testfallpriorisierung aus der Systementwicklung*

Das dynamische Testfallpriorisierungssystem sammelt unterschiedliche Informationen aus der Systementwicklung zur Bestimmung der Fehlerwahrscheinlichkeit in Systemteilen des zu testenden Systems und zur Bestimmung der zu erwartenden Fehlerschwere der möglichen Fehler. Um überhaupt einzelne Systemteile des zu testenden Softwaresystems und Seiteneffekte zwischen diesen bewerten zu können, benötigt das Testfallpriorisierungssystem zunächst grundlegende Informationen über das zu testende Softwaresystem. Die im hier vorgestellten Konzept verwendeten grundlegenden Informationen über das zu testende Softwaresystem sind:

- Vorhandene Systemteile bzw. Softwaremodule (SM) des zu testenden Softwaresystems

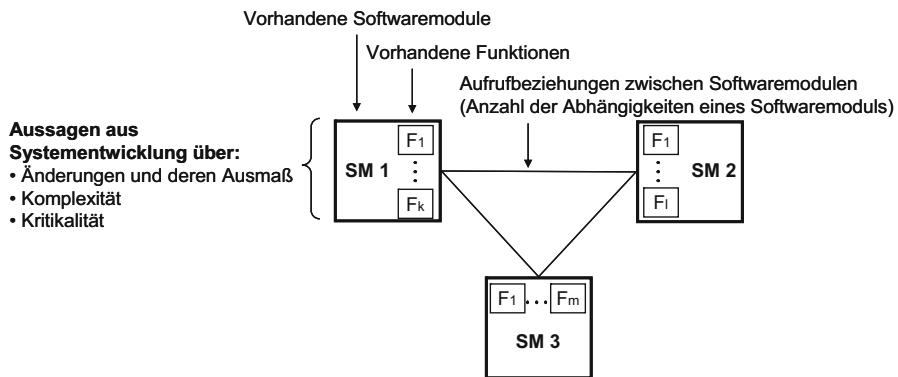


Abb. 16.2 Informationen über das Softwaresystem aus der Systementwicklung

- Funktionen (F) der vorhandenen Softwaremodule
- Aufrufbeziehungen zwischen den Softwaremodulen

Abbildung 16.2 verdeutlicht diese grundlegenden Informationen anhand eines Beispielsystems bestehend aus drei Softwaremodulen. Diese Informationen werden durch das dynamische Testfallpriorisierungssystem aus dem Architekturmodell des zu testenden Softwaresystems bezogen. Dieses Architekturmodell kann dabei in unterschiedlichen Modellierungssprachen vorliegen.

Des Weiteren ergeben sich in der Systementwicklung Informationen bzw. Indikatoren, mit denen man die Fehlerwahrscheinlichkeit in den einzelnen Softwaremodulen bestimmen kann. Folgende dieser Informationen werden im hier vorgestellten Konzept verwendet:

- Geänderte Funktionen bzw. die Anzahl der Änderungen in einem Softwaremodul nach dem letzten Testzyklus
- Komplexität eines Softwaremoduls

Je höher das Ausmaß der Änderungen in einem Softwaremodul bzw. je komplexer ein Softwaremodul ist, desto höher ist die Wahrscheinlichkeit für Fehler in diesem Softwaremodul.

Darüber hinaus besitzt die Systementwicklung auch Informationen bzw. Indikatoren, mit denen die erwartete Schwere möglicher Fehler bestimmt werden kann. Folgende dieser Informationen werden im hier vorgestellten Konzept verwendet:

- Kritikalität eines Softwaremoduls
- Anzahl der vom betrachteten Softwaremodul abhängigen Softwaremodule

Je höher die Kritikalität eines Softwaremoduls ist bzw. je mehr abhängige Softwaremodule ein Softwaremodul hat, desto höher ist die Wahrscheinlichkeit, dass die in diesem Softwaremodul gefundenen Fehler schwerwiegender sind.

Die Informationen aus der Systementwicklung zur Bestimmung der Fehlerwahrscheinlichkeiten werden auch in Abb. 16.2 dargestellt.

Die Informationen über Änderungen der Softwaremodule werden aus Konfigurations- bzw. Änderungsmanagementwerkzeugen, die heute bei der Entwicklung eingesetzt werden, durch das Testfallpriorisierungssystem bezogen. Informationen über die Komplexität und Kritikalität einzelner Softwaremodule werden durch die Entwickler im Architekturmodell und in speziellen Datenbanken abgelegt und werden von dort durch das Testfallpriorisierungssystem eingelesen. Die Informationen über die Anzahl der Abhängigkeiten zu anderen Softwaremodulen werden direkt aus dem Architekturmodell gewonnen.

16.3.2 Informationen zur Testfallpriorisierung aus dem Systemtest und dem Systembetrieb

Das dynamische Testfallpriorisierungssystem sammelt unterschiedliche Informationen aus dem Systemtest und dem Systembetrieb für die Bestimmung der Fehlerwahrscheinlichkeiten in Softwaremodulen, zur Bestimmung der zu erwartenden Fehlerschwere der möglichen Fehler und zur Beurteilung der Fehlerfindungswahrscheinlichkeiten der Testfälle in einzelnen Softwaremodulen. Um einzelne Testfälle und deren Fehlerfindungswahrscheinlichkeit bewerten zu können, benötigt das Testfallpriorisierungssystem zunächst grundlegende Informationen über die Testfälle. Die im hier vorgestellten Konzept verwendeten grundlegenden Informationen über die Testfälle sind:

- Vorhandene Testfälle des Systemtests
- Durch einen Testfall aufgerufene Softwaremodule

Abbildung 16.3 verdeutlicht diese grundlegenden Informationen anhand des Beispielsystems. Diese Informationen werden durch das Testfallpriorisierungssystem aus den im Systemtest eingesetzten Testmanagementwerkzeugen bezogen.

Des Weiteren ergeben sich im Systemtest oder Systembetrieb auch Informationen bzw. Indikatoren, mit denen man die Fehlerwahrscheinlichkeit in den einzelnen Softwaremodulen bestimmen kann. Folgende Informationen werden im hier vorgestellten Konzept verwendet:

- Gefundene Fehler in den einzelnen Softwaremodulen in vorherigen Testzyklen
- Gefundene Fehler in den einzelnen Softwaremodulen im Betrieb des Softwaresystems

Wenn in einem Softwaremodul in vorherigen Testzyklen oder im Betrieb Fehler gefunden wurden, dann ist die Wahrscheinlichkeit hoch, dass weitere Fehler in diesem Softwaremodul existieren bzw. dass eventuell die gefundenen Fehler falsch korrigiert wurden.

Darüber hinaus gewinnt man auch Informationen bzw. Indikatoren aus dem Systemtest und dem Systembetrieb, mit denen die erwartete Schwere möglicher Fehler bestimmt werden kann. Folgende solche Informationen werden im hier vorgestellten Konzept verwendet:

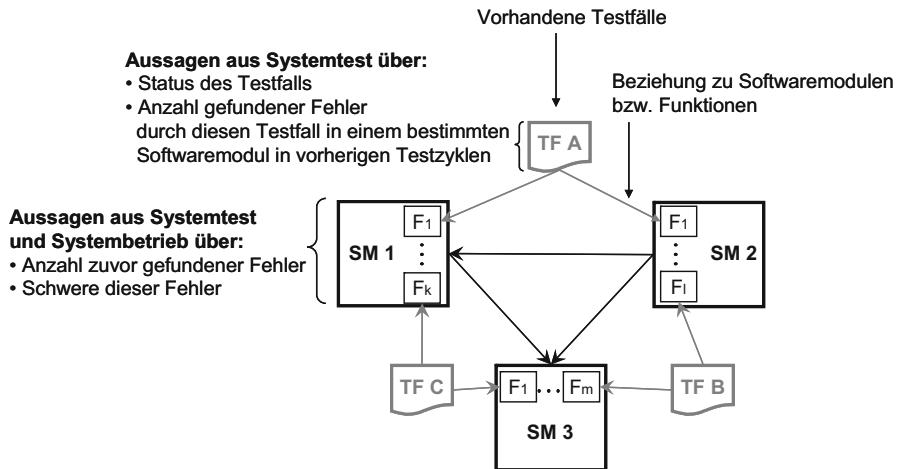


Abb. 16.3 Informationen aus dem Systemtest und dem Systembetrieb

- Schwere der in vorherigen Testzyklen gefundenen Fehler in den einzelnen Softwaremodulen
- Schwere der im Systembetrieb gefundenen Fehler in den einzelnen Softwaremodulen

Wenn die in einem Softwaremodul gefundenen Fehler zuvor als schwerwiegend eingestuft wurden, dann ist die Wahrscheinlichkeit hoch, dass auch neu gefundene Fehler in diesen Softwaremodulen als schwerwiegend eingestuft werden.

Des Weiteren können aus dem Systemtest Informationen zur Bestimmung der Fehlerfindungswahrscheinlichkeit eines Testfalls in einem Softwaremodul bezogen werden. Folgende Informationen werden im hier vorgestellten Konzept verwendet:

- Status des Testfalls, d. h. ist dieser im vorherigen Testzyklus erfolgreich gelaufen oder nicht?
- Anzahl der gefundenen Fehler durch einen Testfall in einem Softwaremodul in vorherigen Testzyklen
- Aufgerufene Funktionen durch den Testfall im Softwaremodul

Testfälle, die im vorherigen Testzyklus nicht erfolgreich gelaufen sind, d. h. Fehler gefunden haben, können diese Fehler weiterhin finden, wenn diese falsch korrigiert wurden. Des Weiteren, wenn ein Testfall in vorherigen Testzyklen in einem Softwaremodul oft Fehler gefunden hat, dann ist die Wahrscheinlichkeit hoch, dass dieser Testfall in diesem Softwaremodul wieder Fehler findet, wenn dieses Softwaremodul geändert wird. Darüber hinaus, wenn ein Testfall geänderte Funktionen in einem Softwaremodul aufruft, hat der Testfall eine höhere Wahrscheinlichkeit Fehler in diesem Softwaremodul zu finden, als ein Testfall, der nicht geänderte Funktionen aufruft.

Die Informationen zur Bestimmung der Fehlerwahrscheinlichkeiten und der Fehlerfindungswahrscheinlichkeiten werden in Abb. 16.3 dargestellt.

Die Informationen über die gefundenen Fehler, deren Schwere und die Testfälle, die diese Fehler gefunden haben sowie über den Status von Testfällen und die durch Testfälle aufgerufenen Funktionen werden aus den im Systemtest eingesetzten Fehlermanagement- und Testmanagementwerkzeugen durch das Testfallpriorisierungssystem bezogen.

16.3.3 Agentenbasierter Ansatz für das dynamische Testfallpriorisierungssystem

Zur Umsetzung des dynamischen Testfallpriorisierungssystems, das die oben genannten dynamischen Informationen von Testzyklus zu Testzyklus bzw. gegebenenfalls während eines Testzyklus auswertet und eine Priorisierung der Testfälle ausgibt, wurde ein agentenbasierter Ansatz gewählt. Merkmal von agentenbasierten Systemen [5–7] ist, dass bei ihnen die Problemstellung in einzelne Softwareagenten abstrahiert wird. Die Softwareagenten können ihre Umgebung wahrnehmen und beeinflussen. Sie können wahrgenommene Informationen hinsichtlich vorgegebener Ziele evaluieren. Durch die Aufteilung von Entscheidungsprozessen auf einzelne Softwareagenten können dynamische Ereignisse einfacher gehandhabt werden. Sie treffen Entscheidungen an Stellen, wo die meisten Informationen über ein Problem liegen bzw. dynamisch durch Interaktion entstehen. Softwareagenten können somit durch Vorgabe von Zielen und Regeln und durch Interaktion mit der Umgebung und anderen Softwareagenten konstruktive Vorschläge für eine Problemstellung erarbeiten. Des Weiteren können sie dynamische Ereignisse und Informationen aktiv wahrnehmen und auf diese durch lokale Aktionen und Interaktionen mit anderen Softwareagenten reagieren. Aufgrund dieser Eigenschaften von Agentensystemen wurde ein agentenbasierter Ansatz für das dynamische Testfallpriorisierungssystem gewählt, mit der grundlegenden Idee, dass die Priorisierung der Testfälle aus der Perspektive der Entitäten des Problemgebiets durchgeführt wird. Dabei werden die Entitäten des Problemgebiets durch Softwareagenten vertreten. Die Entitäten der Testfallpriorisierung sind die Softwaremodule, für die eine Fehlerwahrscheinlichkeit bestimmt werden muss, und sind die Testfälle, für die eine Fehlerfindungswahrscheinlichkeit bestimmt werden muss. Somit wird jedes Softwaremodul und jeder Testfall durch einen Softwareagenten vertreten. Wie die Softwareagenten die Fehlerwahrscheinlichkeiten bestimmen und zusammen eine Priorisierung der Testfälle festlegen, wird im nächsten Abschnitt beschrieben.

16.4 Agentenbasierte Testfallpriorisierung

Für jedes Softwaremodul des zu testenden Systems wird im dynamischen Testfallpriorisierungssystem ein Softwaremodul-Agent und für jeden vorhandenen Testfall ein Testfall-Agent erzeugt. Das globale Ziel der Agenten ist, eine Priorisierung der Testfälle zu finden, die es ermöglicht, in der zur Verfügung stehenden Testzeit, möglichst viele und insbesondere schwerwiegende Fehler zu finden. Jeder Agent trägt zu diesem Ziel bei, indem er aus der Perspektive seiner Entität relevante Werte bestimmt und in Kooperation mit den anderen Agenten die endgültige Priorität festlegt [8–10]. Dies wird in den nächsten Abschnitten näher beschrieben.

16.4.1 Bestimmung der Testwichtigkeit eines Softwaremoduls durch den Softwaremodul-Agent

Der Softwaremodul-Agent vertritt ein Softwaremodul und beantwortet aus der Perspektive des Softwaremoduls folgende Frage: Wie wahrscheinlich ist es, dass man durch Ausführung der Funktionen meines Softwaremoduls Fehler findet und wie schwerwiegend können diese Fehler sein? Er beantwortet diese Fragen, indem er für sein Softwaremodul einen Wert bestimmt, der als Testwichtigkeit (TW) bezeichnet wird. In Abb. 16.4 wird dies für das Softwaremodul SM 1 des Beispielsystems dargestellt, für das die Testwichtigkeit $TW(SM 1)$ bestimmt wird.

Je höher die Testwichtigkeit ist, desto höher muss die Testintensität für das Softwaremodul im nächsten Testzyklus sein. Um die Testwichtigkeit zu bewerten, benutzt der Softwaremodul-Agent verschiedene der in Abschn. 16.3 beschriebenen Informationen bzw. Indikatoren aus der Systementwicklung, dem Systemtest und dem Systembetrieb.

Aus der Systementwicklung werden folgende Informationen verwendet: Anzahl der Änderungen im Softwaremodul, die Komplexität des Softwaremoduls, die

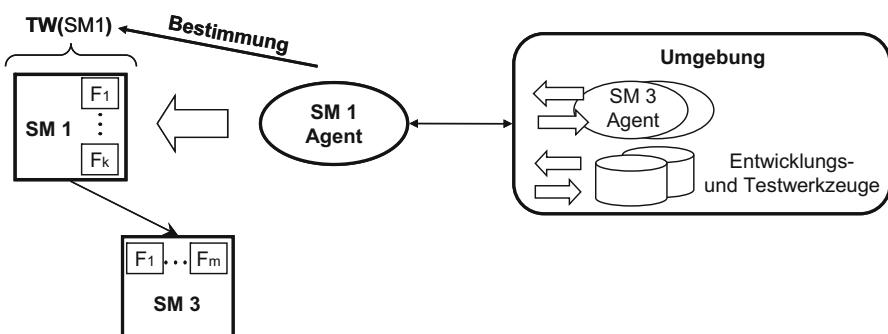


Abb. 16.4 Bestimmung der Testwichtigkeit durch den Softwaremodul-Agent

Kritikalität des Softwaremoduls sowie die Anzahl der vom betrachteten Softwaremodul abhängigen Softwaremodule. Des Weiteren werden auch Informationen über Änderungen in abhängigen Softwaremodulen verwendet, da aufgrund von Seiteneffekten die Fehlerwirkungen dieser Änderungen durch Aufrufe der Funktionen des betrachteten Softwaremoduls gefunden werden können. Im Beispiel in Abb. 16.4 werden somit mögliche Änderungen in Softwaremodul SM 3 in die Bestimmung der Testwichtigkeit von Softwaremodul SM 1 einbezogen. Der Softwaremodul-Agent von Softwaremodul SM 3 informiert den Softwaremodul-Agenten von Softwaremodul SM 1 über mögliche Änderungen. Der Austausch der Informationen der Änderungen der Softwaremodule zwischen den Agenten ermöglicht somit die Berücksichtigung der Seiteneffekte von Änderungen. Alle Informationen aus der Systementwicklung über sein Softwaremodul holt sich ein Softwaremodul-Agent aus den eingesetzten Entwicklungswerkzeugen.

Aus dem Systemtest und dem Systembetrieb verwendet der Softwaremodul-Agent folgende Informationen zur Bestimmung der Testwichtigkeit: Anzahl der gefundenen Fehler im Softwaremodul in vorherigen Testzyklen und im Systembetrieb und die Schwere dieser gefundenen Fehler. Diese Informationen holt sich der Softwaremodul-Agent aus den im Systemtest eingesetzten Testwerkzeugen.

Um aus diesen Informationen die Testwichtigkeit zu bestimmen, verwendet der Softwaremodul-Agent Priorisierungsregeln aufgrund von Erfahrungswissen. Da die zur Bestimmung der Testwichtigkeit verwendeten Informationen nur Indikatoren für die Testwichtigkeit darstellen, ist die aus diesen Informationen abgeleitete Bewertung der Testwichtigkeit immer mit Unsicherheiten behaftet. Des Weiteren sind die Priorisierungsregeln als verbal formulierte Regeln verfügbar. Sie lauten zum Beispiel: „Wo Änderungen durchgeführt wurden oder Seiteneffekte haben, steigt die Wahrscheinlichkeit für Fehler“. Aufgrund dieser Merkmale wurde entschieden, für das Formalisieren des Regelwerks zu Bestimmung der Testwichtigkeit Fuzzy-Logik zu benutzen. Fuzzy-Regelwerke bieten eine gute Möglichkeit, unscharfe menschliche Bewertungsmaßstäbe, Denkmuster und Schlussfolgerungsabläufe nachzubilden [11, 12]. Somit greift der Softwaremodul-Agent zur Bestimmung der Testwichtigkeit auf ein Fuzzy-Regelwerk zu (siehe Abb. 16.5).

Die diskreten Werte der oben genannten Informationen bzw. Indikatoren sind die Eingangswerte des Fuzzy-Regelwerks. Am Ausgang erhält man einen diskreten Wert für die Testwichtigkeit. Ein beispielhafter Regelwerksauszug sieht folgendermaßen aus:

- **WENN** Anzahl Änderungen ist hoch **DANN** Testwichtigkeit ist sehr hoch
- **WENN** Anzahl Änderungen ist mittel **DANN** Testwichtigkeit ist hoch
- **WENN** Anzahl Änderungen ist niedrig **DANN** Testwichtigkeit ist mittel
- **WENN** Anzahl zuvor gefundener Fehler ist hoch **DANN** Testwichtigkeit ist sehr hoch
- Usw.

Die Regeln können je nach Bedarf und verwendeten Informationen verändert und unterschiedlich gewichtet werden.

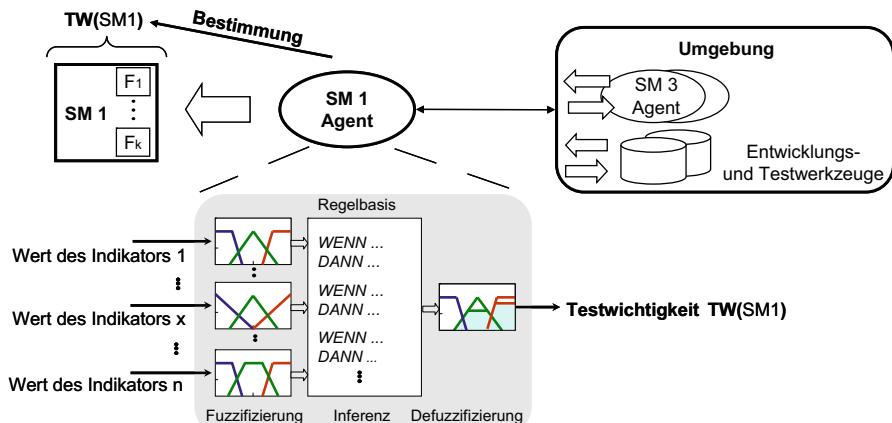


Abb. 16.5 Fuzzy-Regelwerk des Softwaremodul-Agenten

16.4.2 Bestimmung der lokalen Prioritäten und der globalen Priorität eines Testfalls durch den Testfall-Agent

Der Testfall-Agent vertritt einen Testfall und beantwortet aus der Perspektive des Testfalls zunächst folgende Frage: Wie wahrscheinlich ist es, dass mein Testfall durch Aufruf von Funktionen eines bestimmten Softwaremoduls Fehler finden kann und wie schwerwiegend können diese Fehler sein? Er beantwortet diese Fragen, indem er bezüglich jedes Softwaremoduls, bei dem sein Testfall Funktionen ausführt, einen Wert bestimmt, der als lokale Priorität (LP) bezeichnet wird. In Abb. 16.6 wird dies für den Testfall TF B des Beispielsystems dargestellt, für den die lokale Priorität LP(TF B, SM 2) bezüglich des Softwaremoduls SM 2 und die lokale Priorität LP(TF B, SM 3) bezüglich des Softwaremoduls SM 3 bestimmt wird. Je höher die lokale Priorität eines Testfalls bezüglich eines Softwaremoduls, desto höher die Wahrscheinlichkeit, dass der Testfall durch Aufruf der Funktionen dieses Softwaremoduls Fehler, insbesondere schwerwiegende Fehler, findet. Um die lokale Priorität zu bewerten, benutzt der Testfall-Agent bestimmte der oben dargestellten Informationen bzw. Indikatoren aus der Systementwicklung und dem Systemtest.

Aus der Systementwicklung werden die Informationen über die geänderten Funktionen des Softwaremoduls verwendet. Die Informationen über Änderungen der Softwaremodule bekommt der Testfall-Agent von den Softwaremodul-Agenten.

Aus dem Systemtest werden folgende Informationen verwendet: Anzahl und Schwere der gefundenen Fehler durch den Testfall in dem Softwaremodul in vorherigen Testzyklen sowie die durch den Testfall aufgerufenen Funktionen eines Softwaremoduls. Diese Informationen holt sich der Testfall-Agent aus den eingesetzten Testwerkzeugen.

Um aus diesen Informationen die lokale Priorität bezüglich der einzelnen Softwaremodule zu bestimmen, verwendet der Testfall-Agent auch Priorisierungsregeln

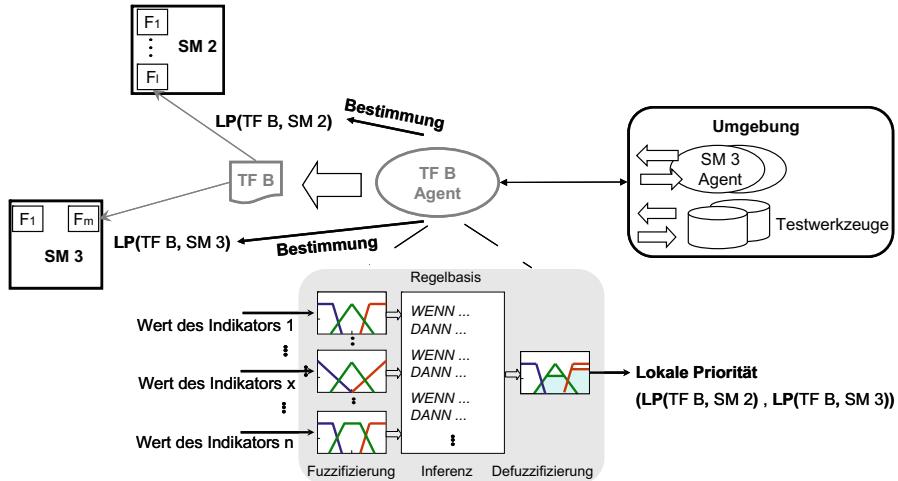


Abb. 16.6 Bestimmung von lokalen Prioritäten durch den Testfall-Agenten

basierend auf Erfahrungswissen. Da die Regeln die gleichen Merkmale besitzen wie die zur Bestimmung der Testwichtigkeit, wird auch hier ein Fuzzy-Regelwerk verwendet, wie in Abb. 16.6 dargestellt.

Ein beispielhafter Regelwerksauszug sieht folgendermaßen aus:

- **WENN** Anzahl gefundenen Fehler durch den Testfall in vorherigen Testzyklen ist hoch **DANN** Lokale Priorität ist sehr hoch
- **WENN** Anzahl gefundenen Fehler durch den Testfall in vorherigen Testzyklen ist mittel **DANN** Lokale Priorität ist hoch
- **WENN** Anzahl gefundenen Fehler durch den Testfall in vorherigen Testzyklen ist niedrig **DANN** Lokale Priorität ist mittel
- **WENN** Anzahl der Änderungen in durch den Testfall aufgerufenen Funktionen ist hoch **DANN** Lokale Priorität ist sehr hoch
- Usw.

Die Regeln können auch hier je nach Bedarf und verwendeten Informationen verändert und unterschiedlich gewichtet werden.

Die Testfall-Agenten bestimmen auf diese Art und Weise die lokalen Prioritäten bezüglich der Softwaremodule und die Softwaremodul-Agenten die Testwichtigkeiten der Softwaremodule. Nun müssen aber die Testfall-Agenten noch folgende Frage beantworten: Wie wahrscheinlich ist es, dass mein Testfall im gesamten Softwaresystem Fehler findet und wie schwerwiegend können diese Fehler sein? Er beantwortet diese Fragen, indem er für seinen Testfall einen Wert bestimmt, der als globale Priorität (GP) bezeichnet wird. In Abb. 16.7 wird dies für den Testfall TF B dargestellt, für den die globale Priorität GP (TF B) bestimmt wird.

Die globale Priorität eines Testfalls widerspiegelt die Wahrscheinlichkeit für die Aufdeckung eines oder mehrerer Fehler im Softwaresystem durch die Ausfüh-

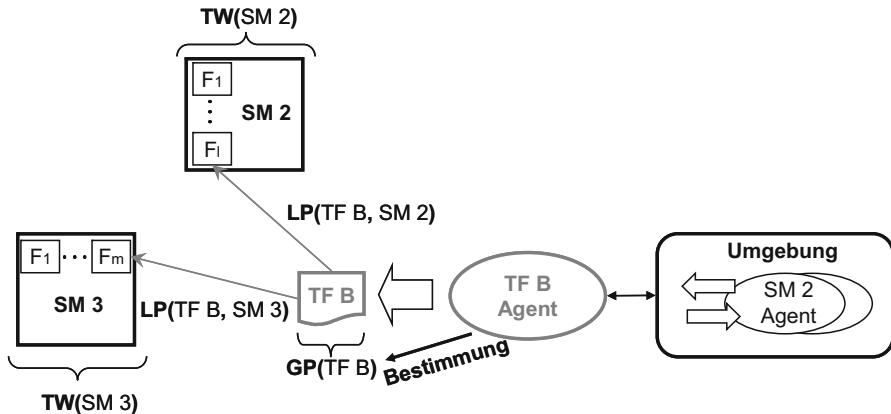


Abb. 16.7 Bestimmung der globalen Priorität durch den Testfall-Agenten

rung dieses Testfalls unter Berücksichtigung der Schwere der möglichen Fehler. Je höher die globale Priorität, desto wahrscheinlicher ist es, dass mehr Fehler und insbesondere schwerwiegende Fehler gefunden werden. Um die globale Priorität zu bewerten, benutzt der Testfall-Agent die von ihm bestimmten lokalen Prioritäten sowie die Testwichtigkeiten der Softwaremodule, die er von den Softwaremodul-Agenten bekommt. Die globale Priorität ist dabei als gewichteter Mittelwert der lokalen Prioritäten definiert, wobei die Gewichte die Testwichtigkeiten sind. Zur Skalierung wird der gewichtete Mittelwert durch die Summe der Testwichtigkeiten geteilt. Die Formel für die globale Priorität, die durch den Testfall-Agenten verwendet wird, ist somit Folgende:

$$GP(TF X) = \frac{\sum_{i=1}^n LP(TFX, SM i) \cdot TW(SM i)}{\sum_{i=1}^n TW(SM i)} \quad (16.1)$$

n Anzahl der Softwaremodule

Für das Beispiel in Abb. 16.7 wird die globale Priorität von Testfall B wie folgt berechnet:

$$GP(TFB) = \frac{(LP(TFB, SM 2) \cdot TW(SM 2)) + (LP(TFB, SM 3) \cdot TW(SM 3))}{TW(SM 1) + TW(SM 2) + TW(SM 3)} \quad (16.2)$$

Nach der Bestimmung der globalen Prioritäten für alle Testfälle durch die Testfall-Agenten wird durch das agentenbasierte dynamische Testfallpriorisierungssystem eine Testfallreihenfolge entsprechend den globalen Prioritäten ausgeben.

16.5 Realisierung und Evaluierung des agentenbasierten dynamischen Testfallpriorisierungssystems

Das agentenbasierte dynamische Testfallpriorisierungssystem wurde basierend auf dem Agentenframework JADE realisiert. Informationen über das zu testende Softwaresystem aus der Systementwicklung bekommt es über ein Architekturmodell, das im XML-Format vorgegeben wird. Die Informationen über Änderungen einzelner Softwaremodule holen sich die Softwareagenten über eine Anbindung zum Konfigurationsmanagementwerkzeug Subversion. Informationen aus dem Systemtest werden bezogen durch eine Anbindung an das Fehlermanagementwerkzeug Bugzilla sowie das Testmanagementwerkzeug Testopia. Es können je nach Bedarf aber alle möglichen Werkzeuge angebunden werden, die Zugriffsschnittstellen für die Informationen bieten.

Zur Evaluierung des Systems wurde die Softwaresteuerung eines industriellen Kaffeeautomaten als zu testendes System herangezogen. Die Steuerung besteht aus 29 Softwaremodulen mit 101 Funktionen. Es liegen für die Steuerung 30 Testfälle vor. Alle Informationen über das zu testende System sowie über die Testfälle, wie sie oben beschrieben wurden, sind in den angebundenen Werkzeugen sowie in einem XML-Architekturmodell abgelegt. Zur Evaluierung sollten Testfälle für einen fortgeschrittenen Testzyklus der Steuerung priorisiert werden. Im vorherigen Testzyklus sind 7 Fehler gefunden worden, die in den angebundenen Werkzeugen dokumentiert wurden. Aufgrund dieser Fehler und weiterer Änderungsaufforderungen sind Änderungen in 11 verschiedenen Softwaremodulen durchgeführt worden. Bei der Durchführung der Änderung wurden mit Absicht 8 Fehler mit unterschiedlicher Schwere in das System injiziert. Dabei wurden die Fehler zufällig verteilt, jedoch unter Berücksichtigung der Wahrscheinlichkeit des Auftretens von Fehlern einer bestimmten Schwere an bestimmten Stellen. Die Priorisierung für den neuen Testzyklus wurde anhand des agentenbasierten dynamischen Testfallpriorisierungssystems durchgeführt und durch 6 Entwickler, die die gleichen Informationen zur Verfügung hatten, d. h. alle Werkzeuge und Architekturmodelle verwenden durften, die die Informationen enthalten. Dabei hatten die Entwickler gute Kenntnisse über das zu testende System und die Werkzeuge. Zum Vergleich der Priorisierungen wurde davon ausgegangen, dass die Testzeit nur für die Durchführung der 8 ersten Testfälle der Priorisierungsliste ausreicht. Es wurde dann geprüft, wie viele Fehler durch die ersten 8 Testfälle gefunden werden, und die Priorisierung mit folgenden Werten verglichen:

- Fehlerentdeckungsgrad: Summe aufgedeckter gewichteter Fehler im Test dividiert durch die Summe aller gewichteter Fehler
- Testeffektivität: Summe aufgedeckter gewichteter Fehler im Test dividiert durch die Anzahl ausgeführter Testfälle.

Die Summe aller gewichteten Fehler war 12. Dies waren die 8 injizierten Fehler gewichtet mit dem Schwerefaktor. Es gab 4 normale Fehler mit Gewichtung 1 und 4 kritische Fehler mit Gewichtung 2. Mit der Priorisierung des agentenbasierten

Tab. 16.1 Resultate der Evaluierung

	Testfallpriorisierungssystem	Manuelle Priorisierung durch die Entwickler (Durchschnittswert)
Priorisierungszeit	< 1 s	15 min
Testeffektivität	1,25	0,5
Fehlerentdeckungsgrad	0,83	0,33

dynamischen Testfallpriorisierungssystems war die Summe aufgedeckter gewichteter Fehler gleich 10 und somit der Fehlerentdeckungsgrad gleich 0,83 und die Testeffektivität gleich 1,25. Mit der manuellen Priorisierung durch die Entwickler wurde im Durchschnitt ein Fehlerentdeckungsgrad von 0,33 und eine Testeffektivität von 0,5 erreicht. Das agentenbasierte dynamische Testfallpriorisierungssystem hat die Priorisierung in weniger als 1 Sekunde durchgeführt, die Entwickler hatten 15 Minuten zur Verfügung und mussten die Zeit immer voll ausschöpfen. Die Ergebnisse der Evaluierung sind in Tab. 16.1 zusammengefasst.

Aus dem Resultat ist zu erkennen, dass der Fehlerentdeckungsgrad und die Testeffektivität durch den Einsatz des agentenbasierten dynamischen Testfallpriorisierungssystems deutlich gesteigert und der zeitliche Aufwand zur Priorisierung stark reduziert werden kann. Bei noch größeren Systemen, die nicht nur aus 29 Softwaremodulen bestehen, sind noch weit größere Differenzen zu erwarten.

Eine weitere Evaluierung wird zurzeit mit Unternehmen aus der Automobilindustrie und der Luft- und Raumfahrtindustrie durchgeführt. Es wurden bzw. werden dafür Prototypen realisiert, die Informationen zur Priorisierung verwenden, die in den Unternehmen verfügbar sind. Die Informationen werden durch Anbindung an die verwendeten Werkzeuge und Datenbanken bezogen. Die Evaluierung wird anhand historischer Projektdaten eines abgelaufenen Projekts durchgeführt. Die aktuellen Ergebnisse zeigen, dass durch den Einsatz des agentenbasierten dynamischen Testfallpriorisierungssystems ein effektiverer Test durchgeführt werden kann, als es mit der manuellen Priorisierung durch die Entwickler und Testmanager heutzutage der Fall ist.

16.6 Zusammenfassung

In diesem Beitrag wurde ein Konzept zur agentenbasierten dynamischen Testfallpriorisierung beim Systemtest vorgestellt, bei dem Softwareagenten Softwaremodule und Testfälle vertreten und aus deren Perspektive in Zusammenarbeit die Testfallpriorisierung durchführen. Das Ziel ist dabei, eine Testfallpriorisierung zu erreichen, die es ermöglicht, in der zur Verfügung stehenden Testzeit möglichst viele und insbesondere schwerwiegende Fehler zu finden. Zur Testfallpriorisierung bestimmen die Softwareagenten Fehlerwahrscheinlichkeiten auf Basis von Informationen,

die bereits im Systemtest vorhanden sind, was die Anwendbarkeit im Systemtest sicherstellt. Je nach Bedarf können die Informationen, die zur Auswertung eingesetzt werden, angepasst werden. Die Softwareagenten werten die Informationen mit Priorisierungsregeln aus, die anhand von Erfahrungswissen in Form von Fuzzy-Logik definiert wurden. Auch die Regeln können je nach Bedarf angepasst werden. Das Agentensystem arbeitet aktiv im Hintergrund. Es registriert die Anpassung von Informationen aufgrund dynamischer Ereignisse und reagiert auf diese mit der Anpassung der Testfallpriorisierung. Das agentenbasierte dynamische Testfallpriorisierungssystem wurde realisiert und evaluiert. Die Evaluierungsergebnisse zeigen eine deutliche Steigerung des Fehlerentdeckungsgrads, der Testeffektivität und eine Reduzierung des zeitlichen Aufwands für die Testfallpriorisierung.

Literatur

- [1] Broekman, B., Notenboom, E.: Testing Embedded Software. Addison-Wesley, London (2005)
- [2] Spillner, A., Linz, T.: Basiswissen Softwaretest. dpunkt.verlag, Heidelberg (2005)
- [3] Spillner, A., Roßner, T., Winter, M., Linz, T.: Praxiswissen Softwaretest – Testmanagement. dpunkt.verlag, Heidelberg (2008)
- [4] Illes, T., Pohlmann, H., Roßner, T., Schlatter, A., Winter, M.: ix Studie – Software-Testmanagement. Heise Zeitschriften Verlag, Hannover (2006)
- [5] VDI/VDE Richtlinie 2653: Agenten in der Automatisierungstechnik, Teil I. Beuth Verlag, Berlin, 2010.
- [6] Mubarak, H.: Developing Flexible Software using Agent-Oriented Software Engineering. IEEE Software 25(5), 12–15 (2008)
- [7] Wooldridge, M., Jennings, N.R.: Intelligent Agents: Theory and Practice. Knowledge Engineering Review 10(2), 115–152 (1995)
- [8] Malz, C., Jazdi, N., Göhner, P.: Prioritization of Test Cases Using Software Agents and Fuzzy Logic. Software Testing, Verification and Validation (ICST). 2012 IEEE Fifth International Conference, S. 483–486 (2012)
- [9] Malz, C., Göhner, P.: Agent-Based Test Case Prioritization. Software Testing, Verification and Validation Workshops (ICSTW). 2011 IEEE Fourth International Conference, S. 149–152 (2011)
- [10] Malz, C., Jazdi, N.: Agent-Based Test Management for Software System Test. Automation Quality and Testing Robotics (AQTR). 2010 IEEE International Conference, S. 1–6 (2010)
- [11] Avineri, E., Köppen, M., Dahal, K., Sunitiyoso, Y., Roy, R.: Applications of Soft Computing – Updating the State of the Art. Springer, Heidelberg (2009)
- [12] Lippe, W.: Soft-Computing. Springer-Verlag, Berlin Heidelberg (2006)

Kapitel 17

Werkzeugunterstützung für die Entwicklung von SPS-basierten Softwareagenten zur Erhöhung der Verfügbarkeit

Daniel Schütz und Birgit Vogel-Heuser

Zusammenfassung Dieser Beitrag behandelt das Vorgehen und die Projektergebnisse der DFG-Transferprojekt KREAagentuse. Aufbauend auf den Ergebnissen des DFG-Projekts AVE wurde eine werkzeugunterstützte Vorgehensweise realisiert, die eine modellbasierte Entwicklung und automatische Implementierung SPS-basierter Softwareagenten ermöglicht. Diese Softwareagenten sind in der Lage, die Verfügbarkeit einer Produktionsanlage durch die Implementierung von Softsensoren zu erhöhen.

17.1 Einleitung

An Produktionssysteme werden hohe Anforderungen in Bezug auf ihre Robustheit gegenüber Störungen und Ausfällen gestellt. Eine der häufigsten Fehlerursachen und Gründe für Stillstände von fertigungstechnischen wie auch verfahrenstechnischen Produktionsanlagen sind defekte oder fehlerhaft arbeitende Sensoren. Das Herunterfahren eines Produktionsprozesses und die Durchführung von Wartungsarbeiten (Ersetzen defekter Sensoren) ist in der Fertigungstechnik eine mögliche Strategie. Im Gegensatz dazu kann diese Strategie innerhalb der Verfahrenstechnik ungeeignet sein, da hier die Prozesse nicht immer gefahrlos heruntergefahren werden können. Eine Produktionsanlage im Falle eines Sensorausfalls in einen stabilen Zustand zu fahren, kann abhängig vom jeweiligen Prozess ebenfalls eine Möglichkeit sein, um weitere Funktionsstörungen zu vermeiden und gegebenenfalls Wartungsarbeiten vorzunehmen. Die Verwendung von redundanten Geräten oder Informationen und eine darauf basierende dynamische Rekonfiguration zur Laufzeit ist ein dritter Weg, um auf Sensorausfälle zu reagieren, ohne den jeweiligen Prozess zwingend abbrechen, herunterfahren oder in einen sicheren Zustand überführen zu

Daniel Schütz (✉), Birgit Vogel-Heuser

Automatisierungstechnik und Informationssysteme, Technische Universität München,

Bolzmannstraße 15, 85748 Garching, Deutschland

e-mail: schuetz@ais.mw.tum.de, vogel-heuser@ais.mw.tum.de

müssen. Die Verwendung von redundanten Geräten kann jedoch die Kosten für die Sensorik einer Anlage mindestens verdoppeln, darüber hinaus kann es in einigen Fällen aufgrund von räumlichen Einschränkungen technisch nicht möglich sein, redundante Sensoren zu installieren.

Davon ausgehend wurde im Rahmen des durch die Deutsche Forschungsgemeinschaft (DFG) geförderten Projekts „Agenten für flexible und verlässliche, eingebettete Echtzeitsysteme“ (AVE) [1] ein Ansatz entwickelt, die Verfügbarkeit von Produktionsanlagen in der Fertigungsautomatisierung sowie in der Prozessautomatisierung mittels redundanten Informationen durch so genannte Softsensoren zu erhöhen [2–4]. Dieser Ansatz integriert das Konzept der Redundanz durch Softsensoren in einen Ansatz für Softwareagenten, die auf handelsüblichen Speicherprogrammierbaren Steuerungen (SPS) implementiert werden können. Die Evaluation des Ansatzes an mehreren verschiedenen Demonstratoren zeigte, dass die Agenten in der Lage sind, Fehler von Sensoren selbstständig zu erkennen und während des laufenden Anlagenbetriebs zu kompensieren. Die entwickelten Softwareagenten ermöglichen mit Hilfe dynamischer Rekonfiguration auch im Fehlerfall einen Weiterbetrieb der Anlagen mit reduzierten Eigenschaften. Die Vorarbeiten umfassten sowohl ein Framework für die Implementierung von Softwareagenten auf Speicherprogrammierbaren Steuerungen in den Sprachen der IEC 61131-3 [5] als auch bereits einzelne Konzepte für Beschreibungsformen und Modellierungsansätze für eine automatische, modellbasierte Implementierung der Softwareagenten. Insbesondere sind dabei das „Redundanzmodell“ und das „Toleranzmodell“ zu nennen. Während innerhalb des Redundanzmodells die analytischen Zusammenhänge zwischen Sensorwerten als Grundlage für Softsensoren modelliert werden, beschreibt das Toleranzmodell die Auswirkungen der Verwendung eines gegebenenfalls weniger präzisen Softsensors auf die Funktionsausführung einer Anlage [4]. Die Modellierungsansätze und Beschreibungsformen basieren dabei auf den Notationen der Systems Modeling Language (SysML), einer objektorientierten Modellierungssprache, die auf die Beschreibung von technischen Systemen angepasst ist.

Innerhalb des DFG-Transferprojekts KREAagentuse [6] integrierte der Lehrstuhl für Automatisierung und Informationssysteme (AIS) der Technischen Universität München gemeinsam mit dem Transferunternehmen Beckhoff Automation GmbH (Beckhoff) die einzelnen Beschreibungsformen zu einem konsistenten Modell für Agentensysteme, aus dem automatisch der Steuerungscode für eine SPS generiert werden kann. Im Bereich der Modellierung von Produktionsanlagen eignet sich die SysML insbesondere zur Integration einer Anforderungsmodellierung [7] sowie einer parametrischen Modellierung von Anlagenmodulen [8]. Die Stärken der SysML wurden in KREAagentuse sowie parallelen Projekten für Anwendungsfälle wie zum Beispiel der energetischen Modellierung von Anlagen [9–10] weiter erschlossen. Im Rahmen des Projekts KREAagentuse wurde auf Basis der im Vorgängerprojekt AVE und weiteren Arbeiten des Lehrstuhls entwickelten Beschreibungsformen, wie zum Beispiel eines gerichteten Graphen für analytische Parameterzusammenhänge eines Anlagenmoduls, eine Werkzeugunterstützung entwickelt. Aus den Ergebnissen von AVE konnte zwar ein großes Potenzial für den Einsatz von autonomen, modellbasiert agierenden Softwareagenten auf der Feld-

ebene (SPS) einer Produktionsanlage identifiziert werden, die Ergebnisse zeigten aber auch, dass eine manuelle Implementierung durch Applikationsingenieure nur sehr schwer möglich und in der industriellen Praxis kaum umsetzbar ist. Die wesentliche Herausforderung in KREAagentuse war daher, einen Modellierungsansatz zu entwickeln, der zum einen die gefundenen Konzepte für die Implementierung von Softwareagenten darstellen kann und zum anderen zur Modellerstellung durch Applikationsingenieure geeignet ist. Weiter musste eine Werkzeugunterstützung entwickelt werden, die diesen Modellierungsansatz durch Editoren für die gewählten Modellierungsdiagramme anwendbar macht und das Modell eines Agentensystems direkt auf Elemente der objektorientierten Erweiterung der IEC 61131-3 abbildet. Dabei wurde die objekt-orientierte Erweiterung der IEC 61131-3 vorausgesetzt, wie sie im Werkzeug CoDeSys V3 der Firma 3S und TwinCAT 3 von Beckhoff umgesetzt wird [11].

Das entwickelte Werkzeug basiert auf einem Plug-in für CoDeSys V3 zur Erstellung von Steuerungscode durch verschiedene Diagramme der Unified Modeling Language (UML), das ebenfalls in Vorarbeiten des Lehrstuhls AIS entwickelt wurde [12]. Um den Applikationsingenieur weiter bei der Entwicklung von Softwareagenten für Speicherprogrammierbare Steuerungen zu unterstützen, wurde zusätzlich ein Vorgehensmodell entwickelt, das an die Struktur der Agentensysteme auf der einen Seite und an das Vorgehen bei der Entwicklung von konventionellem Steuerungscode auf der anderen Seite angepasst ist. Verschiedene domänenspezifische Sichten der Architektur trennen dabei die Entwicklung des Anlagenprozesses, z. B. eines Batch-Rezepts, von der Programmierung der einzelnen Anlagenfunktionen und der Anbindung der Software an die Sensorik und Aktorik einer Anlage. Damit wird die Komplexität der Entwicklung einer Anlagensteuerung in Abschnitte aufgeteilt, die einzelnen Gewerken zugeordnet werden können und das interdisziplinäre Arbeiten in einem gemeinsamen Softwareprojekt ermöglichen.

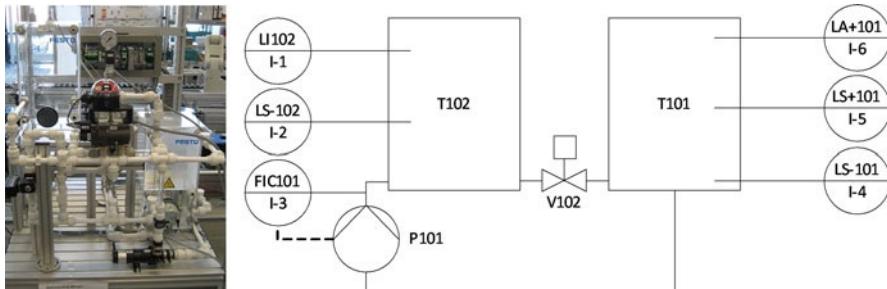
17.2 Erhebung der Anforderungen

Zu Beginn des Projekts KREAagentuse wurden die Ergebnisse des Vorgängerprojekts AVE sowie die Erfahrungen von Beckhoff im Hinblick auf die Zielsetzung des aktuellen Projekts gesammelt und aufbereitet. Auf dieser Grundlage entstand ein Interviewleitfaden, mit dem die Relevanz der Konzepte für die Entwicklung der Steuerungssoftware von automatisierten Anlagen durch vier Kunden von Beckhoff sowie drei Kooperationspartner des Lehrstuhls AIS bewertet wurde. Tabelle 17.1 zeigt die wesentlichen in Vorarbeiten erarbeiteten Konzepte und erzielten Ergebnisse.

Weiter wurde ein Modell, anhand dessen die Konzepte vereinfacht vermittelt werden können, entwickelt, welches auf dem verfahrenstechnischen Anlagenteil des Hybriden Prozessmodells (siehe Abb. 17.1), einer Laboranlage des Lehrstuhls AIS, basiert.

Tab. 17.1 (Teil-)Konzepte für die Implementierung eines SPS-basierten Agentensystems

-
- | | |
|----|---|
| K1 | Aufspaltung der Systemkomplexität durch spezifische Sichten |
| K2 | Zuordnung von Prozessschritten zu ausführenden Ressourcen |
| K3 | Toleranzmodell als Teil der Wissensbasis eines Agenten |
| K4 | Analytische Abhängigkeiten zwischen Sensorwerten (Redundanzmodell) |
| K5 | Entscheidungslogik für die Redundanz auf Basis des Redundanzmodells |
-

**Abb. 17.1** Demonstrationsmodell zur Darstellung des Ansatzes von KREAagentuse

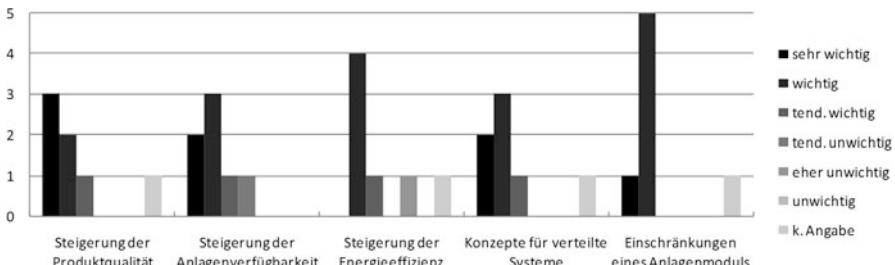
Im Projekt AVE wurde die Systembeschreibung aufgeteilt in die verschiedenen Sichten „Automatisierungssystem“, „Technisches System“ und „Technischer Prozess“ entwickelt, um unter anderem die Beschreibung des Anlagenprozesses von der konkreten Implementierung der Prozessschritte zu entkoppeln (K1). Das Umwälzen eines definierten Volumens einer Flüssigkeit zwischen den zwei Tanks T101 und T102 stellt die übergeordnete Prozessfunktion dar, die einem Ressourcenagenten, der den entsprechenden Anlagenteil steuert, zugeordnet werden kann (K2). Das Toleranzmodell als Teil der Wissensbasis eines Agenten (K3) bildet die Berechnung ab, ob mit einer aktuellen Konfiguration die geforderten Grenzwerte für das umgewälzte Flüssigkeitsvolumen eingehalten werden können. Die analytischen Abhängigkeiten zwischen den Sensoren des Anlagenteils im Redundanzmodell (K4) bilden die Berechnungsgrundlage für Softsensoren. Die Entscheidungslogik, wann ein realer Sensor von einem Agenten als defekt erklärt wird, kann zum Beispiel als ein Mehrheitsentscheid der in einem Tank verbauten Füllstandssensoren erklärt werden (K5).

Die Darstellungen der ausgewählten Konzepte wurden in einen Interviewleitfaden überführt, der es erlaubt diese Konzepte durch Industrievertreter bezüglich der Relevanz der Aspekte für das jeweilige Unternehmen zu bewerten. Tabelle 17.2 zeigt die wesentlichen Fragen des Interviewleitfadens sowie die Zuordnung der Fragen zu den ausgewählten Konzepten.

Die Interviewfragen bezüglich der Steigerung der Produktqualität beziehungsweise der Energieeffizienz wurden als Themen für spätere Forschungsprojekte zusätzlich aufgenommen. Die Energieeffizienz sollte im Rahmen der Interviews gegenüber den Bestrebungen zur Steigerung der Produktqualität sowie den weiteren

Tab. 17.2 Überführung der Anforderungen in einen Interviewleitfaden

Abgefragter Aspekt	Konzepte aus AVE
A Steigerung der Produktqualität	–
B Steigerung der Anlagenverfügbarkeit	K4, K5
C Steigerung der Energieeffizienz	–
D Konzepte für verteilte Systeme	K1, K2
E Einschränkungen eines Anlagenmoduls	K3

**Abb. 17.2** Ergebnisse der geführten Interviews – Gewichtung der Relevanz der Konzepte

abgefragten Konzepten bewertet werden. Die Expertenevaluation wurde als geführtes Interview mit ausgewählten Kunden von Beckhoff aus dem Bereich des Maschinen- und Anlagenbaus durchgeführt. Die Bedeutung der einzelnen Konzepte konnte durch die Industrievertreter mit „sehr wichtig“ bis „unwichtig“ mit insgesamt sieben Stufen bewertet werden.

Die Experten bewerteten die Steigerung der Produktqualität als wichtigste abgefragte Anforderung (vergleiche Abb. 17.2). Die Relevanz des Themas überwiegt nur knapp die der Erhöhung der Anlagenverfügbarkeit; hier bewerteten drei der befragten Unternehmen das Thema als „wichtig“. Bei den Konzepten für verteilte Systeme und modulare Ansätze wurden Defizite heutiger Methoden bei der Strukturierung des Programmcodes angemerkt. Bemängelt wurde vor allem, dass die Modularität, die früher mit eigenständigen Geräten (in Hardware) möglich war, in Zukunft auch mit vielen unabhängigen Applikationsmodulen auf einer Hardwareplattform möglich sein muss. Softwareagenten basieren zwar auf dem Konzept der Modularität, agieren jedoch im Gegensatz zu einfachen Softwaremodulen unabhängiger voneinander. Die breite Anwendung von Modularitätskonzepten innerhalb der Steuerungssoftware von Produktionsanlagen stellt damit eine wichtige Voraussetzung für die erfolgreiche Einführung von Agententechnologien dar. Die Relevanz, die die Experten in den Interviews der Modularität zugesprochen haben kann daher als positiver Trend in Richtung Softwareagenten interpretiert werden.

Vergleichsweise uneinig waren sich die Experten über die Notwendigkeit der Kompensation von Fehlern zur Laufzeit durch vorgegebene Funktionsbausteine bzw. Modelle. Dies wurde vor allem für sicherheitskritische Prozesse als wichti-

gen Aspekt eingestuft. Die Steigerung der Energieeffizienz wurde zwar von vier Unternehmen als „wichtig“, von keinem aber als „sehr wichtig“ bewertet.

17.3 Entwicklung von Modellierungssprache und Werkzeug

Insgesamt bestätigten die Ergebnisse der geführten Interviews die Relevanz der in früheren Arbeiten entwickelten Konzepte mit der Steigerung der Anlagenverfügbarkeit (Redundanzmodell) als ein übergeordnetes Ziel. Die hohe Relevanz in den Interviewergebnissen der Steigerung der Produktqualität und der Festlegung von Einschränkungen eines Anlagenmoduls zeigten weiter, dass eine Implementierung mittels Softsensoren auch geeignete Konzepte zur Verifizierung hinsichtlich dieser Anforderungen (Toleranzmodell) enthalten muss. Die von den Firmen angesprochenen Defizite heutiger Methoden zur Strukturierung verteilter Systeme bekräftigten außerdem den Ansatz der Entwicklung eines modellbasierten Implementierungswerkzeugs.

Nach [12] werden für eine erfolgreiche Systembeschreibung drei Komponenten benötigt: ein Beschreibungsmittel, ein unterstützendes Werkzeug sowie eine Methode. Nachdem im Vorgängerprojekt AVE bereits Methoden und erste Modelle als Beschreibungsmittel von SPS-basierten Agentensystemen entwickelt wurden, galt es im Rahmen von KREAagentuse, eine die gefundenen einzelnen Modelle in eine einheitliche Modellierungssprache zu vereinen, sowie eine entsprechende Werkzeugunterstützung zu realisieren.

17.3.1 Entwicklung der Modellierungssprache auf Basis der SysML

Um eine einheitliche Modellierungssprache für die zu realisierende Werkzeugunterstützung zu entwickeln, wurden die Systems Modeling Language (SysML) und die Darstellungen der Wissensbasis für Agenten aus AVE adaptiert. Zusätzlich wurden weitere Notationselemente auf Grundlage der SysML entwickelt, die generell für die Strukturierung und Entwicklung von Steuerungssoftware notwendig sind.

Damit wurde ein domänenpezifischer Dialekt der SysML für die agentenorientierte Softwareentwicklung durch die Reduktion der Diagramme und Modellierungselemente durchgeführt. Hierfür wurde das vom UML Klassendiagramm abgeleitete Blockdefinitionsdiagramm (BDD) für die Strukturdarstellung des Agentensystems adaptiert. Das Aktivitätsdiagramm (ACT) der SysML wurde für eine Modellierung von Produktionsprozessen und Beschreibung von Prozessanforderungen durch „Requirements“ adaptiert. Das Parameterdiagramm, welches eine Modellierung der Zusammenhänge zwischen Parametern erlaubt, wurde zusätzlich für die Modellierung der beiden Agentenmodelle (Redundanzmodell und Toleranzmodell [4, 13]) erweitert.

Analog zum Element „Klasse“ der UML, dienen in dem entwickelten Dialekt der SysML Blöcke zur Kapselung von Systemelementen. Für die Modellierung von Sensor- bzw. Aktorkomponenten wurden darüber hinaus entsprechende Stereotypen zur Unterscheidung von anderen Systemelementen eingeführt. Um Agenten im Modell von einfachen Bausteinen unterschieden zu können wurde der Stereotyp „Agent“ eingeführt. Das Aktivitätsdiagramm, durch welches ein Produktionsprozess implementiert werden kann, wird innerhalb eines Blocks mit dem Stereotypen „Process“ implementiert.

Das Parameterdiagramm [14] erfüllte alle Anforderungen bezüglich der Modellierung der Wissensbasis eines Agenten, sodass keine Erweiterungen zur Modellierung des Toleranzmodells als Teil dieser Wissensbasis nötig waren [13]. Für die Darstellung des Redundanzmodells musste im Parameterdiagramm zusätzlich der Stereotyp „FunctionalDependency“ eingeführt werden [13], um kennzeichnen zu können, welcher reale Sensor von einem anderen abhängig ist. Da auf dem existierenden UML-Plug-in aufgebaut wurde, wurden die vorhandenen und bereits positiv evaluierten Darstellungen aus dem Bereich der UML wie auch der SysML [15–16] für die Darstellung von Modellen eines Agentensystems adaptiert. Um innerhalb des Editors eine Unterscheidung zwischen der UML und SysML zu erreichen, wurden für abgeleitete wie neu entstandene Diagramme sich abhebende Farbschemata gewählt, wie sie beispielsweise in anderen verbreiteten marktüblichen SysML Editoren (z. B. NoMagic MagicDraw, Altova UModel, etc.) bereits angewendet werden.

Bei der Implementierung der Werkzeugunterstützung konnten umfangreiche Erfahrungen des Lehrstuhls aus der Entwicklung entwickelten UML-Editoren und deren Evaluation mit Industrieunternehmen einfließen. Die Entscheidung die Werkzeugunterstützung in CoDeSys V3 zu realisieren bedeutete ferner, dass das Tool auch zur Programmierung der Steuerungen anderer Hersteller, deren Umgebungen ebenfalls auf CoDeSys basieren, einsetzbar ist. Die Datenhaltung der Modelle agentenorientierter Software wird durch bereits vorhandene Mechanismen der Programmierumgebung CoDeSys umgesetzt. Zur Unterscheidung von agentenspezifischen und konventionellen Anteilen eines Softwareprojekts dienen die entwickelten Diagramme des Ansatzes (siehe Tab. 17.3), insbesondere das Blockdefinitionsdiagramm, in welchem durch die Erweiterungen des Metamodells der SysML agentenspezifische Softwarebausteine deutlich von konventionell implementierten Bausteinen unterschieden werden können.

Entsprechend der entwickelten Modellierungssprache wurden Editoren für Blockdefinitionsdiagramme, Aktivitätsdiagramme und Parameterdiagramme entwickelt. Durch die Erweiterungen des entwickelten Modellierungsansatzes gegenüber der reinen SysML mussten auch diese Editoren gegenüber der üblichen Implementation dieser Diagramme angepasst werden. So unterstützen die marktüblichen SysML Modellierungswerzeuge wie zum Beispiel NoMagic MagicDraw die Modellierung von Requirements und deren Verknüpfung zu Aktivitäten in Aktivitätsdiagrammen nicht; dies wird jedoch durch den Modellierungsansatz gefordert (siehe Tab. 17.3).

Tab. 17.3 Entwickelte Notationen und Editoren für den Modellierungsansatz

Diagramm	Notationselemente	Elementrelationen
Blockdefintiondiagramm (UML Klassendiagramm)	Interface, Block, Constraint, Process, Sensor, Actuator	Implements, Association, Composition
Aktivitätsdiagramm (UML Aktivitätsdiagramm)	Activity, Start/Final Node, Requirement	Control Flow, Data Flow, Containment
Parameterdiagramm (neu hinzugekommen)	Block, Constraint, Port, Variable, Property	BindingConnector, FunctionalDependency

Tab. 17.4 Entwickelte Abbildungsregeln von SysML auf IEC 61131-3 (Auszug)

SysML Modellelement	Anangepasste Stereotypen	IEC 61131-3 Sprachelement
Block	Agent, Process, Sensor, Actuator	Funktionsblock
Constraint-Block	–	Funktion
Port (In)/Port (Out)	–	Eingangsvariable/Ausgangsvariable
Requirement	–	Property (mit Kommentar)

17.3.2 Automatische Übertragung der Modelle in die IEC 61131-3

In Anlehnung an das existierende UML Plug-in für CoDeSys V3 wurden vollständige Abbildungsregeln zwischen dem auf SysML basierenden Modellierungsansatz für Agentensysteme und den Programmiersprachen der IEC 61131-3 entwickelt. Die Vollständigkeit dieser Abbildungsregeln erlaubt die zentrale Datenhaltung durch die in CoDeSys enthaltenen Mechanismen zur Speicherung von Softwaredatenprojekten. Des Weiteren wurden die Abbildungsregeln entwickelt, die eine Codegenerierung aus dem Parameterdiagramm ermöglichen. Die nachfolgende Tabelle (Tab. 17.4) zeigt einen Auszug der Abbildungsregeln und konzentriert sich hierbei auf die Modellierungselemente, die gegenüber dem UML-Plug-in hinzugekommen sind. Die im Modellierungsansatz enthaltenen Diagramme können unterteilt werden in Verhaltensdiagramme und Strukturdiagramme.

Das Blockdefinitionsdiagramm, welches auf UML-Klassendiagramm basiert, stellt ein Strukturdiagramm dar, in dem die Struktur des Agentensystems durch die Zusammenhänge (Kompositionsbeziehungen) einzelner Systemelemente (Blöcke) modelliert werden kann. Für dieses Diagramm musste daher kein Codegenerator erstellt werden, sondern lediglich die Abbildungsregeln angewendet werden, um die entsprechenden Elemente der IEC 61131-3 grafisch durch das Diagramm im Projekt zu erstellen.

Das Aktivitätsdiagramm hingegen stellt ein Verhaltensdiagramm innerhalb der UML und der SysML dar. Daher ist zusätzlich zu den Abbildungsregeln ein Codegenerator notwendig, um die in diesem Diagramm beschriebenen Modellteile in

IEC-Code überführen zu können. Da gegenüber dem UML-Plug-in lediglich „Requirements“ als Diagrammelemente hinzugekommen sind, welche sich nicht auf die Codegenerierung auswirken, konnte der für das UML-Plug-in entwickelte Codegenerator verwendet werden.

Das Parameterdiagramm als weiteres Verhaltensdiagramm stellt die Zusammenhänge zwischen den Parametern einzelner oder mehrerer Module dar. Darüber hinaus dient das Parameterdiagramm auch der Modellierung und Implementierung der Agentenmodelle „Toleranzmodell“ und „Redundanzmodell“.

Für die Agentenbausteine selbst wurden Agentenprototypen entwickelt, in welchen die applikationsunabhängigen Teile der Modellierung eines Agenten bereits implementiert sind. Diese bestehen unter anderem aus der Entscheidungslogik, die auf Grundlage der Informationen eines Redundanzmodells Defekte einzelner Sensoren erkennen und mit Hilfe von Funktionsbausteinen für Softsensoren kompensieren kann. Dazu wurden applikationsunabhängige Codefragmente von prototypischen Implementierungen isoliert, die für eine Implementierung von Softsensoren und damit einhergehende Steigerung der Anlagenverfügbarkeit genutzt wurden. Ebenso wurden Standardbausteine (Funktionsbausteine) für Sensorik- und Aktorikmodule erstellt.

Die Applikationsspezifischen Code-Anteile der Softwareagenten (Redundanzmodell und Toleranzmodell) werden mit dem entwickelten Modellierungsansatz durch die Modellierung von Parameterdiagrammen implementiert. Da sich die Semantik des Diagramms im allgemeinen Fall und dem des Toleranzmodells deutlich von der des Redundanzmodells unterscheidet, mussten zwei unterschiedliche Konzepte für die Codegenerierung angewandt werden. Dies wurde durch die Entwicklung eines Codegenerators mit zwei Ausbaustufen realisiert. Die erste Ausbaustufe übersetzt Parameterdiagramme auf Grundlage der definierten Abbildungsregeln in auf einer SPS lauffähigen Code (IEC 61131-3). Die zweite Ausbaustufe greift darüber hinaus in die Implementierung eines Agenten dergestalt ein, dass aus dem Redundanzmodell die applikationsspezifischen Anteile des Codes erstellt werden. So wird zum Beispiel durch die zweite Ausbaustufe in der Implementierung festgelegt für welche Sensoren redundante Werte zur Kompensation von Ausfällen zur Verfügung stehen.

17.4 Evaluation der Projektergebnisse

Um die Verständlichkeit der SysML-Notation zu untersuchen, wurde eine Usability-Untersuchung mit Probanden durchgeführt. Die Verständlichkeit bezieht sich in dieser Evaluationsmethode auf die Modellinhalte. Die Planung der Versuche folgt den Empfehlungen von Patig [17] bzw. orientiert sich an den Kriterien von Parsons und Cole [18].

Die Untersuchung untersucht Notationen mit gleichem Informationsgehalt (Continuous Function Chart (CFC), Structured Text (ST) und SysML Parameter Diagram (PD)) vergleichend. Die Aufgabe fordert Modellinterpretation und die Teil-

Tab. 17.5 Durchführung der Probandenversuche

Zeit [Min]	VP1	VP2	VP3	VP4	VP5	VP6
5	Vorwissenstest ST					
10	Training ST					
5	Vorwissenstest PD					
10	Training PD					
5	Vorwissenstest CFC					
10	Training CFC					
15	Pause					
10	Test A CFC	Test A CFC	Test A PD	Test A PD	Test A ST	Test A ST
10	Test B ST	Test B PD	Test B ST	Test B CFC	Test B PD	Test B CFC
10	Test C PD	Test C ST	Test C CFC	Test C ST	Test C CFC	Test C PD
5	Fragebogen zur Motivation und Arbeitsbelastung					

nehmer sind Novizen. Als Vergleichsnotationen wurden ST und CFC gewählt, die im Bereich der SPS-Programmierung verbreitet und akzeptiert sind. ST ist eine textuelle, prozedurale Programmiersprache und als Teil der IEC 61131-3 Norm [5] weit verbreitet. CFC ist die graphische SPS-Programmiersprache die dem PD am ähnlichsten ist und im Bereich der Prozessindustrie sowie der Regelungstechnik weit verbreitet ist.

Obwohl CFC keine der in der IEC 61131-3-Norm definierten Sprachen ist, stellt sie eine gängige Erweiterung von IEC-Programmierumgebungen, z. B. TwinCAT und CoDeSys, dar. Diese Sprachen wurden auf Grund ihrer Verbreitung und ihrer vergleichbaren semantischen Aussagekraft zum PD ausgewählt.

Um aussagekräftige Ergebnisse bei der vergleichenden Untersuchung zwischen dem agentenorientierten und dem konventionellen Ansatz zu gewinnen, müssen möglichst alle messbaren Störvariablen erfasst oder ausgeschlossen werden. Dafür wurden Fragebögen zur Vorerfahrung, demografischen Daten und der Motivation der Probanden sowie ein einheitliches Training erstellt. Um Werkzeugeffekte auszuschließen (siehe Friedrich [19]) wurde eine papierbasierte Aufgabenstellung und Bearbeitung vorgesehen.

Die sechs Probanden bearbeiteten drei möglichst ähnliche, aber nicht exakt gleiche Aufgabenstellungen (Aufgabe A, Aufgabe B und Aufgabe C) wobei sowohl die Aufgabenreihenfolge sowie die Notation permutiert werden. Die Versuchsdurchführung erfolgte mit sechs Versuchspersonen (VP1-6) in zwei Permutationen (Aufgabe und Reichenfolge), um Lerneffekte durch die Bearbeitungsreihenfolge auszuschließen wie in Tab. 17.5 dargestellt:

Die Versuche zeigten, dass die Probanden bei gegebenen Parameterdiagrammen besser in der Lage sind, die dargestellten Zusammenhänge nachzuvollziehen und Fragen zu den Inhalten korrekt zu beantworten, als dies bei ST und CFC der Fall war. Somit konnte die Verständlichkeit des Parameterdiagramms der SysML gegenüber ST und CFC positiv evaluiert werden. Das objektive Vorwissen in al-

len drei Notationen lag im unteren zweistelligen Bereich (Mittelwerte: ST 23,6 %, CFC 7,7 %, PD 19,4 %)¹ und bestätigte die geeignete Probandenwahl. Die Mittelwerte der Ergebnisse der Versuchsaufgabe² lagen für das PD mit 68,3 % deutlich über denen von CFC (64,3 %) und ST (63,5 %). Boden- und Deckeneffekte bei der Aufgabenbearbeitung wurden durch die geeignete Wahl der Aufgabenschwierigkeit vermieden. Die Mittelwerte zeigen eine durchwegs positive Tendenz für die Usability des PD im Vergleich zu den etablierten Notationen. Aus anschließend an die Aufgaben geführten Interviews ging hervor, dass die Probanden beim PD die intuitive, eingängige Darstellungsweise von Daten/Informationsflüssen begrüßten.

17.5 Zusammenfassung und Ausblick

Eine der häufigsten Fehlerursachen von automatisierten Produktionsanlagen sind defekte Sensoren, auf die normalerweise das Herunterfahren des Produktionsprozesses und die Durchführung von Wartungsarbeiten (Ersetzen defekter Sensoren) folgen. Eine Alternative kann, abhängig vom jeweiligen Prozess, z. B. das Fahren der Produktionsanlage in einen stabilen Zustand sein, um weitere Funktionsstörungen zu vermeiden. Zur Vermeidung von Produktionsausfällen, ist die Verwendung von redundanten Geräten oder Informationen und eine darauf basierende dynamische Rekonfiguration zur Laufzeit eine Möglichkeit, um auf Sensorausfälle zu reagieren, was jedoch mindestens die Verdoppelung der Kosten der Sensorik einer Anlage zur Folge hätte und in einigen Fällen aufgrund von räumlichen Einschränkungen technisch nicht möglich ist. Der Lösungsansatz, die Verfügbarkeit von Produktionsanlagen mittels redundanter Informationen durch die Implementierung von Softsensoren innerhalb der Steuerungssoftware zu erhöhen, wurde realisiert und evaluiert. Der Ansatz integriert das Konzept der Redundanz durch Softsensoren in eine Architektur für Softwareagenten, die auf handelsüblichen Speicherprogrammierbaren Steuerungen (SPS) implementiert wurden, und ergänzt es durch neu entwickelte prototypische Notationselemente.

Der Ansatz wurde an mehreren Laboranlagen, hinsichtlich einer Steigerung der Anlagenverfügbarkeit, positiv evaluiert werden. Darauf aufbauend wurde ein prototypisches Werkzeug für die modellbasierte Entwicklung und automatische Implementierung der Softwareagenten entwickelt und in eine Standardentwicklungs-umgebung für SPS-Steuerungscode nach IEC 61131-3 integriert. Die Beschreibung der Softwareagenten basiert dabei auf zum Teil erweiterten Notationen und Diagrammen der Systems Modeling Language (SysML). Um unter anderem den für die Softwareagenten entwickelten Modellierungsansatz mit diesem Sprachprofil abbilden zu können, wurde sowohl das Metamodell der SysML erweitert, als auch die existierenden Diagramme adaptiert. Auf Grundlage von Abbildungsregeln zwischen den Notationselementen eines Modells und dem korrespondierenden Steue-

¹ Skala: 0 % keine korrekte oder keine Antwort im Vorwissenstest – 100 % alle Antworten korrekt.

² Skala: 0 % keine korrekte oder keine Antwort in der Aufgabe – 100 % alle Antworten korrekt.

rungscode konnten Codegeneratoren (Modelltransformatoren) entwickelt werden, die eine automatische Implementierung von lauffähigem Steuerungscode ermöglicht. Eine Online-Ansicht unterstützt das Debugging direkt innerhalb der Modelle.

Aus dem SysML-basierten Modellierungsansatz wurde das erweiterte Parameterdiagramm, welches für das Agentenmodell wesentlich erweitert wurde, im Vergleich mit dem Continuous Function Chart (CFC) und der Programmiersprache Structured Text (ST) empirisch mit sechs Probanden, qualitativ evaluiert. Im Parameterdiagramm modellierte Software kann durch nicht am Entwicklungsprozess beteiligte Personen einfacher interpretiert und damit verstanden werden, als dies bei anderen Sprachen der IEC 61131-3 der Fall ist.

Als Ergebnisse dieses Projekts liegen erprobte Beschreibungsmittel (Diagramme und Notationen), ein Werkzeug und eine Methode vor, die es dem Applikationsingenieur ermöglichen agentenorientierte Ansätze in SPS (Beckhoff und CoDeSys-Familie) zu integrieren. Zwar besitzt die Entwicklungsumgebung CoDeSys, in welcher das zu dem Modellierungsansatz gehörende Werkzeug eingebettet wurde, eine sehr hohe Verbreitung, damit sich aber die modellbasierte Entwicklung von Softwareagenten durchsetzen kann, muss auch eine Anwendbarkeit auf die Steuerungen anderer Hersteller wie Siemens, Rockwell oder Phoenix Contact ermöglicht werden. Hierzu trägt die Erstellung des SysML-Profil bei.

Aktuelle Forschungsarbeiten beschäftigen sich ferner mit der Modelltransformation zwischen den im Modellierungsansatz verwendeten Parameterdiagrammen und Blockdiagrammen in Matlab/Simulink. Diese Arbeiten zielen unter anderem auf die Simulation der Modelle von Agentensystemen in dieser Umgebung ab, sodass das Verhalten von Agentensystemen vor ihrer Implementierung innerhalb einer Produktionsanlage durch die Simulation erprobt werden kann. Eine Modelltransformation, die die Überführung von CAD-Daten in SysML-Diagramme ermöglicht, wurde im Rahmen der Beteiligung des Lehrstuhls innerhalb des Exzellenzcluster CoTeSys ebenfalls entwickelt und untersucht. Im Rahmen des Sonderforschungsbereichs 768 wird die Integration verschiedener disziplinspezifischer Modelle in einem erweiterten Metamodell der SysML untersucht werden.

Danksagung Die Autoren bedanken sich bei der Deutschen Forschungsgemeinschaft für die Förderung des Transferprojekts KREAagentuse (Konzeption, Realisierung und Evaluation einer werkzeugunterstützten Vorgehensweise für die Entwicklung von Agentensystemen in der Automatisierungstechnik unter Berücksichtigung der Usability, Förderkennzeichen VO 937/8-1) sowie beim Transferunternehmen Beckhoff Automation GmbH für die gute Zusammenarbeit.

Literatur

- [1] DFG-Projekt: Agenten für flexible und verlässliche eingebettete Echtzeitsysteme (AVE). VO 937/5-1

- [2] Wannagat, A., Vogel-Heuser, B.: Agent oriented software-development for networked embedded systems with real time and dependability requirements the domain of automation. Proc. 17th IFAC World Congress, Seoul, South Korea (2008)
- [3] Wannagat, A., Vogel-Heuser, B.: Increasing flexibility and availability of manufacturing systems – Dynamic reconfiguration of automation software at runtime on sensor faults. Proc. 9th IFAC Workshop on Intelligent Manufacturing Systems (IMS) (2008)
- [4] Wannagat, A.: Entwicklung und Evaluation agentenorientierter Automatisierungssysteme zur Erhöhung der Flexibilität und Zuverlässigkeit von Produktionsanlagen. Dissertation, Technische Universität München (2010)
- [5] IEC 61131-3: Speicherprogrammierbare Steuerungen – Teil 3: Programmiersprachen, (2009)
- [6] DFG-Transferprojekt KREAagentuse. Konzeption, Realisierung und Evaluation einer werkzeugunterstützten Vorgehensweise für die Entwicklung von Agentensystemen in der Automatisierungstechnik unter Berücksichtigung der Usability, Förderkennzeichen VO 937/8-1.
- [7] Schütz, D., Wannagat, A.: Domänen spezifische Modellierung für automatisierungstechnische Anlagen mit Hilfe der SysML. Automatisierungstechnische Praxis (atp) **51**(3), 54–62 (2009)
- [8] Schütz, D.; Vogel-Heuser, B.; Ayeb, M.; Brabetz, L.: Architekturmmodelle zur Bewertung von energetischen Optimierungskriterien – MDA als Bindeglied domänen spezifischer Diagnose. Proc. of Mechatronik 2009, Wiesloch, S. 69–76. (2009)
- [9] Schütz, D., Vogel-Heuser, B.: Modellierung der Verhaltensaspekte automatisierungstechnischer Module von Produktionsanlagen unter Berücksichtigung der energetischen Zusammenhänge. In: U. Jumar (Hrsg.) 11. Fachtagung Entwurf komplexer Automatisierungssysteme (EKA), S. 85–93.
- [10] Schütz, D., Vogel-Heuser, B.: Modellintegration von Verhaltens- und energetischen Aspekten für mechatronische Module. Automatisierungstechnik (at) **59**(1), 33–42 (2010)
- [11] Vogel-Heuser, B.: Automation & Embedded Systems: Effizienzsteigerung im Engineering. kassel university press GmbH, Kassel (2009)
- [12] Schnieder, E.: Methoden der Automatisierung. Vieweg Verlagsgesellschaft, Braunschweig (1999)
- [13] Frank, U., Papenfort, J., Schütz, D.: Real-time capable software agents on IEC 61131 systems – Developing a tool supported method. Proc. of the 18th IFAC World Congress. Mailand, Italien (2011)
- [14] Object Management Group (OMG), Systems Modeling Language (SysML).
- [15] Vogel-Heuser, B., Sommer, K.: A methodological approach to evaluate the benefit and usability of different modeling notations for automation systems. Proc. of the 7th IEEE International Conference on Automation Science and Engineering (CASE), Triest (2011)
- [16] Vogel-Heuser, B., Braun, S., Kormann, B.: Implementation and evaluation of UML as modeling notation in object oriented software engineering for machine and plant automation. Proc. of the 18th IFAC World Congress, Mailand, Italien (2011)
- [17] Patig, S.: A Practical Guide To Testing the Understandability of Notations. In Proc. 5th Asia-Pacific Conference on Conceptual Modeling (APCCM 2008), S. 49–58. Wollongon, Australia (2008)
- [18] Parsons, J., Cole, L.: What do the pictures mean? Guidelines for experimental evaluation of representation fidelity in diagrammatical conceptual modeling techniques. Data & Knowledge Engineering **55**, 327–342 (2005)
- [19] Friedrich, D.: Anwendbarkeit von Methoden und Werkzeugen des konventionellen Softwareengineering zur Modellierung und Programmierung von Steuerungssystemen. Dissertation. Universität Kassel (2009)