Agentenunterstütztes Engineering von Automatisierungsanlagen

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von
Thomas Wagner
aus Gerabronn

Hauptberichter:	Prof. Dr.-Ing. Dr. h. c. Peter Göhner
Mitberichter:	Prof. Dr. rer. nat. habil. Paul Levi
Tag der Einreichung:	04.06.2007
Tag der mündlichen Prüfung:	09.01.2008

Institut für Automatisierungs- und Softwaretechnik
der Universität Stuttgart

2008

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Assistent am Institut für Automatisierungs- und Softwaretechnik (IAS) der Universität Stuttgart.

Mein besonderer Dank gilt meinem Doktorvater und Leiter des Instituts, Herrn Prof. Dr.-Ing. Dr. h. c. Peter Göhner, für die fortwährende Unterstützung und Förderung während der Entstehung dieser Arbeit, die zahlreichen wertvollen Anregungen und konstruktiven Diskussionen sowie die Übernahme des Hauptberichts.

Herrn Prof. Dr. rer. nat. habil. Paul Levi danke ich für das entgegengebrachte Interesse an meiner Arbeit und die Übernahme des Mitberichts.

Allen ehemaligen Kolleginnen und Kollegen am IAS gilt mein Dank für die kollegiale Atmosphäre und die gute Zusammenarbeit. Ganz besonders bedanke ich mich bei Dr. Nasser Jazdi, der mich auf diesen Weg geführt und mich stets freundschaftlich begleitet hat sowie bei Dr. Jan Traumüller, Dr. Stephan Eberle und Gerda Müller für die kritische und gründliche Durchsicht des Manuskripts.

In diesem Zusammenhang möchte ich auch die vielen Studierenden erwähnen, deren Diplom- und Studienarbeiten einen wesentlichen Anteil zum Gelingen der Arbeit beitrugen.

Schließlich möchte ich mich von Herzen bei meiner Familie bedanken, bei meinen Eltern für die langjährige Unterstützung während meiner gesamten Ausbildungszeit und bei meiner Frau Karin für ihr grenzenloses Verständnis, ihre Geduld und ihre Kraft, die entscheidend zum Gelingen dieser Arbeit beigetragen hat.

Stuttgart, im Januar 2008

Thomas Wagner

Inhaltsverzeichnis

Abbildungsverzeichnis ..... **iv**

Tabellenverzeichnis ..... **vi**

Abkürzungsverzeichnis ..... **vii**

Begriffsverzeichnis ..... **viii**

Zusammenfassung ..... **xi**

Abstract ..... **xii**

1 Einleitung und Motivation **1**
1.1 Bedeutung des Engineerings von Automatisierungsanlagen 1
1.2 Problematik des Engineerings 2
1.3 Ziel der aktiven Unterstützung von Engineeringtätigkeiten 3
1.4 Abgrenzung zu ähnlichen Themenstellungen 4
1.5 Gliederung der Arbeit 5

2 Grundlagen des Engineerings von Automatisierungsanlagen **7**
2.1 Automatisierungsanlagen 7
2.2 Engineering von Automatisierungsanlagen 8
2.3 Komponentenbasierter Ansatz im Engineering 9
2.3.1 Der Begriff der Komponente im Engineering 9
2.3.2 Merkmale des komponentenbasierten Ansatzes 11
2.3.3 Vorteile des komponentenbasierten Ansatzes 13
2.4 Vorgehensweise im komponentenbasierten Engineering 13
2.4.1 Erstellung eines Anlagenmodells aus einzelnen Komponenten 14
2.4.2 Erstellung gewerkspezifischer Teilmodelle 16
2.4.3 Wiederverwendung von Teillösungen 18
2.5 Bedeutung des Wissens für Engineeringtätigkeiten 19
2.6 Schwierigkeiten und Herausforderungen 21

3 Bestehende Ansätze zur Unterstützung des Engineerings von Automatisierungsanlagen **23**
3.1 Generelle Möglichkeiten der Rechnerunterstützung 23
3.2 Anforderungen an die Unterstützung des Engineerings 24
3.3 Unterstützung durch Computer Aided Engineering Systeme 26
3.4 Unterstützung durch Komponenten-Frameworks 27
3.5 Unterstützung durch wissensbasierte Systeme 29
3.6 Unterstützung durch agentenbasierte Beratungssysteme 33
3.7 Zusammenfassende Bewertung und Folgerung 36

4 Paradigma der Softwareagenten ... 38
4.1 Einführung in die agentenorientierte Denkweise ... 38
4.2 Grundlegende Konzepte der Agentenorientierung ... 39
4.2.1 Eigenschaften von Softwareagenten ... 39
4.2.2 Aufbau von Softwareagenten ... 40
4.2.3 Interaktionen von Softwareagenten ... 41
4.3 Agentenorientierte Softwareentwicklung ... 42
4.3.1 Agentenorientierte Methoden ... 42
4.3.2 Agentensysteme ... 43
4.4 Vorteile, Eignung und Herausforderungen des Einsatzes von Softwareagenten ... 44
4.5 Einsatz von Softwareagenten in der Automatisierungstechnik ... 46

5 Konzept des agentenunterstützten Engineerings ... 48
5.1 Automatisierung der Komponentenintegration ... 48
5.2 Ansatz eines flexiblen Unterstützungskonzepts ... 50
5.3 Beschreibung des agentenunterstützten Engineerings ... 52
5.3.1 Einsatz von Softwareagenten zur aktiven Unterstützung der Komponentenintegration ... 52
5.3.2 Einsatz von Softwareagenten zur aktiven Unterstützung weiterer Aufgaben ... 59
5.4 Anwendungsfälle der Unterstützung ... 61
5.5 Erweiterter Aufbau der Softwareagenten ... 64
5.6 Vorgehensweise zur Entwicklung eines agentenorientierten Modells der Unterstützung ... 65

6 Agentenorientiertes Modell der Unterstützung ... 67
6.1 Ziele des Agentensystems ... 67
6.1.1 Unterstützung der Komponentenhandhabung ... 67
6.1.2 Unterstützung gewerkspezifischer Tätigkeiten ... 71
6.2 Rollen des Agentensystems ... 72
6.3 Agententypen des Agentensystems ... 74
6.3.1 Agententyp Komponenten-Agent ... 74
6.3.2 Agententyp Bibliotheks-Agent ... 75
6.3.3 Agententyp Aspekt-Agent ... 75
6.4 Struktur des Agentensystems ... 76

7 Konzept zur expliziten Integration von Informationen und Wissen des Ingenieurs ... 78
7.1 Voraussetzung für die Nutzung von Komponenteninformationen und Wissen ... 78
7.2 Formalisierte Beschreibung von Komponenteninformationen und Wissen ... 79
7.2.1 Grundlegender Aufbau von Komponentenbeschreibungen ... 79
7.2.2 Beschreibungsentitäten für Komponenteneigenschaften ... 80
7.2.3 Beschreibungsentitäten für Komponentenschnittstellen ... 81
7.2.4 Beschreibung von Verbindungs- und Parametrierregeln ... 82
7.2.5 Beschreibung von Baugruppen ... 86
7.2.6 Beschreibung der internen Varianten von Teillösungen ... 87
7.2.7 Umsetzung des Beschreibungsmodells ... 90
7.3 Formalisierte Beschreibung komponentenübergreifenden Wissens ... 90

7.4 Bereitstellung, Transfer und Erfassung von Informationen und Wissen ... 91

8 Funktionales Verhalten des Agentensystems ... 93
8.1 Grundsätze der Aktionen und Interaktionen der Agenten ... 93
8.2 Fähigkeiten der Agenten ... 96
8.3 Aktionen und Interaktionen von Komponenten-Agenten ... 97
8.3.1 Interaktion zur Handhabung von komponentenübergreifenden Wechselwirkungen ... 98
8.3.2 Aktionen der Rolle Änderungswächter ... 100
8.3.3 Aktionen der Rolle Konfigurationsmanager ... 103
8.3.4 Aktionen der Rolle Baugruppenmanager ... 106
8.4 Aktionen und Interaktionen von Aspekt-Agenten ... 106
8.4.1 Interaktion zur Verbindungsprüfung und -erstellung ... 106
8.4.2 Aktionen der Rolle Verbindungsmanager ... 107
8.4.3 Aktionen der Rolle Instanzenselektierer ... 109
8.4.4 Aktionen der Rolle Modellierer ... 110
8.4.5 Aktionen der Rolle Teilmodellmanager ... 111
8.5 Aktionen und Interaktionen des Bibliotheks-Agenten ... 111
8.6 Spezifische Merkmale der Interaktionen der Agenten ... 112
8.7 Interaktionen der Agenten mit dem Ingenieur ... 113

9 Realisierung des Unterstützungskonzepts ... 114
9.1 Überblick über das Werkzeugsystem ... 114
9.2 Realisierung des exemplarischen Komponentenmodells ... 114
9.3 Realisierung eines CAE-Werkzeugs ... 115
9.4 Realisierung des Agentensystems ... 116
9.5 Integration des Agentensystems ... 117

10 Anwendungsbeispiel für die Unterstützung ... 119
10.1 Anwendungsbereich Fördertechnikanlagen ... 119
10.2 Aufbau der Beispielanlage 3-Achs-Portal ... 120
10.3 Agentenunterstütztes Engineering des 3-Achs-Portals ... 121
10.3.1 Verbinden von Komponenten ... 122
10.3.2 Änderung der Konfiguration einer Komponente ... 125
10.3.3 Erstellung eines gewerkspezifischen Teilmodells ... 128
10.4 Ergebnisse und Erfahrungen ... 129

11 Zusammenfassung und Ausblick ... 131
11.1 Bewertung des Konzepts ... 131
11.2 Voraussetzungen und Grenzen ... 134
11.3 Ausblick ... 134

Literaturverzeichnis ... 136

Anhang A Notationsübersicht Aktivitätsdiagramme ... 148

Anhang B Regeln des gewerkspezifischen Teilmodells Netzwerkplan ... 149

Abbildungsverzeichnis

Abbildung 2.1: Phasen im Lebenszyklus einer Automatisierungsanlage 7
Abbildung 2.2: Prinzip des komponentenbasierten Ansatzes im Engineering 12
Abbildung 2.3: Beispiel einer Änderung und möglicher Wechselwirkungen 15
Abbildung 2.4: Beispiel für ein gewerkspezifisches Teilmodell Netzwerkplanung 16
Abbildung 2.5: Beispiel für eine Baugruppe 18
Abbildung 2.6: Wissen als wichtige Einflussgröße für die Tätigkeiten beim Engineering 20
Abbildung 3.1: Benutzungsoberfläche des Engineering Systems Comos PT 27
Abbildung 3.2: Anwendungsentwicklung mit Komponenten-Frameworks 28
Abbildung 3.3: Bestandteile eines Expertensystems nach [HaKi86] 31
Abbildung 4.1: Eigenschaften von Softwareagenten 40
Abbildung 4.2: Grundlegender Aufbau von Softwareagenten 41
Abbildung 4.3: Übersicht über die Konzepte agentenorientierter Methoden 43
Abbildung 4.4: FIPA Agent Management Referenzmodell [FIPA04] 44
Abbildung 5.1: Automatisierung der Komponentenintegration 49
Abbildung 5.2: Erweiterung der Komponentenebene um aktive Unterstützungsfunktionalität 49
Abbildung 5.3: Kapselung von Komponenteninstanzen durch Softwareagenten 52
Abbildung 5.4: Grundprinzipien des Konzepts des agentenunterstützten Engineerings 53
Abbildung 5.5: Modularisierung der Unterstützungsfunktionalität 54
Abbildung 5.6: Regelformen zur Beschreibung von technischen Abhängigkeiten 57
Abbildung 5.7: Erweitertes Konzept für den Aufbau von Softwareagenten 64
Abbildung 5.8: Schritte zur Entwicklung eines agentenorientierten Modells der Unterstützung 66
Abbildung 6.1: Ziele zur Unterstützung der Komponentenhandhabung 67
Abbildung 6.2: Zielhierarchie Auswahl und Instanziierung einer Komponente 68
Abbildung 6.3: Zielhierarchie Verbindung von Komponenteninstanzen 68
Abbildung 6.4: Zielhierarchie Anpassung einer Komponenteninstanz 70
Abbildung 6.5: Ziele zur Unterstützung gewerkspezifischer Tätigkeiten 71
Abbildung 6.6: Agententyp Komponenten-Agent: Vertretungsprinzip und Rollen 74
Abbildung 6.7: Agententyp Bibliotheks-Agent: Vertretungsprinzip und Rollen 75
Abbildung 6.8: Agententyp Aspekt-Agent: Vertretungsprinzip und Rollen 76
Abbildung 6.9: Beziehungen und Interaktionen der Agententypen 77
Abbildung 7.1: Entitäten zur Beschreibung einer Komponente 79
Abbildung 7.2: Entitäten zur Beschreibung von Komponenteneigenschaften 80
Abbildung 7.3: Entitäten zur Beschreibung von Komponentenschnittstellen 81
Abbildung 7.4: Beispiel einer Komponentenbeschreibung 82
Abbildung 7.5: Entitäten zur Beschreibung von Verbindungs- und Parametrierregeln 84
Abbildung 7.6: Entitäten zur Beschreibung von Baugruppen 86
Abbildung 7.7: Zusammenhang der Variabilitätsebenen einer wiederverwendbaren Teillösung 87
Abbildung 7.8: Beispiel für Varianten des internen Aufbaus einer Teillösung 88
Abbildung 7.9: Bereitstellung, Transfer und Erfassung von Informationen und Wissen 92
Abbildung 8.1: Einflussbereich und beispielhafte Interaktionen 93
Abbildung 8.2: Menschliche Vorgehensweise zur Handhabung von Wechselwirkungen 94
Abbildung 8.3: Wechselseitige Ermittlung und Prüfung von Wechselwirkungen 98
Abbildung 8.4: Interaktion zur Handhabung komponentenübergreifender Wechselwirkungen 99

Abbildung 8.5: Aktion „Parameteränderung prüfen“ 100
Abbildung 8.6: Aktion „Änderungsauswirkungen ermitteln“ 101
Abbildung 8.7: Aktion „Rückmeldungen verwalten“ 102
Abbildung 8.8: Aktion „Anpassungen komponentenübergreifend umsetzen“ 103
Abbildung 8.9: Aktion „Konsistenzbedingungen und Änderungen weiterleiten“ 103
Abbildung 8.10: Aktion „Externe Konsistenzbedingungen und Änderungen prüfen“ 105
Abbildung 8.11: Aktion „Lokale Anpassungen umsetzen“ 106
Abbildung 8.12: Interaktion zur Verbindungsprüfung und -erstellung 107
Abbildung 8.13: Aktion „Verbindung prüfen“ 108
Abbildung 8.14: Aktion „Komponenteninstanzen auswählen“ 110
Abbildung 8.15: Aktion „Teilmodell bilden“ 110
Abbildung 9.1: Gesamtarchitektur des Werkzeugsystems 114
Abbildung 9.2: Auszug aus einer Komponentenbeschreibung 115
Abbildung 9.3: Benutzungsoberfläche des CAE-Werkzeugs 116
Abbildung 9.4: Management- und Beobachtungswerkzeuge des Agentensystems 117
Abbildung 9.5: Systeminteraktion zwischen CAE-Werkzeug und Unterstützungssystem 118
Abbildung 10.1: Gesamtansicht des Demonstrationsmodells 3-Achs-Portal 120
Abbildung 10.2: Schematischer Aufbau und Materialfluss des 3-Achs-Portals 121
Abbildung 10.3: Anlagenmodell als Ausgangssituation der Anwendungsszenarien 122
Abbildung 10.4: Interne Abläufe des Agentensystems bei Verbindungsprüfung 122
Abbildung 10.5: Ergebnis der Verbindungsprüfung mit Lösungsvorschlag 124
Abbildung 10.6: Agenten-Interaktionen bei Verbindungsprüfung und -umsetzung 124
Abbildung 10.7: Anlagenmodell nach erfolgten Verbindungen und Anpassungen 125
Abbildung 10.8: Interne Abläufe des Agentensystems bei Änderungsprüfung 125
Abbildung 10.9: Ergebnis der Änderungsprüfung mit Lösungsvorschlag 127
Abbildung 10.10: Anlagenmodell nach Parameteränderung und Agenten-Anpassungen 127
Abbildung 10.11: Agenten-Interaktionen bei Teilmodellerstellung (Ausschnitt) 128
Abbildung 10.12: Erstelltes gewerkspezifisches Teilmodell Netzwerkplan 129

Tabellenverzeichnis

Tabelle 3.1: Agententypen im Beratungssystem nach Konrad 34
Tabelle 3.2: Vergleich existierender Ansätze zur Unterstützung des Engineerings 36
Tabelle 4.1: Umsetzung der Prinzipien der Softwaretechnik im agentenorientierten Paradigma 39
Tabelle 5.1: Ausprägungen der agentenorientierten Grundkonzepte im Unterstützungskonzept 55
Tabelle 6.1: Rollen des Agentensystems 73
Tabelle 8.1: Fähigkeiten der Rollen des Agententyps Komponenten-Agent 96
Tabelle 8.2: Fähigkeiten der Rollen des Agententyps Aspekt-Agent 97
Tabelle 8.3: Fähigkeiten der Rollen des Agententyps Bibliotheks-Agent 97
Tabelle 10.1: Beispiele für identifizierte Informationen und Wissen über Komponenten 119
Tabelle 10.2: Ergebnisse der Unterstützung in den Anwendungsszenarien 130

Abkürzungsverzeichnis

ACL	**A**gent **C**ommunication **L**anguage
AMS	**A**gent **M**anagement **S**ystem
AOSE	**A**gent-**O**riented **S**oftware **E**ngineering
CAD	**C**omputer **A**ided **D**esign
CAE	**C**omputer **A**ided **E**ngineering
CAEX	**C**omputer **A**ided **E**ngineering E**X**change
CAN	**C**ontroller **A**rea **N**etwork
CASE	**C**omputer **A**ided **S**oftware **E**ngineering
DF	**D**irectory **F**acilitator
E/A	**E**in-/**A**usgabe
EDDL	**E**lectronic **D**evice **D**escription **L**anguage
EMF	**E**clipse **M**odeling **F**ramework
FIPA	**F**oundation for **I**ntelligent **P**hysical **A**gents
GEF	**G**raphical **E**ditor **F**ramework
ID	**Id**entifikator
JADE	**J**ava **A**gent **DE**velopment Framework
KQML	**K**nowledge **Q**uery and **M**anipulation **L**anguage
LEAP	**L**ightweight **E**xtensible **A**gent **P**latform
LS/TS	**L**iving **S**ystems **T**echnogy **S**uite
MaSE	**M**ulti**a**gent **S**ystems **E**ngineering
MTS	**M**essage **T**ransport **S**ervice
STEP	**St**andard for the **E**xchange of **P**roduct Model Data
UML	**U**nified **M**odeling **L**anguage
XML	E**X**tended **M**arkup **L**anguage

Begriffsverzeichnis

Abstraktion: Verfahren zur Reduzierung der Komplexität eines Problems durch Trennung der für bestimmte Aspekte der Problemlösung wichtigen und unwichtigen Details des Problems.

Agent (Softwareagent): Konzept einer abgrenzbaren Softwareeinheit mit einem definierten Ziel. Ein Agent versucht, dieses Ziel durch autonomes Verhalten zu erreichen und interagiert dabei kontinuierlich mit seiner Umgebung und anderen Agenten.

Agentenorientierung (Agentenorientierte Softwareentwicklung): Paradigma der Softwareentwicklung, bei dem die Struktur und die Funktionalität eines Softwaresystems unter dem Gesichtspunkt einer Menge von Agenten abstrahiert werden.

Agentensystem: Gemäß eines agentenorientierten Softwareentwurfs implementiertes Softwaresystem, bestehend aus nebenläufigen Softwareeinheiten, die eigenständig Aktionen ausführen und diese durch Interaktionen koordinieren.

Aktion: Baustein für die Spezifikation des Verhaltens eines Systems.

Aktor: Einheit zur Umsetzung von Stellinformation tragenden Signalen geringer Leistung in leistungsbehaftete Signale einer zur Prozessbeeinflussung notwendigen Energieform.

Automatisierungsanlage: Technisches System, in dem ein technischer Prozess zur Umformung, Verarbeitung und zum Transport von Materie oder Energie abläuft und dessen physikalische Größen mit technischen Mitteln erfasst und beeinflusst werden.

Baugruppe: Systemspezifischer Baustein, der aus mehreren Komponenten oder Baugruppen besteht. Baugruppen dienen der hierarchischen Gliederung eines komplexen Systems bei der komponentenbasierten Entwicklung.

Bibliothek: Organisierte Sammlung von wiederverwendbaren Bausteinen, die voneinander größtenteils unabhängig sind und nach Bedarf in eigenen Systemen oder Applikationen verwendet werden können.

E/A-Modul: Busankoppel-Modul zum Anschluss von Sensoren und Aktoren an ein Feldbussystem.

Engineering: Ingenieurtätigkeiten zum technischen Entwurf und der Auslegung einer Automatisierungsanlage.

Gewerk: Tätigkeitsfeld einer ingenieurtechnischen Fachdisziplin.

Integration: Zusammensetzung einzelner Bausteine zur Herstellung eines Gesamtsystems.

Interaktion: Spezifikation eines Verhaltens, das als Austausch von Nachrichten zwischen Einheiten beschrieben wird.

Komponente: Systembaustein, der speziell für die mehrfache Verwendung in verschiedenen Systemen oder Applikationen entwickelt wurde und unverändert eingesetzt wird.

Komponentenbasierte Entwicklung: Entwicklung eines Systems auf der Basis bereits vorhandener Komponenten, die unverändert verwendet werden.

Komponentenbeschreibung: Informationstechnische Beschreibung aller anwendungsbezogenen Merkmale einer Komponente. Zu den wichtigsten Bestandteilen gehören die einstellbaren Parameter und die verfügbaren Schnittstellen.

Komponenteninstanz: Individuelle, informationstechnisch angepasste Ausprägung einer Komponentenbeschreibung.

Konfiguration: Bestimmte Ausprägung der Eigenschaften einer Komponenteninstanz.

Konsistenz: Inhaltliche Widerspruchslosigkeit von Informationen bezogen auf die darin beschriebenen realen Zusammenhänge.

Methode: Nachvollziehbare Vorgehensweise zur Erreichung eines Ziels.

Modell: Beschreibung der Eigenschaften eines realen Systems, die aus einer bestimmten Sicht von Interesse sind. Ein Modell umfasst nicht nur eine Auflistung von Eigenschaften des realen Systems, sondern bildet auch immer strukturelle Aspekte ab.

Parametrisierung: Wahl von Variablenwerten, die das Systemverhalten bei gegebener Struktur charakterisieren.

Prinzip: Bewährter Grundsatz für eine Vorgehensweise.

Regel: In der Informatik im Zusammenhang mit regelbasierten Systemen ein formalisierter Konditionalsatz der Form "Wenn A dann B".

Rolle: Abgrenzbare Aufgabe oder Verantwortlichkeit innerhalb eines Softwaresystems.

Schnittstelle: Teil eines Systems oder Systembausteins, das dem Austausch von Informationen, Energie oder Materie mit anderen Systemen oder Systembausteinen dient.

Sensor: Einheit zur Umsetzung von physikalischen oder chemischen Messwerten aus einem technischen Prozess in elektrische oder optische Messsignale, die als Eingabe für das den technischen Prozess steuernde Automatisierungssystem dienen.

Technische Abhängigkeit: Aufgrund der physikalischen Gegebenheiten eines technischen Prozesses bestehender Zusammenhang zwischen den technischen Funktionen einzelner Komponenten, welche zur Beeinflussung des technischen Prozesses eingesetzt werden.

Technische Komponente: Einzelner Baustein der Anlagentechnik, der speziell für die mehrfache Verwendung in verschiedenen Automatisierungsanlagen entwickelt wurde und unverändert eingesetzt werden kann.

Teillösung: Informationstechnische Beschreibung einer Baugruppe, welche speziell für die mehrfache Verwendung in verschiedenen Automatisierungsanlagen abgegrenzt wurde.

Wechselwirkung: Gegenseitiger Einfluss oder das sich gegenseitige Beeinflussen zwischen zwei oder mehreren, ursprünglich unabhängigen Dingen oder Ereignissen.

Wissen: Gesamtheit aller organisierten Informationen mitsamt ihrer wechselseitigen Zusammenhänge, auf deren Grundlage ein (vernunftbegabtes) System handeln kann.

Zusammenfassung

Im Zuge des fortschreitenden globalen Wettbewerbs kommt dem Standort Deutschland zunehmend die Rolle eines „Engineering“-Standorts denn eines Produktionsstandorts zu. Im Bereich der Anlagenautomatisierung werden unter dem Begriff Engineering die Arbeitsprozesse und Tätigkeiten beim technischen Entwurf und der Auslegung von Automatisierungsanlagen zusammengefasst. Die Kosten für das Engineering hängen wesentlich von der Effizienz und Produktivität der menschlichen Arbeitsprozesse und der Qualität der resultierenden Engineeringinformationen ab. Dabei stellt neben methodischen und technologischen Aspekten die Beachtung der technischen Zusammenhänge zwischen den einzelnen Anlagenkomponenten eine große Herausforderung dar. Diese sind sehr vielfältig und für jede Automatisierungsanlage unterschiedlich ausgeprägt. Sie müssen daher im Zuge des Engineerings vollständig erfasst und aufeinander abgestimmt werden, was heute zum überwiegenden Teil manuell erfolgt und hohe Aufwendungen sowie zusätzliche Fehlermöglichkeiten mit sich bringt.

Ausgehend von der modernen komponentenbasierten Vorgehensweise im Engineering und der Beschaffenheit der erstellten Engineeringinformationen wird in der vorliegenden Arbeit ein Ansatz vorgestellt, der zur informationstechnischen Unterstützung des Engineerings von Automatisierungsanlagen dient. Durch geeignete Konzepte wird eine deutliche Reduktion des manuellen Aufwandes und eine vereinfachte Durchführung der menschlichen Tätigkeiten ermöglicht. Dabei wird durch den Einsatz von Softwareagenten eine aktive Form der Unterstützung bereitgestellt, welche sich flexibel an den Ablauf der menschlichen Arbeitsprozesse anpasst. Das Konzept nutzt die vorhandenen Engineeringinformationen und bestehendes Wissen über technische Abhängigkeiten einzelner Komponenten und überträgt diese auf einzelne Softwareagenten. Auf dieser Basis agieren und kooperieren die Softwareagenten parallel zu den Tätigkeiten des Ingenieurs im Hintergrund. Sie sind in der Lage, die beim Engineering entstehenden technischen Zusammenhänge innerhalb der Automatisierungsanlage selbstständig zu erkennen, zu analysieren und geeignete Anpassungen der Engineeringinformationen zu ermitteln. Die Interaktion mit dem Ingenieur erfolgt in Form von entsprechenden Hinweisen und Lösungsvorschlägen, welche auf Wunsch von den Softwareagenten selbstständig umgesetzt werden können. Auf diese Weise werden die individuellen technischen Zusammenhänge einer Automatisierungsanlage bereits auf informationstechnischer Ebene berücksichtigt und ein Großteil der bisher erforderlichen manuellen Tätigkeiten und Überlegungen kann entfallen. Das Konzept wurde darüber hinaus so ausgelegt, dass es zu bisher verwendeten Komponentenmodellen und Werkzeugen kompatibel ist und sich problemlos in bestehende Engineeringprozesse integrieren lässt.

Abstract

In the course of progressive global competition, the role of Germany increasingly shifts from a production location to an „engineering“ location. Within the domain of plant automation under the term engineering the working processes and the activities for technical design and layout of automation plants are summarized. The engineering costs depend substantially on the efficiency and productivity of the human working processes and on the quality of the resulting engineering information. Apart from methodical and technological aspects, the consideration of the technical correlations between the individual plant components constitutes an important challenge. They are multifaceted and take different shape for each automation plant. They must be regarded and coordinated completely in the course of the engineering, which today is predominantly done manually and causes high efforts.

Starting from the modern component-based engineering proceeding and the constitution of the resulting engineering information an approach is presented in this work, which provides a comprehensive information-technical support of the engineering of automation plants. Suitable concepts reduce manual efforts significantly and enable a simpler and more efficient accomplishment of the human activities. Thereby using software agents, an active form of support is made available, which adapts itself in a flexible way to the procedure of human working processes. The concept utilises the existing engineering information and available knowledge about technical dependences of individual plant components and transfers them to several software agents. The software agents act and cooperate in the background of the activities of the engineer in order to investigate autonomously the evolving technical correlations within the automation plant, and to determine suitable adjustments of engineering information. The interactions with the engineer take place in the form of appropriate references and suggestions for solutions, which if desired can be implemented independently by the software agents. Thus, the individual technical correlations of an automation plant are already considered on an information-technical level and the engineer’s manual work and considerations are reduced significantly. Further, the concept can be applied together with existing component models and tools, what simplifies its integration into existing engineering processes.

1 Einleitung und Motivation

1.1 Bedeutung des Engineerings von Automatisierungsanlagen

Automatisierungsanlagen sind heute zentraler Bestandteil verschiedenster Industriebereiche: in der Verfahrenstechnik, der Produktionstechnik, der Logistik oder der Energietechnik. Konfrontiert mit dem steigenden Wettbewerbsdruck auf dem Weltmarkt können sich Industrieunternehmen im Anlagenbau den Forderungen nach einer nachhaltigen Steigerung der Effizienz ihrer Geschäftsprozesse nicht mehr entziehen [WDDD05]. Einen hohen Stellenwert nehmen dabei die Arbeitsprozesse für den technischen Entwurf und die Auslegung von Automatisierungsanlagen ein. Sie werden allgemein unter dem Begriff (Anlagen-) Engineering zusammengefasst. Ziel des Engineerings ist es, ausgehend von den verfügbaren Komponenten der Anlagentechnik eine vollständige Automatisierungsanlage zu entwickeln [LBB+05]. Es bildet die Schnittstelle zwischen Komponentenentwicklung – technische Komponenten wie Sensoren oder Antriebe – und Systementwicklung – die Entwicklung einer Automatisierungsanlage, in der diese Komponenten verwendet werden.

Die Arbeitsprozesse im Engineering zeichnen sich durch Multidisziplinarität, komplizierte Entscheidungsprozesse, ein hohes Maß an Kreativität sowie durch eine große Dynamik aus [MaNa03]. Die vielfältigen Tätigkeiten des Ingenieurs erfordern umfassendes Wissen, werden heute vorwiegend manuell durchgeführt und sind daher mit entsprechenden Arbeitsaufwänden und -kosten verbunden. Über die Hälfte der Aufwendungen in der Planungsphase einer Automatisierungsanlage entfallen auf das Engineering. Die Engineeringkosten haben dadurch mit 15% einen wesentlichen Anteil an den Gesamtkosten im Lebenszyklus einer Automatisierungsanlage und werden zukünftig weiter steigen [Int03a], [Int03b]. Ursache hierfür sind sinkende Hardwarekosten, während die Engineering- und Integrationsaufwände zunehmen [Löwe02].

Die Steigerung der Produktivität und Qualität im Engineering von industriellen Automatisierungsanlagen ist somit eine ständige Herausforderung, um die eigene Wettbewerbsposition zu verbessern [Eppl03], [LBB+05], [Rodi02]. Dies erfordert den Einsatz effizienterer Arbeitsmethoden und Techniken. Zwei wesentliche Maßnahmen werden als Erfolg versprechend angesehen: die Bereitstellung geeigneter Softwarewerkzeuge, welche sowohl auf die technischen Gesichtspunkte als auch auf die spezifischen Anforderungen der Arbeitsabläufe beim Engineering zugeschnitten sind und die Möglichkeit zur flexiblen Wiederverwendung von Engineeringinformationen und -lösungen [AAF03], [AcLö05], [MaNa03].

Heute begleitet rechnergestütztes Engineering mittels Computer Aided Engineering (CAE) Systemen die Automatisierungsanlage über alle Phasen der Lebensdauer [VDE00]. Es stehen eine

Reihe von Softwarewerkzeugen zur Verfügung, die beim Engineering eingesetzt werden. Sie bieten vielfältige, meist grafische Funktionen zur Erstellung von Anlagenmodellen und Engineeringinformationen wie Anlagenstruktur, Verfahrensfließbilder, Elektroplanung und Prozessperipherie. Weitgehend wird auch die Wiederverwendung von Engineeringinformationen unterstützt. Dennoch kann festgestellt werden, dass CAE-Systeme den Problemlösungsprozess des Ingenieurs noch nicht in befriedigender Weise unterstützen [RHA02]. Die Komplexität der Arbeitsprozesse des Ingenieurs (und der dabei entstehenden und verarbeiteten Informationen) bedingt weiterhin einen hohen manuellen Aufwand. Es geht also primär darum, eine geeignete Form der softwaretechnischen Unterstützung für die Aufgaben beim Engineering anzubieten und somit eine Entlastung von manuellen Tätigkeiten zu ermöglichen.

1.2 Problematik des Engineerings

Die Komplexität der Arbeitsprozesse im Engineering wird von mehreren Faktoren beeinflusst. Zum einen werden Automatisierungsanlagen individuell auf Basis spezifischer Kundenwünsche entwickelt [LBB+05]. Die Herausforderung besteht darin, das Zusammenspiel einzelner technischer Komponenten so zu gestalten, dass die geforderte Gesamtfunktionalität der Automatisierungsanlage erfüllt wird. Da Automatisierungsanlagen aufgrund ihrer Größe aus einer Vielzahl technischer Komponenten bestehen und verschiedene fachspezifische Gesichtspunkte zu berücksichtigen sind, entsteht so im Laufe des Engineerings eine Fülle komplex strukturierter Information, die es zu handhaben gilt.

Zum anderen bestehen vielfältige technische Abhängigkeiten zwischen einzelnen Komponenten, die bei ihrer Integration zur gesamten Anlage berücksichtigt werden müssen. Der Ingenieur muss beispielsweise prüfen, ob Komponenten zusammenpassen oder ob die Konfiguration einer Komponente zu den Konfigurationen anderer Komponenten kompatibel ist. Gegebenenfalls müssen dann wiederholt Anpassungen von Komponenten durchgeführt werden, um ihr *konsistentes* Zusammenspiel und somit die korrekte Gesamtfunktionalität der Anlage zu gewährleisten. Dadurch werden zusätzlich zur kreativen Lösungsfindung weitere aufwendige Detailtätigkeiten erforderlich. Sie nehmen einen hohen Stellenwert hinsichtlich der Qualität der Engineeringergebnisse ein, da Engineeringfehler die Kosten im Anlagenbau überproportional in die Höhe treiben [Sche02]. Erschwerend kommt eine hohe Änderungsrate im Engineering hinzu [RHA02], [MaNa03]. Jede Änderung kann aufgrund der technischen Abhängigkeiten zwischen einzelnen Komponenten weitergehende Anpassungen erforderlich machen. Dadurch steigt der Aufwand für die Detailtätigkeiten, die zusätzlich zur kreativen Lösungsfindung anfallen und somit die Wertschöpfung des Anlagenentwicklers beeinträchtigen.

Zur Steigerung der Produktivität im Engineering besteht die Notwendigkeit, den manuellen Aufwand und die Komplexität der menschlichen Tätigkeiten zu reduzieren. Das vollständige Potenzial der informationstechnischen Unterstützung des Ingenieurs ist hier bei weitem noch

nicht ausgeschöpft und stellt eine vielschichtige und facettenreiche Aufgabenstellung dar [MaNa03]. Die gegenwärtig verfügbare Unterstützung durch CAE-Systeme ist passiver Natur, da zwar Funktionen zur Erstellung von Anlagenmodellen und Engineeringinformationen bereitgestellt werden, aber keine Unterstützung für die genannten Detailtätigkeiten zur Berücksichtigung und Handhabung von technischen Abhängigkeiten zwischen Komponenten angeboten wird. Sie obliegen weiterhin dem Ingenieur und müssen manuell durchgeführt werden. Es besteht daher Handlungsbedarf hinsichtlich rechnergestützter Konzepte und Verfahren zur Gestaltung einer aktiven, flexiblen Unterstützung, welche die spezifischen Merkmale der Arbeitsprozesse und die technischen Abhängigkeiten zwischen Komponenten beim Engineering berücksichtigt. Eine solche Unterstützung stellt einen entscheidenden Kostenfaktor bei der Entwicklung von Automatisierungsanlagen dar.

1.3 Ziel der aktiven Unterstützung von Engineeringtätigkeiten

Die Problematik des hohen manuellen Aufwands beim Engineering bildet den Ausgangspunkt der vorliegenden Arbeit. Die zu entwickelnden Konzepte und Verfahren sollen eine geeignete Unterstützung des Ingenieurs bereitstellen, welche die Produktivität der Arbeitsprozesse steigert und die Qualität der Arbeitsergebnisse sicherstellt. Es soll ein Ansatz für ein Unterstützungssystem aufgezeigt werden, das für die manuellen Detailtätigkeiten des Ingenieurs beim Engineering eine geeignete Hilfestellung anbietet und somit die Erstellung von Anlagenmodellen und Engineeringinformationen effizienter und einfacher gestaltet. Dadurch soll die Verlagerung des menschlichen Arbeitsaufwandes auf kreative, wertschöpfende Tätigkeiten ermöglicht werden. Innerhalb dieser Zielsetzung wird folgenden Aspekten ein besonderer Stellenwert beigemessen:

Aktive Unterstützung

Das in dieser Arbeit entstehende Lösungskonzept soll eine Reduktion des manuellen Aufwands beim Engineering ermöglichen. Dazu muss ein Unterstützungssystem eine aktive Rolle im Rahmen der Arbeitsprozesse des Ingenieurs einnehmen. Die sich bei der Durchführung konkreter Tätigkeiten ergebenden situationsspezifischen Problemstellungen sollen selbstständig erkannt und es sollen adäquate Hilfestellungen oder Lösungsvorschläge angeboten werden.

Handhabung von technischen Abhängigkeiten zwischen Komponenten

Eine wesentliche Ursache für den hohen manuellen Aufwand sind Detailtätigkeiten, die durch die Notwendigkeit zur Berücksichtigung von technischen Abhängigkeiten zwischen den einzelnen Komponenten der Automatisierungsanlage entstehen. Gegenstand der Unterstützung soll daher die selbstständige Ermittlung und Handhabung der technischen Abhängigkeiten sein, die im Rahmen des Engineerings der Automatisierungsanlage anfallen. Abhängigkeiten sollen in konstruktiver Weise gehandhabt werden, dahingehend, dass dem Ingenieur Lösungsmöglichkeiten aufgezeigt werden, welche ein korrektes Zusammenspiel der Komponenten sicherstellen.

Anwendbarkeit für den Ingenieur

Die in dieser Arbeit zu entwickelnden Konzepte sollen besonders die Bedürfnisse des Ingenieurs berücksichtigen, der in kreativer Weise eine Aufgabenstellung lösen muss, seine Arbeitsprozesse und Tätigkeiten flexibel an die aktuelle Projektsituation anpasst und daher nicht im Softwarekorsett eines Unterstützungssystems seine Arbeit verrichten möchte [RHA02]. Die Wirkungsweise des Unterstützungssystems darf daher keine Einschränkung der Flexibilität menschlicher Arbeitsprozesse darstellen. Vielmehr ist eine nahtlose, selbstständige Anpassung der Unterstützung an den durch den Ingenieur vorgegebenen Handlungsrahmen zu gewährleisten. Unterstützende Handlungen sollen interaktiv mit dem Ingenieur abgestimmt und an seine Entscheidungen angepasst werden können, sodass eine vollständige Handlungskontrolle besteht [Rodi02]. Lösungsvorschläge für erkannte Problemstellungen müssen einen direkten Bezug zur aktuellen Tätigkeit des Ingenieurs aufweisen und jederzeit nachvollziehbar sein. Der Ingenieur soll sich nicht mit internen Abläufen des Unterstützungssystems beschäftigen müssen. Diese Anforderung ist insofern bedeutsam, da nur verhältnismäßig wenige Experten im Bereich der Automatisierungstechnik auch Softwareexperten sind, die sich mit den internen Abläufen des Unterstützungssystems auskennen [HoRi99].

1.4 Abgrenzung zu ähnlichen Themenstellungen

Die vorliegende Arbeit grenzt an weitere wichtige Themenbereiche, die jedoch bewusst nicht weiter vertieft werden. Diese verwandten Bereiche werden im Folgenden aufgeführt und auf weiterführende Literatur verwiesen.

- *Anwendungsdomäne der Unterstützung*: Es sollen Konzepte für die Unterstützung des Engineerings im Sinne des technischen Entwurfs und der Auslegung von Automatisierungsanlagen erarbeitet werden, und nicht beispielsweise für das Projektmanagement, Datenintegration oder Concurrent Engineering. Diese Themen werden u.a. in [BeWe03], [Eppl03], [FoHi02], [LGB+05], [Rich03] betrachtet. Auch sollen besonders die Engineeringtätigkeiten zur Erstellung eines Anlagenmodells aus technischen Komponenten betrachtet werden und nicht die Tätigkeiten anderer Entwicklungsprozesse, wie z.B. der objektorientierten Softwareentwicklung oder der CAD-Konstruktion. Hier sei auf [Konr99], [Rude98] verwiesen.

- *Entwicklung von technischen Komponenten*: Eine wichtige Fragestellung beim Engineering ist die Abgrenzung und Unterteilung eines Anwendungsbereiches in einzelne technische Komponenten, aus denen eine Automatisierungsanlage aufgebaut werden kann. Die Identifikation sinnvoller Komponenten ist abhängig von einer Vielzahl von Faktoren, beispielsweise der Anwendungsdomäne, den verfügbaren Technologien und der strategischen Marktausrichtung eines Unternehmens. Diese Fragen werden von u.a. von Domain Engineering Methoden aufgegriffen [CzEi00], [MMYA02]. In dieser Arbeit wird davon ausgegangen, dass die benötigten technischen Komponenten für eine Domäne von Automatisierungsanlagen

bereits zur Verfügung stehen und das Engineering einer konkreten Anlage im Vordergrund steht.

- *Komponentenmodelle*: Für das Engineering von Automatisierungsanlagen stehen eine Reihe von Komponentenmodellen zur Verfügung, mit denen einzelne technische Komponenten oder die Engineeringinformationen einer Automatisierungsanlage beschrieben werden können (z.B. STEP [ISO10303], CAEX [CAEX04, DrMu04], EDDL [PROF05]). Die in dieser Arbeit entwickelten Konzepte basieren auf einem allgemeineren Komponentenbegriff [Göhn98], [EbGö04] und sind daher unabhängig von einem spezifischen Komponentenmodell. Es ist aus diesem Grund auch nicht Ziel der Arbeit, ein neues Komponentenmodell als Grundlage für die Anwendbarkeit der entwickelten Konzepte zu definieren. Vielmehr wird aufgrund der Allgemeinheit des zugrunde liegenden Komponentenbegriffs die Übertragbarkeit und Anwendbarkeit der Konzepte auf bestehende Komponentenmodelle gewährleistet.

1.5 Gliederung der Arbeit

Die vorliegende Arbeit ist in insgesamt 11 Kapitel untergliedert. In Kapitel 2 werden zunächst die erforderlichen Grundlagen des Engineerings von Automatisierungsanlagen zusammengetragen. Dabei wird vor allem auf den modernen Ansatz des komponentenbasierten Engineerings eingegangen, der in dieser Arbeit eine wichtige Rolle spielt. Die Merkmale der Tätigkeiten bei komponentenbasierter Vorgehensweise werden diskutiert und die dabei bestehenden Schwierigkeiten und Herausforderungen aufgezeigt.

Ausgehend von den im vorigen Kapitel aufgezeigten Problemstellungen beim Engineering werden in Kapitel 3 die Anforderungen an eine geeignete Unterstützung des Ingenieurs abgeleitet. Es werden existierende Unterstützungsansätze vorgestellt, welche den derzeitigen Stand von Forschung und Praxis auf diesem Gebiet darstellen. Die Ansätze werden hinsichtlich der aufgestellten Anforderungen bewertet und ihre Vorteile und Defizite diskutiert.

In Kapitel 4 wird das Paradigma der Softwareagenten als ein neuer Ansatz zur Entwicklung aktiver, flexibler Softwaresysteme eingeführt und erläutert. Die agentenorientierten Konzepte, Methoden und Technologien werden vorgestellt und es wird untersucht, welche Eigenschaften und Vorteile bei agentenorientiert entwickelten Softwaresystemen hervortreten.

Anschließend wird in Kapitel 5 ein Ansatz zur ganzheitlichen Lösung der Unterstützungsproblematik beim Engineering erarbeitet. Dazu werden zunächst die Automatisierung der Komponentenintegration als Ansatzpunkt zu Unterstützung des Engineerings ermittelt und die Eignung des agentenorientierten Paradigmas als Basis für ein flexibles Unterstützungskonzept betrachtet. Ausgehend von diesen Erkenntnissen wird das Konzept des agentenunterstützten Engineerings entwickelt und alle Kernelemente des Konzepts vorgestellt. Zudem wird als wichtiger Bestandteil des Konzepts auf die Nutzung von Engineeringinformationen und Wissen des Ingenieurs

eingegangen, durch die eine besonders umfassende Unterstützung mittels Softwareagenten möglich wird.

In Kapitel 6, 7 und 8 wird das Konzept des agentenunterstützten Engineerings ausgebaut und verfeinert. Zur Umsetzung der Kernelemente des agentenunterstützten Engineerings wird in Kapitel 6 ein agentenorientiertes Modell der Unterstützung entwickelt. In Kapitel 7 wird untersucht, wie Engineeringinformationen beschaffen sein müssen, um für eine Unterstützung des Ingenieurs durch Softwareagenten genutzt werden zu können. Weiter wird erläutert, wie das Wissen des Ingenieurs unter Zuhilfenahme wissensbasierter Konzepte erfasst und für die flexible, rechnergestützte Verwertung durch Softwareagenten verfügbar gemacht werden kann. In Kapitel 8 wird dann das funktionale Verhalten des agentenorientierten Unterstützungssystems ausführlich behandelt und die benötigten Fähigkeiten der Softwareagenten für die Unterstützung ermittelt. Es wird gezeigt, wie die Softwareagenten zur Erfüllung ihrer Unterstützungsaufgaben untereinander und mit dem Ingenieur interagieren müssen.

In Kapitel 9 wird die Realisierung des agentenorientierten Unterstützungskonzepts in einem Werkzeugsystem beschrieben und die einzelnen Bestandteile des Werkzeugsystems sowie ihr Zusammenspiel werden vorgestellt.

Kapitel 10 widmet sich der praktischen Anwendung und Erprobung des Unterstützungskonzepts. Zur Veranschaulichung wird als konkretes Beispiel das Engineering einer fördertechnischen Anlage gewählt und anhand von Anwendungsszenarien die unterstützende Wirkungsweise des agentenorientierten Unterstützungssystems untersucht. Anschließend werden die dabei erzielten Ergebnisse bewertet.

Abschließend werden in Kapitel 11 nochmals alle wesentlichen Aspekte dieser Arbeit zusammengefasst. Die gewonnenen Erfahrungen werden zusammengetragen und diskutiert sowie zukünftige Anwendungen und Erweiterungen aufgezeigt.

2 Grundlagen des Engineerings von Automatisierungsanlagen

In diesem Kapitel werden die grundlegenden Eigenschaften und Vorgehensweisen des Engineerings von Automatisierungsanlagen beschrieben. Besonderes Augenmerk wird dabei auf den komponentenbasierten Ansatz gelegt, welcher in der aktuellen industriellen Praxis verankert ist und in der hier vorgestellten Arbeit eine zentrale Rolle spielt. Anschließend werden die Merkmale der Tätigkeiten bei komponentenbasierter Vorgehensweise analysiert und die dabei bestehenden Schwierigkeiten und Herausforderungen diskutiert.

2.1 Automatisierungsanlagen

Automatisierungsanlagen sind technische Systeme, in denen ein technischer Prozess zur Umformung, Verarbeitung und zum Transport von Materie oder Energie abläuft [LaGö99a]. Sie bestehen aus elektrischen, mechanischen, pneumatischen oder hydraulischen Einrichtungen, um die physikalischen Größen des technischen Prozesses mit technischen Mitteln zu erfassen und zu beeinflussen. Beispiele für Automatisierungsanlagen sind verfahrenstechnischen Anlagen, Produktionsanlagen, Paketverteilanlagen, Kraftwerke oder Stahlerzeugungsanlagen. Während in der Produktautomatisierung in der Regel eng umgrenzte Automatisierungsaufgaben zu erfüllen sind (beispielsweise die Automatisierung einer Waschmaschine), handelt es sich bei Automatisierungsanlagen um große industrielle Anlagen, in denen der technische Gesamtprozess meist aus einzelnen Teilprozessen besteht und umfangreiche und komplexe Automatisierungsfunktionen auszuführen sind [LaGö99a].

Eine Automatisierungsanlage durchläuft während ihrer Lebenszeit verschiedene Phasen, die in Abbildung 2.1 dargestellt sind. In der ersten Phase erfolgt die vollständige Planung unter technischen, organisatorischen und wirtschaftlichen Gesichtspunkten. Die technischen Aspekte werden durch das Engineering abgedeckt, dessen Ergebnisse auch für die weiteren Phasen genutzt werden. Anschließend erfolgt die Errichtung und die Inbetriebnahme der Anlage. Ihre Betriebsphase kann bis zu 25 Jahre und mehr betragen und beinhaltet auch die Instandhaltungs- und Modernisierungsmaßnahmen [AcLö05]. In der letzten Phase erfolgt die Demontage.

Abbildung 2.1: Phasen im Lebenszyklus einer Automatisierungsanlage

Im Gegensatz zur Produktautomatisierung sind Automatisierungsanlagen häufig Einmal-Systeme, die sich signifikant voneinander unterscheiden [LBB+05]. Sie können nicht, wie z.B.

Waschmaschinen, in großen Stückzahlen produziert und verkauft werden, sondern erfordern die Entwicklung einer individuellen Lösung entsprechend den spezifischen Anforderungen des Kunden. Im Gegensatz zur Produktautomatisierung können die Engineeringkosten nicht auf die Anzahl der verkauften Produkte verteilt werden und sind somit für die Gesamtkosten sehr entscheidend [LaGö99a]. Auch bezieht sich das eingesetzte Wissen der Ingenieure nicht auf ein bestimmtes Produkt, sondern auf die Frage, wie durch die geschickte Kombination einzelner Bauteile ein optimal funktionierendes technisches Gesamtsystem hergestellt werden kann.

2.2 Engineering von Automatisierungsanlagen

Der Begriff "Engineering" ist im Allgemeinen nicht gleichbedeutend mit "Ingenieurtätigkeit". Vielmehr werden darunter im eingeschränkten Sinn all diejenigen Ingenieurtätigkeiten verstanden, die im Vorfeld der eigentlichen Bauausführung von Automatisierungsanlagen stattfinden (vgl. [LaGö99b]). Das Engineering umfasst somit den vollständigen technischen Entwurf und die Auslegung einer Automatisierungsanlage und bildet die Grundlage für deren Beschaffung, Errichtung und Inbetriebnahme.

Das Engineering ist als kreativer Problemlösungsprozess zu verstehen [Lanz99]. Der Ingenieur hat die Aufgabe, aus den verfügbaren technischen Komponenten die geeigneten auszuwählen, sie zu integrieren und ihr Zusammenspiel so zu gestalten, dass die geforderte Gesamtfunktionalität der Automatisierungsanlage erfüllt wird. Im Fokus steht also das „aktive und zielgerichtete Gestalten“ [GeSc96]. Dabei sind verschiedene Fachdisziplinen (so genannte *Gewerke*) wie Verfahrenstechnik, Maschinenbau, Energietechnik, Automatisierungstechnik etc. beteiligt, deren Wissen und Tätigkeiten im Rahmen eines Projekts integriert werden müssen [AcLö05]. Die Arbeitsprozesse im Engineering zeichnen sich daher durch Multidisziplinarität, komplizierte Entscheidungsprozesse, ein hohes Maß an Kreativität sowie durch große Dynamik aus [MaNa03]. Die einzelnen Tätigkeiten des Ingenieurs erfordern umfassendes Wissen und werden vorwiegend manuell durchgeführt.

Im Lebenszyklus einer Anlage hat das rechnergestützte Engineering (CAE) heute vor allem in der Planungsphase besondere Bedeutung [RHA02]. Ziel ist es, dass alle Elemente der Realisierung einer Anlage in der Planungsphase mittels (möglichst) rechnergestützter Methoden abgesichert werden, sodass der korrekte physische Aufbau der Anlage als gesichert betrachtet werden kann. Die zugrunde liegenden informationstechnischen Methoden und Techniken haben im Laufe der Zeit eine deutliche Wandlung erfahren. Zunächst wurden Systeme eingesetzt, welche auf die Engineeringtätigkeiten einzelner Fachdisziplinen zugeschnitten waren. Da die Informationen der Gewerke aber nicht unabhängig voneinander sind, führte dies zu redundanter Datenhaltung in getrennten Werkzeugen, was mit hohem Aufwand bei der Informationserstellung und -pflege verbunden ist, die Gefahr von Informationsinkonsistenzen birgt und die Wiederverwendung von Engineeringinformationen stark einschränkt [RHA02], [AAF03], [LBB+05].

Bedingt durch die Arbeitsteiligkeit aufgrund der verschiedenen Fachdisziplinen und über die einzelnen Phasen der Projektabwicklung kommt dem durchgängigen Informationsmanagement bezüglich der Engineeringinformationen eine zunehmende Bedeutung zu [AcLö05]. Daneben wird die Möglichkeit zur flexiblen Wiederverwendung von Engineeringinformationen und -lösungen heute als wesentlicher Schritt zur Steigerung der Effizienz der Arbeitsprozesse angesehen [AAF03], [LBB+05]. Diese Forderung beruht auf der Erkenntnis, dass trotz ihrer Individualität ein Großteil der Automatisierungsanlagen aus bereits bekannten Verfahren und Komponenten kombiniert wird [AAF03]. Es entstand die Idee, die einzelnen gewerkspezifischen Informationen unter dem übergeordneten Gesichtspunkt der in der Anlage verbauten technischen Komponenten wie Motoren, Ventile oder auch Teilanlagen zu strukturieren. Dies führte zum komponentenbasierten Ansatz im Engineering, der im Folgenden vorgestellt wird.

2.3 Komponentenbasierter Ansatz im Engineering

Moderne Engineeringmethoden beruhen auf einer komponentenbasierten Vorgehensweise. Grundlegendes Ziel des komponentenbasierten Ansatzes ist die einfachere, schnellere und preiswertere Entwicklung eines Systems durch (Mehrfach-)Verwendung vorgefertigter Komponenten [Balz01]. Auch beim Engineering von Automatisierungsanlagen besteht das Ziel, geeignete Komponenten zu entwickeln und in einer Form bereitzustellen, die einen hohen Grad von Wiederverwendung ermöglicht [AAF03]. Im Folgenden wird, ausgehend von den Begriffsdefinitionen anderer Anwendungsdomänen, der Komponentenbegriff im Engineering präzisiert.

2.3.1 Der Begriff der Komponente im Engineering

Komponente als Reusable Asset

In [Szyp98] wird eine allgemeine Definition des Begriffs der Komponente als „Reusable Asset“ – also als wiederverwendbare Einheit – vorgeschlagen, welche die folgenden Eigenschaften aufweist: Bei einer Komponente handelt sich um eine eigenständige, abgeschlossene Einheit, die nicht teilbar ist und ihre Eigenschaften kapselt. Eine Komponente muss mit anderen Komponenten verknüpfbar sein und daher definierte, dokumentierte Schnittstellen aufweisen. Eine Komponente muss mit dem Ziel der Mehrfachverwendbarkeit entwickelt worden sein. Das bedeutet, eine Komponente muss Produktcharakter besitzen und demzufolge beliebig reproduzierbar sein.

Komponente als Softwarebaustein

Eine mehrfach verwendbare Softwarekomponente muss nach [Göhn98], [EbGö04] eine in sich abgeschlossene funktionale Einheit mit vollständig spezifizierten Schnittstellen bilden. Sie muss gut dokumentiert, qualitativ hochwertig und unverändert mehrfach verwendbar sein. Zusätzlich soll eine Softwarekomponente unabhängig von anderen Softwarekomponenten sein, an eine gegebene Aufgabenstellung anpassbar und bezüglich bestimmter Eigenschaften konfigurierbar

sein. Komponentenbasierte Softwareentwicklung ist die Entwicklung von Softwareapplikationen auf der Basis von Softwarekomponenten, die unverändert verwendet werden. Die einzelnen Komponenten werden dabei ausgewählt, konfiguriert und untereinander verbunden [Gunz03]. In [Göhn98] wird die Notwendigkeit der Unterscheidung zwischen Komponenten auf Modellebene (generische Komponente) und Komponenten als binäre Softwarebausteine (Implementierungsebene) hervorgehoben. Eine generische Komponente wird durch ihre Eigenschaften, Parameter und Schnittstellen beschrieben. Mittels generischer Komponenten aus einer Komponentenbibliothek wird ausgehend von den Systemanforderungen ein komponentenbasiertes *Modell* der zu realisierenden Applikation aufgebaut. Der Weg einer generischen Komponente zur konkreten Komponente führt also im Wesentlichen über eine Instanziierung, einer Anpassung durch Konfigurierung und anschließender Codegenerierung.

Komponentendefinition für das Engineering

Technische Komponente: Im Engineering werden unter Komponenten die verfügbaren technischen Bausteine der Anlagentechnik wie Motoren, Ventile oder Förderelemente verstanden. Derartige *technische Komponenten* verfügen über technische Eigenschaften und Schnittstellen, wobei bestimmte Eigenschaften über Parameter konfigurierbar oder optional sind (beispielsweise die Leistung einer Pumpe oder ein optionaler Sensor eines Förderbandes). Über die Ein- und Ausgänge ihrer Schnittstellen können die technischen Komponenten miteinander verbunden werden (beispielsweise die Ein- und Ausgänge für den stofflichen Fluss oder die elektrischen Anschlüsse einer Pumpe).

Im Unterschied zu einer Softwarekomponente stellt eine technische Komponente einen abgrenzbaren materiellen Teil einer Anlage dar.

Komponentenbeschreibung: Für technische Komponenten werden informationstechnische *Komponentenbeschreibungen* zur Verfügung gestellt, welche alle für das Engineering benötigten Informationen über die technischen Eigenschaften und Schnittstellen (Verfahrenstechnik, Mechanik, Elektronik, Steuerung) der realen Komponenten spezifizieren [Löwe02], [AAF03][1]. Durch eine solche informationstechnische Integration aller Informationen über eine technische Komponente wird eine Standardisierung und damit die Mehrfachverwendung von Komponentenbeschreibungen beim Engineering ermöglicht.

Analog zur Softwaretechnik wird durch die Unterscheidung zwischen technischer Komponente und Komponentenbeschreibung die *Abstraktion* der physikalischen Ebene in eine Modellebene erreicht. Das Engineering zum technischen Entwurf und der Auslegung der Automatisierungsanlage findet somit ausschließlich auf Modellebene statt. Die Spezifikation der Automatisierungsanlage in Form eines Modells hat den Vorteil, dass durch die formalisierte Syntax und Semantik eines Modells die vollständige Umsetzbarkeit bei der Realisierung der Anlage gewährleistet ist.

[1] In der Literatur werden Komponentenbeschreibungen auch als technologische Komponenten [Löwe02] oder als Objekte [AAF03] bezeichnet.

2.3.2 Merkmale des komponentenbasierten Ansatzes

Beim komponentenbasierten Ansatz ist zwischen Komponentenentwicklung (Entwicklung für die Mehrfachwendung) und Systementwicklung unter Zuhilfenahme mehrfach verwendbarer Komponenten (Entwicklung mit Mehrfachverwendung) zu unterscheiden [Szyp98]. In der ersten Phase dieses zweigeteilten Entwicklungszyklus entstehen mehrfach verwendbare Produkte [MMYA02]. Bezogen auf den Anlagenbau handelt es sich hierbei um technische Komponenten. Das Engineering entspricht der zweiten Phase und dient zur Entwicklung einer Automatisierungsanlage, in der diese Komponenten verwendet werden. Die informationstechnische Verknüpfung beider Phasen erfolgt durch die Komponentenbeschreibungen, welche von den Herstellern technischer Komponenten erstellt werden und in Komponentenbibliotheken für das Engineering zur Verfügung stehen. Da technische Komponenten grundsätzlich unabhängig voneinander als einzelne, verschaltbare Bausteine zur Verfügung stehen, besteht die Notwendigkeit der Unabhängigkeit der zugehörigen Komponentenbeschreibungen, um sie flexibel im Rahmen des Engineerings verwenden zu können. Dadurch wird die Mehrfachverwendbarkeit einzelner Komponentenbeschreibungen gewährleistet. Es werden Parameter eingesetzt, um eine mehrfach verwendbare Komponentenbeschreibung bei der Instanziierung anpassen zu können. Sie beeinflussen die Eigenschaften der technischen Komponente entsprechend den spezifischen Erfordernissen der zu entwickelnden Automatisierungsanlage.

Beim komponentenbasierten Engineering wird, ausgehend von den Kundenanforderungen, unter Einsatz der Komponentenbeschreibungen aus der Komponentenbibliothek ein komponentenbasiertes Modell der geplanten Automatisierungsanlage aufgebaut. Dazu sind schrittweise geeignete Komponentenbeschreibungen auszuwählen, zu instanziieren und durch Anpassung und Verbindung zu integrieren[2]. Die *Komponenteninstanzen* innerhalb des Anlagenmodells sind individuelle, informationstechnisch angepasste Ausprägungen von Komponentenbeschreibungen, bei denen die anpassbaren Eigenschaften über Parameter konfiguriert und die über die Ein- und Ausgänge ihrer Schnittstellen verbunden sind. Auf diese Weise werden die Informationen über die reale Anlage jeweils den Instanzen der Komponentenbeschreibungen zugeordnet [AAF03]. Typischerweise existieren mehrere Instanzen einer Komponentenbeschreibung innerhalb eines Anlagenmodells, z.B. mehrere Exemplare eines bestimmten Pumpen- oder Fördertyps. Verbindungen beschreiben den Transport von Stoff, Energie, Impuls und Informationen zwischen den technischen Komponenten. Beispiele für Verbindungen sind Rohrleitungen oder die Übergabe von Fördergütern zwischen zwei Förderbändern. Das erstellte Anlagenmodell ist eine Informationsbasis, in der alle die Anlage betreffenden Engineeringinformationen zusammengefasst sind [Eppl03]. Bei der Beschaffung, Errichtung und Inbetriebnahme der Anlage werden die angepassten Komponentenbeschreibungen zur Anpassung und Verschaltung der in der Anlage ver-

[2] In den weiteren Kapiteln dieser Arbeit wird im Kontext der Beschreibung von menschlichen Tätigkeiten beim Engineering stets der Begriff „Komponente“ synonym für „Komponentenbeschreibung“ verwendet, da die informationstechnischen Vorgänge auf Modellebene aus Sicht des Ingenieurs letztlich dem Ziel dienen, reale Komponenten anzupassen und zu integrieren.

bauten technischen Komponenten verwendet. Abbildung 2.2 gibt einen Überblick über das Engineering von Automatisierungsanlagen nach dem komponentenbasierten Ansatz.

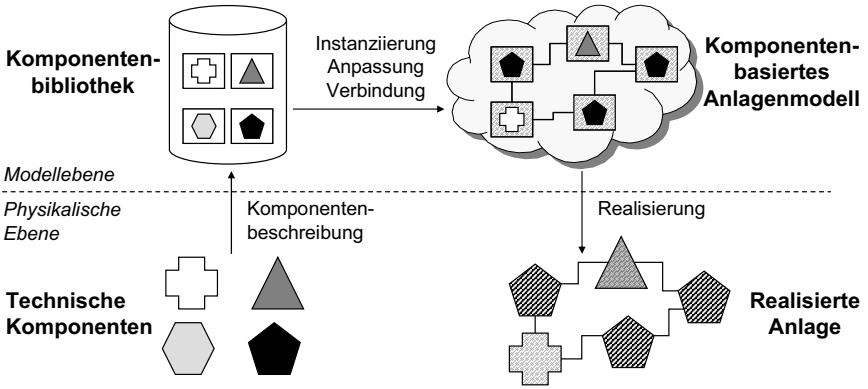


Abbildung 2.2: Prinzip des komponentenbasierten Ansatzes im Engineering

Ein weiteres grundlegendes Merkmal des komponentenbasierten Ansatzes ist die Unabhängigkeit von Komponenten [Göhn98]. Unabhängigkeit besteht in *struktureller* Hinsicht, da eine Komponente von sich aus keine Verbindungen zu externen Komponenten oder Systembestandteilen vorgibt [EbGö04]. Allerdings können von Natur aus funktionale Abhängigkeiten zwischen Komponenten [EbGö04] bestehen, da Komponenten auf die Funktionalität anderer Komponenten angewiesen sein können. So entsteht die Gesamtfunktionalität einer Automatisierungsanlage im Zusammenwirken ihrer technischen Komponenten, die jeweils Teilprozesse des technischen Gesamtprozesses erfassen oder beeinflussen. Daraus resultieren Abhängigkeiten zwischen den technischen Funktionen der Komponenten, die im Rahmen dieser Arbeit als *technische Abhängigkeiten* bezeichnet werden.

Ein Anlagenmodell muss somit nicht nur die kundenspezifischen Anforderungen erfüllen, sondern auch das korrekte Zusammenspiel der verwendeten technischen Komponenten beschreiben. Deshalb sind die technischen Abhängigkeiten zwischen diesen Komponenten bei ihrer *Integration* zur gesamten Anlage zu berücksichtigen. Beispielsweise müssen die Leistung einer Pumpe für die nachfolgende Steigleitung ausreichend ausgelegt oder die Geschwindigkeiten aufeinander folgender Förderbänder einer Förderstrecke aneinander angepasst sein. Dadurch stehen die Konfigurationen einzelner Komponenteninstanzen teilweise in *Wechselwirkung* zueinander.

Die einzelnen Tätigkeiten im Problemlösungsprozesses des Engineerings beziehen sich zunächst auf die Handhabung einzelner Komponenteninstanzen. Aufgrund der technischen Abhängigkeiten zwischen Komponenten sind diese Tätigkeiten aber nicht vollkommen unabhängig voneinander ausführbar. Die bei der Integration entstehenden Wechselwirkungen zu anderen Komponenteninstanzen müssen berücksichtigt werden, um das korrekte Zusammenspiel aller Bestand-

teile der Anlage sicherzustellen. Die Betrachtung der komponentenbasierten Entwicklung als Vorgehen, welches ausschließlich auf dem Zusammenbau von einzelnen, voneinander unabhängigen Komponenten beruht, ist folglich unzureichend und durch praktische Schwierigkeiten belastet [Dujm02]. In [Szyp98] wird dieser Sachverhalt zum Ausdruck gebracht: "It is far too simplistic to assume that components are simply selected from catalogs, thrown together, and magic happens. In reality, the disciplined interplay of components is one of the hardest problems of (software) engineering today."

2.3.3 Vorteile des komponentenbasierten Ansatzes

Analog zum Einsatz von Komponenten bei der Softwareentwicklung verspricht der Einsatz von Komponentenbeschreibungen eine Reihe von Vorteilen beim Engineering:

- *Problemnahe Abstraktion:* Die komponentenorientierte Betrachtung bei Erstellung des Anlagemodells entspricht dem tatsächlichen Aufbau der Anlage aus technischen Komponenten.
- *Datenkonsistenz:* Durch die Integration aller zu einer technischen Komponente gehörenden fachspezifischen Engineeringdaten in einer Komponentenbeschreibung kann die Datenkonsistenz einfacher überprüft und sichergestellt werden.
- *Mehrfachverwendbarkeit:* Einmal erstellte Komponentenbeschreibungen können mehrfach beim Engineering verwendet werden.
- *Kostensenkung:* Durch die mehrfache Verwendung von Komponentenbeschreibungen werden der Aufwand und somit die Kosten für die Erstellung des Anlagenmodells gesenkt.
- *Qualitätssteigerung:* Durch Erstellung des Anlagenmodells mittels Anpassung vorgefertigter Komponentenbeschreibungen werden Engineeringfehler reduziert und die Qualität der Engineeringergebnisse im Vergleich zur Neukonstruktion gesteigert.

2.4 Vorgehensweise im komponentenbasierten Engineering

Ausgehend von den Anforderungen an eine Automatisierungsanlage besteht die Aufgabe des Ingenieurs in der Integration der verfügbaren Komponentenbeschreibungen zu einem Anlagenmodell, sodass ihr Zusammenspiel die benötigte Gesamtfunktionalität der Anlage sicherstellt. Der Problemlösungsprozess des Engineerings besteht aus aufeinander folgenden Elementartätigkeiten, die ausgehend von einer Anforderungsspezifikation einen Lösungszustand erzielen und diesen anhand der Anforderungen überprüfen [Kläg93]. Die Lösungsprozesse und die Tätigkeiten des Ingenieurs bei komponentenbasierter Vorgehensweise dienen der *Erstellung des Anlagenmodells aus einzelnen Komponenten*, der *Erstellung gewerkspezifischer Teilmodelle* und der *Wiederverwendung von Teillösungen*. Sie werden im Folgenden detailliert betrachtet.

2.4.1 Erstellung eines Anlagenmodells aus einzelnen Komponenten

Basistätigkeiten des Ingenieurs

Folgende wesentliche Basistätigkeiten werden bei der Erstellung des komponentenbasierten Anlagenmodells durchgeführt (vgl. [Szyp98]):

1. *Auswählen* von geeigneten Komponenten aus einer Komponentenbibliothek
2. *Instanziieren* der Komponenten
3. *Anpassen* einzelner Komponenteninstanzen durch Konfigurierung ihrer Eigenschaften über die vorgegebenen Parameter
4. *Verbinden* der Eingänge und Ausgänge von Komponenteninstanzen

Mittels dieser Tätigkeiten erstellt der Ingenieur auf Basis seines Wissens und seiner Erfahrung durch eine geeignete Anordnung passender technischer Komponenten eine individuelle Lösung, welche die Anforderungen an die Anlage erfüllt. Die Tätigkeiten dienen zur konstruktiven Umsetzung des gedanklichen Lösungswegs. Hierfür sind im Rahmen der einzelnen Tätigkeiten folgende typische Überlegungen des Ingenieurs erforderlich [Dujm02]:

- Welche Komponenten werden für die gewünschte Anwendung benötigt?
- Wie müssen die einzelnen Komponenten konfiguriert werden?
- Welche Verbindungen müssen zwischen den Komponenten existieren?
- Welche Ein- bzw. Ausgänge müssen für diese Verbindungen verwendet werden?

Auf Basis dieser Überlegungen trifft der Ingenieur konstruktive Entscheidungen und setzt diese um, in dem er jeweils eine (weitere) Komponente auswählt, instanziiert, konfiguriert und mit bereits bestehenden Instanzen verbindet. Auf diese Weise wird schrittweise die technische Struktur der Anlage entworfen, die aus konfigurierten Komponenteninstanzen und Verbindungen besteht. Man spricht von einem „inkrementellen Aufbau“ des Anlagenmodells [Zeid01].

Fehlermöglichkeiten und notwendige Überprüfungen

Bei der Erstellung des Anlagenmodells sind die technischen Abhängigkeiten zwischen den einzelnen Komponenten zu beachten. Ihre Nichtberücksichtigung führt zu einem inkonsistenten, fehlerbehafteten Anlagenmodell. Dabei bestehen folgende mögliche Fehlerursachen [Flei03]:

- Auswahl von Komponenten mit ungeeigneten Eigenschaften
- Falsche Konfigurierung der Komponenteneigenschaften über Komponenten-Parameter
- Falsche oder fehlende Verbindungen von Komponenteninstanzen

Zur Vermeidung von Engineeringfehlern sind bei der Erstellung des Anlagenmodells zusätzlich zur kreativen Problemlösung weitergehende Schritte erforderlich. Um diesen Umstand zu verdeutlichen, wird in [Balz01] die Notwendigkeit der *Überprüfung* aufgeführt, welche parallel zu den anderen Basistätigkeiten erfolgen muss. Typische zusätzliche Überlegungen sind [Dujm02]:

- Welche Komponenten-Kombinationen sind möglich?
- Welche Auswirkungen entstehen durch die Auswahl von Parameterwerten bzw. Optionen?

Die Notwendigkeit der Überprüfung im Rahmen der Basistätigkeiten des Ingenieurs führt zu weiteren *Detailtätigkeiten*, die im folgenden Abschnitt erläutert werden.

Erforderliche Detailtätigkeiten

Detailtätigkeiten dienen dazu, die bei Verbindung und Anpassung von Komponenteninstanzen aufgrund technischer Abhängigkeiten entstehenden Wechselwirkungen zu ermitteln und konstruktiv zu handhaben. Beim Verbinden von Komponenteninstanzen muss vom Ingenieur geprüft werden, ob diese im Rahmen ihrer Verbindungsmöglichkeiten zusammenpassen. Dazu muss er ermitteln, ob *direkte Wechselwirkungen* zwischen ihnen bestehen und ob die Konfigurationen diesbezüglich konsistent zueinander sind. Gegebenenfalls müssen die Konfigurationen der Komponenteninstanzen aneinander angepasst werden. Ist eine geeignete Anpassung nicht möglich, muss eine andere Verbindung gewählt oder möglicherweise eine Komponenteninstanz ausgetauscht werden. Ergänzend zu diesen Überlegungen sind auch die aus der Verbindung und Anpassung der Komponenteninstanzen entstehenden Wechselwirkungen zu weiteren verbundenen Komponenteninstanzen zu berücksichtigen (*indirekte Wechselwirkungen*). Gegebenenfalls müssen dann schrittweise die Konfigurationen weiterer Komponenteninstanzen angepasst werden, um ihr konsistentes Zusammenspiel und somit die gewünschte Gesamtfunktionalität der Anlage zu bewahren. Gleiches gilt für die Änderung der Konfiguration einer Komponenteninstanz zur Anpassung an die projektspezifischen Anforderungen. Abbildung 2.3 zeigt ein Beispiel für mögliche direkte Wechselwirkungen, die durch eine Änderung des Parameters Höhe des Förderbandes (Schritt 1) in erster Folge mit benachbarten Komponenteninstanzen Magazin und Portalkran entstehen können (Schritt 2). Falls dort Anpassungen erforderlich sind (Schritt 3), können in weiterer Folge indirekte Wechselwirkungen mit weiteren jeweils benachbarten Komponenteninstanzen entstehen, die ebenfalls berücksichtigt werden müssen (Schritt 4).

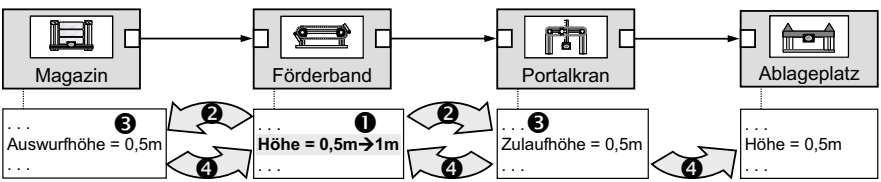


Abbildung 2.3: Beispiel einer Änderung und möglicher Wechselwirkungen

In der Abbildung wird der iterative Charakter der Abläufe deutlich. Beim Anpassen oder Verbinden von Komponenteninstanzen müssen die direkten und indirekten Wechselwirkungen zu weiteren Komponenteninstanzen durch schrittweise Prüfung ermittelt und gegebenenfalls durch entsprechende Anpassung der Konfigurationen berücksichtigt werden. Eine einzelne Basistätigkeit kann somit zu einer Folge weiterer Detailtätigkeiten führen.

2.4.2 Erstellung gewerkspezifischer Teilmodelle

Zusätzlich zum Anlagenmodell sind beim Engineering weitere gewerkspezifische Engineeringinformationen zu erstellen. Für die Integration aller Gewerkinformationen ist entscheidend, inwiefern es eine gewerkübergreifende Struktur in Form einer „führenden Sicht“ (also ein zentrales Gewerk) gibt [LBB+05]. Das komponentenbasierte Anlagenmodell repräsentiert genau diese gewerkübergreifende Struktur unter dem Gesichtspunkt der in der Anlage verbauten technischen Komponenten wie Pumpen, Ventile und Förderbänder (siehe Kapitel 2.2). Es beschreibt die Prozessstruktur, beispielsweise aus Sicht des Gewerks Verfahrenstechnik (verfahrenstechnische Struktur einer Raffinerie) oder Maschinenbau (fördertechnisches Layout einer Logistikanlage). Das komponentenbasierte Anlagenmodell kann somit für die Integration der beteiligten Gewerke genutzt werden [LBB+05]. Es enthält alle zentralen Engineeringinformationen und dient als Ausgangsbasis für die Engineeringtätigkeiten anderer Gewerke [MaNa03], [ScFa05]. Dabei werden weitere gewerkspezifische Engineeringinformationen in Form zusätzlicher Teilmodelle erstellt, welche das Anlagenmodell ergänzen. Dazu gehören z.B. der Entwurf der Automatisierungsstruktur, die Netzwerkplanung, Energieversorgungsplanung und die Einteilung der Bedien- und Notausbereiche. In Abbildung 2.4 ist ein Beispiel für ein gewerkspezifisches Teilmodell Netzwerkplan dargestellt. Es spezifiziert die Zuordnung der Signalanschlüsse der technischen Komponenten des Anlagenmodells zu den E/A-Modulen eines Feldbussystems, um die Übertragung der Prozesssignale vom und zum Automatisierungscomputersystem zu ermöglichen.

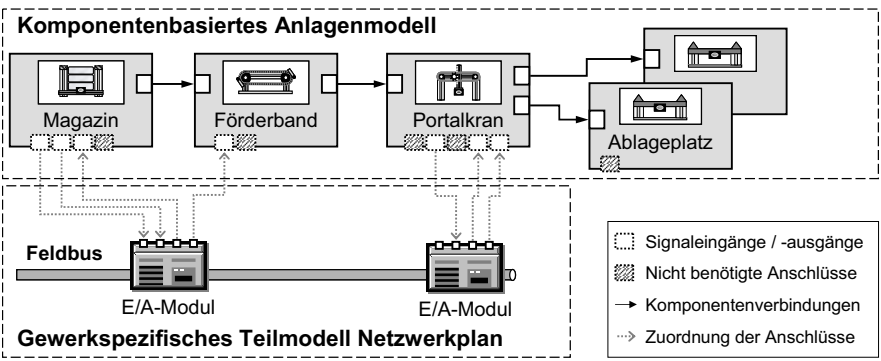


Abbildung 2.4: Beispiel für ein gewerkspezifisches Teilmodell Netzwerkplanung

Zusätzlich kann die Überprüfung und Optimierung des bestehenden Anlagenmodells nach fachspezifischen Gesichtspunkten oder Anforderungen erfolgen. Ein Beispiel hierfür ist die Überprüfung eines fördertechnischen Anlagenmodells auf Durchsatzengpässe und die Durchführung entsprechender Anpassungen.

Tätigkeiten des Ingenieurs

Obwohl sich die erforderlichen Tätigkeiten bei der Erstellung gewerkspezifischer Teilmodelle in fachlicher Hinsicht unterscheiden, können sie zu grundlegenden Typen von Tätigkeiten verallgemeinert werden. Die bestehenden Komponenteninstanzen des Anlagenmodells werden unter fachlichen Aspekten betrachtet und daraus zusätzliche Engineeringinformationen abgeleitet.

- *Erfassen*, *Auswählen*, *Gruppieren* oder *Zuordnen* von Komponenteninstanzen unter gewerkspezifischen Gesichtspunkten
- *Erstellen* zusätzlicher gewerkspezifischer Engineeringinformationen
- *Verbinden* von Komponenteninstanzen unter gewerkspezifischen Gesichtspunkten
- *Anpassen* von Komponenteninstanzen an gewerkspezifische Anforderungen

Die Tätigkeiten des Ingenieurs sollen anhand der Netzwerkplanung verdeutlicht werden. Dabei sind zunächst diejenigen Komponenteninstanzen des Anlagenmodells auszuwählen, welche Signalanschlüsse aufweisen. Weiter ist die Anzahl und die Art (analog, digital, impulsförmig) der benötigten Signalanschlüsse jeder Komponenteninstanz zu ermitteln. Dann erfolgt die Zuordnung der Anschlussleitungen unter Beachtung der fachspezifischen Anforderungen, wie der Berücksichtigung zusammengehöriger Signale, der maximalen Länge der Signalleitungen oder der möglichst optimalen Ausnutzung der verfügbaren Signale der E/A-Module. Nach Erstellung des Teilmodells Netzwerkplanung werden daraus zusätzliche Engineeringinformationen wie Stückliste, Verkabelungsliste und Adressliste abgeleitet.

Fehlermöglichkeiten, notwendige Überprüfungen und Detailtätigkeiten

Bei der Erstellung gewerkspezifischer Teilmodelle bestehen folgende Fehlermöglichkeiten:

- Falsche oder fehlende Berücksichtigung der Informationen aus dem Anlagenmodell
- Falsche oder fehlende Berücksichtigung von gewerkspezifischen Anforderungen

Bei der Instanziierung, Anpassung und Verbindung von Komponenteninstanzen müssen daher die Informationen gewerkspezifischer Teilmodelle überprüft und gegebenenfalls nachgeführt werden, um Inkonsistenzen zwischen dem komponentenbasierten Anlagenmodell und gewerkspezifischen Teilmodellen zu vermeiden. Ebenso kann die Anpassung einer Komponenteninstanz unter einem gewerkspezifischen Gesichtspunkt aufgrund technischer Abhängigkeiten zu Wechselwirkungen mit anderen Komponenteninstanzen im Anlagenmodell führen (siehe Kapitel 2.4.1). So müssen neben den in Kapitel 2.4.1 genannten Detailtätigkeiten zur Handhabung von Wechselwirkungen als weitere Detailtätigkeiten auch die Überprüfung der Auswirkungen auf gewerkspezifische Teilmodelle und gegebenenfalls entsprechende Anpassungen erfolgen.

2.4.3 Wiederverwendung von Teillösungen

Um die hohe Komplexität von Anlagenmodellen besser beherrschen zu können, werden sie für eine übersichtlichere Darstellung zusätzlich in hierarchische Ebenen strukturiert. Als Strukturierungsmittel dienen *Baugruppen*, die einen abgegrenzten Teil der Automatisierungsanlage repräsentieren. Eine Baugruppe entsteht durch das Zusammenfassen einer Menge von verbundenen Komponenteninstanzen und durch die Definition von Ein- und Ausgangsschnittstellen. Dazu werden alle Ein- und Ausgänge von Komponenteninstanzen, die außerhalb der Baugruppe sichtbar sein sollen, als Schnittstelle der Baugruppe zusammengefasst [Flei03]. Abbildung 2.5 zeigt ein Beispiel für eine Baugruppe, ihre Schnittstellen und ihren internen Aufbau.

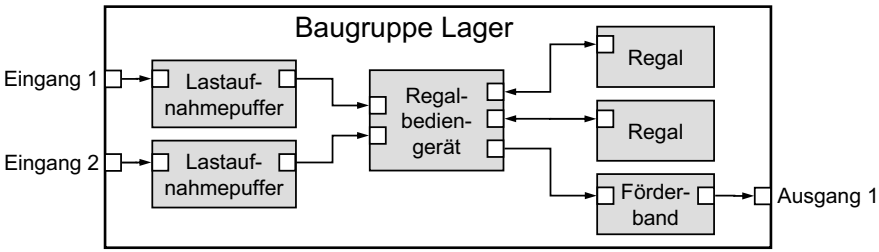


Abbildung 2.5: Beispiel für eine Baugruppe

An ihrer Schnittstelle unterscheidet sich eine Baugruppe nicht von einer einzelnen Komponente. Der interne Aufbau aus Komponenteninstanzen ist durch die Baugruppe gekapselt. Verbindungen zwischen Baugruppen entsprechen damit der Verbindung der jeweiligen Komponenteninstanzen an ihren Ein- bzw. Ausgangsschnittstellen. Zusätzlich können einer Baugruppe übergeordnete Eigenschaften zugewiesen werden, die ihre Funktion genauer charakterisieren [AAF03].

Eine Baugruppe stellt einen applikationsspezifischen, jedoch trotzdem mehrfach verwendbaren Baustein dar [Gunz03]. Handelt es sich um eine häufig (in gleicher oder ähnlicher Form) benötigte Kombination von Komponenten für ein bestimmtes Teilproblem bei Automatisierungsanlagen, kann die Baugruppe als *Teillösung* in einer Bibliothek abgelegt und auf diese Weise (evtl. mit wenigen Änderungen) mehrfach verwendet werden. Eine Teillösung ist die informationstechnische Beschreibung einer Baugruppe. Die Bereitstellung geeigneter Teillösungen, die in möglichst vielen Projekten genutzt werden können, verbessert die Effizienz und Qualität im Engineering [LBB+05]. Bei der Wiederverwendung von Teillösungen müssen diese gegebenenfalls noch an projektspezifische Anforderungen angepasst werden [LaGö99b]. Dies erfordert die geeignete Verbindung und Anpassung ihrer internen Komponenteninstanzen [AAF03].

Tätigkeiten des Ingenieurs

Die Wiederverwendung von Baugruppen als Teillösungen entspricht der Wiederverwendung auf einer höheren Hierarchieebene des Anlagenmodells. Eine Baugruppe verhält sich nach außen

hin wie eine Komponente und kann in den höheren Hierarchieebenen beliebig oft instanziiert werden [Gunz03]. Die erforderlichen Tätigkeiten des Ingenieurs entsprechen daher zunächst grundsätzlich den in Kapitel 2.4.1 genannten Basistätigkeiten. Allerdings repräsentiert eine Baugruppe ein abstraktes Modellkonstrukt, das physikalisch nur in Form seiner Bestandteile, den technischen Komponenten, existiert. Ihre konkrete Ausprägung definiert sich daher aus der Ausprägung ihrer internen Struktur in Form einzelner Komponenteninstanzen, deren Konfigurationen und Verbindungen. Infolgedessen muss bei der Handhabung einer Baugruppe als wiederverwendete Teillösung zusätzlich ihre interne Struktur berücksichtigt werden:

- *Instanziieren* einer wiederverwendeten Teillösung als Baugruppe inklusive aller Komponenten ihrer internen Struktur
- *Verbinden* von Baugruppen durch Verbinden der jeweiligen Komponenteninstanzen an ihren Ein- bzw. Ausgangsschnittstellen
- *Anpassen* einer Baugruppe an projektspezifische Anforderungen durch Anpassen der internen Struktur: Anpassen oder Verbinden von internen Komponenteninstanzen, Hinzufügen oder Entfernen von Komponenteninstanzen

Wurde beispielsweise die Baugruppe Lager aus Abbildung 2.5 als mehrfach verwendbare Teillösung in einer Bibliothek abgelegt, kann sie nun in einem weiteren Projekt instanziiert werden. Wird im aktuellen Projekt eine höhere Kapazität des Lagers benötigt, muss geprüft werden, ob hierfür eine Anpassung der internen Struktur erforderlich ist, z.B. durch Instanziierung weiterer Regal-Komponenten und ihrer Verbindung mit dem Regalbediengerät. Ist im aktuellen Projekt nur eine Zulaufförderstrecke für das Lager vorgesehen, so kann die entsprechende Lastaufnahmepuffer-Komponente am Eingang des Lagers entfallen.

Fehlermöglichkeiten, notwendige Überprüfungen und Detailtätigkeiten

Da die Tätigkeiten des Ingenieurs zu Handhabung einer Baugruppe und zur Anpassung ihrer internen Struktur den Basistätigkeiten aus Kapitel 2.4.1 entsprechen, gelten hier ebenso die erforderlichen typischen Überlegungen und die Fehlermöglichkeiten. Sie entfallen für diejenigen Bestandteile, bei denen keine Anpassungen erforderlich sind. Gleiches gilt für die aus Basistätigkeiten und Überlegungen resultierenden Detailtätigkeiten: Die durch Verbindung und Konfiguration von Komponenteninstanzen entstehenden Wechselwirkungen sind durch schrittweise Prüfung zu ermitteln und mittels geeigneter Anpassungen konstruktiv zu handhaben.

2.5 Bedeutung des Wissens für Engineeringtätigkeiten

In den vorangegangen Abschnitten wurde deutlich, dass zusätzlich zu den Basistätigkeiten des komponentenbasierten Engineerings weitergehende Überlegungen und Detailtätigkeiten erfor-

derlich sind, um entstehende Wechselwirkungen zu erkennen und zu handhaben. Die grundlegenden Schritte sind hier nochmals zusammengefasst:

- *Erkennen von Wechselwirkungen*: Der Ingenieur muss die technischen Abhängigkeiten kennen, um Wechselwirkungen zwischen Komponenteninstanzen erkennen zu können.

- *Handhaben von Wechselwirkungen*: Der Ingenieur muss die Konfiguration von Komponenteninstanzen prüfen und gegebenenfalls anpassen, um ihr korrektes Zusammenspiel zu gewährleisten. Dabei entsteht eine hohe Dynamik, da diese Anpassungen weitere indirekte Wechselwirkungen zur Folge haben können, die ihrerseits erkannt und gehandhabt werden müssen.

Bei jeder einzelnen Tätigkeit werden Engineeringinformationen erstellt, ergänzt oder verändert [Balz01]. Zusätzlich benötigt der Ingenieur bei Durchführung seiner Tätigkeiten und Überlegungen auch *Wissen* [Balz01]. Abbildung 2.6 verdeutlicht den Einfluss des Wissens auf die Tätigkeiten des Ingenieurs, welcher in der folgenden Aufzählung genauer untersucht wird.

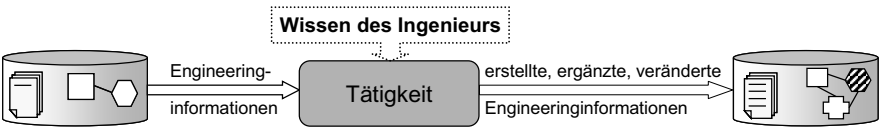


Abbildung 2.6: Wissen als wichtige Einflussgröße für die Tätigkeiten beim Engineering

- Für die richtige Verwendung einer Komponente muss der Ingenieur ihre Eigenschaften, ihr Verhalten und die Bedingungen für ihren korrekten Einsatz kennen [Luce02]. Dazu gehören alle Informationen über ihre Parameter und Schnittstellen und das komponentenspezifische Wissen über die durch technische Abhängigkeiten bestimmten Kombinations- und Anpassungsmöglichkeiten einer Komponente.

- Bei der Erstellung gewerkspezifischer Teilmodelle werden Informationen über die bestehenden Komponenteninstanzen des Anlagenmodells, ihre Parameterwerte und Verbindungen genutzt. Zudem wird komponentenübergreifendes Wissen eingesetzt, um die Auswahl, Gruppierung oder Zuordnung, Verbindung und Anpassung von Komponenteninstanzen unter gewerkspezifischen Gesichtspunkten vornehmen zu können.

- Bei der Handhabung von Baugruppen sind Informationen über ihre Eigenschaften und Schnittstellen sowie über ihren internen Aufbau aus Komponenteninstanzen erforderlich [AAF03]. Zusätzlich wird Wissen über die Anpassungsmöglichkeiten des internen Aufbaus und über technische Abhängigkeiten der internen Komponenten benötigt.

Der Ingenieur verknüpft bei seinen Tätigkeiten Informationen und Wissen situationsspezifisch im Kontext der jeweils betrachteten Komponentenkonfigurationen und -verbindungen, ermittelt dabei Wechselwirkungen und leitet daraus notwendige konstruktive Maßnahmen ab.

2.6 Schwierigkeiten und Herausforderungen

Die in den vorangehenden Abschnitten erläuterten Detailtätigkeiten zur Erkennung und Handhabung von Wechselwirkungen müssen begleitend zur kreativen Problemlösung – dem Finden der optimalen, kundenspezifischen Lösung – durchgeführt werden. Sie haben teilweise Routinecharakter und einen geringeren kreativen Anteil. Sie sind aber mit hohem manuellem Aufwand verbunden, wodurch die Produktivität und somit die Wertschöpfung des Anlagenentwicklers eingeschränkt wird. Außerdem nehmen sie einen hohen Stellenwert hinsichtlich der Qualität der Engineeringergebnisse ein, da eine fehlerhafte Durchführung die Funktion der Anlage stark beeinträchtigen kann. Der hohe manuelle Aufwand resultiert aus zwei wesentlichen Ursachen:

1. Informationsverlust zwischen Komponentenentwicklung und Engineering

Komponentenentwicklung und Engineering sind meist zwei getrennte Prozesse, die von unterschiedlichen Personen (häufig in unterschiedlichen Unternehmen oder Abteilungen) durchgeführt werden [LBB+05]. Die Konsequenz aus dieser Tatsache ist, dass der Ingenieur beim Engineering nicht über dieselben Informationen und dasselbe Wissen wie der Komponentenentwickler verfügt. Insbesondere das Wissen über die technischen Abhängigkeiten könnte bei der Entwicklung einer technische Komponente aus den grundlegenden Überlegungen über ihren Verwendungskontext zwar ermittelt werden, dem Ingenieur stehen für das Engineering aber nur die Informationen über die technischen Parameter und Schnittstellen einer Komponente in Form der Komponentenbeschreibung zur Verfügung. Diese Informationen sind für die korrekte Verwendung einer Komponente jedoch nicht ausreichend [Luce02]. Daher muss der Ingenieur das zusätzlich benötigte Wissen über technische Abhängigkeiten von sich aus ermitteln, z.B. durch Zurückgreifen auf eigene Fachkenntnisse oder durch Lesen zusätzlicher Dokumentation (falls diese verfügbar ist) und im individuellen Kontext der geplanten Anlage anwenden. Ähnlich verhält es sich bei der Wiederverwendung von Baugruppen. Dem Ingenieur stehen zwar die strukturellen Informationen über den internen Aufbau der Baugruppe zur Verfügung, nicht aber das benötigte Wissen über mögliche Anpassungsvarianten für unterschiedliche Einsatzbereiche [AAF03]. In jedem Fall ist die Akquisition und Anwendung des zusätzlich benötigten Wissens zur sachgemäßen Verwendung von Komponenten ein manueller Vorgang. Es besteht also ein Informationsverlust zwischen Komponentenentwicklung und Engineering. Die Vollständigkeit der Beschreibung von Komponenten hinsichtlich ihrer Verwendung ist nicht gegeben, womit sich in der Praxis die Produktivität durch manuelle Vorgänge verringert und die Fehlerträchtigkeit (z.B. inkompatible Komponentenkonfigurationen oder Schnittstellenprobleme) erhöht.

2. Häufige Änderungen und hohe Dynamik der Tätigkeiten

Häufige Änderungen aufgrund veränderter Anforderungen oder unvorhergesehener Revisionen zur Behebung eines erkannten Problems oder zur Umsetzung einer Verbesserungsmöglichkeit sind im Engineering die Regel [RHA02], [MaNa03]. Jede Änderung ist mit Anpassungen von Komponentenkonfigurationen oder -verbindungen verbunden. Durch die Möglichkeit von

Wechselwirkungen zwischen Komponenteninstanzen und mit gewerkspezifischen Teilmodellen werden die entsprechenden Detailtätigkeiten erforderlich, was zu einer hohen Dynamik von Prüfungen und Anpassungen führt. In [LaGö99b] wird dieser Sachverhalt verdeutlicht: „Die Schwierigkeit bei komplexen Systemen besteht nun darin, bei einer Änderung an einer Stelle zu erkennen, an welchen anderen Stellen diese Änderung sich auswirkt und diese Auswirkungen dann entsprechend zu berücksichtigen."

Folglich ist die bloße Bereitstellung von Komponentenbeschreibungen für ein erfolgreiches komponentenbasiertes Engineering von Automatisierungsanlagen nicht ausreichend. Zur Steigerung von Effizienz und Produktivität beim Engineering bei gleichzeitiger Sicherstellung der Qualität der Engineeringergebnisse ist eine umfassende Unterstützung des Ingenieurs erforderlich, welche den manuellen Aufwand bei den Detailtätigkeiten reduziert.

In diesem Kapitel wurden die für das Verständnis der vorliegenden Arbeit relevanten Grundlagen und Begriffe vorgestellt. Der Schwerpunkt lag dabei auf der komponentenbasierten Vorgehensweise im Engineering. Dabei wurde festgestellt, dass technische Abhängigkeiten zwischen Komponenten bestehen, die bei der Erstellung des Anlagenmodells berücksichtigt werden müssen. Dadurch werden zusätzlich zu den Basistätigkeiten der komponentenbasierten Vorgehensweise weitere Detailtätigkeiten notwendig, um die entstehenden Wechselwirkungen zwischen Komponenteninstanzen zu erkennen und zu handhaben. Diese Detailtätigkeiten sind mit hohem manuellem Aufwand verbunden. Als Ursachen wurden der Informationsverlust zwischen Komponentenentwicklung und Engineering sowie die Dynamik von Änderungen und Wechselwirkungen identifiziert. Zur Aufwandsreduzierung besteht die Notwendigkeit einer geeigneten Unterstützung des Ingenieurs bei seinen Tätigkeiten. Ausgehend von diesen Grundlagen und der in Kapitel 1.3 formulierten Zielsetzung der Arbeit wird im nächsten Kapitel der Stand der Technik bei der Unterstützung des Engineerings analysiert und bewertet.

3 Bestehende Ansätze zur Unterstützung des Engineerings von Automatisierungsanlagen

Im vorangehenden Kapitel wurde dargelegt, dass das Engineering von Automatisierungsanlagen auf der Basis von Komponenten erfolgt und dass eine geeignete Unterstützung des Ingenieurs bei den dabei anfallenden Tätigkeiten notwendig ist. In diesem Kapitel werden ausgehend von einer Betrachtung der generellen Möglichkeiten der Rechnerunterstützung zunächst konkrete Anforderungen für die Unterstützung des komponentenbasierten Engineerings definiert. Im Anschluss daran werden verschiedene Ansätze der Unterstützung vorgestellt. Die Ansätze stellen den derzeitigen Stand der Technik dar. Abschließend erfolgt auf der Basis der Anforderungen eine Bewertung der Ansätze hinsichtlich ihrer Eignung und eine Diskussion der Ergebnisse.

3.1 Generelle Möglichkeiten der Rechnerunterstützung

Als ein wesentliches Merkmal des Engineerings von Automatisierungsanlagen wurde in Kapitel 2 die kreative Lösungsfindung für ein gegebenes Problem identifiziert: das Entwickeln einer zweckmäßigen Kombination aus geeigneten technischen Komponenten, welche die Kundenanforderungen erfüllt und gleichzeitig das korrekte Zusammenspiel der Komponenten sicherstellt. Grundsätzlich kann eine Steigerung von Effizienz, Produktivität und Qualität durch eine möglichst umfassende Rechnerunterstützung erreicht werden [MaNa03]. Die Versuche der Forschung in den 80er Jahren haben jedoch zu der Erkenntnis geführt, dass Entwicklungsprozesse nicht *vollständig automatisierbar* sind [Herz92], [Bind94]. Die kreativen, schöpferischen Tätigkeiten und Entscheidungsprozesse sind dem Menschen vorbehalten [Fuch00]. Dieser Umstand kommt in folgendem Zitat aus [Booc94a] zum Ausdruck: „A tool cannot tell us that we ought to invent a new class so as to simplify our class structure … good design always comes from good designers, not from tools.“ Die Möglichkeiten der Rechnerunterstützung liegen vielmehr darin, den Ingenieur bei diesen Aufgaben zu entlasten, die menschlichen Fähigkeiten zu ergänzen und menschliche Schwächen auszugleichen, wie z.B. das Vergessen von Einzelheiten, die unsystematische Vorgehensweise, die mangelnde Übersicht über eine Vielzahl von Zusammenhängen, Fakten und Werten, die sich zudem noch laufend ändern [LaGö99b]. Nach [LaGö99b] können mittels einer Rechnerunterstützung u.a. folgende allgemeine Ziele angestrebt werden: Vermeidung von Spezifikations- und Entwurfsfehlern, Prüfungen zum Erkennen von Fehlern, Entlastung von lästigen Routinearbeiten und Beherrschbarkeit der Komplexität. Im Rahmen der vorliegenden Arbeit sollen diese Ziele durch die Unterstützung der im Rahmen des kreativen Problemlösungsprozesses im Engineering anfallenden Detailtätigkeiten verwirklicht werden, welche zur Sicherstellung des korrekten Zusammenspiels der technischen Komponenten einer Automatisierungsanlage erforderlich sind.

3.2 Anforderungen an die Unterstützung des Engineerings

Ausgehend von der in Kapitel 1 formulierten Zielsetzung der Arbeit und den in Kapitel 2 analysierten Merkmalen, Schwierigkeiten und Herausforderungen beim Engineering stellt sich nun die Frage, welche Form der Unterstützung für eine Entlastung des Ingenieurs von Detailtätigkeiten bei der Handhabung von Komponenten und ihren Abhängigkeiten benötigt wird. Dabei sind Vorgaben aus zweierlei Sichten zu beachten:

- *Vorgaben aus technischer Sicht:* Die Unterstützung muss die technischen Randbedingungen von Automatisierungsanlagen berücksichtigen [LaGö99b]. Dazu gehören insbesondere die Eigenschaften von technischen Komponenten und ihre technischen Abhängigkeiten.
- *Vorgaben aus Sicht des Anwenders:* Die Unterstützung muss sich an der menschlichen Vorgehensweise bei Problemlösungsprozessen im Engineering orientieren [MaNa03].

Daraus ergeben sich folgende konkrete Anforderungen an die Unterstützung des Engineerings:

Aktive Unterstützung bei der Integration von Komponenten

Ein Unterstützungssystem soll eine aktive Rolle im Rahmen der Arbeitsprozesse des Ingenieurs einnehmen. Die sich bei den Tätigkeiten des komponentenbasierten Engineerings ergebenden situationsspezifischen Problemstellungen sollen selbstständig erkannt und alle bestehenden Zusammenhänge ermittelt werden. Der Ingenieur bleibt zu keinem Zeitpunkt sich selbst überlassen und muss sich keine Gedanken machen, ob wichtige Prüfungen vergessen wurden. Wie in Kapitel 2 deutlich wurde, ist es dazu insbesondere erforderlich, aktiv alle Wechselwirkungen zwischen Komponenteninstanzen zu ermitteln und zu prüfen, die bei der Erstellung des komponentenbasierten Anlagenmodells aufgrund von technischen Abhängigkeiten entstehen.

Konstruktive Unterstützung

Die Unterstützung soll nicht nur situationsspezifische Problemstellungen erkennen und darauf hinweisen, sondern dem Ingenieur zielgerichtet konstruktive Hilfestellungen oder Vorschläge zur Problemlösung anbieten. Die Lösungsvorschläge sollen dabei nicht allgemeiner Art sein, sondern konkrete Lösungen für die aktuelle Entwicklungssituation darstellen. Sie sollen sich auf die konstruktive Handhabung von Wechselwirkungen zur Sicherstellung des konsistenten Zusammenspiels der technischen Komponenten beziehen. Es sollen situationsspezifisch Möglichkeiten aufgezeigt werden, welche Konfigurationen oder Verbindungen von Komponenteninstanzen zur Berücksichtigung entstandener Wechselwirkungen angepasst werden müssen. Weiter soll eine selbstständige Umsetzung konstruktiver Lösungsvorschläge möglich sein.

Interaktive begleitende Problemlösung

Wie in Kapitel 2 deutlich wurde, ist das Engineering für den Ingenieur ein iterativer Prozess, der schrittweise zu einem individuellen Anlagenmodell führt: Neue Komponenteninstanzen werden

hinzugefügt, konfiguriert und verbunden, bestehende Konfigurationen oder Verbindungen verändert. Die Problemlösung soll daher begleitend zu den Tätigkeiten des Ingenieurs erfolgen. Der Ingenieur soll mittels der Unterstützung in kleinen, überschaubaren und nachvollziehbaren Schritten zu einer Lösung gelangen. Zudem ist bei der Lösungsumsetzung eine direkte Kooperation mit dem Ingenieur erforderlich, da die aus einer spezifischen Problemstellung abgeleiteten unterstützenden Aktivitäten nicht zwangsläufig gewünscht oder eindeutig sind. Beispielsweise führen Anpassungen unter Umständen zu Fortpflanzungseffekten in Form weiterer Anpassungen, die nicht unbedingt der menschlichen Absicht entsprechen. Daneben gibt es möglicherweise Lösungsvarianten, welche unterschiedliche Konsequenzen haben. Können Lösungsvarianten durch das Unterstützungssystem nicht selbstständig qualifiziert werden, sind fehlende Informationen und Entscheidungen situationsspezifisch mit dem Ingenieur abzustimmen.

Berücksichtigung der Änderungsdynamik

Ein Kernproblem, das vielen Schwierigkeiten in Automatisierungsprojekten zugrunde liegt, besteht darin, die Auswirkungen von Änderungen in den Griff zu bekommen [LaGö99b]. Anpassungen an einer Stelle haben aufgrund von direkten und indirekten Wechselwirkungen oft Auswirkungen auf andere Teile des Anlagenmodells, die entsprechend erkannt und berücksichtigt werden müssen. Daher besteht die Anforderung, lokale Änderungen und Wechselwirkungen komponentenübergreifend zu einer passenden, vollständigen Lösung zu verknüpfen und dabei bestehende bzw. neu entstehende Wechselwirkungen dynamisch zu handhaben. Die Lösungsvorschläge müssen die Systematik des schrittweisen menschlichen Vorgehens bei der Handhabung von Wechselwirkungen berücksichtigen, um ihre Verständlichkeit zu gewährleisten. Zudem kann es sinnvoll sein, die Auswirkungen einer Änderung zunächst nur in einem abgegrenzten Bereich zu handhaben, bevor weitergehende Wechselwirkungen ermittelt werden, um die Komplexität möglicher Wechselwirkungen für den Ingenieur überschaubar zu halten.

Gewährleistung der Flexibilität menschlicher Arbeitsprozesse

Das Unterstützungssystem darf keine Einschränkung der Flexibilität menschlicher Arbeitsprozesse darstellen. Eine detaillierte Arbeitsplanung mit engen Vorgaben ist im Engineering nicht erwünscht, weil sie die Kreativität der Ingenieure mit offensichtlich negativen Folgen für die Engineeringergebnisse behindern würde [MaNa03]. Vielmehr ist eine nahtlose, selbstständige Anpassung der Unterstützung an den durch den Ingenieur vorgegebenen Handlungsrahmen zu ermöglichen. Der Ingenieur soll die Unterstützung als Hilfe, nicht als Zwang empfinden. Insbesondere ist die uneingeschränkte Wahl- und Verwendungsmöglichkeit von technischen Komponenten zu gewährleisten. Manuelle Eingriffe in Unterstützungs- bzw. Lösungsprozesse sind jederzeit zu ermöglichen und zu berücksichtigen. Aus Benutzersicht handelt es sich daher bei der Unterstützung um eine „ad hoc“-Reaktion [Konr99]. Folglich muss das Unterstützungssystem in der Lage sein, auf beliebige Entwicklungssituationen zu reagieren, d.h. auch auf Situationen, auf die es nicht explizit vorbereitet wurde.

Im Folgenden werden existierende Ansätze zur Unterstützung des Engineerings untersucht. Nicht alle Ansätze wurden in der Praxis für das Engineering von Automatisierungsanlagen angewendet, sondern manche auch innerhalb anderer Anwendungsdomänen. Sie sind für die vorliegende Arbeit dennoch von Interesse, da eine grundsätzliche Übertragbarkeit auf das Engineering besteht.

3.3 Unterstützung durch Computer Aided Engineering Systeme

Computer Aided Engineering (CAE) fasst alle Möglichkeiten der Computerunterstützung von Arbeitsprozessen der Ingenieure zusammen. Im Rahmen dieser Arbeit werden CAE-Systeme betrachtet, die als Entwicklungsumgebung im Engineering eingesetzt werden. Eine Entwicklungsumgebung stellt alle benötigten Hilfsmittel für die Entwicklung zur Verfügung, bestehend aus Spezifikationssprachen zur Beschreibung von Entwicklungssachverhalten, aus Datenbanksystemen zur Speicherung dieser Sachverhalte sowie aus Softwarewerkzeugen zur Unterstützung der Entwicklungstätigkeiten [LaGö99b]. CAE-Systeme für das Engineering stellen im Allgemeinen folgende Funktionen bereit [LaGö99b]:

- Grafische Editoren zur Erstellung des Anlagenmodells und zur Spezifikation der für den Anlagenaufbau benötigten Engineeringinformationen (z.B. R&I-Fließbilder, Anlagenlayout, Gerätelisten, Montagepläne, Schaltpläne usw.)
- Gewährleistung der Aktualität und Konsistenz aller Informationen durch Generierung aller Ausgaben (Listen, Dokumente usw.) aus einer Projektdatenbank
- Konfigurationsmanagement für alle Komponenten der Anlage und die Möglichkeit zur Rückdokumentation

Heutige CAE-Systeme für das Engineering unterstützen die in Kapitel 2 beschriebene komponentenbasierte Vorgehensweise. Sie stellen Komponentenbeschreibungen für eine Vielzahl technischer Komponenten in Bibliotheken zur Verfügung, ermöglichen den komponentenbasierten Aufbau des Anlagenmodells, seine hierarchische Strukturierung und die Erstellung gewerkspezifischer Teilmodelle. Beispiele sind ComosPT der Firma Innotec (siehe Abbildung 3.1), SmartPlant der Firma Intergraph und promis engine/sigraph CAE der Fa. TCS.

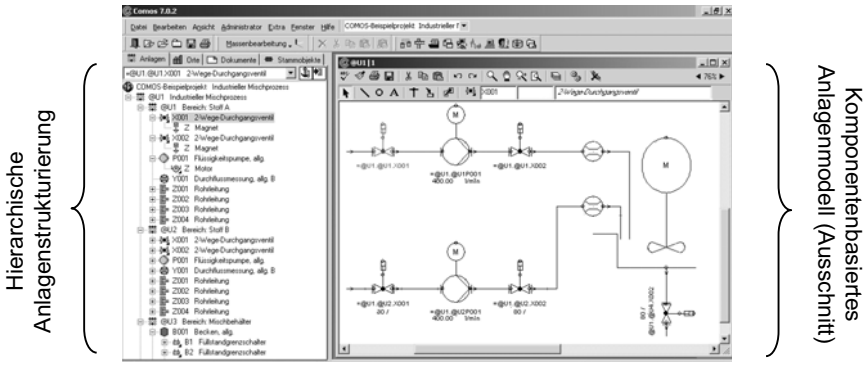


Abbildung 3.1: Benutzungsoberfläche des Engineering Systems Comos PT

Eine Übersicht über verschiedene CAE-Systeme und ihre Funktionen findet sich in [RHA02].

Bewertung

CAE-Systeme für komponentenbasiertes Engineering ermöglichen die flexible Gestaltung des Anlagenmodells aus einzelnen Komponentenbeschreibungen und die Erstellung aller benötigten Engineeringinformationen. Sie bieten vielfältige Funktionen zur Unterstützung der Tätigkeiten, wie z.B. Modellnavigation oder die Wiederverwendung von Teillösungen [INNO03]. Weiter unterstützen sie bereits beim Zusammensetzen der Komponenten während des Engineerings die Prüfung statischer Eigenschaften, wie z.B. den Schnittstellentyp bei der Verbindung von Anschlüssen oder die Einhaltung von Wertebereichen bei der Konfigurierung der Parameter. Dennoch kann festgestellt werden, dass CAE-Systeme den Problemlösungsprozess des Ingenieurs noch nicht in befriedigender Weise unterstützen [RHA02]. Die genanten Prüfungen dienen zur Sicherstellung der Datenkonsistenz und finden auf einer *syntaktischen* Ebene statt. Technische Abhängigkeiten befinden sich jedoch auf *semantischer* Ebene und können somit nicht erkannt werden (vgl. [Dujm02]). So bieten CAE-Systeme zwar vielfältige Unterstützung und teilweise auch eine interaktive Problemlösung bei den oben genannten Funktionen und Prüfungen an, jedoch keine Unterstützung hinsichtlich der Berücksichtigung von Wechselwirkungen zwischen Komponenteninstanzen. Der manuelle Aufwand bei Detailtätigkeiten zur Handhabung von Wechselwirkungen bleibt aus Sicht des Ingenieurs bestehen.

3.4 Unterstützung durch Komponenten-Frameworks

Der Ansatz von Frameworks entstand aus der Erkenntnis, dass in verschiedenen Anwendungen einer Domäne immer wieder dieselben prinzipiellen Problemlösungen realisiert werden. Ein Framework ist ein Anwendungsrahmen, welcher die Architektur und den Kontrollfluss darauf aufbauender Anwendungen definiert [JoFo88]. Der Entwickler bekommt durch das Framework eine domänenspezifische, generische Lösung an die Hand, die er durch Parametrisierungen und

Erweiterungen zu seiner speziellen Anwendung vervollständigen kann. Im Rahmen dieser Arbeit ist speziell der Ansatz des Komponenten-Frameworks interessant, der in [Dujm02] vorgestellt wird. Nach [Dujm02] wird durch ein Komponenten-Framework eine Architektur für einen bestimmten Problembereich definiert und somit ein Rahmen für das Zusammenspiel einer definierten Menge von Komponenten festgelegt. Jedes Komponenten-Framework ist durch zwei obligatorische Bestandteile gekennzeichnet:

- *Komponenten*, welche die Funktionalität für die frameworkbasierten Anwendungen realisieren und einer
- *Frameworkbeschreibung*, welche den Aufbau möglicher Anwendungen beschreibt, die aus den verfügbaren Komponenten des Frameworks erstellt werden können.

Die Frameworkbeschreibung definiert alle Möglichkeiten für die Auswahl, Konfiguration und Verbindung einer Menge von Komponenten. Sie kann zudem als Eingangsinformation für eine automatisierte Werkzeugumgebung (Instanziierungswerkzeug) genutzt werden, durch die der Ingenieur umfassende Unterstützung bei der Anwendungsentwicklung erhält: Mittels geführter Dialoge können die Entwurfsentscheidungen aus den (in der Frameworkbeschreibung) vorgegebenen Alternativen ausgewählt werden, bis die konkrete Anwendung vollständig spezifiziert ist.

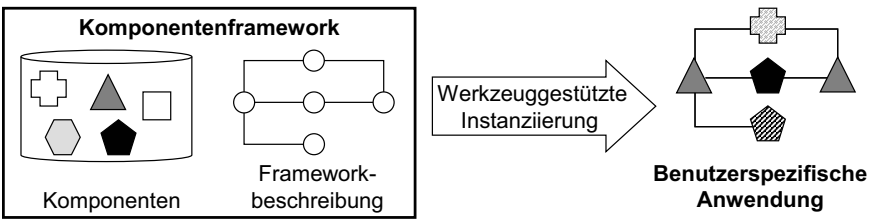


Abbildung 3.2: Anwendungsentwicklung mit Komponenten-Frameworks

Bewertung

Komponenten-Frameworks bieten für definierte Anwendungsbereiche ein effizientes Wiederverwendungskonzept. Sie ermöglichen die Wiederverwendung von Architektur, Komponenten und Kontrollfluss. Durch die Ausrichtung auf eine bestimmte Anwendungsdomäne ist das Wissen über diese Domäne im Framework gekapselt [Dujm02]. Der in Kapitel 2 genannte Informationsverlust zwischen Komponentenentwicklung und Engineering kann so vermieden werden. Der Ingenieur muss sich keine Gedanken über die Abhängigkeiten zwischen Komponenten machen, da diese bereits im Rahmen der durch die Frameworkbeschreibung vorgegebenen Komponentenvariabilitäten und -kombinationen berücksichtigt sind. Zudem erhält er durch das Instanziierungswerkzeug eine aktive und konstruktive Unterstützung bei Verwendung und Handhabung der Komponenten und wird schrittweise und interaktiv durch den Problemlösungsprozess bei der Entwicklung einer Anwendung geführt.

Obwohl der Ansatz des Komponenten-Frameworks eine umfassende Unterstützung für den Ingenieur bietet, ist er auch mit Nachteilen verbunden. Das benötigte Wissen über technische Abhängigkeiten ist nur *implizit* durch die in der Frameworkbeschreibung definierten Komponentenvariabilitäten und -kombinationen vorhanden. In der Frameworkbeschreibung sind nur diejenigen technische Abhängigkeiten berücksichtigt, die im Rahmen der vom Frameworkentwickler definierten Varianten bestehen. Möchte der Ingenieur die vom Frameworkentwickler vorgesehenen Pfade der Problemlösung verlassen, andere Komponenten verwenden oder manuelle, nicht vordefinierte Anpassungen oder Verbindungen durchführen, so erhält er keinerlei Unterstützung hinsichtlich der Handhabung entstehender, direkter Wechselwirkungen. Somit muss der Ingenieur deutliche Einschränkungen in seiner Flexibilität und kreativen Problemlösung hinnehmen. Zusätzlich können aus den manuellen Anpassungen indirekte Wechselwirkungen innerhalb des bestehenden Komponenten-Frameworks entstehen, die nicht in der Frameworkbeschreibung vorgesehen wurden. Sie müssen ebenfalls manuell gehandhabt werden, und das dazu benötigte Wissen über technische Abhängigkeiten muss manuell erfasst und angewendet werden. Damit ist auch die geforderte Berücksichtigung der Änderungsdynamik nur innerhalb des Frameworkrahmens gegeben. Die Ursache für diese Einschränkungen liegt in der Tatsache begründet, dass das in der Frameworkbeschreibung implizit enthaltene Wissen nicht automatisch auf Zusammenhänge außerhalb der vorgesehenen Lösungsvarianten angewendet werden kann. Aufgrund der hohen Individualität der Lösungen beim Engineering von Automatisierungsanlagen sind diese jedoch meist nicht in einem festen Anwendungsrahmen fassbar. Die Unterstützung des Ingenieurs durch Komponenten-Frameworks ist daher nur sehr eingeschränkt und nur für abgegrenzte, überschaubare Teillösungen geeignet.

3.5 Unterstützung durch wissensbasierte Systeme

Wissensbasierte Systeme sollen das Wissen eines Fachgebiets auf einem Rechner verfügbar machen [ScHu87]. Ein besonderes Kennzeichen wissensbasierter Systeme ist die klare Trennung des Fachbereichswissens von den Verarbeitungsstrategien zur Auswertung des Wissens [Perm90]. Im Gegensatz zu konventionellen Softwaresystemen, bei denen das Wissen zur Problemlösung (z.B. Berechnung eines Ergebnisses aus gegebenen Eingangsinformationen) implizit in Form fester Verarbeitungsanweisungen *abgebildet* ist, wird beim wissensbasierten Ansatz das Wissen über mögliche Lösungsschritte *explizit* beschrieben [RuNo95]. Ein Verarbeitungsalgorithmus versucht dann, ausgehend von den gegebenen Eingangsinformationen durch dynamische Verkettung einzelner Lösungsschritte ein zulässiges Ergebnis zu erreichen.

Wissen und Wissensrepräsentation

Allgemein ist „Wissen als begründete Information zu verstehen, aus der wir Ereignisse vorhersagen können.“ [Umst98]. Bei wissensbasierten Systemen wird von folgendem konkretisierten Wissensbegriff ausgegangen: Wissen bezeichnet die Gesamtheit aller organisierten Informatio-

nen mitsamt ihrer wechselseitigen Zusammenhänge, auf deren Grundlage ein (vernunftbegabtes) System handeln kann [Wiki06a]. Wissen beschreibt ein extrahiertes Abbild der Wirklichkeit, bestehend aus einer Menge von Aussagen über die reale Welt als nachgebildetes Modell innerhalb einer Rechneranwendung [HeHe90]. Es sind folgende allgemeine Merkmale festzustellen:

- Dem Wissen liegen Informationen zugrunde.
- Diese Informationen müssen derart aufeinander bezogen sein, dass sie in sich stimmig sind.
- Neben der inneren Übereinstimmung muss sich Wissen in Übereinstimmung mit den wahrnehmbaren Bedingungen einer Umwelt befinden.

Grundlegend sind zwei Arten von Wissen zu unterscheiden [Lehm04]:

- *Fachbereichswissen (deklaratives Wissen)* repräsentiert das statische Wissen über die im Anwendungsbereich verfügbaren Wissensobjekte ("Was"). Es zeichnet sich dadurch aus, dass es explizit formalisierbar ist, und kann mithilfe geeigneter Repräsentationsformen präzise beschrieben, aufgezeichnet und verlustfrei wieder- und weitergegeben werden.

- *Problemlösungswissen (operatives bzw. prozedurales Wissen)* beschreibt, wie das Fachbereichswissen anzuwenden ist ("Wie") und muss als Algorithmus abgebildet werden. Es wird auch als Handhabungswissen bezeichnet.

Um ein wissensbasiertes System zu entwickeln, muss das Fachbereichswissen aus einer Domäne isoliert, in eine deklarative Form gebracht und nach generellen, anwendungsunabhängigen Prinzipien strukturiert werden. Es kann dann deklarativ in einer Wissensbasis repräsentiert werden. Die Wissensbasis umfasst die Gesamtheit des anwendungsspezifischen Wissens, bestehend aus Einzeltatsachen und generellen Zusammenhängen. Zusammenhänge sind klassifikatorisch-definitorisch (z.B. Abstraktion "Ist-Ein" und Aggregation "Teil-Von") oder empirisch-kausal (z.B. Folgerung "aus A folgt B") [Lehm04]. Das Erstellen einer Wissensbasis erfolgt in den drei Schritten Wissensakquisition, Wissensmodellierung und Wissensoperationalisierung.

Drei Hauptfunktionalitäten wissensbasierter Systeme lassen sich unterscheiden: *Wissensrepräsentation*, *Wissensverarbeitung* und *Wissensableitung*. Die Wissensrepräsentation kann je nach Einsatzzweck und Aufgabenstellung unterschiedlich erfolgen, z.B. regelbasiert, fallbasiert, framebasiert, constraintbasiert, mithilfe von semantischen Netzen oder neuronalen Netzen. Eine Sortierung und Klassifikation erfolgt durch Begriffshierarchien wie Taxonomien, Ontologien, und Dictionaries. Eine Übersicht über die Eigenschaften der verschiedenen Repräsentationsformen findet sich in [Lunz94]. Die erstellte Wissensbasis enthält die deklarativen Wissensanteile in expliziter Form und ist von den prozeduralen Anteilen des Wissens getrennt. Somit wird eine anwendungsunabhängige Interpretation und eindeutige Rückübersetzung des Wissens ermöglicht. Die automatisierte Wissensverarbeitung und -ableitung aus der Wissensbasis erfolgt mit einer Problemlösungskomponente (so genannte Inferenzmaschine). Im wissensbasierten Ansatz werden dabei spezielle, anwendungsspezifische Prozeduren zur Problemlösung durch generelle

Mechanismen ersetzt [Lehm04]. Der Verarbeitungsalgorithmus der Inferenzmaschine basiert somit auf generischen Problemlösungsstrategien. Es werden Teilhypothesen gebildet, die gegen die Inhalte der Wissensbasis überprüft und gegebenenfalls verworfen werden. Diese Vorgänge werden als Schlussfolgerungen bezeichnet und wiederholen sich, bis eine (bzw. mehrere oder keine) Lösung gefunden wird. Zusätzlich legen Kontrollstrategien fest, in welcher Weise Schlussfolgerungen verkettet werden, z.B. Vorwärtsverkettung oder Rückwärtsverkettung [Runo95]. Die generischen Problemlösungsstrategien der Inferenzmaschine werden in prozeduraler Form wie in herkömmlichen Programmen formuliert [Perm90].

Anwendungsbereiche für den wissensbasierten Ansatz sind Spracherkennung, maschinelles Lernen, Programmgenerierung, Planungssysteme, Computerspiele, Benutzungsoberflächen und Expertensysteme. Letztere sind im Rahmen der vorliegenden Arbeit von besonderem Interesse.

Expertensysteme

Expertensysteme sind wissensbasierte Systeme mit spezieller fachspezifischer Problemlösefähigkeit. Es handelt sich um Programme, mit denen das Spezialwissen und die Schlussfolgerungsfähigkeit qualifizierter Fachleute auf eng begrenzten Aufgabengebieten nachgebildet werden soll. Die Form der Wissensrepräsentation, die sich dafür besonders eignet, ist die der logischen Wissensrepräsentation durch *Regeln*. Regeln haben die Form „Wenn X, dann Y“, z.B. wenn die Temperatur zu niedrig ist, dann Heizleistung erhöhen [Pupp91].

Mittels Expertensystemen können spezielle Probleme automatisch oder in Zusammenarbeit mit dem Benutzer gelöst werden [Lehm04]. Expertensysteme bestehen aus einer Wissensbasis zur Speicherung von Expertenwissen und einer Inferenzkomponente zur Auswertung des Wissens [Perm90]. Die Wissensbasis setzt sich aus Regeln und Fakten zusammen. Regeln repräsentieren das anwendungsunabhängige Expertenwissen und definieren die durchführbaren Schlussfolgerungen in der Wissensbasis [Konr99]. Fakten repräsentieren die fallspezifischen Daten, die sich auf das jeweilige Anwendungsproblem beziehen. Zusätzlich werden nach [HaKi86] noch ein Wissenserwerbssubsystem, ein Erklärungssubsystem und eine Benutzungsschnittstelle benötigt. Abbildung 3.3 zeigt die Bestandteile eines Expertensystems nach [HaKi86].

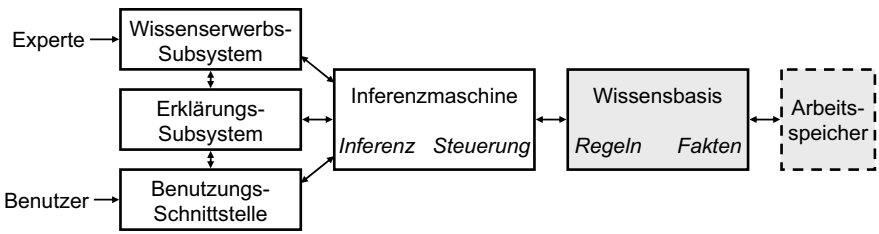


Abbildung 3.3: Bestandteile eines Expertensystems nach [HaKi86]

Expertensysteme werden als Beratungssysteme eingesetzt. Ziel ist es, entweder einen menschlichen Experten zu ersetzen, indem dessen Funktion als Ratgeber für Nicht-Experten vollständig eingenommen wird, oder die Rolle eines Assistenten für einen menschlichen Wissens- und Entscheidungsträger einzunehmen, d.h. einen menschlichen Experten bei seiner Arbeit zu unterstützen (vgl. [Jack99]). Die Benutzung eines Expertensystems erfolgt nach dem Frage-Antwortprinzip: Der Benutzer gibt die Informationen (Fakten) über eine anwendungsspezifische Situation als Fragestellung ein und erhält ein oder mehrere plausible Lösungsvorschläge. Die Lösungsvorschläge werden von der Inferenzmaschine durch Verknüpfung der eingegebenen Fakten mit den anwendungsunabhängigen Regeln der Wissensbasis mittels der vorgegebenen, generischen Lösungsstrategien berechnet. Berechnete Lösungen sind immer vollständig, d.h., sie sind für alle Regeln der Wissensbasis gültig. Daher besteht auch die Möglichkeit, dass der Benutzer keine Antwort erhält, wenn keine vollständig gültige Lösung gefunden wird. Beispiele für Expertensystemanwendungen existieren in der medizinischen und technischen Diagnose, in Businesssystemen, im Bildungsbereich, aber auch in der Entwicklungsunterstützung.

Bewertung

Wissensbasierte Systeme können zur Unterstützung der Entwicklung durch Nutzung von Wissen über die Anwendungsdomäne eingesetzt werden. Somit kann mittels wissensbasierten Systemen auch der Informationsverlust zwischen Komponentenentwicklung und Engineering reduziert werden. Das Wissen über technische Abhängigkeiten kann bei der Komponentenentwicklung in expliziter Form spezifiziert und dem Ingenieur zur Unterstützung seiner Tätigkeiten verfügbar gemacht werden. Ein wesentlicher Vorteil des wissensbasierten Ansatzes ist, dass der Lösungsweg für eine Problemstellung nicht explizit kodiert werden muss. Vielmehr können bei einer hinreichend spezifizierten Wissensbasis Lösungen für beliebige Probleme eines Bereichs gefunden werden. Dadurch zeichnen sich wissensbasierte Systeme durch eine hohe Flexibilität bei der Benutzung aus. Der Ingenieur kann in einer beliebigen Situation eine Anfrage an das System stellen und wird durch Ausgabe von Empfehlungen unterstützt [Konr99]. Diese Empfehlungen kann er annehmen und umsetzen, oder aber er kann sie verwerfen.

Ein wesentliches Merkmal von wissensbasierten Systemen ist, dass der Ingenieur die Initiative ergreifen muss, um die Unterstützung zu aktivieren. Das System fungiert lediglich als Berater und interagiert nicht zusammen mit dem Ingenieur oder anderen Systemen. Das bedeutet, dass die Unterstützung nur dann stattfindet, wenn der Ingenieur sie explizit anfordert. Auch die Umsetzung der Vorschläge in konstruktive Maßnahmen muss durch den Ingenieur selbst erfolgen, da wissensbasierte Systeme nicht handeln, sondern lediglich Handlungsmöglichkeiten vorschlagen. Ursache hierfür ist, dass es sich um isolierte, geschlossene Systeme handelt. Sie sind nur über den Benutzer mit der Umwelt verbunden und verfügen über keine Schnittstellen, über die sie ihre Umgebung unmittelbar wahrnehmen bzw. auf diese einwirken können. Wissensbasierte Systeme sehen also kein ereignisorientiertes Verhalten vor, vielmehr stehen die logischen Be-

ziehungen von Daten und Regeln im Vordergrund. Sie können nicht gezielt auf bestimmte Veränderungen der Umwelt (relevante Ereignisse) reagieren [Laub93].

Eine weitere Einschränkung ist, dass wissensbasierte Systeme monozentrisch orientiertes Problemlösen darstellen. Ihnen fehlen Komponenten, die für die Planung, Orientierung und Koordination sowie für die Kommunikationsfähigkeit, Allokation und Steuerung des Problemlösungsprozesses sorgen [Bech93]. Auch die Erklärungsfähigkeit ist nur teilweise vorhanden, sehr einfach und wenig selektiv [Lehm04]. Daher bieten wissensbasierte Systeme nur vollständige, globale Lösungen an. Es findet keine interaktive, begleitende Problemlösung statt, der Weg zur Lösung kann nicht direkt nachvollzogen oder beeinflusst werden. Die Verständlichkeit einer Lösung kann so nicht gewährleistet werden, da die Systematik des schrittweisen Vorgehens des Ingenieurs zur Handhabung von einzelnen Komponenteninstanzen und ihren Wechselwirkungen durch wissensbasierte Systeme nicht berücksichtigt wird. Der Ingenieur kann den Lösungsweg nicht beeinflussen und „strategische“ Teilentscheidungen treffen, z.B. dass Änderungen der Konfigurationen von Komponenteninstanzen nur teilweise oder in einer bestimmten Variante ausgeführt werden, um davon ausgehend weitergehende, indirekte Wechselwirkungen zu ermitteln. Vielmehr ist die Problemlösungsstrategie des wissensbasierten Systems generischer Art und kann nicht in Kooperation mit dem Ingenieur an die spezifische Situation angepasst werden. Es ist auch nicht möglich, die Berücksichtigung der Auswirkungen einer Änderung zunächst nur für einen bestimmten Teilbereich des Anlagenmodells zu begrenzen, um die durch die hohe Dynamik von Änderungen entstehende Komplexität möglicher Wechselwirkungen für den Ingenieur überschaubar zu halten.

3.6 Unterstützung durch agentenbasierte Beratungssysteme

In [Konr99] wurde der Ansatz eines agentenbasierten, aktiven Beratungssystems für die objektorientierte Entwicklung von Softwaresystemen bei Automatisierungsprojekten vorgeschlagen. Ziel des Ansatzes ist es, einen Entwickler mit wenig Erfahrung in objektorientierten Methoden während eines Entwicklungsprojekts *aktiv* durch Ratschläge zu unterstützen. Hierdurch soll der Kenntnisstand des Entwicklers erweitert werden, sodass er effizienter Software höherer Qualität erstellen kann. Zum Beratungsinhalt gehören Grundprinzipien der Softwareentwicklung, Wissen bezüglich der geeigneten Anwendung objektorientierter Konzepte sowie domänenspezifische Kenntnisse (mit Schwerpunkt auf Aspekten des zeitlichen Verhaltens von Automatisierungssystemen). Das für die Beratung benötigte Wissen wird aus den Erfahrungen routinierter Entwickler sowie aus Literaturquellen gewonnen und mithilfe der Konzepte wissensbasierter Systeme abgebildet.

Um eine aktive Beratung zu ermöglichen und somit einen der Nachteile wissensbasierter Systeme zu überwinden, wurde das Beratungssystem als Agentensystem realisiert. Dabei wird in [Konr99] von folgendem Verständnis von Agentensystemen ausgegangen: „Agentensysteme

sind Netzwerke von weitgehend autonom operierenden, lose gekoppelten Problemlösern, die an einem gemeinsamen Problem arbeiten. Ein Agent stellt eine gegenüber seiner Umwelt abgeschlossene Einheit dar, die über lokales Wissen zur Lösung von Teilproblemen verfügt. Eine Gesamtproblemlösung wird durch die Interaktion zwischen den Agenten erreicht. Das benötigte Wissen ist auf die einzelnen Agenten verteilt, sodass Steuerung, Datenhaltung und Problemlösungsverfahren dezentralisiert sind.“ Mittels des Agenteneinsatzes wird die aktive Erteilung von Ratschlägen gemäß folgendem Ablauf realisiert:

1. *Beobachtung* des Entwicklers bei der Erstellung und Bearbeitung objektorientierter Modelle, sowie Auswertung der Modellinformationen aus einem CASE-Werkzeug
2. *Klassifikation und Bewertung* von Modellen mithilfe allgemeiner, wissensbasiert spezifizierter Entwurfsheuristiken und -mustern
3. *Erkennung* von konkreten Entwurfsproblemen, Entscheidung über Ratschläge
4. *Beratung* durch Hinweise auf Fehler, Verbesserungsvorschläge in Form von Hinweisen auf geeignete Entwurfsprinzipien/ -muster

Die Ratschläge werden in Form von vorgefertigten, statischen Hypertext-Dokumenten ausgegeben. Falls der Entwickler Interesse hat, kann er die erstellten Ratschläge lesen, er kann sie beachten (d.h. manuell umsetzen) oder auch verwerfen.

Softwareagenten werden dabei für zwei Aufgaben eingesetzt: zum einen zur aktiven Beobachtung der Aktivitäten des Benutzers, zum anderen, um zu ermitteln, wann welcher Ratschlag an den Entwickler sinnvoll ist. Dazu wurden folgende aufgabenbezogene Agententypen eingeführt:

Tabelle 3.1: Agententypen im Beratungssystem nach Konrad

Agententyp	**Aufgabe**
Klassifikations-Agent	Beinhaltet das Wissen eines Experten. Für unabhängige Wissensbereiche (z.B. Strukturierung, Zeitverhalten) werden verschiedene Agenten eingesetzt. Ihre Aufgabe ist die Analyse und Klassifikation von Entwicklungssituationen.
Dialog-Agent	Übernimmt die Auswahl und Übermittlung von Meldungen an den Benutzer.
Meldungsverteiler	So genannte „Spione“ erfassen und melden die Benutzeraktivitäten im CASE-Werkzeug. Dieser Agent verteilt die Ereignisse an die Klassifikations-Agenten.
Metawissen	Repräsentiert das Wissen über Abhängigkeiten zwischen Teilmodellen.

Das agentenbasierte Beratungssystem erkennt beispielsweise Gestaltungsfehler in so genannten Objekt-Kommunikationsdiagrammen. In der Methode wird eine hierarchische Kommunikationsstruktur empfohlen, in der Objekte eindeutig einer Architekturebene zugeordnet werden und Kommunikation nur zwischen benachbarten Ebenen stattfindet. Objekte, die eine Hardwareschnittstelle repräsentieren, sollen möglichst nur mit genau einem anderen Objekt kommu-

nizieren, auf jeden Fall aber nicht mit Objekten aus verschiedenen Ebenen. Derartige Entwurfsheuristiken werden als Regeln in der Wissensbasis einzelner Klassifikationsagenten abgelegt. Anschließend sind die Agenten in der Lage, vom Entwickler erstellte Modelle zu analysieren und entsprechende Hinweise und Ratschläge in Form von Hypertext-Dokumenten zu erteilen. Der Inhalt der Ratschläge entspricht dabei genau den genannten Entwurfsheuristiken. Weiter werden dem Entwickler beispielhafte Vorschläge gegeben, wie die dem Ratschlag zugrunde liegende Entwurfsheuristik in ein Modell umgesetzt werden kann.

Bewertung

Der Ansatz des agentenbasierten Beratungssystems nach Konrad ermöglicht die Unterstützung des Entwicklers durch Nutzung von Expertenwissen nach dem wissensbasierten Ansatz. Zudem findet eine aktive Beratung statt, indem Softwareagenten als "Mittler" zwischen den Ingenieurtätigkeiten in einem CASE-Werkzeug und dem wissensbasierten System eingesetzt werden. Die Aufgabe der Softwareagenten ist es, eine Entwicklungssituation selbstständig zu erkennen, durch flexible Kombination ihres jeweiligen Expertenwissens zu klassifizieren und entsprechende Ratschläge in Form von Meldungen an den Entwickler zu geben. Der Inhalt der Beratung besteht aus allgemeinen Entwurfsheuristiken und -mustern, die bei der Erstellung von Modellen zu beachten sind. Dabei besteht keinerlei Einschränkung der Flexibilität des menschlichen Vorgehens, die Beratung findet parallel zu den Tätigkeiten des Entwicklers statt und ist an die jeweils aktuelle Entwicklungssituation angepasst.

Allerdings handelt sich bei dem vorgeschlagenen Ansatz um ein Beratungssystem, nicht um ein Unterstützungssystem im Sinne der Anforderungen dieser Arbeit. Zwar wird eine konkrete Problemsituation durch die Softwareagenten aktiv erkannt und zugehörige Lösungsvorschläge erteilt, diese sind jedoch allgemeiner Art. Es handelt sich um vorgefertigte Beispiele zur Umsetzung von Entwurfsheuristiken und -mustern, die nicht spezifisch an die konkrete Problemsituation angepasst sind. Es werden lediglich generelle Handlungsmöglichkeiten zur Problembehebung vorgeschlagen, ihre spezifische Abbildung auf die aktuelle Entwicklungssituation und die konstruktive Umsetzung erfolgt manuell durch den Entwickler. Es findet also keine konstruktive Unterstützung statt. Die angebotene Unterstützung bezieht sich auch nicht auf einzelne, konkrete Tätigkeiten zur Handhabung von Komponenten (hier: Objekte), sondern auf Grundprinzipien bei der Gestaltung von Modellen. Ziel des Ansatzes ist es, den Kenntnisstand des Entwicklers zu erweitern und nicht, konkrete routinemäßige Detailtätigkeiten zu unterstützen. Daher werden nur die Ergebnisse von Tätigkeiten in Form des entstandenen Modells analysiert. Weiter werden Lösungsvorschläge stets isoliert voneinander betrachtet und die bei Änderungen durch Abhängigkeiten entstehende Dynamik wird nicht berücksichtigt. Hat der Entwickler einen Lösungsvorschlag umgesetzt, erfolgen eine erneute Prüfung und evtl. weitere Ratschläge.

3.7 Zusammenfassende Bewertung und Folgerung

Die Ergebnisse der Untersuchung bestehender Ansätze zu Unterstützung des Engineerings von Automatisierungsanlagen und ihre zusammenfassende Bewertung gemäß der aufgestellten Anforderungen werden in Tabelle 3.2 zusammengefasst. Jeder Ansatz wird danach beurteilt, ob die zugrunde liegende Anforderung gut (+), teilweise (0) oder schlecht (-) erfüllt wird.

Tabelle 3.2: Vergleich existierender Ansätze zur Unterstützung des Engineerings

	Aktive Unterstützung	**Konstruktive Unterstützung**	**Interaktive Problemlösung**	**Änderungs-dynamik**	**Flexibilität des Benutzers**
Computer Aided Engineering Systeme	0	0	0	0	+
Komponenten-Frameworks	+	+	+	-	-
Wissensbasierte Systeme	-	0	-	0	+
Agentenbasierte Beratungssysteme	+	-	0	-	+

Während alle beschriebenen Ansätze interessante Lösungen für Teile der Unterstützungsproblematik anbieten, gibt es keinen Lösungsansatz, der alle Anforderungen vollständig abdeckt. So stellen heutige *CAE-Systeme* eine ingenieurgerechte Werkzeugunterstützung für die Anwendung des komponentenbasierten Engineerings von Automatisierungsanlagen, bieten umfangreiche grafische Modellierungsfunktionen und sind zudem in der industriellen Praxis weit verbreitet. Sie weisen jedoch Defizite hinsichtlich der in dieser Arbeit angestrebten Unterstützung von Detailtätigkeiten zur Berücksichtigung von technischen Abhängigkeiten auf. Der Ansatz der *Komponenten-Frameworks* bietet zwar Vorteile hinsichtlich der Anforderungen an eine aktive und konstruktive Unterstützung sowie die interaktive, begleitende Problemlösung. Technische Abhängigkeiten zwischen Komponenten werden ebenfalls berücksichtigt. Die Unterstützung ist jedoch auf eine vorgegebene Menge von Komponenten, Variabilitäten und Kombination beschränkt. Komponenten-Frameworks weisen dadurch nicht die erforderliche Flexibilität und Änderungsdynamik für den gesamten Problemlösungsprozess des Engineerings auf, bieten jedoch interessante Konzepte für die Wiederverwendung und Anpassung abgegrenzter Teillösungen. Beim *wissensbasierten Ansatz* kann das Wissen des Komponentenentwicklers über technische Abhängigkeiten in expliziter Form spezifiziert und somit für den Ingenieur zur Unterstützung seiner Tätigkeiten beim Engineering verfügbar gemacht werden, ohne die Flexibilität der menschlichen Arbeitsprozesse einzuschränken. Es bestehen jedoch keine Möglichkeiten der aktiven Unterstützung, der konstruktiven Umsetzung von Lösungsvorschlägen, der interaktiven Problemlösung und bezüglich einer *nachvollziehbaren* Berücksichtigung der Änderungsdynamik bei der Handhabung von Komponenten und ihren Wechselwirkungen. Durch den Einsatz

von Softwareagenten im *agentenbasierten Beratungssystem* kann eine aktive Beratung ermöglicht werden, welche sich flexibel an die menschlichen Arbeitsprozesse anpasst. Das konzipierte Beratungssystem betrachtet jedoch keine einzelnen Tätigkeiten, sondern vielmehr die Ergebnisse von Tätigkeiten in Form der erstellten Modelle. Es kann daher nicht für die Unterstützung der spezifischen menschlichen Tätigkeiten zur Handhabung von Komponenten ausgelegt werden. Weiter wurde der Einsatz von Softwareagenten konzeptionell auf die Analyse und Klassifikation von Problemen beschränkt. So wurde zwar begleitend zu den Entwicklungstätigkeiten eine aktive, flexible Beratung, aber noch keine konstruktive und dynamische Unterstützung von konkreten Tätigkeiten erreicht.

Es stellt sich somit die Frage nach einem umfassenden Konzept, das die Vorteile bestehender Ansätze integriert und gleichzeitig die Unterstützungsproblematik in ihrer Gesamtheit betrachtet. Ein geeignetes Konzept muss den Komponentenbezug der Tätigkeiten und Überlegungen des Ingenieurs berücksichtigen. Es muss für konkrete Tätigkeiten eine aktive Unterstützung und konstruktive Lösungsvorschläge anbieten, ohne die Flexibilität einzuschränken, und die Dynamik von Wechselwirkungen und Anpassungen angemessen berücksichtigen.

Einen geeigneten Ausgangspunkt stellt der komponentenbasierte Ansatz und seine werkzeugseitige Umsetzung in existierenden CAE-Systemen dar. Daher ist die technische Integration eines Unterstützungssystems mit CAE-Systemen anzustreben. Der wissensbasierte Ansatz ermöglicht die explizite Spezifikation von Wissen über technische Abhängigkeiten und stellt somit eine wichtige Grundlage für die Ermittlung und Handhabung von Wechselwirkungen zur Verfügung. Aus dem Ansatz des agentenbasierten Beratungssystems wurde erkennbar, dass der Einsatz von Softwareagenten eine Reihe von Vorteilen hinsichtlich der Aktivität und Flexibilität der Unterstützung mit sich bringt. Daher soll genauer untersucht werden, ob Softwareagenten nicht nur für eine aktive Beratung durch Fehleranalyse und Hinweise auf vorhandenes Wissen, sondern darüber hinaus für eine umfassende, konstruktive Unterstützung und interaktive Problemlösung bei Engineeringtätigkeiten genutzt werden können.

In diesem Kapitel wurden die Anforderungen ermittelt, die an eine umfassende Unterstützung des Ingenieurs beim Engineering bestehen. Existierende Lösungsansätze wurden vorgestellt und hinsichtlich ihrer Eignung für eine solche Unterstützung bewertet. Auf der Grundlage dieser Betrachtungen wurde deutlich, dass bestehende Lösungsansätze immer nur Teilaspekte der Unterstützung des Ingenieurs aufgreifen. Aus diesem Grund wird ein neues Lösungskonzept benötigt, welches die Unterstützungsproblematik in ihrer Gesamtheit betrachtet. Im folgenden Kapitel wird das Paradigma der Softwareagenten als Ansatz zur Entwicklung aktiver, flexibler und dynamischer Systeme eingeführt. Danach wird in Kapitel 5 ein neues Konzept vorgestellt, das unter Berücksichtigung aller Anforderungen die umfassende Unterstützung des Ingenieurs auf der Grundlage von Softwareagenten ermöglicht.

4 Paradigma der Softwareagenten

In diesem Kapitel wird das Paradigma der Softwareagenten eingeführt und definiert. Es werden die Konzepte, Methoden und Werkzeuge vorgestellt, die für eine agentenorientierte Softwareentwicklung bereitstehen. Weiter wird erläutert, wie auf Basis von Softwareagenten aktive und flexible Softwaresysteme entwickelt werden können. Ausgehend von einer generellen Betrachtung der Merkmale agentenorientierter Software werden ihre Einsatzmöglichkeiten in der Automatisierungstechnik untersucht und der Bezug zur Unterstützung des Engineerings von Automatisierungsanlagen hergestellt.

4.1 Einführung in die agentenorientierte Denkweise

Eine Ursache für die vielfach bekannten Probleme in der heutigen Softwareentwicklung ist die hohe Komplexität der zu realisierenden Softwaresysteme. In den letzten drei Jahrzehnten hat die Softwaretechnik ein zunehmend besseres Verständnis für die Eigenschaften und Zusammenhänge der Komplexität von Software erreicht [EbGö04]. Heute ist anerkannt, dass *Verteiltheit* und *Interaktion* die hervorstechenden Merkmale komplexer Softwaresysteme sind [Simo96]: Moderne Softwaresysteme bestehen häufig aus vielen verteilten Elementen, die getrennt voneinander ablaufen und untereinander in vielfältiger und komplexer Weise interagieren. Dabei wächst die Anzahl möglicher verschiedener Abhängigkeiten zwischen den Elementen des Systems sehr viel schneller als die Anzahl der Elemente [Paru98]. Die Schwierigkeit besteht in der Beherrschung der komplexen Abläufe und Verhaltensweisen solcher Systeme. Daher liegt der Schwerpunkt der Softwaretechnik zunehmend darin, Instrumente zu entwickeln, mit denen komplexe Softwaresysteme besser verstanden, modelliert und implementiert werden können. Dabei haben sich grundlegende, allgemein gültige *Prinzipien* herausgebildet: die Abstraktion zur Modellierung einer Problemstellung, die Zerlegung des Gesamtsystems in seine Bestandteile und ihre Anordnung zu einer Struktur [Booc94], [Balz01]. Zur Anwendung der Prinzipien wurden verschiedene Paradigmen der Softwareentwicklung konzipiert, die jeweils unterschiedliche Gesichtspunkte bei der Abstraktion, Zerlegung und Strukturierung eines Softwaresystems in den Vordergrund stellen. Zu den bekanntesten zählen die strukturierte Softwareentwicklung [Balz01] und die objektorientierte Softwareentwicklung [Booc94]. Eine neue Herangehensweise zur Entwicklung verteilter Softwaresysteme mit komplexem, schwer überschaubarem Gesamtverhalten ist das Paradigma der *Agentenorientierung* [LPB03]. Für die systematische und zielgerichtete Entwicklung solcher Systeme stellt das agentenorientierte Paradigma Konzepte, Methoden und Werkzeuge zur Verfügung. Dabei werden die Struktur und die Funktionalität eines Softwaresystems unter dem Gesichtspunkt einer Menge von Agenten[3] betrachtet [Weiß01], d.h.

[3] Im Folgenden wird der Begriff *Agent* stets im Sinne von *Softwareagent* verwendet. Eine allgemeine Taxonomie des Agentenbegriffes findet sich in [FrGr96].

unter dem Aspekt des Zusammenwirkens flexibel interagierender, autonomer Softwareeinheiten. [Schi05] verdeutlicht die zugrunde liegende Denkweise: „Die Komplexität des Gesamtsystems macht es erforderlich, dieses modular aus Teillösungen aufzubauen, wobei jede Teilaufgabe und ihre Schnittstelle genauestens spezifiziert wird. Wenn nun diese Teillösungen mit Autonomie ausgestattet werden und sie asynchron und unabhängig voneinander der Lösung ihrer Aufgabe nachgehen, hat man es bereits (oft ohne es zu wissen) mit einem Agentensystem zu tun.“ Tabelle 4.1 stellt die Prinzipien der Softwaretechnik den Schwerpunkten des agentenorientierten Paradigmas gegenüber [WGU03].

Tabelle 4.1: Umsetzung der Prinzipien der Softwaretechnik im agentenorientierten Paradigma

Prinzip	**Bedeutung**	**Agentenorientiertes Paradigma**
Abstraktion	Beschreibung der wesentlichen Aspekte einer Problemstellung unter einem bestimmten Blickwinkel	Fokus auf der Beschreibung verteilter Strukturen, lokaler Dynamik und Wechselwirkungen zwischen Teilprozessen
Zerlegung (Modularisierung)	Aufteilung des Systems in kleinere überschaubare Bestandteile, die isoliert voneinander betrachtet werden	Betonung der selbstständigen Aufgabenerfüllung durch einzelne, möglichst unabhängige Elemente (Agenten)
Strukturierung (Organisation)	Anordnung der Bestandteile und Beschreibung ihrer Beziehungen	Schwerpunkt auf flexiblen Beziehungen und dynamischen Strukturen

4.2 Grundlegende Konzepte der Agentenorientierung

Bei der agentenorientierten Softwareentwicklung wird ein Softwaresystem als eine Menge von autonomen Einheiten betrachtet, die selbstständig innerhalb ihres Entscheidungsrahmens handeln und dabei vorgegebene Ziele verfolgen. Sie können miteinander flexibel interagieren und durch Verhandlungen kooperieren, um ihre individuellen Ziele zu erreichen. Zentrales Abstraktionsmittel bei der agentenorientierten Betrachtung eines Sachverhalts ist das Konzept des Softwareagenten. Im Rahmen dieser Arbeit wird folgende allgemein anerkannte Definition zugrunde gelegt (vgl. [Jenn00], [WoJe95], [Wool97], [Weiß01]):

Softwareagent: In der agentenorientierten Softwareentwicklung ist ein Agent das Konzept einer abgrenzbaren Softwareeinheit mit einem definierten Ziel. Ein Agent versucht, dieses Ziel durch autonomes Verhalten zu erreichen und interagiert dabei kontinuierlich mit seiner Umgebung und anderen Agenten.

4.2.1 Eigenschaften von Softwareagenten

Der Definition von Softwareagenten liegen verschiedene Eigenschaften zugrunde, welche die *Grundkonzepte* der Agentenorientierung darstellen (siehe Abbildung 4.1) [WGU03]:

- *Kapselung*: Zustand und Verhalten sind in einer Einheit, dem Agenten, zusammengefasst.
- *Zielorientierung*: Ein Agent orientiert sein Verhalten an bestimmten Zielen, die er zu erreichen versucht. Der Weg zum Ziel ist dabei nicht im Einzelnen vorgegeben, sondern wird durch den Agenten im Laufe seiner Aktionen festgelegt.
- *Autonomie*: Ein Agent handelt selbstständig bei der Erfüllung der übertragenen Aufgaben. Autonomie bedeutet die Kontrolle über den internen Zustand und das Verhalten: Es werden eigene Entscheidungen bezüglich der im Einzelnen auszuführenden Aktionen getroffen.
- *Aktivität:* Ein Agent verfolgt aktiv seine übertragenen Aufgaben. Diese Aktivität setzt sich zusammen aus *Reaktivität* und *Proaktivität*: Ein Agent reagiert auf seine Umwelt, er nimmt Informationen auf und leitet daraus eigene Aktionen ab. Über das rein reaktive Verhalten hinaus hat ein Agent hat die Fähigkeit, ohne Aufruf oder Einfluss von außen zielgerichtet und vorausschauend zu handeln (Eigeninitiative).
- *Interaktion*: Agenten interagieren miteinander, um eine Aufgabe zu lösen, individuelle Ziele zu erreichen oder um Abhängigkeiten untereinander zu handhaben.
- *Persistenz*: Ein Agent verfügt über einen dauerhaften Kontrollfluss, der unabhängig ist von externer Aktivierung. Er behält seinen inneren Zustand während seines Lebenszyklus bei.

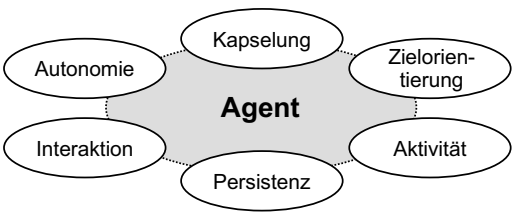


Abbildung 4.1: Eigenschaften von Softwareagenten

In der agentenorientierten Softwareentwicklung wird eine Problemstellung unter Anwendung der Grundkonzepte in einzelne Agenten abstrahiert, um so verteilte Informationen, Funktionalität und Entscheidungsprozesse beschreiben zu können. Auf diese Weise wird ein agentenorientiertes Modell entwickelt, welches den Aufbau einzelner Agenten und ihr Verhalten beschreibt.

4.2.2 Aufbau von Softwareagenten

Zur Modellierung von Softwareagenten sind verschiedene grundlegende Bestandteile erforderlich, welche die genannten Grundkonzepte unterstützen [GUW04]. Die Fähigkeit, selbst zu entscheiden, was wann zu tun ist und auf welche Weise, bildet eine fundamentale Eigenschaft eines Softwareagenten. Das Verhalten wird daher prinzipiell durch die *Ziele* bestimmt, die der Agent erreichen soll. Ziele können entweder vom Entwickler definiert werden oder zur Ausführungs-

zeit von einem Benutzer in Form eines Auftrages übertragen werden. Um eigene Entscheidungen bezüglich der auszuführenden Aktionen treffen zu können, muss ein Softwareagent über alle benötigten Informationen über und aus seiner Umgebung verfügen [Paru98]. Diese werden im *Umgebungsmodell* beschrieben. Die möglichen Aktionen, die ein Softwareagent ausführen kann, werden durch seine Fähigkeiten festgelegt. *Fähigkeiten* beschreiben zum einen die möglichen internen Aktionen des Softwareagenten und zum anderen die möglichen Interaktionen mit seiner Umgebung. Abbildung 4.2 verdeutlicht den grundlegenden Aufbau von Softwareagenten.

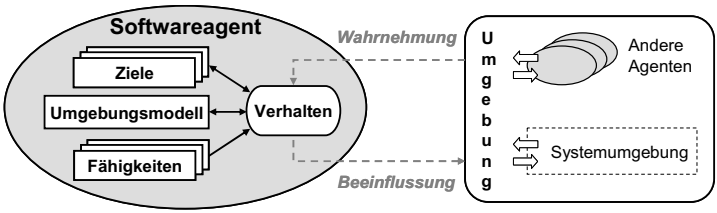


Abbildung 4.2: Grundlegender Aufbau von Softwareagenten

Das *Verhalten* eines Softwareagenten besteht aus der Erfassung der Informationen aus seiner Umgebung, der Evaluierung dieser Informationen hinsichtlich der vorgegebenen Ziele, sowie die Auswahl und Ausführung geeigneter Aktionen und Interaktionen auf Basis der Fähigkeiten.

4.2.3 Interaktionen von Softwareagenten

Die Interaktion von Softwareagenten mit ihrer Umgebung besteht darin, diese wahrzunehmen und durch Handlungen auf Basis der eigenen Fähigkeiten und entsprechend ihrer Ziele zu beeinflussen. Die Umgebung kann dabei aus anderen Softwareagenten bestehen oder Teil der Systemumgebung sein (siehe Abbildung 4.2). Softwareagenten tauschen dabei untereinander nicht nur einzelne Datenfelder oder Variablen aus, sondern können mittels Interaktionen auch Aufgaben oder Anfragen an andere Agenten verteilen bzw. koordinieren, über Ressourcen verfügen oder Verhandlungen durchführen [Paru98]. Die Interaktionen zwischen Softwareagenten erfolgen durch den Austausch von Nachrichten und finden somit auf einer Ebene mit variabler Semantik statt [Müll04]. Die Interaktion durch Nachrichtenaustausch hat den Vorteil, dass der Informationsaustausch als eigentlicher Zweck von Interaktion von der Kommunikation getrennt werden kann [Eber05]. Dadurch wird eine höhere Flexibilität von Interaktionen und eine größere strukturelle Unabhängigkeit der Agenten voneinander erreicht, als dies bei der Interaktion auf einer niedrigeren Abstraktionsebene mit fester Semantik (z.B. in Form von Funktions- oder Methodenaufrufen) der Fall ist [Gehm99], [Jenn00]. Eine Nachricht besteht aus ihrem eigentlichen Inhalt (*content*) und weiteren Attributen, welche der Kommunikation und dem Nachrichtentransport zugeordnet sind (z.B. *Sender* und *Receiver*). Agentenkommunikationssprachen wie FIPA-ACL [FIPA02b] oder KQML [KQML93] legen den grundlegenden Aufbau von Nachrichten fest. Sie definieren verschiedene Nachrichtentypen, wobei der Inhalt der Nachricht nicht

festgelegt ist. Mögliche Nachrichtentypen sind Informationsübertragung und Dienstanfrage (*inform*, *query*, *subscribe*, *request*), Verhandlung (*call-for-proposal*, *propose*), Zustimmung und Ablehnung (*agree*, *refuse*) [Gehm99].

Interaktionen zwischen Softwareagenten setzen sich häufig aus einer aufgabenorientierten Sequenz von Nachrichten zusammen, so genannte Konversationen, die zur Erreichung eines bestimmten Ziels verfolgt werden. Zur Standardisierung häufig benötigter Konversationen existieren Interaktionsprotokolle. Interaktionsprotokolle definieren Muster von Nachrichten und Nachrichtensequenzen, die im Verlauf einer Konversation auftreten können [Paru98]. Es gibt einen umfangreichen Satz an Standardinteraktionsprotokollen, z.B. das Query Protokoll, das Contract-Net Protokoll und Protokolle für verschiedene Verhandlungsformen.

4.3 Agentenorientierte Softwareentwicklung

Die Aufgabe der agentenorientierten Softwareentwicklung ist es, alle Entwicklungsphasen für ein agentenorientiertes System zu unterstützen, ausgehend von der agentenorientierten Analyse der Problemstellung über den Entwurf bis hin zur Implementierung.

4.3.1 Agentenorientierte Methoden

Um Software agentenorientiert zu entwickeln, ist daher eine grundsätzlich andere Denk- und Herangehensweise als bei herkömmlichen Methoden der Softwareentwicklung erforderlich, die speziell an den agentenorientierten Ansatz angepasst ist [Jenn00]. Der Schwerpunkt liegt dabei auf Prozessen und Abläufen, die zwischen den aktiven Elementen des Softwaresystems – den Agenten – stattfinden. Mehrere agentenorientierte Entwicklungsmethoden wurden als Ergebnis von Forschungsarbeiten konzipiert und stehen für die Anwendung zur Verfügung[4]. Sie stellen spezielle Analyse- und Entwurfskonzepte, Notationen und Vorgehensweisen zur Unterstützung des Entwicklers bereit, um die Problemstellung systematisch zu abstrahieren und ein Modell auf Basis der agentenorientierten Grundkonzepte zu entwickeln.

Die Methoden verwenden hierzu methodische Konzepte wie Zielanalyse, Rollenanalyse und Interaktionsmodelle (siehe Abbildung 4.3). Zunächst werden durch eine Zielanalyse die Anforderungen an das Softwaresystem in detaillierte Systemziele überführt. Auf diese Weise werden Ziele und Aufgaben des Softwaresystems in den Fokus der Analyse gestellt. Im zweiten Schritt werden Ziele in so genannten *Rollen* zusammengefasst. Rollen definieren die Verantwortlichkeiten und Aufgaben, die innerhalb eines Softwaresystems vorkommen. [Kend99]. Im dritten Schritt werden die aktiven Elemente des Softwaresystems ermittelt, als Agententypen modelliert und die entsprechenden Ziele und Rollen zugeordnet. Im vierten Schritt werden die konkreten

[4] Zu diesen Methoden gehören Gaia [WJK00], MaSE [WoDe01], MASSIVE [Lin01], MESSAGE [EKC+01], PASSI [CoPo02], Prometheus [PaWi02] und Tropos [GMP02]. Übersichten über agentenorientierte Methoden finden sich in [IGG99], [Weiß02], [HeGi05], Vergleiche und Evaluierungen von Methoden in [DaWi04] [StSh05].

Fähigkeiten zur Erfüllung der Rollen und die benötigten Informationen des Umgebungsmodells spezifiziert. Dazu werden die Aktionen und Interaktionen der Agenten mithilfe von Sequenz-, Aktivitäts- oder Zustandsdiagrammen entworfen. Neben methodeneigenen Notationsformen existieren auch spezielle Erweiterungen der UML für die Modellierung agentenorientierter Software [Baue02]. Ein anschauliches Beispiel für die Anwendung einer agentenorientierten Methode zur Entwicklung einer Steuerungssoftware für ein technisches System wird in [UWG03] vorgestellt, weitere Beispiele finden sich in [WJK00], [CoPo02], [Stec03], [HWH04].

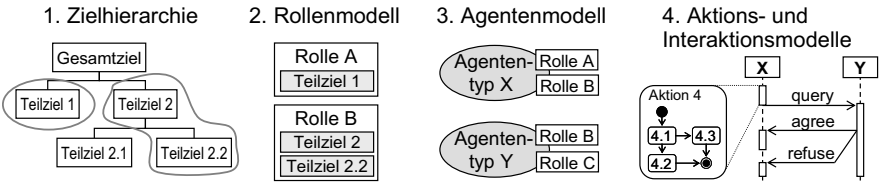


Abbildung 4.3: Übersicht über die Konzepte agentenorientierter Methoden

Das erstellte agentenorientierte Modell kann anschließend als Softwaresystem implementiert werden. Zur Laufzeit entsteht dann das Gesamtverhalten des Softwaresystems aus dem dynamischen Zusammenspiel der einzelnen Agenten, die aktiv die ihnen übertragenen Ziele verfolgen, situationsabhängig verschiedene Rollen einnehmen und flexibel miteinander interagieren.

4.3.2 Agentensysteme

Für die Implementierung eines agentenorientierten Softwareentwurfes in ein agentenorientiertes Softwaresystem – als *Agentensystem* bezeichnet – existieren Standards und Technologien, welche die Umsetzung der agentenorientierten Grundkonzepte bei der Implementierung unterstützen. Standards definieren die Architektur von Agentensystemen, Technologien unterstützen den Programmierer bei der Implementierung von Agenten, ihrer Aktionen und Interaktionen.

Architektur von Agentensystemen

Agentensysteme bestehen aus mehreren Agenten, welche unterschiedliche Aufgaben übernehmen und kooperieren, um ihre Aufgaben zu lösen. Die flexible Interaktion zwischen Agenten impliziert die dynamische Ermittlung anderer Agenten bzw. ihrer angebotenen Dienste und ein geeignetes Kommunikationsmedium. Die Standardisierung dieser architektonischen Elemente führt zu besserer Entwicklungsunterstützung und höherer Kompatibilität von Agentensystemen. Ein weithin anerkannter Architekturstandard wurde von der *Foundation for Intelligent Physical Agents* (FIPA) entwickelt. In der *Abstract Architecture Specification* [FIPA02a] und der *Agent Management Specification* [FIPA04] wurde das in Abbildung 4.4 gezeigte Referenzmodell definiert, das den Aufbau und die erforderlichen Bestandteile eines Agentensystems festlegt.

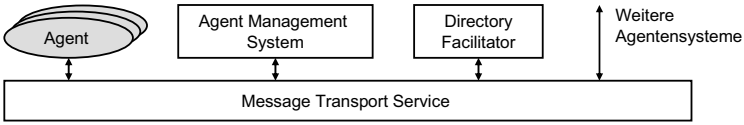


Abbildung 4.4: FIPA Agent Management Referenzmodell [FIPA04]

Jedes Agentensystem besitzt ein *Agent Management System* (AMS), welches ein Verzeichnis mit den Bezeichnern und Kommunikationsadressen aller Agenten im Agentensystem führt. Es bietet einen Auskunftsdienst („White Pages“) an, über den Agenten ihre Kommunikationspartner ermitteln können. Der *Directory Facilitator* (DF) stellt einen “Gelbe Seiten”-Dienst („Yellow Pages“) dar, über den Agenten ihre Dienste anmelden und die Dienste anderer Agenten abfragen können. Der *Message Transport Service* (MTS) ist eine Kommunikationsinfrastruktur, die von Agenten zum Nachrichtenaustausch verwendet wird. Neben den Kommunikationsmechanismen wird durch den MTS auch die Agentenkommunikationssprache festgelegt.

Technologien für die Implementierung von Agentensystemen

Zu den Technologien für die Implementierung von Agentensystemen zählen agentenorientierte Programmiersprachen, Klassenbibliotheken sowie Entwicklungs- und Laufzeitumgebungen [GUW04]. In der Praxis kommen vor allem Letztere zum Einsatz. Sie ermöglichen die Implementierung des Agentensystems auf hoher Abstraktionsebene, da es auf der Basis vorhandener Programmbibliotheken und -infrastrukturen realisiert wird. Dazu gehören auch integrierte Laufzeitkomponenten (so genannte Agentenplattformen) für Kommunikation, Agentenverwaltung, Dienstverwaltung und Überwachung. Die Aufgabe des Entwicklers beschränkt sich auf die Implementierung der anwendungsspezifischen Funktionalität. Eine weit verbreitete Entwicklungs- und Laufzeitumgebung ist das Open Source Projekt JADE (Java Agent DEvelopment Framework)[5] [JADE]. JADE unterstützt die Entwicklung von Agentensystemen durch eine Bibliothek mit umfangreichen Lösungen für wiederkehrende Aufgaben bei der Realisierung von Agentensystemen, einer zum FIPA-Standard konformen Agentenplattform und Werkzeuge für die Konfiguration, Verwaltung und das Debugging von Agentensystemen.

4.4 Vorteile, Eignung und Herausforderungen des Einsatzes von Softwareagenten

Die wesentliche Aufgabe eines Paradigmas der Softwareentwicklung besteht darin, durch Überbrückung der inhaltlichen Distanz zwischen Problembereich (Beschreibung der Problemstellung) und Lösungsbereich (das ausführbare Programm) die Entwicklung zu vereinfachen [Jenn00]. Das agentenorientierte Paradigma bietet hierzu besonders geeignete Hilfsmittel zum Verständnis komplexer Aufgabenstellungen. Durch die agentenorientierten Grundkonzepte wer-

[5] Weitere Beispiele sind AGENTBUILDER [BUILDER], FIPA-OS [FIPAOS], JACK [JACK], JATLite [JATLITE], LEAP [LEAP], LS/TS [LSTS] und ZEUS [ZEUS]. Übersichten finden sich in [Mang02] und [PBL05].

den nicht nur die realen Entitäten des Problembereichs wie bei der Objektorientierung in den Mittelpunkt der Modellierung gestellt, sondern auch ihr „Zweck“, d.h. ihre Ziele und Aufgaben explizit berücksichtigt und integriert [PBC01]. Dadurch wird eine Verschiebung der konzeptionellen Basis der Softwareentwicklung vom Lösungsbereich in den Problembereich erreicht [Jenn00]. Die konzeptionelle Aufteilung von Zielen, Funktionalität und Entscheidungsprozessen auf autonome Softwareeinheiten ermöglicht eine einfachere Handhabung der Komplexität bei der Entwicklung von verteilten Softwaresystemen. Sie führt zudem zu einer geringeren strukturellen Kopplung zwischen den Elementen des Softwaresystems: Indem das Wissen über die Problemstellung mittels Zielen in die Agenten integriert wird, können Entscheidungen an der Stelle gefällt werden, an der die meisten Informationen über ein Problem vorliegen bzw. durch dynamische Interaktionen entstehen. Dabei besteht die Möglichkeit, die beste Lösung für ein bestehendes Problem dynamisch unter Beteiligung anderer Agenten zu ermitteln.

Ein entscheidender Unterschied des agentenorientierten Ansatzes zur herkömmlichen Softwareentwicklung ist die Tatsache, dass die Gesamtheit der Systemstruktur und des Systemverhaltens nicht zwingend zur Entwurfszeit vollständig spezifiziert werden muss, sondern sich zur Laufzeit auf Basis der aktuellen Situation dynamisch bildet und flexible Interaktionen im Rahmen festgelegter Variationen stattfinden können. Durch eine agentenorientierte Entwicklung kann die Berücksichtigung der Menge möglicher Abläufe im Gesamtverhalten des Softwaresystems aus der Entwurfsphase in die Laufzeit verschoben und so die Komplexität des Entwurfes reduziert werden [Jenn00]. Diese Reduktion der Komplexität ist darin begründet, dass zur Entwurfszeit lediglich die Aufgabenbereiche der einzelnen Agenten und die Möglichkeiten ihrer Interaktionen vorgegeben werden müssen. Somit müssen nicht alle möglichen Einzelsituationen betrachtet und damit nicht der vollständige Lösungsraum spezifiziert werden. Das Gesamtverhalten ergibt sich aus der Kombination der konkreten Einzelverhaltensweisen der Agenten und der konkreten Interaktionen zwischen ihnen. In diesen Punkten folgt die Agentenorientierung in besonderem Maße dem Prinzip der Modularisierung und verbessert somit auch allgemeine Qualitätsmerkmale wie Skalierbarkeit, Änderbarkeit, Erweiterbarkeit und Wiederverwendbarkeit [Jenn00]. Aufgrund dieser Vorteile ist die Agentenorientierung insbesondere für Problemdomänen geeignet, die folgende charakteristische Merkmale aufweisen [Paru98], [Weiß01]:

- *Natürliche logische Verteilung*: Eine Anwendung ist in abgrenzbare „Prozesse“ aufteilbar, die unterschiedliche Aufgaben erfüllen, jedoch Abhängigkeiten untereinander besitzen.

- *Strukturelle Veränderbarkeit*: Die möglichen Wechselbeziehungen zwischen den Elementen eines Softwaresystems sind nicht vollständig (oder nur mit großem Aufwand) zur Entwicklungszeit spezifizierbar oder zur Laufzeit besonders starken Änderungen unterworfen.

- *Komplexe, variable Abläufe*: Die vollständige Festlegung des Systemverhaltens als Kette von Einzelschritten ist zur Entwicklungszeit ist nicht praktikabel, es besteht die Notwendigkeit zur flexiblen und dynamischen Anpassung der Abläufe zur Laufzeit.

- *Umfangreiche Koordinationsprozesse*: Zur flexiblen Verteilung von Ressourcen, Aufgaben oder Leistungen sowie zur Handhabung von Wechselwirkungen oder gegensätzlichen Interessen einzelner Systemelemente sind umfangreiche Koordinationsprozesse erforderlich.

Die Flexibilität agentenorientierter Systeme ist mit dem Nachteil einer eingeschränkten Vorhersagbarkeit und Analysierbarkeit verbunden. Die Verteilung von Entscheidungsprozessen auf autonome Einheiten führt zu einer Variabilität im Verhalten des Agentensystems, die nicht vollständig a priori vorhersagbar ist. Es gibt verschiedene Möglichkeiten, die Variabilität des Systemverhaltens zu beschränken und somit unerwünschten Effekten geeignet zu begegnen. Zum einen können mögliche Interaktionen zwischen Agenten durch Interaktionsprotokolle sowie durch die Einschränkung möglicher organisatorischer Strukturen begrenzt und parallele Abläufe durch Synchronisierungsmechanismen koordiniert werden. Zum anderen können globale Beschränkungen in Form von zusätzlichen Zielen der Agenten vorgegeben werden [UWG+04].

4.5 Einsatz von Softwareagenten in der Automatisierungstechnik

Wie im vorangehenden Abschnitt deutlich wurde, ist besonderes Potenzial beim Einsatz von Softwareagenten überall dort zu erwarten, wo in verteilten Strukturen lokale Dynamik und Selbstständigkeit von Bedeutung sind und zugleich vielfältige, systemübergreifende Abhängigkeiten zwischen Teilprozessen das Verhalten des Gesamtsystems wesentlich beeinflussen. Für die dezentrale, heterogene Welt der Automatisierungstechnik, die durch zunehmend komplexere Abläufe sowie die Notwendigkeit zu mehr Flexibilität, Robustheit, Skalierbarkeit, Anpassbarkeit und Integrationsfähigkeit geprägt ist, erscheint dieser Ansatz besonders geeignet. Mithilfe von Agenten ist es möglich, die notwendige Flexibilität und Anpassungsfähigkeit von Automatisierungssystemen systematisch zu entwerfen. Damit kann ein längst fälliger Wechsel der Strukturen in der Automatisierungstechnik vollzogen werden – weg von hierarchischen, statischen Systemen, hin zu flexiblen, dezentralen Netzwerken aus autonomen, kooperierenden Elementen [Happ04]. Die folgenden Anwendungsbeispiele verdeutlichen, wie Lösungen auf Basis eines agentenorientierten Ansatzes vorteilhaft in technischen Systemen eingesetzt werden:

- *Agentenorientiertes Produktionsleitsystem:* Heutige Produktionsanlagen unterliegen Anforderungen an eine große Stückzahl- und Variantenflexibilität, hohe Produktivität und Verfügbarkeit durch Robustheit gegenüber Fehlern. Ein agentenorientierter Ansatz bietet hier eine dezentrale, flexible Lösung der Ablaufplanung: Produktionsstationen, Werkstücke und Transporteinheiten werden durch Agenten repräsentiert, welche den Produktionsablauf entsprechend der einlaufenden Werkstücke, dem Auslastungsgrad und dem Fehlerzustand der Anlage dynamisch koordinieren [Buss98]. Ansätze und Projekte zu agentenorientierten Produktionsleitsystemen werden in [Paru98], [ShNo99], [TöWo01], [KlLü03], [BCB+01] und

[Pöss06] präsentiert. Im Rahmen einer industriellen Realisierung wurde die höhere Flexibilität und Verfügbarkeit agentenorientierter Produktionsleitsysteme nachgewiesen [SuBu01].

- *Agentenorientiertes Energiemanagement für Kraftfahrzeuge*: In modernen Kraftfahrzeugen ist ein Energiemanagement erforderlich, welches die Sicherheit der Energieversorgung, insbesondere für sicherheitsrelevante elektronische Systeme, gewährleistet. Ein agentenorientierter Ansatz bietet eine Lösung mit dezentraler Entscheidungsfindung. Dabei wird jedes Verbrauchersteuergerät durch einen autonomen Agenten vertreten. Die Verteilung der verfügbaren elektrischen Energie erfolgt nicht mehr zentral und statisch wie bei herkömmlichen Energiemanagementsystemen, sondern durch dynamische Verhandlungen zwischen den verteilten Verbraucher-Agenten. Damit kann ohne die Notwendigkeit einer redundanten zentralen Entscheidungseinheit die erforderliche hohe Sicherheit der Energieversorgung bei gleichzeitiger großer Flexibilität und Skalierbarkeit gewährleistet werden [HWH04].

Weitere Einsatzbeispiele für Softwareagenten in der Automatisierungstechnik sind Prozessüberwachungs- und Diagnosesysteme [WaWa97], [PlWe99], [KNS99], [Fodo02], [KMC+03], Selbstkonfiguration von Produktionsanlagen [Dete03], [PDS03], Robotik [Levi05], [Raff05] und selbstoptimierende verteilte Regelungs- und Steuerungssysteme [SFB614], [Gau05].

In diesem Kapitel wurde das Paradigma der Softwareagenten vorgestellt und erläutert. Auf Basis des agentenorientierten Ansatzes können flexible Lösungen für komplexe Problemstellungen entwickelt werden. Die genannten Beispiele aus der Automatisierungstechnik veranschaulichen, dass Softwareagenten erfolgreich für den flexiblen, anpassungsfähigen Betrieb von Automatisierungssystemen eingesetzt werden können. Ein weiteres Anwendungsgebiet ist die Unterstützung der Systementwicklung, d.h. des Engineerings selbst [Paru98]. Das diesbezügliche Potenzial von Softwareagenten wird in der aktuellen Trendprognose des Feldafinger Kreises zu industriell relevanten Entwicklungen hervorgehoben: Softwareagenten können Routineaufgaben übernehmen, indem sie komplexe, verteilte Umgebungen und Prozesse integrieren, beobachten und analysieren sowie durch Planung, Entscheidungsfindung und Verhandlung dem Menschen assistieren [WaWe05]. In dieser Aussage wird das *Auftragsprinzip* als wichtiges Merkmal für die Einsatzmöglichkeiten von Softwareagenten deutlich: Sie übernehmen oder unterstützen komplexe menschliche Aufgaben mit Routinecharakter. Ein erfolgreiches Beispiel hierfür sind agentenorientierte Logistiksysteme, bei denen menschliche Dispositionsaufgaben von Softwareagenten übernommen werden [Dann06].

Im folgenden Kapitel werden die wesentlichen Problemstellungen beim Engineering herausgearbeitet und Ansätze für eine geeignete Unterstützung ermittelt. Darauf aufbauend wird ein Konzept entwickelt, das auf dem agentenorientierten Paradigma basiert und eine umfassende Unterstützung des Engineerings ermöglicht.

5 Konzept des agentenunterstützten Engineerings

Ziel einer Unterstützung des Engineerings ist die Reduktion von manuellem Aufwand und Komplexität bei der Erstellung von Anlagenmodellen. In diesem Kapitel wird ausgehend von einer Betrachtung der Problematik der Komponentenintegration ein Ansatz für ein flexibles Unterstützungskonzept zur Entlastung des Ingenieurs ermittelt. Darauf aufbauend wird das Konzept des agentenunterstützten Engineerings eingeführt und erläutert. Zunächst werden die Grundidee der Kapselung von Komponenten durch Softwareagenten und die wesentlichen Eigenschaften des Konzepts erarbeitet. Es wird die Notwendigkeit zur Nutzung von Informationen und Wissen des Ingenieurs für die Unterstützung diskutiert und ein Ansatz auf der Grundlage wissensbasierter Konzepte entwickelt. Anschließend wird die Unterstützung weiterer komponentenübergreifender Aufgaben des Ingenieurs durch das Konzept betrachtet. Abschließend werden die Wirkungsweise des Konzepts anhand von Anwendungsfällen veranschaulicht und wichtige Aspekte bezüglich der Umsetzung des Konzepts angesprochen: der erforderliche Aufbau von Softwareagenten und die Vorgehensweise bei der Entwicklung eines agentenorientierten Modells der Unterstützung.

5.1 Automatisierung der Komponentenintegration

Der komponentenbasierte Ansatz erhöht die Produktivität und die Qualität beim Engineering und hat sich in der Praxis bewährt. Er soll daher als Ausgangspunkt dieser Arbeit dienen und durch ein geeignetes Unterstützungskonzept ergänzt werden. Im komponentenbasierten Ansatz erfolgt die Erstellung der Engineeringinformationen durch Auswahl, Instanziierung, Anpassung und Verbindung einzelner Komponenten im Anlagenmodell. Komponenten sind die *Handlungsobjekte* bei den Tätigkeiten des Ingenieurs. Die wesentliche Schwierigkeit besteht dabei in der Problematik der Komponentenintegration, weil sich die Komponenten gegenseitig beeinflussen: Bei jeder Verbindung und Anpassung von Komponenteninstanzen können aufgrund von technischen Abhängigkeiten Wechselwirkungen entstehen, welche sich auf andere Komponenteninstanzen auswirken. Diese müssen für die Integration jeder einzelnen Komponenteninstanz dahingehend geprüft und gehandhabt werden, dass die Konsistenz aller Engineeringinformationen des Anlagenmodells sichergestellt ist. Die dazu erforderlichen Tätigkeiten sind mit hohem manuellen Aufwand und hoher Komplexität verbunden.

Durch eine Automatisierung der Komponentenintegration kann der manuelle Aufwand und die Komplexität im Engineering reduziert und somit eine wesentliche Entlastung des Ingenieurs erreicht werden. Dies erfordert die Einführung aktiver Unterstützungsfunktionalität, welche bei der Komponentenintegration, d.h. bei jeder Verbindung oder Anpassung einer Komponentenin-

stanz die erforderlichen Prüf- und Anpassungsschritte selbstständig erkennt und ausführt. Abbildung 5.1 stellt die Automatisierung der Komponentenintegration beispielhaft dar.

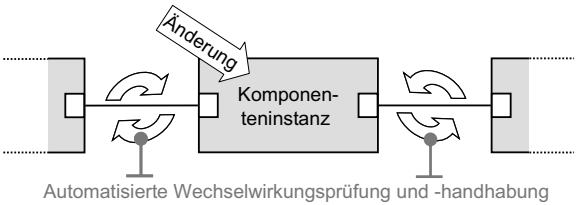


Abbildung 5.1: Automatisierung der Komponentenintegration

Bisher werden Komponenten beim Engineering als informationstechnische Beschreibungsbausteine eingesetzt, welche alle für die Realisierung einer Anlage erforderlichen technischen Informationen beinhalten. Für eine Automatisierung der Komponentenintegration beim Engineering muss daher die Komponentenebene um die benötigte aktive Unterstützungsfunktionalität erweitert werden (siehe Abbildung 5.2). Folgende Aufgaben müssen durch die Unterstützungsfunktionalität bei der Integration einer Komponente bewerkstelligt werden:

- Ermitteln der Auswirkungen auf andere Komponenteninstanzen des Anlagenmodells
- Überprüfen der Wechselwirkungen auf Inkonsistenzen
- Anpassen der Engineeringinformationen der betroffenen Komponenteninstanzen

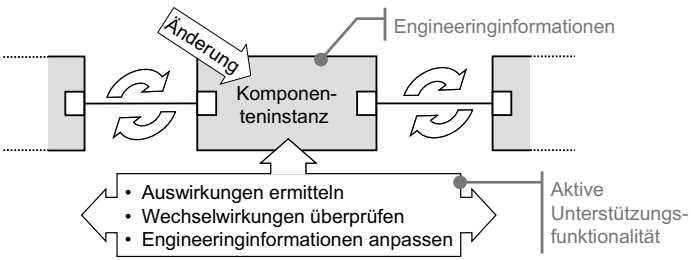


Abbildung 5.2: Erweiterung der Komponentenebene um aktive Unterstützungsfunktionalität

Hierbei muss betont werden, dass die einzelnen Aufgaben der Unterstützungsfunktionalität nicht unabhängig voneinander sind, da durch Anpassungen weitere Wechselwirkungen entstehen können, die wiederum ermittelt, geprüft und durch Anpassungen gehandhabt werden müssen.

Randbedingungen für das Unterstützungskonzept

Die Unterstützungsfunktionalität zur automatisierten Komponentenintegration muss in die menschlichen Handlungsprozesse beim Engineering integriert werden. Kennzeichnend für das komponentenbasierte Engineering sind dabei die graduelle Änderung der Engineeringinformationen und die Dynamik des Ablaufs bei der Erstellung der Engineeringinformationen:

- Beliebige Komponenten werden hinzugefügt oder gelöscht.
- Beliebige Komponenten werden verbunden oder Verbindungen gelöscht.
- Beliebige Komponenten werden geändert.

Diese Variabilität ist für die Unterstützung insofern von besonderer Bedeutung, als dass sowohl die Informationsbasis der Unterstützung als auch die möglichen Wirkungszusammenhänge von Komponentenabhängigkeiten nicht a priori bestimmbar sind, sondern sich im Verlauf der Erstellung des Anlagenmodells beständig verändern. Daraus ergeben sich folgende Randbedingungen für das Unterstützungskonzept:

- Die Unterstützung muss unabhängig von den verfügbaren Komponenten der Komponentenbibliothek und unabhängig von den im Anlagenmodell verwendeten Komponenten erfolgen.
- Die Unterstützung muss sich permanent an den aktuellen Entwicklungsstand und an die Vorgehensweise des Ingenieurs anpassen, selbstständig die aktuelle Problemstellung erkennen und mit der jeweils passenden Unterstützungsfunktionalität eingreifen.
- Die Unterstützung muss Hinweise und Lösungsvorschläge an beliebige Problemstellungen angleichen können, sodass deren Bedeutung und Zusammenhang mit der aktuellen Tätigkeit des Ingenieurs ohne zusätzliche Überlegungen nachvollziehbar sind.
- Die Unterstützung muss in der Lage sein, ihr Verhalten an Vorgaben des Ingenieurs anzupassen. Lösungsvorschläge und dazugehörige Eingriffe dürfen nur mit Zustimmung des Ingenieurs wirksam werden, andernfalls müssen sie vollständig verworfen werden.

Zusammenfassend besteht die Notwendigkeit, die Unterstützung permanent an die sich ändernde Informationsbasis und die sich ändernde Unterstützungssituation anzupassen, ohne externe Konfiguration oder Kontrolle zu erfordern. Ein Unterstützungskonzept muss diese dynamischen Veränderungen berücksichtigen und die dafür notwendige Flexibilität gewährleisten.

5.2 Ansatz eines flexiblen Unterstützungskonzepts

Unter Flexibilität wird allgemein die Fähigkeit zur Anpassung an die Gegebenheiten der Umwelt verstanden. Übertragen auf die Unterstützungsproblematik bedeutet Flexibilität also die Möglichkeit, die Unterstützung im Verlauf des Engineerings an den aktuellen Unterstützungskontext anzupassen. Betrachtet man die Eigenschaften des komponentenbasierten Engineerings, ergeben sich verschiedene Aspekte hinsichtlich der Flexibilität des Unterstützungskonzepts:

- Ein komponentenbasiertes Anlagenmodell weist eine verteilte Struktur aus einzelnen Komponenteninstanzen auf, die individuell aufgebaut wird und graduellen, variablen Änderungen unterworfen ist. Entsprechend ist eine *flexible Anpassung* der Unterstützung *an die strukturellen Veränderungen des Anlagenmodells* erforderlich.

- Die dynamische Änderung von Wechselwirkungen zwischen den Komponenteninstanzen des Anlagenmodells, abhängig von ihren spezifischen Konfigurationen und Kombinationen, erfordert die *flexible Anpassung* der Unterstützung *an die Abläufe bei der Komponentenintegration*. Hierbei ist zu beachten, dass diese Anpassung der Abläufe in *koordinierter Weise* erfolgen muss, um sämtliche Änderungen und daraus wiederum entstehende Wechselwirkungen konsistent zu handhaben.

Die Herausforderung bei der Entwicklung des Unterstützungskonzepts besteht in der Bewerkstelligung der erforderlichen Abläufe und Verhaltensweisen der Unterstützung, insbesondere, wenn diese sich flexibel an die aktuelle Unterstützungssituation anpassen soll. Der Ansatz einer statischen Lösung, bei der Struktur und Verhalten der Unterstützungsfunktionalität fest vorgegeben werden müssen, würde bedingen, dass alle möglichen Unterstützungssituationen im Voraus berücksichtigt werden müssen. Dies hat einen hohen Aufwand und hohe Komplexität bei der Entwicklung des Unterstützungskonzepts zur Folge.

Um diese Nachteile zu vermeiden, muss das Unterstützungskonzept auf einem Ansatz aufbauen, der eine flexible und anpassungsfähige Lösung ermöglicht. Für dieses Problem sind aus der Softwaretechnik Ansätze bekannt, welche die Entwicklung flexibler, anpassungsfähiger Lösungen ermöglichen, ohne die Vielfalt aller möglichen Betriebssituationen bei der Entwicklung berücksichtigen und konkrete Reaktionen für jede mögliche Betriebssituation spezifizieren zu müssen. Stattdessen wird die Gesamtaufgabe in entkoppelbare Teilaufgaben zerlegt und unabhängigen funktionalen Einheiten zugeordnet. Zudem werden Mechanismen für das Zusammenspiel der funktionalen Einheiten spezifiziert, welche es ermöglichen, dass Einzelreaktionen auf eine Betriebssituation dynamisch kombiniert werden und in ein zielgerichtetes Gesamtverhalten münden. Infolgedessen muss nicht mehr der gesamte Lösungsraum spezifiziert werden, sondern lediglich generische Teillösungen, welche sich zur Laufzeit *eigenständig* entsprechend der aktuellen Betriebssituation zusammenfügen. Die Komplexität der Entwicklung einer flexiblen Lösung wird somit reduziert und beherrschbarer gemacht [Jenn00].

In der Softwaretechnik stehen diese Aspekte im Zentrum des Paradigmas der agentenorientierten Softwareentwicklung, das in Kapitel 4 vorgestellt wurde. Der agentenorientierte Ansatz bietet Hilfsmittel für die systematische Entwicklung von Lösungen, die Flexibilität und Selbstanpassungsfähigkeit im laufenden Betrieb aufweisen und eignet sich für Problembereiche mit den charakteristischen Merkmalen *natürliche logische Verteilung*, *strukturelle Veränderbarkeit*, *komplexe, variable Abläufe* und *umfangreiche Koordinationsprozesse*. Damit deckt der agentenorientierter Ansatz alle oben genannten Eigenschaften eines flexiblen Unterstützungskonzepts für das Engineering ab. Der in Kapitel 4.5 formulierte Grundgedanke, Softwareagenten zur Unterstützung des Menschen einzusetzen, indem sie komplexe, verteilte Umgebungen integrieren sowie durch Entscheidungsfindung und Verhandlung assistieren, stellt somit einen geeigneten Ausgangspunkt dar, um ein Unterstützungskonzept für das Engineering zu entwickeln. Es soll daher als „agentenunterstütztes Engineering“ bezeichnet werden.

5.3 Beschreibung des agentenunterstützten Engineerings

5.3.1 Einsatz von Softwareagenten zur aktiven Unterstützung der Komponentenintegration

In den vorangehenden Abschnitten wurden die Automatisierung der Komponentenintegration als Gegenstand der Unterstützung und das Paradigma der Softwareagenten als Basis für ein flexibles Unterstützungskonzept identifiziert. Im Folgenden wird erläutert, wie Softwareagenten eingesetzt werden können, um als aktive, selbstständige Auftragnehmer des Ingenieurs bei der Komponentenintegration die Erkennung und Handhabung der Komponentenabhängigkeiten vollständig zu übernehmen.

5.3.1.1 Vertretung von Komponenteninstanzen durch Softwareagenten

Für eine Integration der Unterstützung in die menschlichen Handlungsprozesse müssen automatische Prüf- und Anpassungsschritte tätigkeitsbezogen erfolgen. Da die menschlichen Tätigkeiten sich stets auf die Handhabung einzelner Komponenten beziehen, muss *jede einzelne Komponente* im Anlagenmodell um die benötigte aktive Unterstützungsfunktionalität erweitert werden. Dabei sollen die bisherigen Komponenten nicht verändert werden, weil dies die Abkehr von bestehenden Komponentenbibliotheken und Engineering Systemen und somit einen erheblichen Hinderungsgrund für den Einsatz des Unterstützungskonzepts bedingen würde. Dies kann vermieden werden, wenn die benötigte aktive Unterstützungsfunktionalität einer Komponente durch einen Softwareagenten als aktive funktionale Einheit, welche die Komponente in ihrer bisherigen Form ergänzt, gekapselt wird. Daraus ergibt sich die Grundidee des agentenunterstützten Engineerings, die in Abbildung 5.3 grafisch veranschaulicht ist:

- Jede Komponenteninstanz des Anlagenmodells wird durch einen Softwareagenten vertreten und ihre Engineeringinformationen durch den Softwareagenten gekapselt.
- Der Softwareagent kapselt die aktive Unterstützungsfunktionalität zur Integration der Komponenteninstanz.
- Er führt alle erforderlichen Prüf- und Anpassungsschritte selbstständig durch.

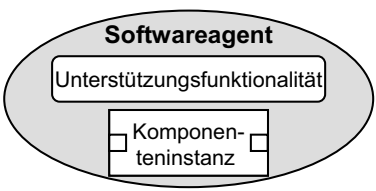


Abbildung 5.3: Kapselung von Komponenteninstanzen durch Softwareagenten

Aus dieser zentralen Idee und den weiteren Anforderungen an das Unterstützungskonzept ergeben sich folgende Grundprinzipien des Konzepts des agentenunterstützten Engineerings:

- *Auslagerung der Unterstützungsfunktionalität*: Die Unterstützungsfunktionalität wird aus den Komponenten des Anlagenmodells heraus in eine Unterstützungsebene verlagert und dort durch Softwareagenten gekapselt. Damit kann die Unterstützungsfunktionalität im Hintergrund der menschlichen Tätigkeiten ablaufen und es können temporäre Lösungen auf der Unterstützungsebene entwickelt werden, die mit dem Ingenieur abgestimmt und abhängig von seiner Entscheidung auf der Komponentenebene umgesetzt oder verworfen werden.

- *Modularisierung der Unterstützungsfunktionalität*: Die Unterstützungsfunktionalität wird modularisiert, d. h. in einzelne Softwareagenten zerlegt, die unabhängig voneinander agieren und zur Abstimmung von Unterstützungsprozessen zusammenarbeiten. Eine solche Modularisierung ermöglicht die flexible Anpassung der Unterstützungsebene an die strukturellen Änderungen des Anlagenmodells und die Änderungen von Komponentenwechselwirkungen.

- *Abbildung der Charakteristika der Komponentenintegration*: Die Ausprägung der Eigenschaften der Unterstützungsebene erfolgt durch die Abbildung aller grundlegenden Charakteristika der Komponentenintegration auf die Eigenschaften von Softwareagenten. Dadurch werden die Softwareagenten in die Lage versetzt, konkrete Problemstellungen aktiv zu erkennen und die situationsspezifische Bestimmung der geeigneten Unterstützungsfunktionalität und ihres Ablaufs vorzunehmen.

- *Nutzung von Informationen und Wissen*: Die bei den Aufgaben zur Komponentenintegration benötigten Informationen und das benötigte Wissen werden ermittelt und den Softwareagenten zugeordnet. Dadurch wird eine selbstständige Durchführung der Unterstützungsaufgaben durch die Softwareagenten ermöglicht.

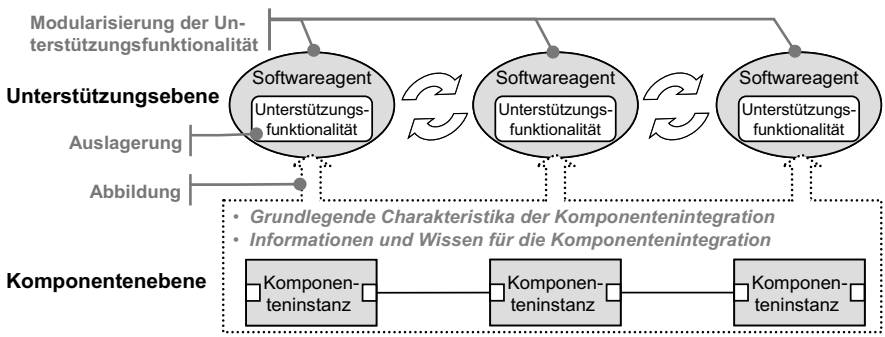


Abbildung 5.4: Grundprinzipien des Konzepts des agentenunterstützten Engineerings

5.3.1.2 Modularisierung der Unterstützungsfunktionalität

Die Aufgaben der Unterstützungsfunktionalität, *Ermitteln der Auswirkungen von Änderungen*, *Überprüfen der Wechselwirkungen* und *Anpassen der Engineeringinformationen*, müssen auf der Unterstützungsebene bewerkstelligt werden. Um die Flexibilität der Unterstützung zu gewährleisten, müssen diese Aufgaben so modularisiert und auf Softwareagenten übertragen werden, dass jeder Softwareagent über lokale Selbstständigkeit bei der Handhabung der von ihm vertretenen Komponenteninstanz verfügt. Lokale Selbstständigkeit ist bei denjenigen Aufgaben der Unterstützung gegeben, bei denen nur die Eigenschaften einzelner Komponenteninstanzen betrachtet werden. Demnach werden jedem Softwareagenten die Unterstützungsaufgaben *Überprüfen* und *Anpassen* der vertretenen Komponenteninstanz übertragen, zusammen mit den Zielen hinsichtlich der Sicherstellung der Konsistenz ihrer Eigenschaften. Diese Aufgaben können von ihm unabhängig von anderen Softwareagenten durchgeführt werden. Abhängigkeiten treten dann auf, wenn bei der Änderung einer Komponenteninstanz Wechselwirkungen zu anderen Komponenteninstanzen entstehen, die berücksichtigt werden müssen. Die Unterstützungsaufgabe *Ermittlung der Auswirkungen* muss daher auf die Interaktionen der Softwareagenten abgebildet werden und wird durch Kooperation zwischen den Softwareagenten bewerkstelligt. Die Ziele der Softwareagenten hinsichtlich der Sicherstellung der Konsistenz müssen dazu um die Betrachtung der Auswirkungen auf andere Komponenteninstanzen erweitert werden. Abbildung 5.5 veranschaulicht die Modularisierung der Unterstützungsfunktionalität.

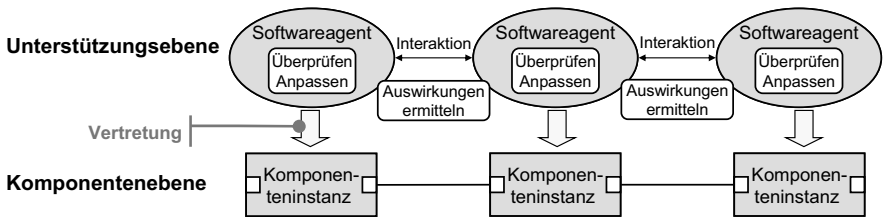


Abbildung 5.5: Modularisierung der Unterstützungsfunktionalität

Die Strukturierung der Unterstützungsebene muss die Basis für die Handhabung der Komponentenabhängigkeiten bereitstellen und zudem die flexible Anpassung der Unterstützungsebene ermöglichen. Dies wird erreicht, indem die Beziehungen zwischen Softwareagenten aus den Strukturen des Anlagenmodells und den Komponentenabhängigkeiten abgeleitet werden. Dadurch werden die Komponentenabhängigkeiten auf der Unterstützungsebene explizit sichtbar. Auf dieser Basis können die Softwareagenten durch Interaktionen die bei der Integration einzelner Komponenten entstehenden Wechselwirkungen prüfen und alle erforderlichen zusätzlichen Anpassungen ermitteln und abstimmen. Die dabei benötigte dynamische Anpassung der Abläufe geschieht mittels der Agenteninteraktionen. Die flexible Anpassung der Unterstützungsebene an Veränderungen des Anlagenmodells und an Veränderungen der Komponentenwechselwirkungen erfolgt durch die dynamische Anpassung der Beziehungen zwischen den Softwareagenten.

5.3.1.3 Abbildung der Charakteristika der Komponentenintegration

Wenn die Unterstützungsebene ihre Unterstützungsaufgaben aktiv wahrnehmen soll, muss sie Eigenschaften aufweisen, die alle Aspekte der Komponentenintegration abdecken. Damit die Unterstützung unabhängig von den verwendeten Komponenten erfolgen kann, müssen diese Eigenschaften aus den grundlegenden Charakteristika der Komponentenintegration und nicht aus spezifischen Belangen einzelner Komponenteninstanzen abgeleitet werden. Zur Ausprägung der Eigenschaften stehen die in Kapitel 4.2.1 eingeführten Grundkonzepte von Softwareagenten zur Verfügung. Die entsprechenden Ausprägungen der Agenteneigenschaften im Unterstützungskonzept sind in der folgenden Tabelle dargestellt.

Tabelle 5.1: Ausprägungen der agentenorientierten Grundkonzepte im Unterstützungskonzept

Agentenorientierte Grundkonzepte	Ausprägung der Agenteneigenschaften im Unterstützungskonzept
Zielorientierung	Ziele zur Sicherstellung der Konsistenz bei der Integration der vertretenen Komponenteninstanz durch Berücksichtigung aller Wechselwirkungen
Kapselung	Tätigkeitsrelevante Informationen über die vertretene Komponenteninstanz und Wissen über Abhängigkeiten Fähigkeiten zur Prüfung und Anpassung der Komponenteninstanz
Autonomie	Selbstständige Handhabung der lokalen Prüfungen und Anpassungen der Engineeringinformationen der vertretenen Komponenteninstanz
Aktivität	Erkennen menschlicher Tätigkeiten und der entstehenden Änderungen Selbstständige Ermittlung von Wechselwirkungen sowie von Lösungen zu ihrer Handhabung
Interaktion	Systematische, schrittweise Ermittlung der Änderungsauswirkungen, Identifikation der Wechselwirkungen und Abstimmung von Anpassungen Kooperation mit dem Ingenieur zur interaktiven Problemlösung
Persistenz	Berücksichtigung der Historie von Änderungen und Wechselwirkungen Kennen des inneren Zustands, d.h. des Verlaufs der Aufgabenerfüllung

Die ermittelten Agenteneigenschaften im Unterstützungskonzept bilden alle Aspekte der Unterstützung der Komponentenintegration ab und dienen als Grundlage für die Entwicklung eines agentenorientierten Modells der Unterstützung, welche in Kapitel 7 erfolgt.

5.3.1.4 Nutzung von Informationen und Wissen des Ingenieurs

Für eine selbstständige Durchführung aller Aufgaben der Komponentenintegration müssen auf der Unterstützungsebene folgende Informationen und folgendes Wissen verfügbar sein:

- Informationen über die Parameter und Schnittstellen von Komponenten
- Informationen über die Konfiguration und die Verbindungen von Komponenteninstanzen

- Wissen über die technischen Abhängigkeiten von Komponenten

Selbstständigkeit ist zudem nur dann gegeben, wenn die erforderlichen Informationen und das Wissen zum Zeitpunkt einer Unterstützungshandlung bereits vorliegen bzw. selbstständig abgeleitet werden können und nicht vom Ingenieur erfragt werden müssen. Informationen und Wissen müssen daher rechnergestützt abgebildet und durch die Softwareagenten gekapselt werden.

Abbildung der Informationen über Komponenteninstanzen

Die benötigten Informationen über Komponentenparameter und -schnittstellen sind bereits in der Komponentenbeschreibung abgebildet und können von den Softwareagenten übernommen werden. Die Informationen über die Konfigurationen und Verbindungen einzelner Komponenteninstanzen sind im Anlagenmodell abgebildet und können ebenfalls übernommen werden. Im Gegensatz zu den Komponenteninformationen werden die Informationen über Komponenteninstanzen bei den Tätigkeiten des Ingenieurs verändert. Sie müssen daher bei jeder Änderung einer Komponenteninstanz bei den betreffenden Softwareagenten aktualisiert werden.

Abbildung des Wissens über technische Abhängigkeiten von Komponenten

Wie in Kapitel 2.5 deutlich wurde, ist das Wissen über technische Abhängigkeiten von Komponenten bisher nicht informationstechnisch abgebildet. Durch eine informationstechnische Abbildung des Wissens darf die Unabhängigkeit der Komponenten nicht eingeschränkt werden. Das Wissen muss daher komponentenbezogen und verwendungsunabhängig abgebildet werden. Zudem muss es so spezifiziert werden, dass eine selbstständige Bezugsetzung zwischen Informationen und zugehörigem Wissen möglich wird. Diese Anforderungen werden durch wissensbasierte Konzepte erfüllt. Deshalb sieht das agentenunterstützte Engineering vor, das Wissen über die technischen Abhängigkeiten einer Komponente unter Zuhilfenahme wissensbasierter Konzepte abzubilden, in expliziter Form zu spezifizieren und mit den Komponenteninformationen zu integrieren.

Ein geeigneter Ansatzpunkt zur Abbildung des Wissens über die technischen Abhängigkeiten einer Komponente ist die *explizite* Beschreibung ihrer internen Abhängigkeiten (Anpassungsmöglichkeiten einer Komponente) und ihrer externen Abhängigkeiten (Verbindungsmöglichkeiten zu anderen Komponenten) [CzEi00], [KoCa00]. Um sie abzubilden, ist die Komponentenbeschreibung um die Spezifikation von *Konsistenzbedingungen* zu erweitern. Konsistenzbedingungen dokumentieren die Voraussetzungen, deren Einhaltung für die korrekte und bestimmungsgemäße Funktion einer Komponente in einem Verwendungskontext erforderlich ist. Um dabei das Merkmal der strukturellen Unabhängigkeit von Komponentenbeschreibungen zu erhalten, darf die Spezifikation von Kontextbedingungen für externe Abhängigkeiten nicht mittels struktureller Verweise auf einen spezifischen Verwendungskontext, d.h. auf andere Komponenten, erfolgen. Stattdessen müssen die Abhängigkeiten zwischen Komponenten indirekt ausgedrückt werden. Ein geeignetes Konzept ist die Spezifikation von Konsistenzbedingungen in

Form anonymer Eigenschaften, welche die Komponente von ihrer (noch unbekannten) Umgebung erfordert [SMBV03]. Für jede Instanz dieser Komponente gilt dann: Verbundene Komponenteninstanzen müssen Eigenschaften aufweisen, welche die Konsistenzbedingungen erfüllen.

Dabei ist zu berücksichtigen, dass technische Abhängigkeiten abhängig von der Konfiguration einer Komponenteninstanz relevant oder irrelevant sein können. Wurde beispielsweise ein Förderelement für einen stetigen Materialtransport konfiguriert, dürfen nachfolgende Förderelemente den Weitertransport nicht blockieren. Im Fall einer solchen externen Abhängigkeit einer Komponente wird die als Konsistenzbedingung definierte Eigenschaft von der Umgebung nur dann gefordert, wenn komponentenintern eine bestimmte Voraussetzung (Vorbedingung) erfüllt ist, d.h. wenn ein Komponentenparameter einen bestimmten Wert aufweist. Um diesen Umstand auszudrücken, werden Konsistenzbedingungen in Regeln eingebettet, die ihre Gültigkeit festlegen. Komponenteninterne Abhängigkeiten, d.h. die Abhängigkeit eines Parameters vom Wert eines anderen Parameters derselben Komponente, werden ebenfalls durch Regeln spezifiziert. Somit können technische Abhängigkeiten durch zwei Repräsentationsformen explizit, d.h. formalisiert abgebildet werden (siehe Abbildung 5.6):

- Verbindungsregeln beschreiben die externen Abhängigkeiten einer Komponente von ihrer Umgebung und legen somit die Verbindungsmöglichkeiten mit anderen Komponenten fest.
- Parametrierregeln beschreiben die internen Abhängigkeiten zwischen Komponentenparametern und legen somit die Anpassungsmöglichkeiten der Komponentenkonfiguration fest.

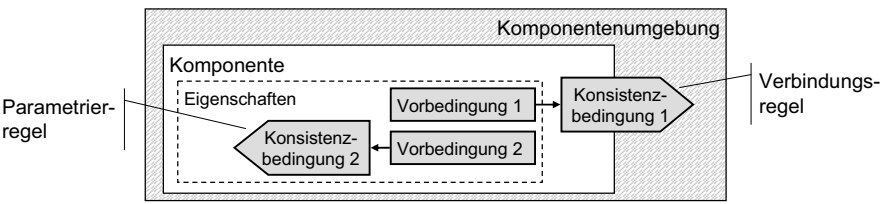


Abbildung 5.6: Regelformen zur Beschreibung von technischen Abhängigkeiten

Durch diese Abbildung unter Verwendung von Konsistenzbedingungen und Regeln erfolgt die Spezifikation des Wissens über technische Abhängigkeiten komponentenbezogen, und die Komponentenunabhängigkeit bleibt gewährleistet. Das Wissen kann auf formalisierte Weise spezifiziert und mit den Informationen über die Parameter und Schnittstellen in der Komponentenbeschreibung integriert werden. Es wird beim Engineering explizit verfügbar und kann durch die Softwareagenten aus der Komponentenbeschreibung übernommen werden. Damit kapselt ein Softwareagent neben den Informationen über die vertretene Komponenteninstanz auch das Wissen über ihre technischen Abhängigkeiten. Bei der Komponentenintegration verknüpft er Wissen und Informationen situationsspezifisch, um Wechselwirkungen zu identifizieren und Anpassungen zu ermitteln.

5.3.1.5 Übertragbarkeit des Konzepts auf die Integration von Baugruppen

Ein komponentenbasiertes Anlagenmodell kann nicht nur aus einzelnen Komponenteninstanzen bestehen, sondern auch aus Baugruppen, die eine Menge verbundener Komponenteninstanzen auf einer höheren Hierarchieebene zusammenfassen. Die Wiederverwendung von Baugruppen als Teillösungen zielt auf die Nutzbarmachung von bereits erfolgreich eingesetzten Komponentenkombinationen ab. Da sich Baugruppen in der Außensicht nicht von einzelnen Komponenten unterscheiden, werden sie wie Komponenten verwendet und gehandhabt. Dabei besteht dieselbe Integrationsproblematik wie bei einzelnen Komponenten: Entstehende Wechselwirkungen zwischen Baugruppen müssen geprüft und gehandhabt werden. Die in den vorangegangenen Unterkapiteln erläuterten Grundprinzipien des agentenunterstützten Engineerings können daher in gleicher Weise zur Automatisierung der Baugruppenintegration angewendet werden:

- Baugruppen werden wie Komponenteninstanzen durch einen Softwareagenten vertreten.
- Es gelten dieselben Unterstützungsaufgaben und Agenteneigenschaften.
- Informationen und Wissen über die Baugruppe werden im Softwareagenten gekapselt.

Bei der *Anpassung* einer Baugruppe durch einen Softwareagenten müssen allerdings nicht nur ihre Eigenschaften, sondern zusätzlich ihre interne Struktur berücksichtigt und ebenfalls angepasst werden. Die benötigten Informationen über Baugruppen umfassen daher nicht nur ihre Eigenschaften und Schnittstellen, sondern auch die Details ihrer internen Struktur: die internen Komponenteninstanzen, ihre Verbindungen und Konfigurationen. Insgesamt umfasst das Wissen über Baugruppen:

- Wissen über die internen und externen technischen Abhängigkeiten (Abhängigkeiten auf der Hierarchieebene der Baugruppen). Dieses Wissen kann wie bei Komponenten durch die in Kapitel 5.3.1.4 vorgestellten Konzepte abgebildet und formalisiert spezifiziert werden.

- Wissen über die Anpassungsmöglichkeiten der internen Struktur der Baugruppe in Abhängigkeit von den Ausprägungen ihrer Eigenschaften: alternative Komponenten, Verbindungen und Konfigurationen. Dieses Wissen muss ebenfalls abgebildet und die Baugruppe um Eigenschaften erweitert werden, die eine definierte Menge von Varianten ihres internen Aufbaus ausprägen lassen [Wagn05]. Dabei können die technischen Abhängigkeiten zwischen den internen Komponenten bereits berücksichtigt werden, indem nur sinnvolle Varianten definiert werden. Die Varianten werden durch eine explizite, formalisierte Beschreibung in Bezug zu den sie ausprägenden Baugruppeneigenschaften gesetzt, sodass der Bezug zwischen möglichen Konfigurationen und den Varianten des internen Aufbaus gegeben ist.

Das Wissen muss zusammen mit den Eigenschaften und Strukturinformationen in die Baugruppenbeschreibung integriert werden. Damit wird es beim Engineering explizit verfügbar. Es kann durch die Softwareagenten aus der Baugruppenbeschreibung übernommen und die zusätzlichen Anpassungen der internen Struktur der Baugruppe können selbstständig durchgeführt werden.

5.3.2 Einsatz von Softwareagenten zur aktiven Unterstützung weiterer Aufgaben

Wesentlich für das Konzept des agentenunterstützten Engineering ist, dass durch die Vertretung von Komponenteninstanzen durch Softwareagenten die Informationen des Anlagenmodells und ihre Änderungen auf der Unterstützungsebene aktiv verwaltet werden und Komponentenabhängigkeiten explizit sichtbar werden. Auf dieser Basis kann die Unterstützungsebene um Elemente zur aktiven Unterstützung weiterer Engineeringaufgaben erweitert werden.

5.3.2.1 Unterstützung gewerkspezifischer Tätigkeiten

Die Erstellung gewerkspezifischer Teilmodelle oder die fachspezifische Optimierung des Anlagenmodells sind weitere Aufgaben, die beim Engineering durchgeführt werden müssen. Die Aufgaben werden ausgehend von den Informationen des Anlagenmodells und unter einem bestimmten fachlichen Aspekt ausgeführt, wie die zwei folgenden Beispiele verdeutlichen:

- *Netzwerkplanung*: In fördertechnischen Anlagenmodellen wird ausgehend von der bestehenden Komponentenstruktur (Förderbänder, Verteiler, Puffer) ein Teilmodell Netzwerkplan abgeleitet, das die benötigten dezentralen E/A-Module, ihre Zuordnung zu den Signalanschlüssen der Komponenten und das Kommunikationsnetzwerk beschreibt.

- *Verriegelungsplanung*: In verfahrenstechnischen Anlagenmodellen wird aus bestimmten typischen Kombinationen von Komponenten (Behälter, Pumpen, Ventile) ein Teilmodell mit Engineeringinformationen über bestimmte komponentenübergreifende Überlaufschutz-Verriegelungen abgeleitet, die den ordnungsgemäßen Betrieb der Anlage sicherstellen.

In den beiden Beispielen wird deutlich, dass bei gewerkspezifischen Aufgaben häufig komponentenübergreifende Zusammenhänge, d.h. spezifische Komponentenkombinationen und -konfigurationen im Anlagenmodell betrachtet werden. Sie sind daher ebenfalls von einer Integrationsproblematik geprägt, die zusätzliche Komplexität hervorruft und manuelle Aufwendungen erforderlich macht: Gewerkspezifische Informationen sind abhängig von den Informationen des Anlagenmodells. Die Abhängigkeit ist dynamisch, da Änderungen im Anlagenmodell Auswirkungen auf gewerkspezifische Informationen haben können, die geprüft und durch Anpassungen gehandhabt werden müssen. Die Abhängigkeit kann bidirektional sein, da bei gewerkspezifischen Aufgaben auch Anpassungen des Anlagenmodells möglich sind, deren Auswirkungen geprüft und gehandhabt werden müssen. Eine Entlastung des Ingenieurs kann erreicht werden, indem diese Abhängigkeiten ebenfalls auf der Unterstützungsebene aktiv gehandhabt werden. Dazu ist folgende zusätzliche Unterstützungsfunktionalität erforderlich:

- Auswahl von Komponenteninstanzen des Anlagenmodells oder Identifikation von komponentenübergreifenden Zusammenhängen und Erstellung zusätzlicher Informationen durch Interpretation ausgewählter Komponenteninformationen oder Zusammenhänge.

- Anpassung von Komponentenkonfigurationen und -kombinationen im Anlagenmodell unter gewerkspezifischen Gesichtspunkten.

- Sicherstellung der Konsistenz der gewerkspezifischen Informationen zu den Engineeringinformationen des Anlagenmodells durch beiderseitige Ermittlung und Überprüfung der Auswirkungen von Änderungen.

Die Integration dieser Unterstützungsfunktionalität in die Unterstützungsebene erfolgt durch Einführung zusätzlicher Softwareagenten, welche gewerkspezifische Teilmodelle vertreten und die zusätzliche Unterstützungsfunktionalität und die gewerkspezifischen Informationen kapseln. Die Modularisierung der zusätzlichen Unterstützungsfunktionalität folgt den in Kapitel 5.3.1.2 vorgestellten Prinzipien. Dadurch werden die Abhängigkeiten zwischen Komponenteninstanzen und gewerkspezifischen Teilmodellen auf der Unterstützungsebene explizit sichtbar und die Ermittlung der Auswirkungen von Änderungen erfolgt entsprechend durch Zusammenarbeit der Softwareagenten. Die in Kapitel 5.3.1.3 ermittelten Agenteneigenschaften gelten in gleicher Weise für die zusätzlichen Softwareagenten und beziehen sich hier auf die Vertretung von gewerkspezifischen Teilmodellen.

Für eine selbstständige Durchführung der gewerkspezifischen Unterstützungsfunktionalität muss auf der Unterstützungsebene das dafür erforderliche Wissen verfügbar sein. Es bezieht sich auf komponentenübergreifende Zusammenhänge für Auswahl, Gruppierung, Zuordnung, Verbindung und Anpassung von Komponenteninstanzen des Anlagenmodells und für die Ableitung gewerkspezifischer Engineeringinformationen. Dieses Wissen muss ebenfalls rechnergestützt abgebildet und durch die Softwareagenten gekapselt werden.

Dabei kann die Tatsache genutzt werden, dass bei gewerkspezifischen Aufgaben häufig wiederkehrende Muster angewendet werden, die entweder schriftlich dokumentiert sind oder als Erfahrungswissen vorliegen. Sie können durch einen regelbasierten Ansatz anwendungsunabhängig und formalisiert abgebildet werden [ScFa05], [SFDH06]. Durch Regeln können Muster von Komponentenkombinationen und -konfigurationen definiert werden, aus denen in festgelegter Weise die Engineeringinformationen in gewerkspezifischen Teilmodellen abgeleitet werden:

- Regeln für die Auswahl von Komponenteninstanzen des Anlagenmodells oder für die Identifikation von komponentenübergreifenden Zusammenhängen

- Regeln für Generierung zusätzlicher Informationen durch Interpretation ausgewählter Komponenteninformationen oder Zusammenhänge

Die folgenden Beispiele veranschaulichen gewerkspezifische Regeln für die oben erläuterten Bereiche der Netzwerkplanung und der Verriegelungsplanung [Yilm06], [ScFa05]:

- *Netzwerkplanung*: In Regeln wird die Zuordnung der Signalanschlüsse der Komponenteninstanzen im Anlagenmodell an die E/A-Module des Kommunikationsnetzwerks festgelegt.

- *Verriegelungsplanung*: Durch Regeln werden Muster für Komponentenanordnungen definiert, welche eine Überlaufschutz-Verriegelung erforderlich machen. Für die identifizierten Komponentenanordnungen in einem Anlagenmodell wird mittels weiterer Regeln die Erstellung von Überlaufschutz-Funktionen festgelegt.

Durch die Abbildung mittels Regeln wird das benötigte Wissen für die Unterstützung der Integration gewerkspezifischer Teilmodelle und der fachspezifischen Optimierung des Anlagenmodells rechnergestützt und anwendungsunabhängig verfügbar. Es muss spezifiziert und in einer gewerkspezifischen Regelbasis abgelegt werden. Damit wird es beim Engineering explizit verfügbar, kann durch die Softwareagenten gekapselt und zur selbstständigen Durchführung ihrer gewerkspezifischen Unterstützungsaufgaben angewendet werden.

5.3.2.2 Unterstützung der Komponentenauswahl

Durch die bisherigen Elemente des Unterstützungskonzepts werden die Informationen des Anlagenmodells und die gewerkspezifischer Teilmodelle für die Unterstützung des Ingenieurs durch Softwareagenten genutzt. Die Unterstützung kann zusätzlich erweitert und verbessert werden, indem auch die Informationen der Komponentenbibliothek in die Unterstützungsfunktionalität miteinbezogen werden. Wird beispielsweise durch die Softwareagenten festgestellt, dass eine Komponenteninstanz nicht über die vom Ingenieur vorgesehenen Verbindungen mit anderen Komponenteninstanzen integriert werden kann, so kann eine weitergehende Unterstützung dahingehend erfolgen, dass auf Basis des ermittelten Integrationsproblems automatisch in der Bibliothek nach geeigneten Komponenten gesucht wird, welche als „Zwischenkomponente“ eine konsistente Integration ermöglichen. Für eine solche Unterstützungsfunktionalität müssen die Informationen über die verfügbaren Komponenten der Bibliothek auf der Unterstützungsebene verfügbar sein. Dies wird erreicht, indem die Komponentenbibliothek und die zusätzliche Unterstützungsfunktionalität zur Komponentenauswahl durch einen eigenen Softwareagenten gekapselt werden.

5.4 Anwendungsfälle der Unterstützung

Bevor die Wirkungsweise des Konzepts anhand wichtiger Anwendungsfälle der Unterstützung veranschaulicht wird, wird zunächst der prinzipielle Verlauf der Unterstützung betrachtet:

Prinzipieller Verlauf der Unterstützung

Die Softwareagenten agieren permanent im Hintergrund der Tätigkeiten des Ingenieurs. Jede Tätigkeit wird von den Softwareagenten als Auftrag erfasst und – abhängig von der aktuellen Entwicklungssituation – werden die unmittelbar resultierenden Auswirkungen ermittelt und die Konsistenz überprüft. Werden Inkonsistenzen erkannt, erfolgt direkt im Anschluss an die menschliche Tätigkeit ein entsprechender Hinweis, zusammen mit einem Vorschlag für geeigne-

te Anpassungen, über deren Umsetzung der Ingenieur entscheiden kann. Die Umsetzung erfolgt wiederum durch die Softwareagenten. Nach erfolgter Umsetzung ist der Auftrag der Softwareagenten abgeschlossen. Somit hat der Ingenieur die Gewähr, beim Übergang zur nächsten Tätigkeit von einem konsistenten Zustand der Engineeringinformationen ausgehen zu können.

Anwendungsfall 1: Verbindung zweier Komponenteninstanzen

Die Verbindung zweier Komponenteninstanzen durch den Ingenieur wird von den beiden betroffenen Softwareagenten als „Auftrag“ wahrgenommen, die Kompatibilität der beiden Komponenteninstanzen zu prüfen und sicherzustellen. Dazu verknüpfen sie durch Interaktion wechselseitig ihre jeweiligen Verbindungsregeln mit den Informationen über die Komponentenkonfigurationen. Damit ermitteln sie die Wechselwirkungen, identifizieren Inkonsistenzen und ermitteln – unter Miteinbeziehung der internen Parametrierregeln – die erforderlichen Anpassungen der betroffenen Komponentenkonfiguration. Dabei handhabt jeder Softwareagent die Prüfung und Anpassung der vertretenen Komponenteninstanz selbstständig, über die Interaktion werden lediglich die Informationen über mögliche Wechselwirkungen ausgetauscht. Die Interaktion ist abgeschlossen, wenn keine Inkonsistenzen (mehr) auftreten oder einer der beteiligten Softwareagenten eine Inkonsistenz nicht durch Anpassungen im Rahmen der Konfigurationsmöglichkeiten der vertretenen Komponenteninstanz auflösen kann. Hinweise an den Ingenieur erfolgen dann, wenn Inkonsistenzen identifiziert wurden. Lösungsvorschläge erfolgen, falls die Inkonsistenzen durch geeignete Anpassungen aufgelöst werden können. Die für den Ingenieur wahrnehmbare Unterstützung besteht dabei aus folgenden Hinweisen bzw. Lösungsvorschlägen:

- Die Verbindung der Komponenteninstanzen kann erfolgen, es sind jedoch Anpassungen an ihren Konfiguration(en) erforderlich, um die Konsistenz zu wahren. Zusätzlich werden als Lösungsvorschlag die erforderlichen Anpassungen präsentiert und nach Zustimmung auf der Komponentenebene umgesetzt.
- Die Verbindung der Komponenteninstanzen führt zu Inkonsistenzen, die nicht durch geeignete Anpassungen aufgelöst werden können.
- Falls die beiden Komponenteninstanzen ohne Anpassungen zueinander kompatibel sind, erfolgt auch keine Rückmeldung durch die Softwareagenten.

Anwendungsfall 2: Änderung der Konfiguration einer Komponenteninstanz

Die Änderung der Konfiguration einer Komponenteninstanz wird vom betroffenen Softwareagenten als Auftrag wahrgenommen, die konsistente Umsetzung der Änderung sicherzustellen. Dazu überprüft er zunächst die interne Konsistenz der Komponentenkonfiguration auf Basis seiner Parametrierregeln und ermittelt erforderliche zusätzliche Anpassungen. Anschließend werden mögliche Auswirkungen der Änderung(en) auf andere, verbundene Komponenteninstanzen durch Interaktionen mit den jeweiligen Softwareagenten dynamisch ermittelt und geprüft. Dazu werden die Informationen über mögliche Wechselwirkungen zwischen den Soft-

wareagenten weitergeleitet, wechselseitig mit den jeweiligen komponentenspezifischen Informationen verknüpft und Inkonsistenzen identifiziert. Im Fall von Inkonsistenzen verhandeln die Softwareagenten über die notwendigen Änderungen der Komponentenkonfigurationen. Alle in Interaktionen beteiligten Softwareagenten handhaben die Informationen und deren Änderungen für die vertretene Komponenteninstanz in lokaler Selbstständigkeit und gewährleisten dabei die komponenteninterne Konsistenz. Die für den Ingenieur wahrnehmbare Unterstützung besteht dabei aus folgenden Hinweisen bzw. Lösungsvorschlägen:

- Die Änderung der Komponentenkonfiguration erfordert zusätzliche Parameteranpassungen und / oder die Anpassung der Konfigurationen weiterer Komponenteninstanzen. Zusätzlich werden als Lösungsvorschlag die erforderlichen Anpassungen präsentiert und nach Zustimmung auf der Komponentenebene umgesetzt.
- Die Änderung der Komponentenkonfiguration führt zu Inkonsistenzen, die nicht durch geeignete Anpassungen aufgelöst werden können.
- Bestehen keine Auswirkungen, erfolgt auch keine Rückmeldung durch die Softwareagenten.

Anwendungsfall 3: Erstellung eines gewerkspezifischen Teilmodells

Die Erstellung eines gewerkspezifischen Teilmodells erfolgt mittels direkter Beauftragung des entsprechenden Softwareagenten durch den Ingenieur. Der Softwareagent interagiert mit den Softwareagenten der Komponenteninstanzen im Anlagenmodell, um auf Basis seines gewerkspezifischen Wissens die erforderlichen Informationen über die vorhanden Komponentenkonfigurationen und -kombinationen zu ermitteln und daraus die Informationen des gewerkspezifischen Teilmodells abzuleiten. Dies führt zu folgenden Lösungsvorschlägen:

- Vorschlag der aus gewerkspezifischer Sicht relevanten Informationen des Anlagenmodells
- Vorschlag der benötigten gewerkspezifischen Informationen

Anwendungsfall 4: Anpassung eines gewerkspezifischen Teilmodells

Wie das Anlagenmodell können auch gewerkspezifische Teilmodelle durch den Ingenieur verändert werden. Ein solche Änderung wird vom betroffenen Softwareagenten als Auftrag wahrgenommen, die konsistente Umsetzung der Änderung sicherzustellen. Dazu überprüft er auf Basis seines gewerkspezifischen Wissens die gewerkspezifischen Informationen und ihre Zusammenhänge mit den Informationen des Anlagenmodells, mit folgenden möglichen Ergebnissen:

- Hinweis auf unberücksichtigte Informationen des Anlagenmodells und Lösungsvorschlag
- Hinweis auf unberücksichtigte gewerkspezifische Anforderungen und Lösungsvorschlag

Die Überprüfung und Anpassung eines gewerkspezifischen Teilmodells ist auch bei Änderungen im Anlagenmodell erforderlich. Hierbei handelt es sich also um eine Erweiterung der Anwendungsfälle 1 und 2: Bei Verbindungen oder Änderungen von Komponenteninstanzen wer-

den zusätzlich die Softwareagenten gewerkspezifischer Teilmodelle informiert, um die Auswirkungen aus gewerkspezifischer Sicht zu prüfen. Die Unterstützung erfolgt durch Hinweise auf die Auswirkungen von Änderungen des Anlagenmodells auf gewerkspezifische Informationen und Vorschläge der erforderlichen Anpassungen.

5.5 Erweiterter Aufbau der Softwareagenten

Die Softwareagenten müssen die Informationen und das Wissen über Komponenteninstanzen und gewerkspezifische Teilmodelle kapseln und diese situationsspezifisch bei der Durchführung ihrer Unterstützungsaufgaben verknüpfen. Dazu muss ihr Aufbau aus Kapitel 4.2.2 um Bestandteile zur Kapselung von Informationen und Wissen erweitert werden (siehe Abbildung 5.7).

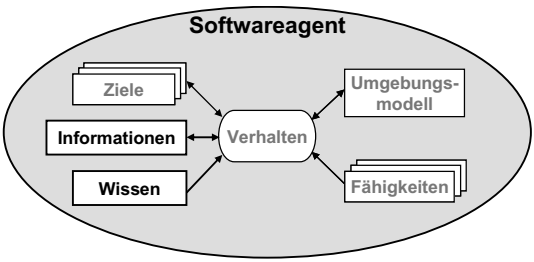


Abbildung 5.7: Erweitertes Konzept für den Aufbau von Softwareagenten

Bei ihrer Initialisierung erfassen die Softwareagenten die jeweiligen Informationen und das benötigte Wissen und initialisieren damit ihre internen Informationen und ihr Wissen. Die im Zuge menschlicher Tätigkeiten oder Agenten-Aktionen veränderbaren Informationen des Anlagenmodells und gewerkspezifischer Teilmodelle werden von den jeweiligen Softwareagenten fortlaufend erfasst und abgeglichen.

Die Verwendung wissensbasierter Konzepte zur Repräsentation der technischen Abhängigkeiten von Komponenten und des fachspezifischen komponentenübergreifenden Wissens ermöglicht die vollständige Trennung des Wissens der Softwareagenten von ihren Problemlösungsfähigkeiten (Aktionen und Interaktionen). Somit ist die Unabhängigkeit ihrer Aktionen und Interaktionen von der Realisierung der Wissensbasis gewährleistet.

Das Umgebungsmodell eines Agenten enthält die Beziehungen zu anderen Agenten und wird im Rahmen von Interaktionen der Agenten oder bei Veränderungen des Anlagenmodells dynamisch aktualisiert. Beziehungen im Umgebungsmodell bestehen zwischen Agenten verbundener Komponenteninstanzen und zu Agenten, die von Änderungen der jeweils vertretenen Komponenteninstanz abhängig sind. Im Rahmen der Agenten-Interaktionen zur Ermittlung und Prüfung von Wechselwirkungen und der Abstimmung von Änderungen entstehen zudem temporär weitere Beziehungen, die nach Abschluss der Interaktionen wieder aufgelöst werden.

5.6 Vorgehensweise zur Entwicklung eines agentenorientierten Modells der Unterstützung

Zur Umsetzung des beschriebenen Konzepts muss ein agentenorientiertes Modell entwickelt werden, welches die Struktur und das Verhalten der Softwareagenten vollständig beschreibt, so dass es in Form eines Agentensystems realisiert werden kann. Bei der Entwicklung eines agentenorientierten Modells wird die Problemstellung in einzelne Agenten abstrahiert, welche abgrenzbare Ziele und Aufgaben verfolgen und dabei interagieren, um ihre Aktionen abzustimmen. In Kapitel 5.3.1.2 wurde deutlich, dass dabei eine Zerlegung der gesamten Unterstützungsfunktionalität in die Lösungsstrategien einzelner Agenten und in die Abläufe ihrer Interaktionen erforderlich ist, sodass eine zielführende Zusammenarbeit bei der Erfüllung von Unterstützungsaufgaben ermöglicht wird. Das agentenorientierte Modell besteht daher aus zwei Ebenen:

- Mikroebene: Konzeption der einzelnen Agententypen, ihrer Ziele, Rollen und Fähigkeiten
- Makroebene: Konzeption des Aufbaus des Agentensystems, bestehend aus den Beziehungen und Interaktionen der verschiedenen Agententypen

Vorgehensweise auf der Mikroebene

Um die Ziele und Aufgaben beim komponentenbasierten Engineering auf die Ziele und Rollen der Softwareagenten zu übertragen, werden die agentenorientierten Analysekonzepte Ziel- und Rollenanalyse angewendet. Durch Zusammenfassung von kohärenten Zielen und Rollen werden die verschiedenen benötigten Typen von Softwareagenten zur Vertretung von Komponenteninstanzen, gewerkspezifischen Teilmodellen und der Komponentenbibliothek definiert und ihnen die jeweiligen Informations- und Wissensbereiche zugeordnet. Weiter werden die für die einzelnen Rollen erforderlichen Fähigkeiten der Softwareagenten spezifiziert.

Vorgehensweise auf der Makroebene

Zur Definition des Zusammenspiels der Softwareagenten werden ihre erforderlichen Interaktionen ermittelt, um durch Kooperation aus Änderungen entstehende Wechselwirkungen zwischen Handlungsobjekten zu erkennen und durch koordinierte, systematische Abstimmung von Aktionen geeignete Anpassungen zu ermitteln. Dabei muss der Ablauf der Interaktionen die Dynamik von Anpassungen und Wechselwirkungen berücksichtigen. Zur Sicherstellung der Koordiniertheit von Interaktionen ist es zudem notwendig, einen organisatorischen Rahmen für die Struktur des Agentensystems zu spezifizieren und somit festzulegen, welche Agenten grundsätzlich mit welchen anderen Agenten auf welche Weise interagieren können. Über die Interaktionen zur Abstimmung der Aktionen einzelner Agenten hinaus sind Interaktionen zur Kooperation mit dem Ingenieur zu entwickeln, um Entscheidungen situationsspezifisch abzustimmen. Abbildung 5.8 fasst die Schritte zur Entwicklung des agentenorientierten Modells zusammen.

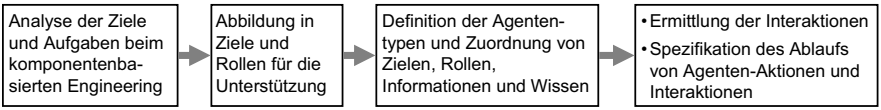


Abbildung 5.8: Schritte zur Entwicklung eines agentenorientierten Modells der Unterstützung

In diesem Kapitel wurden die Grundzüge des Konzepts des agentenunterstützten Engineerings vermittelt. Wesentliche Kernpunkte des Konzepts sind:

- Vertretung von Komponenteninstanzen durch Softwareagenten auf der Unterstützungsebene
- Kapselung von Komponenteninformationen und -wissen durch die Softwareagenten
- Aktive, selbstständige Durchführung der Komponentenintegration durch Softwareagenten
- Flexible Anpassung der Softwareagenten an Änderungen des Anlagenmodells und der Komponentenwechselwirkungen

Auf dieser Basis wurde das Konzept um die Unterstützung der Gewerkintegration und um die Berücksichtigung der Informationen der Komponentenbibliothek durch zusätzliche Softwareagenten erweitert.

Die Vertretung der Handlungsobjekte menschlicher Tätigkeiten und die Kapselung der erforderlichen Informationen und des Wissens führen zu lokaler Selbstständigkeit einzelner Softwareagenten. Konkrete Tätigkeiten des Ingenieurs, wie das Einfügen neuer Komponenten, das Erstellen von Verbindungen, die Änderung von Parametern oder die Erstellung eines gewerkspezifischen Teilmodells, werden von den betroffenen Agenten als „Auftrag“ wahrgenommen, und die entsprechenden Prüf- und Anpassungsschritte werden auf das Handlungsobjekt angewendet. Mittels Interaktionen werden die Auswirkungen von Änderungen ermittelt und weitere erforderliche Anpassungen abgestimmt.

Die Entlastung des Ingenieurs durch das Konzept besteht darin, dass er komponentenbasierte Anlagenmodelle und gewerkspezifische Teilmodelle erstellen kann, ohne die möglichen vielfältigen Auswirkungen einzelner Änderungen berücksichtigen zu müssen. Die Berücksichtigung der Auswirkungen von Änderungen und die Ermittlung geeigneter Anpassungen werden vollständig von den Softwareagenten auf der Unterstützungsebene übernommen. Lediglich wenn Änderungen Inkonsistenzen der Engineeringinformationen verursachen und Anpassungen zu ihrer Auflösung erforderlich sind, werden diese durch situationsbezogene Meldungen der Softwareagenten wieder in den menschlichen Problemlösungsprozess eingebracht.

In den folgenden Kapiteln werden wichtige Teilaspekte zur Umsetzung der Elemente des Konzepts aufgegriffen und präzisiert. Im nächsten Kapitel wird ein agentenorientiertes Modell der Unterstützung entwickelt. In Kapitel 7 wird die formalisierte Spezifikation und integrierte Beschreibung von Informationen und Wissen betrachtet. In Kapitel 8 werden das funktionale Verhalten der Softwareagenten ausführlich behandelt und ihre Fähigkeiten präzisiert.

6 Agentenorientiertes Modell der Unterstützung

Ausgehend von den im vorangegangenen Kapitel erarbeiteten Konzept des agentenunterstützten Engineerings wird in diesem Kapitel ein agentenorientiertes Modell der Unterstützung entwickelt. Dabei wird die Vorgehensweise agentenorientierter Methoden angewendet. Zunächst werden die Ziele der Tätigkeiten des Ingenieurs analysiert und direkt in die Ziele eines Agentensystems zu ihrer Unterstützung abgebildet. Zusammengehörige Ziele werden anschließend zu Rollen zusammengefasst. Weiter werden alle für die Unterstützung benötigten Agententypen und ihre erforderlichen Interaktionen ermittelt.

6.1 Ziele des Agentensystems

Ziele beschreiben kein bestimmtes Szenario, sondern dienen zur Ermittlung und Strukturierung der gesamten Unterstützungsfunktionalität, unabhängig von einer konkreten Situation. Die Ziele werden durch Betrachtung der grundlegenden Charakteristika der Aufgaben beim komponentenbasierten Engineering ermittelt und in detailliertere Teilziele untergliedert. Dabei wird zwischen den zwei Bereichen *Unterstützung der Komponentenhandhabung* und *Unterstützung gewerkspezifischer Tätigkeiten* unterschieden. Da Baugruppen wie Komponenten behandelt werden können, gelten die Ziele zur Unterstützung der Komponentenhandhabung in gleicher Weise auch für die Unterstützung der Baugruppenhandhabung. Es werden lediglich zusätzliche Ziele aufgeführt, die durch die Berücksichtigung der internen Struktur von Baugruppen entstehen.

6.1.1 Unterstützung der Komponentenhandhabung

In Abbildung 6.1 werden die grundlegenden Ziele eines Agentensystems zur Unterstützung der Tätigkeiten bei der Handhabung einzelner Komponenten aufgeführt.

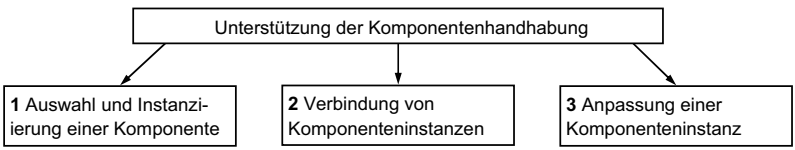


Abbildung 6.1: Ziele zur Unterstützung der Komponentenhandhabung

Ziel 1: Auswahl und Instanziierung einer Komponente

Die Unterstützung hat zum Ziel, eine zum Anwendungsfall passende Komponente in der Komponentenbibliothek zu finden und sie zu instanziieren. Abbildung 6.2 zeigt die Verfeinerung des Ziels in Teilziele.

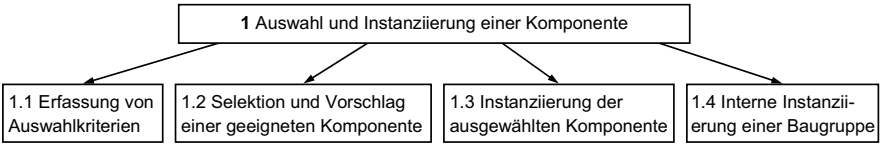


Abbildung 6.2: Zielhierarchie Auswahl und Instanziierung einer Komponente

- *Teilziel 1.1 Erfassung von Auswahlkriterien*: Auswahlkriterien definieren die benötigten technischen Eigenschaften und Schnittstellen der gesuchten Komponente. Aus dem Verwendungskontext der auszuwählenden Komponente (z.B. zu welchen anderen, bereits bestehenden Komponenteninstanzen eine Verbindung erfolgen soll) sind die bestehenden technischen Abhängigkeiten für die Auswahl zu berücksichtigen.

- *Teilziel 1.2 Selektion und Vorschlag einer geeigneten Komponente*: Die Auswahlkriterien sind mit den Beschreibungen der Eigenschaften und Schnittstellen der in der Komponentenbibliothek verfügbaren Komponenten zu vergleichen und geeignete Komponenten zu identifizieren. Gegebenenfalls sind Rückfragen beim Ingenieur zur Erfassung weiterer Informationen oder zur manuellen Auswahl bei gleichwertigen Auswahlmöglichkeiten erforderlich.

- *Teilziel 1.3 Instanziierung der ausgewählten Komponente*: Bei der Instanziierung ist im Anlagenmodell ein Exemplar der ausgewählten Komponente zu erstellen und zu initialisieren.

- *Teilziel 1.4 Interne Instanziierung einer Baugruppe*: Bei der Instanziierung einer wiederverwendbaren Teillösung als Baugruppe müssen zusätzlich alle internen Komponenten der Baugruppe instanziiert sowie die vorgegebenen Verbindungen und Konfigurationen der internen Komponenteninstanzen initialisiert werden.

Ziel 2: Verbindung von Komponenteninstanzen

Zur Unterstützung einer Verbindung von Komponenteninstanzen sind neben prüfenden Maßnahmen auch konstruktive Maßnahmen notwendig, um ihr konsistentes Zusammenspiel sicherzustellen. Diese können in weitere Teilziele verfeinert werden (siehe Abbildung 6.3):

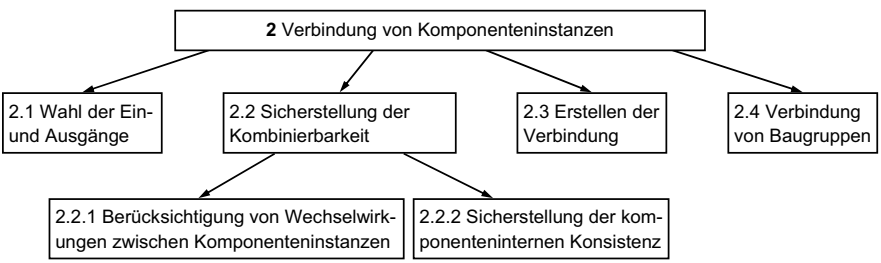


Abbildung 6.3: Zielhierarchie Verbindung von Komponenteninstanzen

- *Teilziel 2.1 Wahl der Ein- und Ausgänge*: Hierbei ist zu prüfen, ob die für die Verbindung zu verwendenden Ein- und Ausgänge auf Schnittstellenebene zusammenpassen und ob diese frei sind. Gegebenenfalls sind andere verfügbare Ein- und Ausgänge zu wählen.

- *Teilziel 2.2 Sicherstellung der Kombinierbarkeit der Komponenteninstanzen*: Vor Erstellen einer Verbindung muss sichergestellt werden, dass die Komponenteninstanzen im Rahmen ihrer durch die technischen Abhängigkeiten beschränkten Verbindungsmöglichkeiten kombinierbar sind. Dazu muss ermittelt werden, ob sich aus den technischen Abhängigkeiten konkrete Wechselwirkungen zwischen den Konfigurationen der Komponenteninstanzen ergeben (siehe Teilziele 2.1.1 und 2.2.2). Wenn sich keine Wechselwirkungen ergeben oder im Fall von Wechselwirkungen keine Inkonsistenzen bestehen, kann die Verbindung erfolgen. Bestehen Inkonsistenzen und sind geeignete Anpassungen der Konfigurationen der Komponenteninstanzen zu ihrer Auflösung möglich, kann die Verbindung unter Berücksichtigung der Anpassungen ebenfalls erfolgen. Wird bei der Prüfung festgestellt, dass durch Anpassungen nicht alle Inkonsistenzen aufgelöst werden können, besteht ein Konflikt zwischen den Komponentenkonfigurationen. Die Komponenteninstanzen passen nicht zusammen und eine direkte Verbindung kann nicht erfolgen. Um den Verbindungsauftrag dennoch zu erfüllen, soll geprüft werden, ob durch Einfügen einer geeignet angepassten dritten Komponente oder einer Kombination von Komponenten als „Zwischenkomponente“ eine indirekte Verbindung der beiden Komponenteninstanzen möglich ist und somit das ursprüngliche Ziel erreicht werden kann. Gegebenenfalls sind Rückfragen beim Ingenieur zur Erfassung weiterer Informationen, zur Abstimmung erforderlicher Änderungen oder zur manuellen Auswahl bei gleichwertigen Anpassungsmöglichkeiten erforderlich.

- *Teilziel 2.2.1 Berücksichtigung von Wechselwirkungen zwischen Komponenteninstanzen*: Wechselwirkungen zwischen den Komponenteninstanzen sind zu ermitteln und zu prüfen. Dazu müssen die technischen Abhängigkeiten jeder Komponenteninstanz mit der Konfiguration der jeweils anderen verglichen werden. Werden Wechselwirkungen erkannt, muss ermittelt werden, ob dadurch Inkonsistenzen bestehen und wie die Konfigurationen angepasst werden können, um die Konsistenz zwischen den Komponenteninstanzen sicherzustellen.

- *Teilziel 2.2.2 Sicherstellung komponenteninterner Konsistenz*: Es ist zu prüfen, ob komponenteninterne Abhängigkeiten zwischen Parametern bestehen, ob diese bei einer Änderung derselben berücksichtigt werden müssen und ob eine geeignete Anpassung möglich ist.

- *Teilziel 2.3 Erstellen der Verbindung*: Die entsprechenden Ein- und Ausgänge der Komponenteninstanzen sind zu verbinden. Wurden zuvor notwendige Anpassungen der Konfigurationen ermittelt, sind diese durch Setzen der entsprechenden Parameterwerte auszuführen (siehe *Ziel 3 Anpassung einer Komponenteninstanz*).

- *Teilziel 2.4 Verbindung von Baugruppen*: Bei der Verbindung von Baugruppen sind zusätzlich die jeweiligen Komponenteninstanzen an den Ein- und Ausgangsschnittstellen der Baugruppen zu wählen und diese unter Berücksichtigung der vorigen Teilziele zu verbinden.

Ziel 3: Anpassung einer Komponenteninstanz

Vor der Durchführung einer Änderung der Konfiguration einer Komponenteninstanz muss durch die Unterstützung sichergestellt werden, dass die Konsistenz des Anlagenmodells gewährleistet bleibt. Dabei müssen Wechselwirkungen berücksichtigt werden, die durch die Anpassung der Komponenteninstanz entstehen können. Abbildung 6.4 zeigt die Hierarchie der Teilziele.

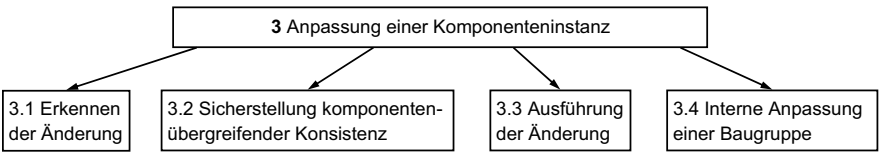


Abbildung 6.4: Zielhierarchie Anpassung einer Komponenteninstanz

- *Teilziel 3.1 Erkennen der Änderung*: Eine vom Ingenieur beabsichtige Änderung muss erkannt und die Maßnahmen zur Prüfung ihrer Auswirkungen müssen eingeleitet werden.

- *Teilziel 3.2 Sicherstellung komponentenübergreifender Konsistenz*: Es muss geprüft werden, ob durch die Änderung Wechselwirkungen zu verbundenen Komponenteninstanzen bestehen, bzw. neu entstehen. Dazu müssen die Verbindungen der Komponenteninstanz ermittelt und für jede verbundene Komponenteninstanz die Auswirkungen der Änderung geprüft werden. Dabei sind ebenfalls die *Teilziele 2.2.1 Berücksichtigung von Wechselwirkungen zwischen Komponenteninstanzen* und *2.2.2 Sicherstellung komponenteninterner Konsistenz* zu beachten. Es muss sichergestellt werden, dass alle Wechselwirkungen geprüft werden, alle Inkonsistenzen identifiziert werden und durch entsprechende Anpassungen aufgelöst werden können. Andernfalls verursacht die Änderung einen Konflikt mit anderen Komponentenkonfigurationen, über den der Ingenieur informiert werden muss. Werden erforderliche Änderungen an verbundenen Komponenteninstanzen ermittelt, können von diesen Komponenteninstanzen ausgehend wiederum weitere indirekte Wechselwirkungen zu den dort verbundenen Komponenteninstanzen bestehen, bzw. durch die Änderung neu entstehen. Um diese zu erkennen und zu berücksichtigen, sind schrittweise die entsprechenden Prüfungen durchzuführen und bei Inkonsistenzen die notwendigen Anpassungen der Konfigurationen zu ermitteln. Diese Schritte sind dann nicht erforderlich, wenn an einer Komponenteninstanz keine Änderungen notwendig sind. Wird bei den fortgesetzten Prüfungen ein Konflikt zwischen Konfigurationen erkannt, ist der Ingenieur darüber zu informieren.

- *Teilziel 3.3 Ausführung der Änderung*: Wurden alle Wechselwirkungen und notwendigen Anpassungen ermittelt, sind nach Rücksprache mit dem Ingenieur die ermittelten Anpassun-

gen der Konfigurationen aller betroffenen Komponenteninstanzen durch Setzen der entsprechenden Parameterwerte auszuführen.

- *Teilziel 3.4 Interne Anpassung einer Baugruppe*: Die Anpassung einer Baugruppe erfolgt mittels Anpassung ihrer internen Struktur. Dabei ist zu ermitteln, welche Anpassungen erforderlich sind, um die gewünschten Eigenschaften der Baugruppe sicherzustellen. Gegebenenfalls sind interne Komponenteninstanzen anzupassen (Teilziel 3), weitere Komponenteninstanzen oder Verbindungen hinzuzufügen oder zu entfernen (Teilziele 1.3 und 2).

6.1.2 Unterstützung gewerkspezifischer Tätigkeiten

Gewerkspezifische Tätigkeiten unterscheiden sich in fachlicher Hinsicht. Die Ziele des Agentensystems sollen jedoch unabhängig von fachspezifischen Gesichtspunkten sein, um eine Unterstützung beliebiger Gewerke zu ermöglichen. Daher werden aus den in Kapitel 2.4.2 erläuterten *grundlegenden* Arten von Tätigkeiten bei der Erstellung gewerkspezifischer Teilmodelle allgemeine, fachlich unabhängige Teilziele zu ihrer Unterstützung abgeleitet, welche in Abbildung 6.5 dargestellt sind.

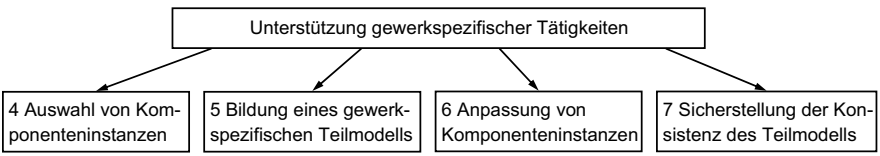


Abbildung 6.5: Ziele zur Unterstützung gewerkspezifischer Tätigkeiten

- *Teilziel 4 Auswahl von Komponenteninstanzen*: Hierzu müssen die gewerkspezifischen Auswahlkriterien erfasst werden. Durch Auswertung der Konfiguration und der Schnittstellen aller bestehenden Komponenteninstanzen des Anlagenmodells sowie dem anschließenden Vergleich mit den Auswahlkriterien sind diejenigen Komponenteninstanzen zu selektieren, die für die Bildung des gewerkspezifischen Teilmodells relevant sind.

- *Teilziel 5 Bildung eines gewerkspezifischen Teilmodells*: Hierzu sind Gruppierungen oder Zuordnungen bestehender Komponenteninstanzen unter gewerkspezifischen Gesichtspunkten und die Erstellung gewerkspezifischer Engineeringinformationen durchzuführen. Die Konkretisierung des Teilziels erfolgt abhängig von der fachlichen Aufgabe des Gewerks.

- *Teilziel 6 Anpassung bzw. Verbindung von Komponenteninstanzen*: Änderungen der Konfiguration bestehender Komponenteninstanzen des Anlagenmodells können auch aus einem gewerkspezifischen Gesichtspunkt heraus initiiert werden. Dabei muss ebenfalls sichergestellt werden, dass die Konsistenz des Anlagenmodells gewährleistet bleibt und die Wechselwirkungen zwischen Komponenteninstanzen berücksichtigt werden (siehe Ziele 2 und 3).

- *Teilziel 7 Sicherstellung der Konsistenz des Teilmodells*: Änderungen an Komponenteninstanzen im Anlagenmodell müssen erkannt werden, da diese eine Aktualisierung des gewerkspezifischen Teilmodells erforderlich machen können. Daraufhin ist die Ermittlung und Überprüfung von Wechselwirkungen zum gewerkspezifischen Teilmodell erforderlich, wobei die Wechselwirkungen aus den für die Bildung des Teilmodells relevanten gewerkspezifischen Gesichtspunkten ermittelt werden können. Wenn das gewerkspezifische Teilmodell von der Änderung betroffen ist, ist eine entsprechende Aktualisierung notwendig. Bei manuellen Änderungen eines Teilmodells ist ebenfalls eine Überprüfung und gegebenenfalls eine Anpassung an die fachspezifischen Anforderungen erforderlich.

6.2 Rollen des Agentensystems

Ausgehend von dem in Kapitel 5.3.1.2 entwickelten Prinzip der Modularisierung der Unterstützungsfunktionalität werden aus den Zielen der Unterstützung die Rollen des Agentensystems ermittelt. Für die einzelnen Rollen werden konkrete, abgrenzbare Verantwortlichkeiten definiert.

- *Rolle Komponentenverwalter*: Verwalten aller Komponenten und wiederverwendbaren Teillösungen der Komponentenbibliothek, Erfassen ihrer Eigenschaften und Schnittstellen.
- *Rolle Komponentensucher*: Auswählen und Instanziieren von Komponenten und wiederverwendbaren Teillösungen aus der Komponentenbibliothek.
- *Rolle Verbindungsmanager*: Verbinden von Komponenteninstanzen, Sicherstellen der Kombinierbarkeit, Lösen von Verbindungskonflikten durch Zwischenkomponenten.
- *Rolle Schnittstellenprüfer*: Ermitteln der Verbindungsmöglichkeiten zweier Komponenteninstanzen auf Schnittstellenebene.
- *Rolle Konfigurationsmanager*: Verwalten der Konfiguration einer Komponenteninstanz und Sicherstellung ihrer Konsistenz durch Berücksichtigung der komponenteninternen Wechselwirkungen und der Wechselwirkungen zu anderen Komponenteninstanzen.
- *Rolle Änderungswächter*: Ermitteln und Handhaben der Auswirkungen von Änderungen der Konfiguration einer Komponenteninstanz auf andere Komponenteninstanzen und gewerkspezifische Teilmodelle.
- *Rolle Baugruppenmanager*: Handhaben einer Baugruppe, Verwalten und Anpassen ihrer internen Struktur.
- *Rolle Instanzenselektierer:* Ermitteln der relevanten Komponenteninstanzen des Anlagenmodells für ein gewerkspezifisches Teilmodell.

- *Rolle Modellierer:* Erstellen eines gewerkspezifischen Teilmodells aus den Informationen des Anlagenmodells, Initiieren von fachspezifischen Anpassungen der Komponenteninstanzen des Anlagenmodells, Prüfen und Anpassen eines gewerkspezifischen Teilmodells.
- *Rolle Teilmodellmanager:* Verwalten der Abhängigkeiten eines Teilmodells von den Informationen des Anlagenmodells, Handhaben der Auswirkungen von Änderungen.

Für die identifizierten Rollen ist festzulegen, welche der im vorangegangenen Abschnitt identifizierten Ziele bzw. Teilziele durch die Rolle zu erfüllen sind. Dabei werden Ziele mit hoher Kohäsion derselben Rolle zugeordnet. Ziele, die in mehrere Verantwortlichkeitsbereiche fallen, werden auf die entsprechenden Rollen aufgeteilt (siehe Tabelle 6.1).

Tabelle 6.1: Rollen des Agentensystems

Rolle	**Zugeordnete Ziele / Teilziele**
Komponentenverwalter	Ziel 1: Auswahl und Instanziierung einer Komponente Teilziel 1.2 Selektion und Vorschlag einer geeigneten Komponente
Komponentensucher	Ziel 1: Auswahl und Instanziierung einer Komponente Teilziel 1.1 Erfassung von Auswahlkriterien Teilziel 1.2 Selektion und Vorschlag einer geeigneten Komponente Teilziel 1.3 Instanziierung der ausgewählten Komponente
Verbindungsmanager	Ziel 2 Verbindung von Komponenteninstanzen Teilziel 2.2 Sicherstellung der Kombinierbarkeit der Komponenteninstanzen Teilziel 2.4 Verbindung von Baugruppen
Schnittstellenprüfer	Teilziel 2.1 Wahl der Ein- und Ausgänge
Konfigurations-manager	Ziel 3 Anpassung einer Komponenteninstanz Teilziel 2.2.1 Berücksichtigung von Wechselwirkungen zwischen Komponenteninstanzen Teilziel 2.2.2 Sicherstellung komponenteninterner Konsistenz Teilziel 2.3 Erstellen der Verbindung
Änderungswächter	Ziel 3 Anpassung einer Komponenteninstanz Teilziel 3.1 Erkennen der Änderung Teilziel 3.2 Sicherstellung komponentenübergreifender Konsistenz Teilziel 3.3 Ausführung der Änderung Teilziel 7 Sicherstellung der Konsistenz des Teilmodells
Baugruppenmanager	Teilziel 1.4 Interne Instanziierung einer Baugruppe Teilziel 2.4 Verbindung von Baugruppen Teilziel 3.4 Interne Anpassung einer Baugruppe
Instanzenselektierer	Teilziel 4 Auswahl von Komponenteninstanzen

Rolle	Zugeordnete Ziele / Teilziele
Modellierer	Teilziel 5 Bildung eines gewerkspezifischen Teilmodells Teilziel 6 Anpassung von Komponenteninstanzen
Teilmodellmanager	Teilziel 7 Sicherstellung der Konsistenz des Teilmodells

6.3 Agententypen des Agentensystems

Das Agentensystem besteht aus verschiedenen Typen von Agenten, die jeweils abgegrenzte Unterstützungsaufgaben übernehmen. [WaGö05]. Jeder Agententyp übernimmt die zur Unterstützung der jeweiligen Tätigkeiten erforderlichen Rollen, verfolgt die zugehörigen Ziele und kapselt die zu ihrer Erfüllung benötigten Informationen und das benötigte Wissen. Dadurch wird eine hohe Kohäsion zwischen Rollen, Zielen, Informationen und Wissen erreicht.

6.3.1 Agententyp Komponenten-Agent

Agenten vom Typ *Komponenten-Agent* vertreten einzelne Komponenteninstanzen und übernehmen die aktive Unterstützungsfunktionalität für die Komponentenintegration. Dazu werden ihnen die Rollen des *Konfigurationsmanagers* und des *Änderungswächters* zugewiesen. Jeder Komponenten-Agent verfolgt die den Rollen zugeordneten Ziele, um die von ihm vertretene Komponenteninstanz in ihre Umgebung zu integrieren und dabei die Wechselwirkungen zu anderen Komponenteninstanzen zu berücksichtigen. Er kapselt die Informationen über die vertretene Komponenteinstanz und das Wissen über ihre internen und externen Abhängigkeiten. Abbildung 6.6 links zeigt die Vertretung von Komponenteninstanzen durch den Agententyp Komponenten-Agent und seine Rollen.

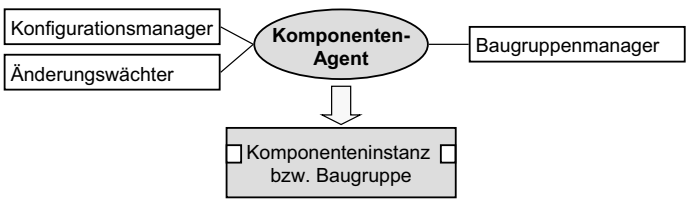


Abbildung 6.6: Agententyp Komponenten-Agent: Vertretungsprinzip und Rollen

Baugruppen werden ebenfalls durch einen eigenen Komponenten-Agenten vertreten, da sie grundsätzlich vom Ingenieur als abgeschlossener Baustein gehandhabt werden und bei ihrer Integration technische Abhängigkeiten auf Baugruppenebene berücksichtigt werden müssen. Zusätzlich muss bei Handhabung einer Baugruppe ihre interne Struktur in Betracht gezogen werden. Dazu übernimmt der Komponenten-Agent, der die Baugruppe vertritt, die Rolle des *Baugruppenmanagers* und kapselt die Informationen und das Wissen über die Details der inter-

nen Struktur der Baugruppe (siehe Abbildung 6.6 rechts). Die internen Komponenteninstanzen der Baugruppe werden durch eigene Komponenten-Agenten vertreten. Es entsteht eine Agentengruppe, die durch den Baugruppenmanager koordiniert wird. Eine Anpassung der Baugruppe führt der Baugruppenmanager auf der Basis seines Variantenwissens durch Koordination der Anpassungen der internen Komponenteninstanzen durch ihre Komponenten-Agenten aus.

6.3.2 Agententyp Bibliotheks-Agent

Für die Erfassung und Verwaltung der Informationen aus der Komponentenbibliothek sowie für das Finden und Instanziieren geeigneter Komponenten oder wiederverwendbarer Teillösungen wird der Agententyp *Bibliotheks-Agent* eingeführt. Er übernimmt die Rollen des *Komponentenverwalters* und *Komponentensuchers* (siehe Abbildung 6.7). Zusätzlich ermittelt er in der Rolle des *Schnittstellenprüfers* auf Basis der Komponenteninformationen in der Bibliothek die Verbindungsmöglichkeiten von Komponenteninstanzen auf Schnittstellenebene. Durch die Zuordnung dieser Rolle zum Bibliotheks-Agenten können bei der Abstimmung von Schnittstellen auch die Informationen über die Schnittstellen weiterer Komponenten mit einbezogen werden (z.B. bei der Auflösung von Verbindungskonflikten).

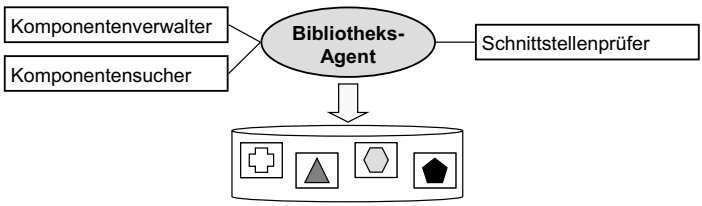


Abbildung 6.7: Agententyp Bibliotheks-Agent: Vertretungsprinzip und Rollen

6.3.3 Agententyp Aspekt-Agent

Bei der Erstellung gewerkspezifischer Teilmodelle oder der fachspezifischen Optimierung des Anlagenmodells handelt es sich um Aufgaben, bei denen komponentenübergreifende Zusammenhänge betrachtet werden und komponentenübergreifendes Wissen angewendet wird. Die Aufgaben werden unter einem bestimmten fachlichen Aspekt ausgeführt. Die Umsetzung der diesbezüglichen Unterstützungsziele des Agentensystems erfolgt durch den Agententyp *Aspekt-Agent*. Ein Aspekt-Agent übernimmt die gewerkspezifische Unterstützungsfunktionalität und kapselt fachspezifische, komponentenunabhängige Informationen und komponentenunabhängiges Wissen. Ihm werden die Rollen *Instanzenselektierer*, *Modellierer* und *Teilmodellmanager* zugewiesen, um die Informationen des Anlagenmodells zu analysieren, um daraus zusätzliche Engineeringinformationen zu erstellen, bzw. bei Änderungen des Anlagenmodells zu aktualisieren, sowie um Komponenteinstanzen unter fachlichen Gesichtspunkten anzupassen.

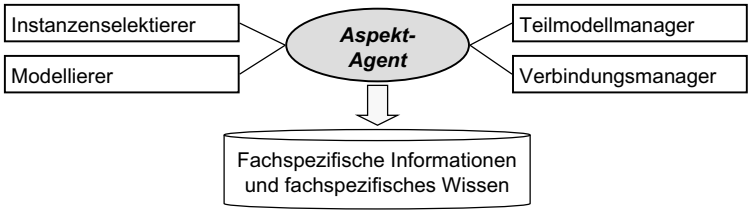


Abbildung 6.8: Agententyp Aspekt-Agent: Vertretungsprinzip und Rollen

Dem Agententyp Aspekt-Agent wird ebenfalls die Rolle des *Verbindungsmanagers* zugewiesen, da es sich bei dieser Rolle ebenfalls um eine komponentenübergreifende Aufgabe handelt, von der mindestens zwei Komponenteninstanzen betroffen sind. Somit existiert auch für das Anlagenmodell ein Aspekt-Agent, der die Rolle des Verbindungsmanagers einnimmt. Zudem können Verbindungen auch aus einem gewerkspezifischen Gesichtspunkt heraus initiiert werden. Durch die Rolle des Verbindungsmanagers werden die Abläufe zur Prüfung von Wechselwirkungen und Abstimmung der Konfigurationen der betroffenen Komponenteninstanzen eingeleitet und koordiniert. Die Durchführung erfolgt durch die jeweiligen Komponentenagenten in ihren Rollen als Konfigurationsmanager und Änderungswächter.

6.4 Struktur des Agentensystems

Mittels Interaktionen zwischen Softwareagenten werden Informationen ausgetauscht, Aufgaben verteilt und koordiniert sowie die Auswirkungen lokaler Aktionen ermittelt und abgestimmt. Für die ermittelten Agententypen ist daher festzulegen, welche Beziehungen grundsätzlich bestehen können und welche Interaktionen über diese Beziehungen durchgeführt werden können.

In Kapitel 5.3.1.2 wurde deutlich, dass bei der Unterstützung der Komponentenintegration und der Gewerkintegration die Aufgabe der Ermittlung von Änderungsauswirkungen auf Interaktionen abgebildet werden muss. Hierzu wird die Interaktion *Ermittlung komponentenübergreifender Wechselwirkungen* zwischen Komponenten-Agenten und zwischen Komponenten-Agenten und Aspekt-Agenten eingeführt. Mittels dieser Interaktion werden die jeweiligen Informationen und das Wissen der verschiedenen Agenten wechselseitig verknüpft, somit die situationsspezifisch entstandenen Wechselwirkungen ermittelt und Änderungen zur Auflösung von erkannten Inkonsistenzen abgestimmt.

Durch die Interaktionen *Informationsabfrage* von Aspekt-Agenten mit Komponenten-Agenten werden die Informationen über die Komponenteninstanzen des Anlagenmodells erfasst, Komponenteninstanzen selektiert und daraus gewerkspezifische Engineeringformationen erstellt bzw. überprüft. Mittels der Interaktion *Anpassung* können Komponenteninstanzen durch Aspekt-Agenten angepasst werden.

Ein Aspekt-Agent, der die Rolle des Verbindungsmanagers einnimmt, koordiniert die Verbindung von Komponenteninstanzen durch die Interaktionen *Verbindungsprüfung und Erstellung* mit den betroffenen Komponenten-Agenten. Die Kompatibilität der Komponenteninstanzen auf Schnittstellenebene wird dabei durch Interaktion mit dem Bibliotheks-Agenten ermittelt, ebenso die Suche nach geeigneten Zwischenkomponenten im Fall eines Verbindungskonflikts zwischen zwei Komponenteninstanzen. Der Bibliotheks-Agent initialisiert zudem Komponenten-Agenten bei ihrer Instanziierung mit den Informationen und dem Wissen aus den Komponentenbeschreibungen in der Bibliothek. Abbildung 6.9 zeigt die Beziehungen zwischen den Agententypen und die beschriebenen Interaktionen.

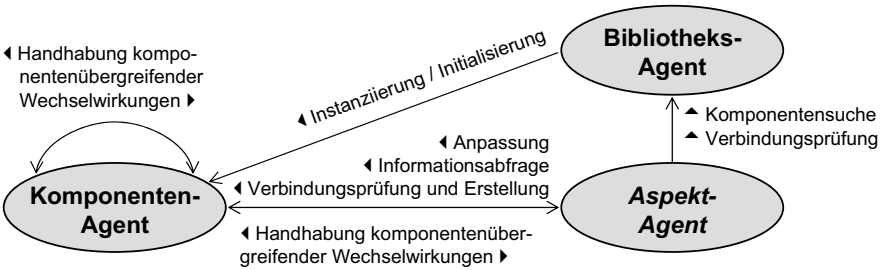


Abbildung 6.9: Beziehungen und Interaktionen der Agententypen

In diesem Kapitel wurde zur Umsetzung des Konzepts des agentenunterstützten Engineerings ein agentenorientiertes Modell der Unterstützung entwickelt. Dazu wurden die Ziele und Rollen des Agentensystems zur Unterstützung der Aufgaben beim komponentenbasierten Engineering ermittelt. Durch Zusammenfassung kohärenter Ziele, Rollen, Informationen und Wissen wurden die Agententypen Komponenten-Agent, Bibliotheks-Agent und Aspekt-Agent spezifiziert, die auf die Unterstützung konkreter Tätigkeiten spezialisiert sind. Weiter wurden die möglichen Beziehungen und die erforderlichen Interaktionen zwischen den Agententypen definiert.

Im folgenden Kapitel wird zunächst das Konzept zur integrierten Beschreibung von Informationen und vom Wissen des Ingenieurs ausgearbeitet. In Kapitel 8 wird das funktionale Verhalten des Agentensystems beschrieben. Dabei werden Aktionen für jede Rolle der Softwareagenten und die Abläufe ihrer Interaktionen spezifiziert. Zudem werden die erforderlichen Interaktionen der Softwareagenten zur Kooperation mit dem Ingenieur ermittelt.

7 Konzept zur expliziten Integration von Informationen und Wissen des Ingenieurs

Ein wichtiger Bestandteil für eine umfassende Unterstützung des Ingenieurs durch Softwareagenten ist die Nutzung von Informationen und Wissen des Ingenieurs. In diesem Kapitel wird das Konzept zur integrierten Beschreibung von Informationen und Wissen unter Zuhilfenahme wissensbasierter Konzepte ausführlich erläutert. Dabei werden alle erforderlichen Informationen und alles erforderliche Wissen über Komponenten, über wiederverwendbare Teillösungen und über komponentenübergreifende Zusammenhänge betrachtet. Abschließend werden die Möglichkeiten zur Bereitstellung von Informationen und Wissen diskutiert.

7.1 Voraussetzung für die Nutzung von Komponenteninformationen und Wissen

Die selbstständige Bewerkstelligung von Unterstützungsaufgaben durch Softwareagenten erfordert die rechnergestützte Nutzung von Komponenteninformationen und Wissen über technische Abhängigkeiten. Die Informationen über Parameter und Schnittstellen von Komponenten sind heute bereits in Komponentenbeschreibungen abgelegt, deren einheitlicher Aufbau durch Komponentenmodelle festgelegt ist. Die explizite Beschreibung des Wissens über technische Abhängigkeiten kann, wie in Kapitel 5 erläutert, auf der Grundlage regelbasierter Konzepte erfolgen.

Die rechnergestützte Nutzung von Informationen und Wissen ist jedoch nicht generell möglich, sondern nur, wenn gewisse Voraussetzungen erfüllt sind: Zum einen müssen Informationen und Regeln in formalisierter Form vorliegen und mit einer eindeutigen Semantik belegt sein. Andernfalls ist eine rechnergestützte Erfassung und Verknüpfung nicht möglich. Zum anderen müssen die interne Struktur einer Komponente und die Zugehörigkeit von Informationen und Regeln zu den internen Bestandteilen der Komponente explizit sichtbar sein. Sonst lässt sich nicht entscheiden, für welche Zusammenhänge innerhalb oder zwischen Komponenten sie relevant sind und die Menge bestehender Wechselwirkungen kann nicht bestimmt werden.

Zusammengenommen ergibt sich hieraus folgende Aussage: Die Nutzung von Komponenteninformationen und Wissen bedingt, dass sie formalisiert, eindeutig und zuordenbar sind. Angesichts dieser Überlegungen stellt sich nun die Frage, in welcher Weise ein Komponentenmodell gestaltet sein muss, um auf seiner Basis Komponentenbeschreibungen bereitzustellen, welche den genannten Voraussetzungen genügen. Die Spezifikation eines standardisierten Komponentenmodells mit entsprechender Syntax und Semantik würde jedoch die Verwendbarkeit des Unterstützungskonzepts auf dieses Komponentenmodell beschränken. Stattdessen sollen die erforderlichen Vorgaben auf der Metamodellebene definiert werden. Metamodelle sind Sprach-

mittel, mit denen der Aufbau von Modellen explizit beschrieben werden kann [Eppl03]. Ein Metamodell kann allgemeine Strukturregeln definieren, die für alle Modelle einer Anwendungsdomäne gelten, ohne die Syntax und die Semantik der Modelle festzulegen. Es wird lediglich vorgegeben, welche Entitäten von Informationen vorkommen müssen, welche Attribute sie aufweisen, welche Beziehungen zwischen den Klassen zu bestehen haben und welche Ausprägungen von Entitäten, Attributen und Beziehungen zulässig sind. Derartige Modelle werden auch als *Strukturmetamodelle* bezeichnet [Eppl03]. Die Beschreibung der Elemente eines Strukturmetamodells ist wiederum mit einem allgemeinen *Basismetamodell* möglich, z. B. dem UML-Klassendiagramm [OMG04] oder dem XMLSchema-Schema [W3C06].

Im Folgenden wird daher ein *Beschreibungsmodell* als Strukturmetamodell entworfen, welches den grundlegenden Aufbau und die notwendige semantische Ausdrucksstärke von Komponentenmodellen im Hinblick auf die Unterstützung definiert: die erforderlichen Komponenteninformationen, ihre Strukturierung und ihre Integration mit dem Wissen über Komponentenabhängigkeiten. Zum einen wird dadurch sichergestellt, dass Komponentenbeschreibungen – unabhängig vom gewählten Komponentenmodell – konform mit den Voraussetzungen der rechnergestützten Nutzung von Informationen und Wissen sind. Zum anderen bildet das Beschreibungsmodell die Basis für die Aktionen der Agenten, mit welchen sie die für die Unterstützung benötigten Informationen und das benötigte Wissen aus den Komponentenbeschreibungen erfassen und verknüpfen, um Wechselwirkungen zu ermitteln, zu prüfen, Inkonsistenzen zu identifizieren und Anpassungen zu ermitteln.

7.2 Formalisierte Beschreibung von Komponenteninformationen und Wissen

7.2.1 Grundlegender Aufbau von Komponentenbeschreibungen

Abbildung 7.1 zeigt die grundlegenden Elemente einer Komponentenbeschreibung. Eine Komponente wird zunächst durch ihren Namen (`ComponentName`) charakterisiert und eindeutig identifiziert. Der Instanzname (`InstanceName`) erlaubt die spätere Benennung und Identifikation von Komponenteninstanzen im Anlagenmodell. Wesentliche Bestandteile einer Komponente sind ihre nach außen verfügbaren Anschlusspunkte (`Port`) und Eigenschaften (`Feature`). Diese Entitäten werden in den folgenden Abschnitten erläutert.

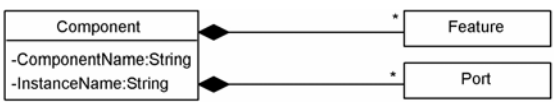


Abbildung 7.1: Entitäten zur Beschreibung einer Komponente

7.2.2 Beschreibungsentitäten für Komponenteneigenschaften

Die Eigenschaften einer Komponente umfassen ihre Charakteristika und ihre Variationspunkte. Neben rein technischen Eigenschaften, wie die Leistung einer Pumpe oder die Geschwindigkeit eines Förderantriebs, kann es sich hierbei auch um funktionale Eigenschaften handeln, wie beispielsweise die Eigenschaften einer fördertechnischen Komponente „*ermöglicht einen stetigen Materialtransport*“ oder „*kann maximal ein Fördergut gleichzeitig aufnehmen*“. Es lassen sich drei Arten von Komponenteneigenschaften unterscheiden (siehe Abbildung 7.2):

- *Merkmale* (`Property`) beschreiben diejenigen Charakteristika einer Komponente, die üblicherweise in der Dokumentation der Komponente informell wiedergegeben werden, in formalisierter Weise. Bei Merkmalen handelt es sich um komponentenspezifische Konstanten, die einen vordefinierten Wert besitzen und nicht für einzelne Instanzen der Komponente konfiguriert werden können (z.B. Kapazität = 1).

- *Parameter* (`Parameter`) ermöglichen die Konfiguration von Komponenteneigenschaften durch die individuelle Eingabe von Werten. Diese Konfiguration erfolgt spezifisch für jede Komponenteninstanz. Zusätzlich werden Informationen über den Standardwert, erlaubte Werte und Wertebereiche angegeben.

- *Optionen* (`Option`) dienen zur Auswahl optionaler Eigenschaften einer Komponente.

Komponenteneigenschaften werden durch einen Namen (`Name`) charakterisiert. Eigenschaften können wiederum weitere Eigenschaften enthalten, sodass die Beschreibung hierarchisch strukturierter Attribute möglich ist. Merkmale (`Property`) verfügen über einen Typ (`Type`) und einen konstanten Wert (`FixedValue`). Parameter weisen ebenfalls einen Typ (`Type`) auf. Der Parameterwert (`Value`) erlaubt die Konfiguration von Komponenteninstanzen im Anlagenmodell. Zusätzlich ist die Angabe der zulässigen Wertemenge (`ValueSet`) oder des zulässigen Wertebereichs (`ValueRange`) möglich. Optionen weisen ein Attribut `Selected` auf, das bei der Konfiguration von Komponenteninstanzen gesetzt werden kann.

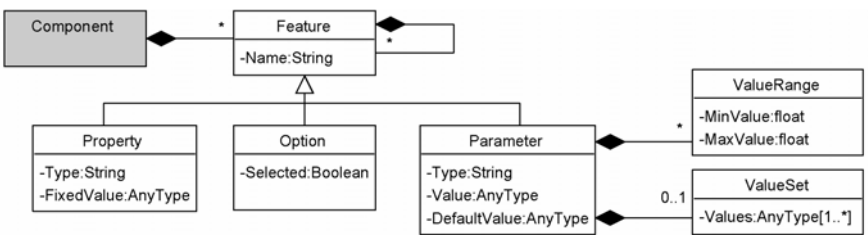


Abbildung 7.2: Entitäten zur Beschreibung von Komponenteneigenschaften

Alle strukturbezogenen Informationen einer Komponente sind nach ihrer Instanziierung nicht mehr änderbar, die Anpassung einer Komponenteninstanz erfolgt ausschließlich durch die Angabe von Parameterwerten und die Auswahl von Optionen.

7.2.3 Beschreibungsentitäten für Komponentenschnittstellen

Schnittstellen definieren die Verbindungsmöglichkeiten einer Komponente. Sie bestehen aus einem oder mehreren Anschlusspunkten (`Port`), über die die instanziierte Komponente mit den Anschlusspunkten anderer Komponenteninstanzen verbunden werden kann. Anschlusspunkte werden durch eine `PortID` identifiziert und können zusätzlich einen Namen (`PortName`) besitzen. Für jeden Anschlusspunkt wird der Schnittstellentyp (`InterfaceType`) festgelegt. Schnittstellentypen charakterisieren die Art des Anschlusspunktes wie Signalfluss oder Materialfluss. Zudem kann die Möglichkeit vorgesehen werden, den Anschlusspunkt bei der Konfiguration einer Komponenteninstanz je nach Bedarf zu aktivieren oder zu deaktivieren (`Activation`). Anschlusspunkte werden in Eingänge (`IncomingPort`) und Ausgänge (`OutgoingPort`) unterschieden. Komponenteninstanzen können verbunden werden, indem zwischen einem Ausgangs-Anschlusspunkt und einem Eingangs-Anschlusspunkt des gleichen Schnittstelentyps eine Verbindung (`Connection`) erstellt wird. Anschließend sind die beiden Anschlusspunkte über die Verbindung miteinander assoziiert.

Anschlusspunkten können ebenfalls Eigenschaften zugeordnet werden. Auf diese Weise kann der Gültigkeitsbereich von Komponenteneigenschaften präzisiert werden. Eigenschaften, die einem Anschlusspunkt zugeordnet sind, können nur genutzt werden, wenn eine Verbindung mit diesem Anschlusspunkt besteht. Eigenschaften „innerhalb“ der Komponente stehen für alle Verbindungen zur Verfügung. Die Gesamtmenge der Eigenschaften einer Komponente ergibt sich aus den Eigenschaften von Anschlusspunkten und denen der Komponente selbst. Die Eigenschaften von Anschlusspunkten und die der Komponente sind disjunkt, d.h., eine Eigenschaft kann nur exklusiv zugeordnet sein. In Abbildung 7.3 wird dies durch die Verwendung von Kompositionsbeziehungen verdeutlicht.

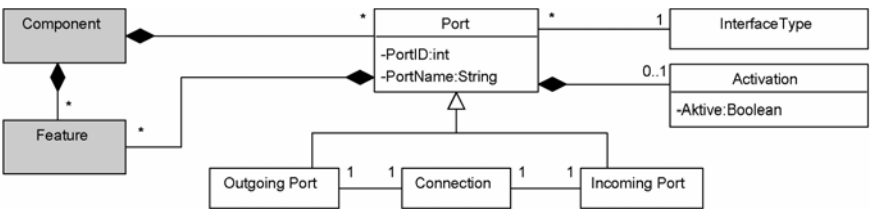


Abbildung 7.3: Entitäten zur Beschreibung von Komponentenschnittstellen

Abbildung 7.4 zeigt das Beispiel einer Komponentenbeschreibung für eine Komponente Förderband. Zur Veranschaulichung wurden die Informationen grafisch angeordnet. In der Mitte sind

die komponenteninternen Eigenschaften zusammen mit ihren möglichen Werten dargestellt. Links und rechts sind die Eingangs- und Ausgangsanschlusspunkte und die ihnen jeweils spezifisch zugeordneten Eigenschaften dargestellt. Anhand des Parameters „Flussrichtung“ wird die Bedeutung der Zuordnung von Eigenschaften zu Anschlusspunkten deutlich: Das Förderband kann Förderobjekte sowohl in horizontaler Richtung (z.B. von einem anderen Förderband) als auch in vertikaler Richtung (z.B. von einem Hebekran) aufnehmen und abgeben. Dabei können bei Materialzufluss und -abfluss unterschiedliche Richtungen auftreten.

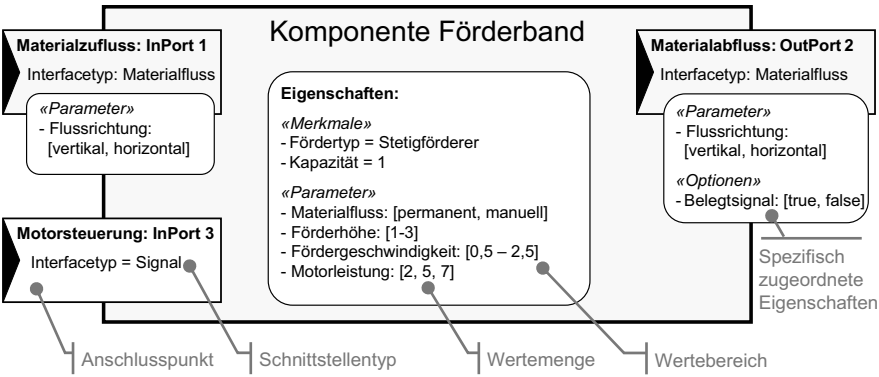


Abbildung 7.4: Beispiel einer Komponentenbeschreibung

7.2.4 Beschreibung von Verbindungs- und Parametrierregeln

In Kapitel 5.3.1.4 wurde erläutert, wie das Wissen über technische Abhängigkeiten mithilfe von Verbindungs- und Parametrierregeln formalisiert spezifiziert werden kann. Dieses Wissen muss in Regelform (`Rule`) in die Komponentenbeschreibung integriert und den Komponenteninformationen zugeordnet werden.

Bedeutung von Verbindungs- und Parametrierregeln

Verbindungs- und Parametrierregeln beschreiben die technischen Abhängigkeiten einer Komponente. Sie werden als zusammengesetzte logische Aussagen formuliert, deren Gültigkeit erfüllt sein muss, damit ein Anlagenmodell richtig konfiguriert ist. Sie bestehen aus zwei aussagenlogischen Sätzen A und B, die durch eine so genannte materiale Implikation „$\rightarrow$“ (Konditional oder auch Wenn-Dann-Verknüpfung) zu einer neuen Aussage verbunden sind [Wiki06b]. Die materiale Implikation $A \rightarrow B$ besagt, dass wenn A gilt, muss auch B gelten. Allgemein wird A als Vordersatz (Antezedens) der Regel und B als Nachsatz (Konsequenz) bezeichnet. Bei der materialen Implikation ist A eine hinreichende, aber keine notwendige Bedingung für B: Eine Aussage $A \rightarrow B$ ist demnach genau dann falsch, wenn A wahr ist und B nicht wahr. In jedem anderen Fall ist $A \rightarrow B$ wahr, d.h., wenn A nicht wahr ist, kann B einen beliebigen Wahrheits-

wert aufweisen. Eine Regel ist genau dann erfüllt, wenn die enthaltene Implikation wahr ist. Die Regel „Wenn es regnet, dann ist die Straße nass“ ist also nur dann nicht erfüllt, wenn es regnet, die Straße aber nicht nass ist. Umgekehrt kann die Straße durchaus nass sein, auch wenn es nicht regnet. Im Folgenden werden zur Verdeutlichung die in Kapitel 5.3.1.4 eingeführten Begriffe „Vorbedingung“ für die Antezedens und „Konsistenzbedingung“ für die Konsequenz der Implikation in Verbindungs- und Parametrierregeln verwendet. Es ergibt sich folgende Regelform: Vorbedingung → Konsistenzbedingung, oder umgangssprachlich ausgedrückt: „Wenn die Vorbedingung erfüllt ist, dann muss auch die Konsistenzbedingung erfüllt sein“.

In Verbindungs- und Parametrierregeln sind Vorbedingung und Konsistenzbedingung jeweils logische Aussagen, die sich auf Komponenteneigenschaften und ihre Werte beziehen. Sie stellen einen Vergleich dar, z.B. `Geschwindigkeit > 5`, `Materialfluss = permanent` oder `Optionaler-Sensor = TRUE` (gewählt). Auf diese Weise kann dargestellt werden, dass wenn die in der Vorbedingung der Regel angegebene Eigenschaft einen bestimmten Wert aufweist, auch die in der Konsistenzbedingung angegebene Eigenschaft einen bestimmten Wert aufweisen muss. Bei der in der Konsistenzbedingung angegebenen Eigenschaft handelt es sich entweder um eine Eigenschaft derselben Komponente (Parametrierregel) oder um eine Eigenschaft einer (noch unbekannten) anderen Komponente (Verbindungsregel).

Für Verbindungs- und Parametrierregeln gelten folgende Bedeutungszusammenhänge:

- Verbindungs- und Parametrierregeln sind gültig, wenn ihre Vorbedingung erfüllt ist. Existiert keine Vorbedingung, so ist eine Regel immer gültig.
- Verbindungs- und Parametrierregeln sind erfüllt, wenn sie gültig sind und auch ihre Konsistenzbedingung erfüllt ist.
- Die Konsistenz einer Komponenteninstanz ist gegeben, wenn alle ihre gültigen Verbindungs- und Parametrierregeln erfüllt sind.
- Ein Anlagenmodell ist konsistent, wenn die Konsistenz für alle enthaltenen Komponenteninstanzen gegeben ist.

Beschreibungsentitäten für Verbindungs- und Parametrierregeln

Abbildung 7.5 zeigt die erforderlichen Strukturelemente für Regeln zur Beschreibung technischer Abhängigkeiten von Komponenten. Regeln bestehen generell aus einer Konsistenzbedingung (`ConsistencyRequirement`) und optional aus einer Vorbedingung (`PreCondition`), welche die Gültigkeit der Regel einschränkt: Wenn die Vorbedingung nicht erfüllt ist, wird die Konsistenzbedingung nicht gefordert.

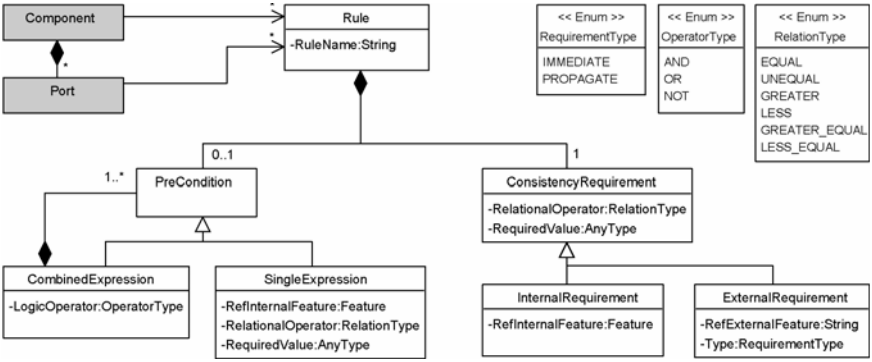


Abbildung 7.5: Entitäten zur Beschreibung von Verbindungs- und Parametrierregeln

Vorbedingung einer Regel

Eine einzelne Vorbedingung ist ein elementarer logischer Ausdruck (`SingleExpression`), der eine referenzierte Komponenteneigenschaft (`RefInternalFeature`) mit einem Wert vergleicht. Der Vergleich der Eigenschaft mit einem Wert erfolgt durch einen der logischen Vergleichsoperatoren (`RelationOperator`) GLEICH, UNGLEICH, GRÖSSER, KLEINER, GRÖSSER_ODER_GLEICH, KLEINER_ODER_GLEICH. Der zu vergleichende Wert wird entweder durch Angabe eines individuellen Wertes oder durch die Angabe einer weiteren Komponenteneigenschaft spezifiziert, sodass auch Vorbedingungen der Form „WENN *Eigenschaft1* GLEICH *Eigenschaft2*“ möglich sind.

Für komplexere Vorbedingungen können elementare Ausdrücke (`SingleExpression`) mittels logischer Verknüpfungsoperatoren wie NICHT, UND, ODER zu zusammengesetzten Aussagen verbunden werden (Kompositionalität). Diese werden durch die Entität `CombinedExpression` repräsentiert. Eine zusammengesetzte Aussage kann Elementaraussagen oder weitere zusammengesetzte Aussagen enthalten. Auf diese Weise können mehrere Vorbedingungen zu beliebig verschachtelten logischen Ausdrücken kombiniert werden [Wu04].

Konsistenzbedingung einer Regel

Die Konsistenzbedingung der Regel besteht ebenfalls aus einer referenzierten Eigenschaft, einem logischen Vergleichsoperator (`RelationOperator`) und dem geforderten Wert. Die Unterscheidung zwischen Parametrier- und Verbindungsregeln erfolgt durch die Differenzierung der referenzierten Eigenschaft. Bei Parametrierregeln handelt es sich um eine interne Konsistenzbedingung (`InternalRequirement`), die sich auf eine Eigenschaft derselben Komponente (`RefInternalFeature`) bezieht. Bei Verbindungsregeln handelt es sich um eine externe Konsistenzbedingung (`ExternalRequirement`), die sich auf eine Eigenschaft möglicher verbundener Komponenten bezieht (`RefExternalFeature`). Daher wird die externe

Eigenschaft über ihren Namen referenziert. Der in der Konsistenzbedingung geforderte Wert wird ebenfalls entweder durch Angabe eines individuellen Wertes oder durch die Angabe einer weiteren Komponenteneigenschaft spezifiziert, sodass auch Konsistenzbedingungen der Form „*externe Eigenschaft1* muss GLEICH *interner Eigenschaft2* sein“ möglich sind.

Bei Verbindungsregeln wird der Bezugspunkt der externen Konsistenzbedingung durch Angabe ihres Typs (`RequirementType`) genauer spezifiziert. Externe Konsistenzbedingungen vom Typ IMMEDIATE müssen durch direkt verbundene Komponenteninstanzen erfüllt werden. Externe Konsistenzbedingungen vom Typ PROPAGATE sind ungebunden: Sie müssen von irgendeiner anderen Komponenteninstanz erfüllt werden, zu der eine Verbindung – auch über mehrere Komponenteninstanzen hinweg – besteht. Damit kann die Tatsache ausgedrückt werden, dass für die korrekte Funktionalität einer Komponente eine Eigenschaft in ihrer Umgebung gewährleistet werden muss, unabhängig davon, wo diese Eigenschaft erfüllt ist.

Regeln können wie Eigenschaften der Komponente selbst oder einem ihrer Anschlusspunkte zugeordnet sein. Auf diese Weise kann der Gültigkeitsbereich einer Regel präzisiert werden. Zugeordnete Parametrierregeln gelten dann nur innerhalb der Komponente oder innerhalb eines Anschlusspunktes. Einem Anschlusspunkt zugeordnete Verbindungsregeln gelten für andere Komponenteninstanzen, die mit diesem Anschlusspunkt verbunden sind. Verbindungsregeln, die einer Komponente zugeordnet werden, gelten für alle mit Anschlusspunkten der Komponente verbundenen Komponenteninstanzen.

Beispiele für Verbindungs- und Parametrierregeln

Die folgenden Beispiele von Verbindungs- und Parametrierregeln für die Komponente aus Abbildung 7.4 verdeutlichen den Aufbau von Verbindungs- und Parametrierregeln. Die verwendete Pseudo-Notation ist frei gewählt und entspricht den Vorgaben des Beschreibungsmodells.

Beispiel einer Verbindungsregel im Eingangs-Anschlusspunkt 1:

$$\texttt{IF Materialfluss = permanent THEN Belegterkennung}_{\texttt{ext}}\texttt{ = TRUE}$$

Beschreibung: Wenn ein unterbrechungsfreier Materialtransport gewährleistet werden soll, dann muss die zuführende Komponente einen Sensor zu Belegterkennung aufweisen, um ein Übergabesignal erkennen zu können.

Beispiel einer Verbindungsregel im Ausgangs-Anschlusspunkt 2:

$$\texttt{IF Materialfluss = permanent THEN Flussrichtung}_{\texttt{ext}}\texttt{ = horizontal}$$

Beschreibung: Ein unterbrechungsfreier Materialtransport kann nur gewährleistet werden, wenn die nachfolgende Komponente in horizontaler Richtung weitertransportiert (bei vertikaler Flussrichtung müsste das Förderband für die Materialabnahme gestoppt werden).

Beispiel einer Verbindungsregel im Ausgangs-Anschlusspunkt 2 ohne Vorbedingung:

$\texttt{Förderhöhe}_{\texttt{ext}}$ = Förderhöhe

Beschreibung: Die Förderhöhe der nachfolgenden Komponente muss den gleichen Wert wie die eigene Förderhöhe aufweisen. Da keine Vorbedingung angegeben ist, gilt die Regel immer.

Beispiel einer komponenteninternen Parametrierregel:

```
IF Fördergeschwindigkeit > 2 THEN Motorleistung > 5
```

Beschreibung: Wenn die Fördergeschwindigkeit > 2 m/s sein soll, dann muss eine Motorleistung > 5 kW gewählt werden.

7.2.5 Beschreibung von Baugruppen

Abbildung 7.6 zeigt die grundlegenden Elemente einer Baugruppenbeschreibung. Eine als wiederverwendbare Teillösung definierte Baugruppe (`Composite`) verhält sich nach außen wie eine Komponente und kapselt ihren internen Aufbau. Der interne Aufbau besteht aus mehreren internen Elementen (`CompositeElement`). Bei internen Elementen handelt es sich entweder um instanziierte, konfigurierte und verbundene Komponenten (`ComponentInstance`) oder um weitere Baugruppen. Durch diese Verwendung des Kompositum-Musters [GHJV95] ergibt sich eine hierarchische Aufbaustruktur. Auf der untersten Hierarchieebene müssen interne Elemente letztendlich als Komponenteninstanzen ausgeführt sein.

Wie Komponenten verfügen Baugruppen über Eigenschaften und Schnittstellen, bestehend aus einem oder mehreren Anschlusspunkten, über die die als Baugruppe instanziierte Teillösung auf der entsprechenden Hierarchieebene mit den Anschlusspunkten von anderen Baugruppen verbunden werden kann. Da eine Baugruppe ein abstraktes Modellkonstrukt darstellt, das physikalisch nur in Form seiner internen Komponenteninstanzen existiert und die Verbindung von Baugruppen eine Verbindung der Komponenteninstanzen an ihren Schnittstellen entspricht, müssen ihre Anschlusspunkte (`CompositePort`) auf die Anschlusspunkte interner Komponenteninstanzen abgebildet werden. Dies erfolgt über die Referenz `MappingPort`.

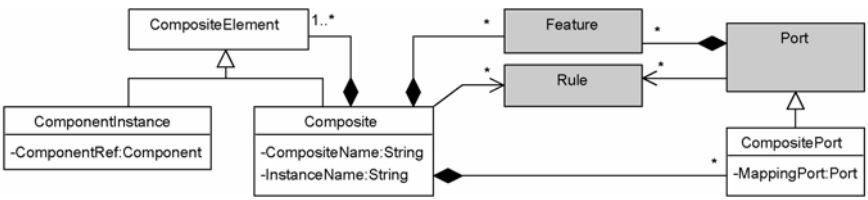


Abbildung 7.6: Entitäten zur Beschreibung von Baugruppen

7.2.6 Beschreibung der internen Varianten von Teillösungen

In Kapitel 5.3.1.5 wurde deutlich, dass eine Automatisierung der Integration von Baugruppen als Teillösungen durch die Abbildung des Wissens über die Varianten des internen Aufbaus zur Anpassung an wechselnde Anforderungen erreicht werden kann. Dabei müssen sowohl die invariablen als auch die variablen Elemente beschrieben werden. Die invariablen Elemente des internen Aufbaus können bereits mittels der im vorherigen Abschnitt vorgestellten Entitäten spezifiziert werden. Damit wird die Standard-Baugruppenvariante der Teillösung, bestehend aus instanziierten, konfigurierten Komponenten und ihren Verbindungen, festgelegt.

Beschreibung des Wissens über mögliche Varianten

Das Wissen über alternative Varianten einer Zusammenstellung von Komponenten kann nach [Dujm02] in drei Ebenen unterschieden werden:

- Auf der Ebene der *Architekturvariabilität* werden alternative Verbindungsstrukturen zwischen den internen Komponenten betrachtet. Zur Kategorie der Architekturvariabilitäten gehören auch weitere Instanziierungen von Komponenten, optionale Komponenten und die Zuordnung von anderen Komponenteninstanzen zu den Schnittstellen der Teillösung.
- Auf der Ebene der *Komponentenvariabilität* werden alternative Komponenten innerhalb einer Verbindungsstruktur betrachtet. Hierbei wird die Absicht verfolgt, Komponenten mit ähnlicher Funktion gegeneinander austauschen zu können.
- Auf der Ebene der *Instanzenvariabilität* werden alternative Konfigurationen einzelner Komponenteninstanzen betrachtet.

Mit diesen drei Ebenen werden alle Variationsmöglichkeiten für Baugruppenvarianten der wiederverwendbaren Teillösung berücksichtigt. Die Ebenen sind hierarchisch angeordnet: Innerhalb einer Architekturvariante können weitere Komponentenvarianten vorliegen, eine Komponentenvariante kann weitere Instanzenalternativen beinhalten. Abbildung 7.7 veranschaulicht den Zusammenhang zwischen den drei Variabilitätsebenen. Eine Baugruppenvariante der Teillösung kann aus Varianten der drei Variabilitätsebenen in beliebiger Kombination bestehen.

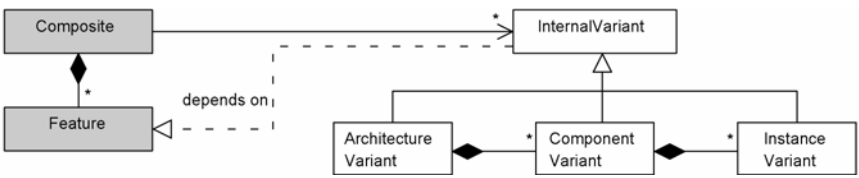


Abbildung 7.7: Zusammenhang der Variabilitätsebenen einer wiederverwendbaren Teillösung

Abbildung 7.8 zeigt einige Beispiele für Baugruppenvarianten am Beispiel einer Teillösung Lager. Architekturvariante A sieht ein weiteres Regal innerhalb des Lagers vor. Komponentenva-

riante B definiert ein 2-fach Regalbediengerät als alternative Komponente zum bestehenden Regalbediengerät. Architekturvariante C1 sieht ein zusätzliches Förderband als Bypass zum Regalbediengerät und Instanzenvariante C2 die Konfiguration der optionalen Komponenteneigenschaft „Scanner“ in der Komponente Lastaufnahmepuffer 1.

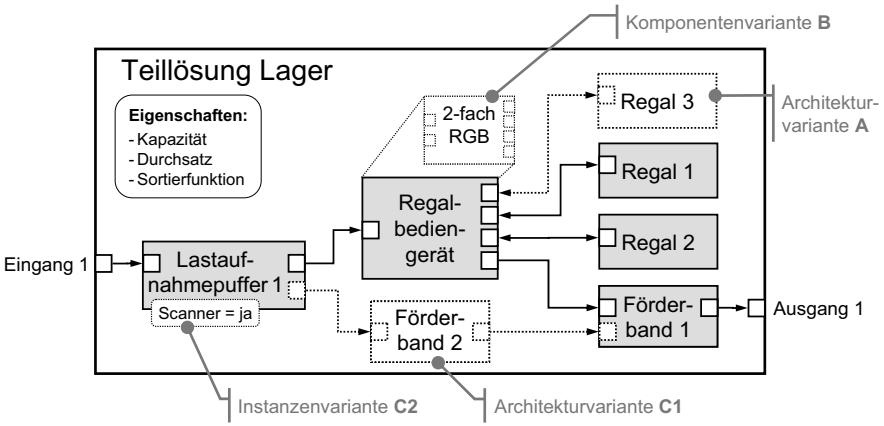


Abbildung 7.8: Beispiel für Varianten des internen Aufbaus einer Teillösung

Bei der Spezifikation von Baugruppenvarianten einer Teillösung können die technischen Abhängigkeiten zwischen den internen Komponenten bereits berücksichtigt werden, indem auf allen Variabilitätsebenen nur zulässige Varianten vorgesehen werden. Die Verbindungs- und Parametrierregeln der internen Komponenten sind jedoch dann von Bedeutung, wenn bei einer als Baugruppe instanziierten und in einer der vorgesehenen Varianten ausgeprägten Teillösung zusätzliche, benutzerspezifische Komponenten ergänzt werden sollen.

Beschreibung des Wissens über Variantenwahl

Um eine selbstständige Anpassung einer Baugruppe durch Softwareagenten zu ermöglichen, müssen auch die Konfigurationsmöglichkeiten für die Anpassung des internen Aufbaus mit den Informationen über die Teillösung integriert werden. Dazu müssen die in den jeweiligen Variabilitätsebenen definierten Varianten in Bezug zu den übergeordneten Baugruppeneigenschaften gesetzt werden. Dadurch kann die konkrete Ausprägung ihres internen Aufbaus in Abhängigkeit von den Ausprägungen von Baugruppeneigenschaften erfolgen. Im Beispiel aus Abbildung 7.8 bestehen folgende Zusammenhänge der Varianten zu den Eigenschaften des Lagers.

1. Ist die Kapazität des Lagers größer 20 und kleiner 31 Einheiten, wird als Architekturvariante ein drittes Regal gewählt, da jedes Regal 10 Einheiten aufnimmt (Architekturvariante A).

2. Soll der Durchsatz des Lagers mehr als 60 Einheiten pro Stunde betragen, muss als Komponentenvariante das Regalbediengerät durch ein 2-fach Regalbediengerät ersetzt werden, das eine schnellere Einlagerung ermöglicht (Komponentenvariante B).

3. Wird die optionale Sortierfunktion des Lagers gewählt, wird als Instanzenvariante von Lastaufnahmepuffer 1 der Einbau des optionalen Scanners zur Einheitenerkennung konfiguriert. Zudem wird als Architekturvariante ein weiteres Förderband parallel zum Regalbediengerät gewählt, um nicht einzulagernde Einheiten aus Lastaufnahmepuffer 1 schneller zum Lagerausgang transportieren zu können (Architekturvariante C1 mit Instanzenvariante C2).

Die Beschreibung des Wissens über die geeignete Variantenwahl muss eine eindeutige Zuordnung von Eigenschaftsausprägungen zu Varianten ermöglichen. Dabei ist die Möglichkeit von 1:n Abbildungen zu berücksichtigen, da die Wahl einer bestimmten Variante durchaus von einer Kombination von Eigenschaften abhängen kann. Daher ist für die Spezifikation von Varianten-Konfigurationsmöglichkeiten ebenfalls ein regelbasierter Ansatz zu wählen. Da solche *Konfigurationsregeln* aber keine Anforderung, sondern eine Handlungsableitung darstellen, ist für den Inhalt der Regel die metasprachliche Implikation A ⊢ B zu wählen [Wiki06b]. Sie besagt: „Aus A folgt B“. A wird Prämisse und B Konklusion (Folgerung) genannt. Dabei definiert die Prämisse einer Konfigurationsregel die Ausprägungen von Eigenschaften mit aussagelogischen Sätzen (z.B. `Eigenschaft = Wert`). Die Beschreibung einer Kombination von Eigenschaften erfolgt mittels logischer Verknüpfungsoperatoren (analog zur Beschreibung von Vorbedingungen). Der Folgerungsteil der Konfigurationsegel spezifiziert die zu wählende Baugruppenvariante der Teillösung, wenn die Prämisse erfüllt ist. Das folgende Beispiel (in Pseudo-Notation) für die Komponentenvariante B der Teillösung Lager verdeutlicht den Aufbau:

```
IF Kapazität > 20 AND Durchsatz > 60
THEN wähle Komponentenvariante B
```

Die vorgestellte Form zur Beschreibung von Anpassungsmöglichkeiten wiederverwendbarer Teillösungen entspricht dem Ansatz der Komponentenframeworks [Dujm02]. In Kapitel 3.7 wurde deutlich, dass sich dieser Ansatz für klar abgrenzbare Teilbereiche mit definierbarer, überschaubarer Variantenmenge und klarer Zuordenbarkeit einzelner Varianten eignet. Dieser Umstand ist bei der Definition wiederverwendbarer Teillösungen zu berücksichtigen. Die Wahl einer zu großen oder funktional unzusammenhängenden Baugruppe mit vielen Baugruppenvarianten als wiederverwendbare Teillösung führt zu einem hohen Aufwand und Komplexität bei der Spezifikation des Wissens über die Variantenwahl, welcher den Wiederverwendungsnutzen in der Gesamtbetrachtung aufhebt. Die im Rahmen der Arbeit gesammelten praktischen Erfahrungen in industriellen Projekten haben gezeigt, dass ein Wiederverwendungsnutzen dann noch gegeben ist, wenn eine wiederverwendbare Teillösung aus ca. 20% variablen und 80% invariablen Anteilen besteht.

7.2.7 Umsetzung des Beschreibungsmodells

Das Beschreibungsmodell stellt ein Strukturmetamodell für konkrete anwendungsspezifische Komponentenmodelle dar. Diese können eine spezifische Syntax und Semantik und durchaus auch zusätzliche Strukturen und Informationen enthalten. Durch die Vorgaben des Beschreibungsmodells wird jedoch gewährleistet, dass alle für die Unterstützungsfunktionalität der Softwareagenten erforderlichen Informationen und Wissen formalisiert und mit hinreichender informationstechnischer Aussagekraft verfügbar sind.

Für die praktische Umsetzung des Beschreibungsmodells für Baugruppenvarianten und Varianten-Konfigurationsregeln in wiederverwendbaren Teillösungen stehen bereits geeignete Konzepte aus den Forschungsbereichen der Frameworkentwicklung und der Produktlinienentwicklung zur Verfügung [Dujm02], [Krue02], [Beuc05]. In [Dujm02] wurde eine XML-basierte Spezifikationssprache für die formalisierte Beschreibung von Komponenten-Frameworks vorgestellt, welche alle drei Variabilitätsebenen des Variantenwissens berücksichtigt. Damit kann sie auch für die Beschreibung wiederverwendbarer Teillösungen eingesetzt werden.

7.3 Formalisierte Beschreibung komponentenübergreifenden Wissens

In Kapitel 5.3.2.1 wurde deutlich, dass das komponentenübergreifende Wissen für die Unterstützung der Erstellung gewerkspezifischer Teilmodelle und der fachspezifischen Optimierung des Anlagenmodells durch einen regelbasierten Ansatz rechnergestützt und anwendungsunabhängig verfügbar gemacht werden kann. Gewerkspezifische Auswahl-, Erstellungs- oder Überprüfungsregeln stellen eine Handlungsanweisung dar, bei der die Regelprämisse ein Muster von Komponentenkombinationen und -konfigurationen beschreibt und die Regelkonklusion eine Aktion (Auswertungsfunktion), welche auf diese Klasse anzuwenden ist. Das folgende Beispiel aus [ScFa05] zeigt, wie die Ermittlung der erforderlichen Überlaufschutz-Verriegelungen für alle Behälter eines verfahrenstechnischen Anlagenmodells durch eine Regel abgebildet werden kann. Zum besseren Verständnis sind die Regeln in natürlichsprachlicher Form dargestellt.

```
WENN  ein Behälter (mindestens) einen Zulauf besitzt,
UND   der Zulauf durch ein Stellglied (Ventil, Pumpe)
      verschlossen werden kann,
UND   der Füllstand des Behälters erfassbar ist,
DANN  erstelle eine auf der Füllstandsmessung und einem
      geeigneten Stellglied basierende Überlaufschutzfunktion.
```

Das Wissen über die Erstellung gewerkspezifischer Teilmodelle wird gegebenenfalls durch projektunabhängige Informationen ergänzt. Im vorliegenden Beispiel sind dies die Informationen über die verfügbaren Stellglieder und die geforderten Überlaufschutzfunktionen.

Durch den regelbasierten Ansatz ist eine einfache Änderbarkeit oder Erweiterbarkeit des spezifizierten Fachwissens gewährleistet. Zudem kann das spezifizierte Fachwissen nicht nur zur Erstellung gewerkspezifischer Teilmodelle, sondern auch zur Überprüfung von manuellen Änderungen oder Ergänzungen an Teilmodellen genutzt werden. Voraussetzung für die Anwendbarkeit des regelbasierten Ansatzes ist die Verfügbarkeit explizit beschriebener Engineeringinformationen des Anlagenmodells, um ihre rechnergestützte Auswertbarkeit zu ermöglichen. Komponentenmodelle, deren Informationen und Informationsstrukturen die Merkmale des in Kapitel 7.2 eingeführten Beschreibungsmodells erfüllen, gewährleisten eine einfache Auswertbarkeit und eindeutige Zuordnung der Engineeringinformationen eines Anlagenmodells.

7.4 Bereitstellung, Transfer und Erfassung von Informationen und Wissen

Die Bereitstellung von Informationen und Wissen über Komponenten und über gewerkspezifische Zusammenhänge geschieht durch Domänenexperten (Abbildung 7.9 links). Für Komponenten sind die Eigenschaften, Schnittstellen, Verbindungs- und Parametrierregeln zu spezifizieren. Für Baugruppen, die als wiederverwendbare Teillösung genutzt werden, sind der interne Aufbau, seine Varianten und die zugehörigen Konfigurationsregeln zu spezifizieren. Für Gewerke sind die spezifischen Auswahl-, Erstellungs- oder Überprüfungsregeln und die projektunabhängigen Gewerkinformationen zu spezifizieren. Hierbei handelt es sich um einen einmaligen, initialen Aufwand, da die erstellen Spezifikationen verwendungsunabhängig sind und somit beim Engineering unverändert mehrfach wiederverwendet werden können. Der Transfer von Informationen und Wissen erfolgt durch die Ablage der erstellten Komponentenbeschreibungen in der Komponentenbibliothek bzw. durch die Erstellung fachspezifischer Wissensbasen.

Damit werden Informationen und Wissen beim Engineering explizit verfügbar, können durch die Softwareagenten rechnergestützt erfasst und für ihre unterstützenden Handlungen genutzt werden (Abbildung 7.9 rechts). Komponenteninformationen und -wissen werden bei der Instanziierung einer Komponente von einem Agenten des Agententyps Komponenten-Agent statisch erfasst und fortan bei seinen Agenten-Aktionen und -Interaktionen berücksichtigt. Die Informationen über die Konfigurationen und Verbindungen einzelner Komponenteninstanzen sind im Anlagenmodell spezifiziert und somit ebenfalls rechnergestützt verfügbar. Im Gegensatz zu den Komponenteninformationen werden die Informationen über Komponenteninstanzen bei den Tätigkeiten des Ingenieurs verändert. Daher müssen sie vor der Ausführung unterstützender Handlungen dynamisch aus dem Anlagenmodell erfasst werden. Gewerkinformationen und Wissen werden bei der Initialisierung von Agenten des Typs Aspekt-Agent übernommen und im Rahmen seiner Aktivitäten angewendet. Besteht die Aufgabe des Aspekt-Agenten in der Erstellung eines gewerkspezifischen Teilmodells, kann dieses auch durch manuelle Eingriffe des Ingenieurs verändert oder ergänzt werden. Daher müssen die aktuellen Informationen des Teilmodells dynamisch erfasst werden. Der Bibliotheks-Agent erfasst bei seiner Initialisierung die für

Komponentenauswahl und Schnittstellenprüfung erforderlichen Informationen über Eigenschaften und Schnittstellen der in der Komponentenbibliothek verfügbaren Komponenten.

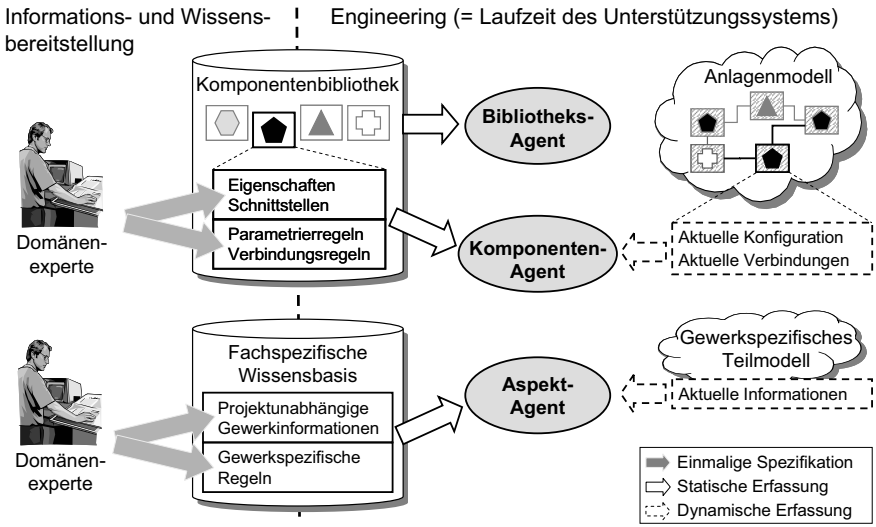


Abbildung 7.9: Bereitstellung, Transfer und Erfassung von Informationen und Wissen

Da die Softwareagenten bei ihren Interaktionen zur Handhabung von Wechselwirkungen ihre jeweiligen Informationen und ihr Wissen untereinander verknüpfen müssen, ist bei der Beschreibung von Komponenteninformationen und -wissen für alle Komponenten ein gemeinsames Vokabular zu verwenden. Die Dokumentation eines gemeinsamen, semantisch fundierten Vokabulars zur Abstimmung mehrerer Komponentenexperten wird durch die Verwendung von Begriffsmodellen, so genannter Ontologien, erleichtert und systematisiert [Müll04].

In diesem Kapitel wurde ein Beschreibungsmodell zur formalisierten, integrierten Beschreibung von Komponenteninformationen und -wissen entwickelt. Durch die Verwendung regelbasierter Konzepte kann Wissen über die technischen Abhängigkeiten von Komponenten und über die Varianten von wiederverwendbaren Teillösungen explizit und anwendungsneutral spezifiziert werden und steht so für das Engineering zur Verfügung. Damit wird der Informationsverlust zwischen Komponentenentwicklung und Engineering überwunden und eine effiziente Unterstützung durch Softwareagenten ermöglicht. Daneben wurde gezeigt, wie fachspezifisches komponentenübergreifendes Wissen durch regelbasierte Konzepte für die rechnergestützte Verwertung verfügbar gemacht werden kann. Der Ingenieur wird beim Engineering normalerweise das beschriebene Wissen nicht lesen, sondern durch die Softwareagenten unterstützt, welche das Wissen rechnergestützt verwerten. Die dafür notwendigen Aktionen und Interaktionen der Softwareagenten werden im folgenden Kapitel ausführlich beschrieben.

8 Funktionales Verhalten des Agentensystems

Gegenstand dieses Kapitels ist die Ausarbeitung der informationstechnischen Prozesse des Agentensystems zur Umsetzung der in den vorangegangen beiden Kapiteln erarbeiteten Konzepte. Zunächst werden die Grundsätze und die Randbedingungen für die verteilten Aktionen und Interaktionen der Softwareagenten des Unterstützungssystems diskutiert. Daraus werden die erforderlichen Fähigkeiten für die verschiedenen Rollen der Softwareagenten ermittelt und der Ablauf ihrer Aktionen und Interaktionen ausführlich erläutert. Abschließend werden wichtige Merkmale von Interaktionen betrachtet und die Kooperation mit dem Ingenieur vorgestellt.

8.1 Grundsätze der Aktionen und Interaktionen der Agenten

Durch die Kapselung von Informationen und Wissen verfügen Agenten über lokale Selbstständigkeit bei der Ausführung ihrer Aktionen innerhalb ihres Einflussbereiches zur individuellen Anpassung der von ihnen vertretenen Komponenteninstanzen oder gewerkspezifischen Teilmodelle. Mittels Interaktionen erfolgt die koordinierte Ermittlung entstehender Wechselwirkungen und entsprechender Anpassungsmöglichkeiten sowie ihre Umsetzung durch die betroffenen Softwareagenten. Zur Verdeutlichung zeigt Abbildung 8.1 die Einflussbereiche und Interaktionen in einer exemplarischen Systemsituation.

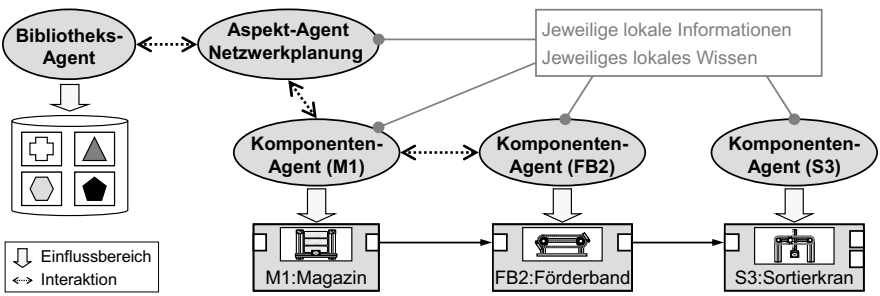


Abbildung 8.1: Einflussbereich und beispielhafte Interaktionen

In Kapitel 6.4 wurden bereits die erforderlichen Beziehungen zwischen den drei Agententypen Komponenten-Agent, Aspekt-Agent und Bibliotheks-Agent ermittelt und die Interaktionszwecke für jede Beziehung identifiziert. Im Folgenden werden ausgehend von der menschlichen Vorgehensweise die Grundsätze des Ablaufs von Aktionen und Interaktionen abgeleitet.

Adaption der menschlichen Vorgehensweise zur Wechselwirkungshandhabung

Die Prinzipien der Abläufe innerhalb des agentenorientierten Unterstützungssystems werden durch Übertragung der Vorgehensweise des Ingenieurs ermittelt. In Kapitel 2.4 wurden die Ü-

Überlegungen und Detailtätigkeiten zur Handhabung von Wechselwirkungen diskutiert, die bei den Tätigkeiten des komponentenbasierten Engineerings erforderlich sind. Abbildung 8.2 zeigt den Ablauf der menschlichen Vorgehensweise und verdeutlicht die Zusammenhänge zwischen verschiedenen Detailtätigkeiten und Überlegungen.

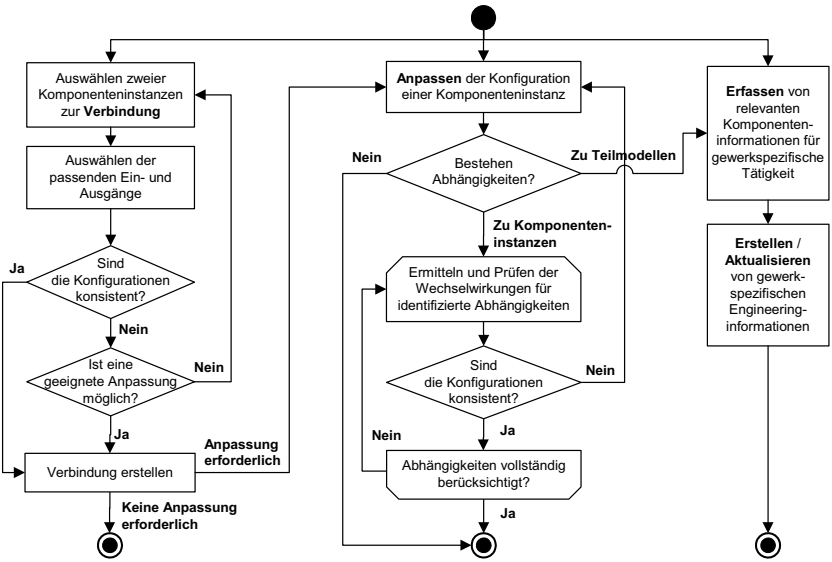


Abbildung 8.2: Menschliche Vorgehensweise zur Handhabung von Wechselwirkungen

In der Abbildung wird der iterative Charakter des menschlichen Vorgehens deutlich. Ausgehend von einer bei der Durchführung einer Tätigkeit entstehenden Veränderung von Engineeringinformationen werden die Wechselwirkungen zu anderen Engineeringinformationen geprüft und diese gegebenenfalls angepasst, um Inkonsistenzen aufzulösen. Ausgehend von diesen Anpassungen werden schrittweise weitergehende Wechselwirkungen ermittelt und Anpassungen durchgeführt, bis die Konsistenz aller Engineeringinformation sichergestellt ist. Bei der Übertragung dieses Vorgehens auf die Abläufe innerhalb des agentenorientierten Unterstützungssystems ist zu beachten, dass die Softwareagenten zwar autonom handeln, um Wechselwirkungen zu prüfen, Inkonsistenzen zu identifizieren und die erforderlichen Anpassungen zu ermitteln, die Entscheidung über die Umsetzung von Anpassungen jedoch stets dem Ingenieur obliegen soll.

Aufgaben von Aktionen und Interaktionen

Die Softwareagenten des Unterstützungssystems nehmen menschliche Tätigkeiten als Auftrag wahr und verfolgen ihre jeweiligen Unterstützungsziele, indem sie die spezifizierten Rollen einnehmen. In jeder Rolle besitzen sie verschiedene Fähigkeiten, die bei ihren Unterstützungshandlungen eingesetzt werden. Fähigkeiten bestehen in Aktionen und Interaktionen, mittels derer die

Agenten die in Kapitel 6.1 identifizierten Teilziele zur Berücksichtigung technischer Abhängigkeiten erfüllen. Bei ihren Aktionen müssen sie erkennen, an welcher Stelle Wechselwirkungen auftreten können (vgl. Abbildung 8.2). Davon ausgehend muss der betroffene Agent entsprechende Interaktionen einleiten, die folgende grundlegenden Aufgaben erfüllen:

- *Ermittlung der betroffenen Agenten*: Ausgehend von einer menschlichen Tätigkeit wie dem Verbinden oder der Konfiguration von Komponenteninstanzen werden von dem beauftragten Komponenten-Agenten die Agenten aller durch die Änderung möglicherweise betroffenen anderen Komponenteninstanzen oder gewerkspezifischen Teilmodelle ermittelt und in die Interaktionen zur Handhabung von Wechselwirkungen mit einbezogen.

- *Ermittlung und Prüfung von Wechselwirkungen*: Die jeweils relevanten lokalen Konfigurationen und Konsistenzbedingungen werden zwischen den Agenten situationsabhängig ausgetauscht und miteinander verknüpft. Dabei werden Wechselwirkungen ermittelt, geprüft und inkonsistente Komponentenkonfigurationen oder Teilmodellinformationen identifiziert.

- *Abstimmung von möglichen Anpassungen*: Jeder Agent ermittelt zunächst lokal mögliche Anpassungen zur Auflösung der Inkonsistenzen. Dabei sind auch die durch Parametrierregeln spezifizierten Anforderungen an die komponenteninterne Konsistenz zu berücksichtigen. Da aus den Anpassungen weitere indirekte Wechselwirkungen entstehen können, werden wiederum die entsprechenden Interaktionen zu ihrer Ermittlung und Prüfung eingeleitet.

- *Vollständige Umsetzung von Anpassungen*: Nachdem alle Wechselwirkungen ermittelt und von allen betroffenen Agenten geeignete Anpassungen unter Berücksichtigung der weitergehenden indirekten Wechselwirkungen identifiziert wurden, werden diese dem Ingenieur als Lösungsvorschlag präsentiert, abgestimmt und anschließend umgesetzt. Dazu gehören Anpassungen der Konfiguration von Komponenteninstanzen, das Erstellen von Verbindungen zwischen Komponenteninstanzen und die Anpassung gewerkspezifischer Teilmodelle.

Randbedingungen von Aktionen und Interaktionen

Aus den Zielen des Unterstützungssystems und den allgemeinen Anforderungen aus Kapitel 3.2 ergeben sich zusätzliche Randbedingungen an die Aktionen und Interaktionen der Agenten:

- *Interaktionen zur Behandlung von Inkonsistenzen*: Um eine konstruktive Unterstützung anzubieten, sollen die Interaktionen der Agenten die Sicherstellung der Komponentenkonsistenz bei Wechselwirkungen durch Erkennung und Aufhebung von Inkonsistenzen ermöglichen. Konflikte bei nicht auflösbaren Inkonsistenzen müssen erkannt und die entsprechenden Zusammenhänge dem Ingenieur nachvollziehbar vermittelt werden.

- *Koordinierter Ablauf von Aktionen und Interaktionen*: Um eine umfassende Nachvollziehbarkeit der vom Agentensystem vorgeschlagenen Lösung zu gewährleisten, sollen die verteilten Aktionen der Agenten und ihre Interaktionen in koordinierter Weise ablaufen. Der

Ablauf soll die Systematik des schrittweisen menschlichen Vorgehens bei der Handhabung von Wechselwirkungen widerspiegeln.

- *Ermöglichen manueller Entscheidungen*: Die Umsetzung einer vorgeschlagenen Lösung soll – abhängig vom Wunsch des Ingenieurs – vollständig oder nur für manuell ausgewählte Anpassungen erfolgen. Das vollständige Verwerfen der Lösung soll ebenfalls möglich sein.
- *Gewährleistung von Vollständigkeit und Endlichkeit:* Der Ingenieur soll bei Verwendung des Unterstützungssystems sicher sein, dass keine wichtigen Schritte vergessen wurden. Die Agenten-Interaktionen müssen daher sicherstellen, dass alle tatsächlich vorhandenen Wechselwirkungen erkannt und berücksichtigt werden. Bei der Umsetzung bzw. dem Verwerfen einer Lösung müssen alle entsprechenden Änderungen einbezogen werden. Zudem muss gewährleistet sein, dass die Agenten-Interaktionen stets in definierter Weise terminieren.
- *Einschränkung des Lösungsbereiches*: Es soll durch manuelle Vorgaben des Ingenieurs möglich sein, die Reichweite der Agenten-Interaktionen zu beschränken, um die Komplexität der Zusammenhänge überschaubar zu halten.

8.2 Fähigkeiten der Agenten

Ausgehend von den Zielen und Rollen der einzelnen Agententypen und ihren gekapselten Informationen und Wissen können nun ihre Fähigkeiten in Form von Aktionen und Interaktionen ermittelt werden. Sie werden in Tabelle 8.1, Tabelle 8.2 und Tabelle 8.3 den jeweiligen Rollen der Agententypen zugeordnet.

Tabelle 8.1: Fähigkeiten der Rollen des Agententyps Komponenten-Agent

Rolle	**Aktionen**	**Interaktionen**
Änderungs-wächter	Parameteränderung prüfen Änderungsauswirkungen ermitteln Rückmeldungen verwalten Anpassungen komponentenübergreifend umsetzen	Handhabung komponentenübergreifender Wechselwirkungen Beobachteranmeldung
Konfigura-tionsmanager	Konsistenzbedingungen und Änderungen weiterleiten Externe Konsistenzbedingungen und Änderungen prüfen Lokale Anpassungen umsetzen Verbindung erstellen oder löschen	Handhabung komponentenübergreifender Wechselwirkungen Komponentenübergreifende Verbindungsprüfung und -erstellung Informationsabfrage
Baugruppen-manager	Baugruppe intern anpassen Baugruppenverbindung umsetzen	

Tabelle 8.2: Fähigkeiten der Rollen des Agententyps Aspekt-Agent

Rolle	Aktionen	Interaktionen
Verbindungs-manager	Verbindung prüfen Zwischenkomponente ermitteln Verbindung entfernen	Komponentenübergreifende Verbindungsprüfung und –erstellung Zwischenkomponente anfordern
Instanzen-selektierer	Komponenteninstanzen auswählen	Informationsabfrage
Modellierer	Teilmodell bilden Teilmodell prüfen Komponenteninstanzen anpassen	
Teilmodell-manager	Abhängigkeiten verwalten Teilmodell aktualisieren	Beobachteranmeldung Handhabung komponentenübergreifender Wechselwirkungen

Tabelle 8.3: Fähigkeiten der Rollen des Agententyps Bibliotheks-Agent

Rolle	Aktionen	Interaktionen
Komponenten-verwalter	Suchinformationen initialisieren Suchinformationen aktualisieren	Zwischenkomponente anfordern
Komponenten-sucher	Komponente finden Komponente instanziieren	
Schnittstellen-prüfer	Schnittstellen prüfen	Komponentenübergreifende Verbindungsprüfung und -erstellung

Standardaktionen der Agenten wie das Verarbeiten von Informationsabfragen oder Anmeldungen wurden hier nicht aufgelistet. Der Ablauf der einzelnen agenteninternen Aktionen und ihr Zusammenwirken bei den Agenten-Interaktionen ist Gegenstand der folgenden Abschnitte. Zur grafischen Veranschaulichung werden UML Aktivitäts- und Sequenzdiagramme verwendet.

8.3 Aktionen und Interaktionen von Komponenten-Agenten

Vorgehensweise zur Ermittlung und Prüfung von Wechselwirkungen

Ein wichtiges Ziel der Komponenten-Agenten ist die Sicherstellung der Konsistenz zu verbundenen Komponenteninstanzen. Dazu sind die Wechselwirkungen zwischen den Konfigurationen zu ermitteln, zu prüfen und Anpassungsmöglichkeiten zur Vermeidung von Inkonsistenzen zu identifizierten. Abbildung 8.3 zeigt das Prinzip der wechselseitigen Ermittlung und Prüfung von Wechselwirkungen. Der prinzipielle Ablauf besteht aus folgenden Schritten:

- Die aktuell gültigen Verbindungsregeln jeder Komponenteninstanz (für den betroffenen Anschlusspunkt) werden ermittelt.
- Die Konsistenzbedingungen der gültigen Verbindungsregeln werden mit den Eigenschaften des Anschlusspunkts der jeweils anderen Komponenteninstanz verknüpft. Alternativ werden die Eigenschaften innerhalb der Komponente (die für alle Anschlusspunkte verfügbar sind), verwendet. Eigenschaften anderer Anschlusspunkte bleiben unberücksichtigt, da sie für die betrachtete Verbindung nicht relevant sind.
- Bei Inkonsistenzen aufgrund nicht erfüllter Konsistenzbedingungen wird die Anpassung der betroffenen Eigenschaften ermittelt. Dabei sind die Parametrierregeln sowie die zulässigen Werte von Eigenschaften der jeweiligen Komponenteninstanzen zu berücksichtigen. Zudem können durch Anpassungen weitere Verbindungsregeln Gültigkeit erlangen und müssen in die wechselseitige Prüfung mit einbezogen werden.
- Alle ermittelten Anpassungen sind erneut mit den externen Konsistenzbedingungen und internen Parametrierregeln abzustimmen und weitere Anpassungen zu ermitteln, bis alle Inkonsistenzen aufgehoben sind oder Konflikte durch nicht erfüllbare Regeln erkannt werden. Ermittelte Anpassungen werden nicht direkt ausgeführt, sondern als *temporäre Änderungen* zwischengespeichert. So kann ein vollständiges Verwerfen sichergestellt werden.

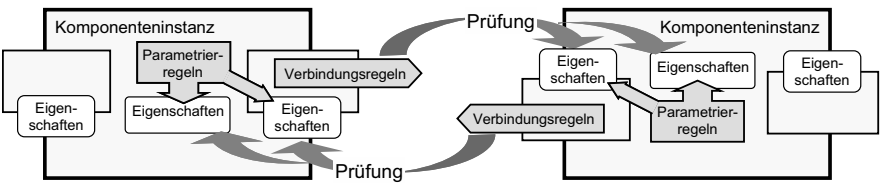


Abbildung 8.3: Wechselseitige Ermittlung und Prüfung von Wechselwirkungen

Zur besseren Übersicht werden im Folgenden zunächst die Agenten-Interaktionen zur systematischen Handhabung komponentenübergreifender Wechselwirkungen erläutert und anschließend die Aktionen einzelner Komponenten-Agenten zur Wechselwirkungsprüfung betrachtet.

8.3.1 Interaktion zur Handhabung von komponentenübergreifenden Wechselwirkungen

Die Interaktion zur Handhabung komponentenübergreifender Wechselwirkungen legt einen koordinierten Ablauf aller Agenten-Aktionen nach einer erkannten Änderung fest. Sie dient zur Sicherstellung der komponentenübergreifenden Konsistenz durch Wechselwirkungsprüfung, Änderungsermittlung und -umsetzung. Wird eine Änderung der Konfiguration einer Komponenteninstanz durch den Änderungswächter des betreffenden Komponenten-Agenten erkannt, leitet er die Interaktion ein. Für die koordinierte Ermittlung der Änderungsauswirkungen ist eine Ab-

stimmung aller betroffenen Agenten erforderlich. Dazu übernimmt der für die auslösende Änderung verantwortliche Komponenten-Agent die Koordination der Aktionen aller beteiligten Agenten. Abbildung 8.4 zeigt den Ablauf der Interaktion und die Aktionen der einzelnen Komponenten-Agenten. Die Interaktion wird durch das Weiterleiten der Änderungen und der Konsistenzbedingungen an alle direkt abhängigen Agenten angestoßen. Neben den benachbarten Komponenten-Agenten werden auch alle abhängigen Aspekt-Agenten informiert. Deren darauf folgende Aktionen sind in Abbildung 8.4 nicht darstellt und werden in Kapitel 8.4 erläutert. Alle betroffenen Komponenten-Agenten prüfen in ihrer Rolle als Konfigurationsmanager die übermittelten Konsistenzbedingungen und Änderungen, identifizieren Inkonsistenzen, ermitteln – falls möglich – erforderliche Änderungen und senden ihre Bestätigung oder Konfliktmeldung an den auslösenden Komponenten-Agenten zurück (Abbildung 8.4 Mitte). Komponenten-Agenten, die erforderliche Änderungen für ihre Komponenteninstanz ohne Konflikt ermittelt haben, sind für die Ermittlung weitergehender, indirekter Wechselwirkungen verantwortlich. Dazu werden alle von ihnen abhängigen Komponenten-Agenten an den Änderungswächter des koordinierenden Komponenten-Agenten zurückgemeldet sowie die jeweiligen Konsistenzbedingungen und Änderungen entsprechend weitergeleitet. Alle im weiteren Verlauf der Interaktion betroffenen Komponenten-Agenten verfahren nach demselben Muster (Abbildung 8.4 rechts). Nachdem alle betroffenen Komponenten-Agenten dem auslösenden Komponenten-Agenten eine Bestätigung oder Konfliktmeldung gesendet haben, ist die Ermittlung der komponentenübergreifenden Wechselwirkungen abgeschlossen. Die Ergebnisse können mit dem Ingenieur abgestimmt und vollständig für alle betroffenen Komponenten-Agenten umgesetzt werden.

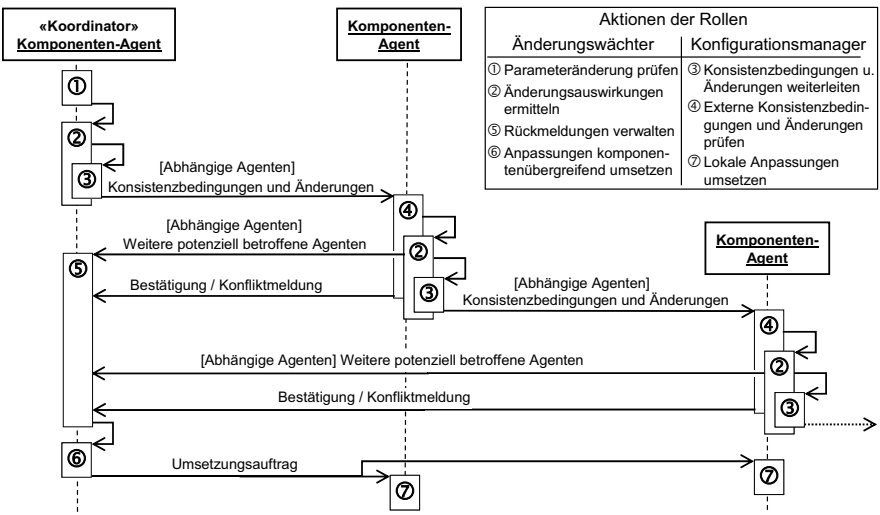


Abbildung 8.4: Interaktion zur Handhabung komponentenübergreifender Wechselwirkungen

In Abbildung 8.4 wird die Adaption des schrittweisen menschlichen Vorgehens auf die Interaktionen der Agenten deutlich: Ausgehend von einer Änderung werden zunächst die direkten Wechselwirkungen mit allen benachbarten Komponenteninstanzen geprüft, Inkonsistenzen identifiziert und erforderliche Änderungen ermittelt. In der Folge werden schrittweise für die jeweiligen Änderungen die indirekten Wechselwirkungen mit allen zur betroffenen Komponenteninstanz benachbarten Komponenteninstanzen geprüft. Die Interaktionen setzen sich fort, bis alle Wechselwirkungen konsistent berücksichtigt sind oder ein Konflikt zwischen den Konfigurationen zweier Komponenteninstanzen erkannt wurde. Wird durch einen Komponenten-Agenten ein Konflikt festgestellt, erfolgt keine Änderungsweiterleitung, sodass die Interaktion dort endet.

8.3.2 Aktionen der Rolle Änderungswächter

Aufgabe des Änderungswächters ist das Erkennen von Änderungen an der Konfiguration der vertretenen Komponenteninstanz und die Sicherstellung komponentenübergreifender Konsistenz durch Koordination der Wechselwirkungshandhabung.

8.3.2.1 Parameteränderung prüfen

Auslöser der Aktion ist eine Parameteränderung an der Konfiguration einer Komponenteninstanz durch den Ingenieur, die durch den Änderungswächter des betreffenden Komponenten-Agenten erkannt und als Auftrag wahrgenommen wird. Alternativ kann der Änderungsauftrag durch einen Aspekt-Agenten erfolgen. Zunächst werden die internen Wechselwirkungen für die Komponenteninstanz geprüft. Treten hierbei bereits Konflikte auf, ist keine Interaktion mit anderen Agenten erforderlich. Andernfalls wird die Ermittlung der Änderungsauswirkungen für alle abhängigen Agenten angestoßen. In Abbildung 8.5 ist der Ablauf der Aktion dargestellt.

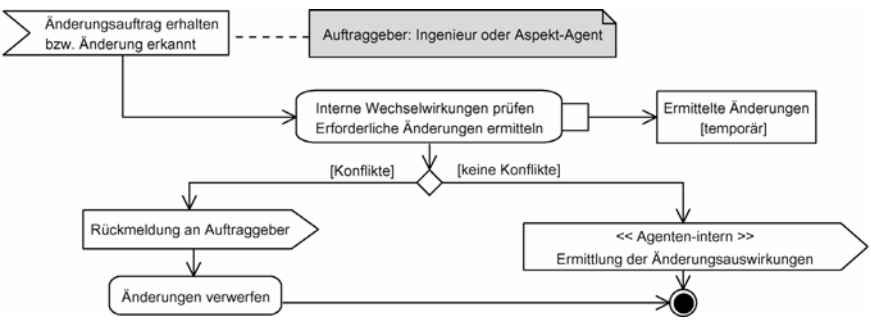


Abbildung 8.5: Aktion „Parameteränderung prüfen“

8.3.2.2 Änderungsauswirkungen ermitteln

Für die Ermittlung der Änderungsauswirkungen ist eine Weiterleitung der Änderungen und der eigenen Konsistenzbedingungen an alle von der Änderung potenziell betroffenen Softwareagenten erforderlich. Zunächst werden deshalb aus dem Umgebungsmodell die Abhängigkeiten ermittelt und die potenziell betroffenen Softwareagenten selektiert, abhängig vom Gültigkeitsbereich der Änderung. Betrifft die Änderung Eigenschaften, die einem Anschlusspunkt der Komponente zugeordnet sind, so ist nur der dort verbundene Komponenten-Agent potenziell von der Änderung betroffen, da die Eigenschaften nicht für Verbindungen über andere Anschlusspunkte genutzt werden können. Änderungen an komponenteninternen Eigenschaften können alle verbundenen Komponenten-Agenten betreffen. Zusätzlich können alle von der Komponenteninstanz abhängigen Aspekt-Agenten von der Änderung betroffen sein (siehe Kapitel 8.4.5). Sie werden ebenfalls über die Änderung informiert. Abbildung 8.6 zeigt den Ablauf der Aktion.

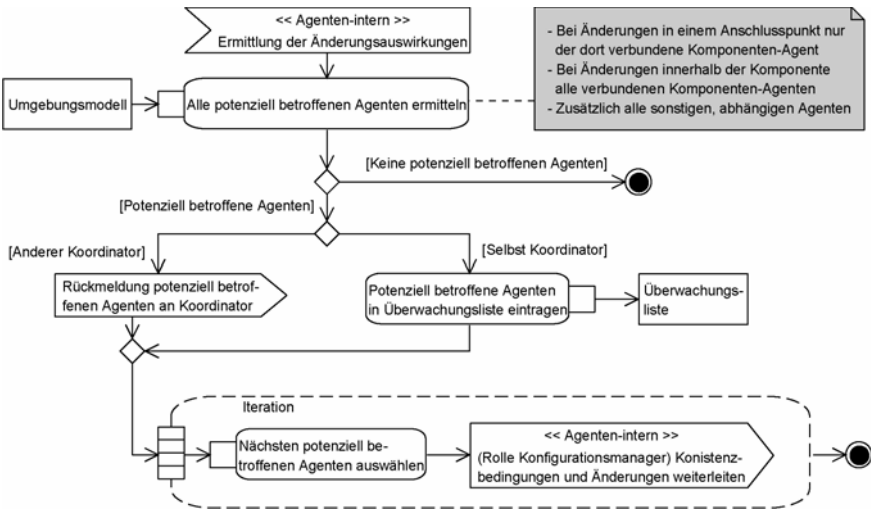


Abbildung 8.6: Aktion „Änderungsauswirkungen ermitteln“

Zudem unterscheidet der Komponenten-Agent, ob er selbst oder ein anderer Komponenten-Agent Auslöser der Änderung ist. Im ersten Fall ist er für die Koordination der Interaktion verantwortlich und legt eine Überwachungsliste an, in der alle potenziell betroffenen Agenten und später ihre Rückmeldungen eingetragen werden. Im zweiten Fall werden alle potenziell betroffenen Agenten an den Koordinator gemeldet. Anschließend erfolgt die Benachrichtigung aller potenziell betroffenen Agenten durch das Weiterleiten der Änderungen und der eigenen Konsistenzbedingungen. Dazu wird die entsprechende Aktion des Konfigurationsmanagers angestoßen.

8.3.2.3 Rückmeldungen verwalten

Der Koordinator der Interaktion zur Handhabung komponentenübergreifender Wechselwirkungen erfasst die von einer Änderung betroffenen Agenten und ihre Prüfungsergebnisse. Er erhält von anderen Komponenten-Agenten, die eine Anpassung ihrer Komponenteninstanz ermittelt haben, die Rückmeldung über alle weiteren betroffenen Komponenten-Agenten und ergänzt seine Überwachungsliste. Zudem erhält er von jedem betroffenen Komponenten-Agenten nach erfolgter Prüfung der Wechselwirkungen eine Bestätigung (inklusive der ermittelten Änderungen) oder eine Konfliktmeldung (mit Konfliktursache) und ergänzt die Überwachungsliste entsprechend. Sind die Rückmeldungen aller als betroffen gemeldeten Komponenten-Agenten erfolgt, ist die Prüfungsphase der Interaktion abgeschlossen und die komponentenübergreifende Umsetzung der ermittelten Anpassungen wird angestoßen. Abbildung 8.7 zeigt den gesamten Ablauf.

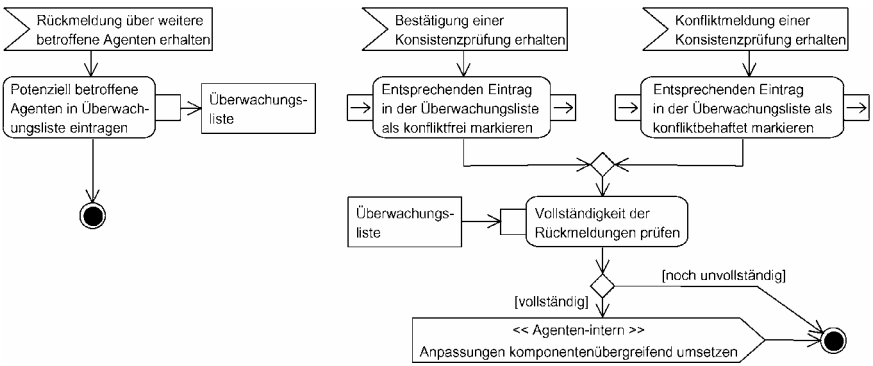


Abbildung 8.7: Aktion „Rückmeldungen verwalten“

8.3.2.4 Anpassungen komponentenübergreifend umsetzen

Alle nach einer Änderung ermittelten Anpassungen zur Sicherstellung der komponentenübergreifenden Konsistenz oder aber gemeldete Konflikte werden vom koordinierenden Komponenten-Agenten dem Ingenieur präsentiert und mit ihm abgestimmt. Der Ingenieur trifft die Entscheidung, ob die Anpassungen vollständig, gar nicht oder nur für bestimmte Komponenteninstanzen erfolgen sollen. Anschließend wird an jeden in der Überwachungsliste enthaltenen Agenten der entsprechende Umsetzungsauftrag gesendet (siehe Abbildung 8.8).

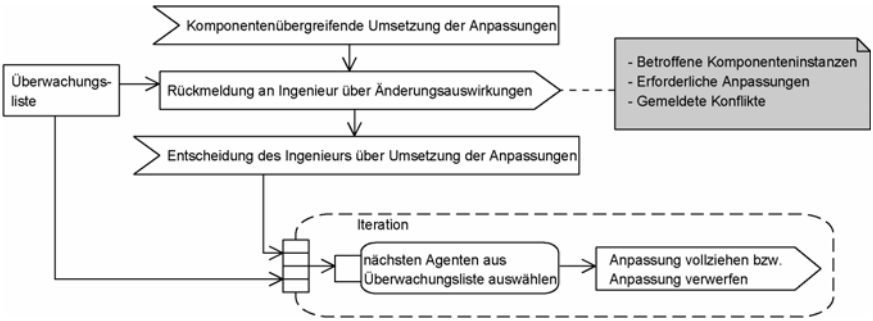


Abbildung 8.8: Aktion „Anpassungen komponentenübergreifend umsetzen“

8.3.3 Aktionen der Rolle Konfigurationsmanager

Die Aufgabe der Rolle Konfigurationsmanager besteht in der Handhabung und Anpassung der Konfiguration einer Komponenteninstanz unter Berücksichtigung der Konsistenz zu verbundenen bzw. zu verbindenden Komponenteninstanzen. Sie verfügt über verschiedene Aktionen zur Umsetzung des beschriebenen Ablaufs zur Ermittlung und Handhabung von Wechselwirkungen.

8.3.3.1 Konsistenzbedingungen und Änderungen weiterleiten

Das Weiterleiten von Konsistenzbedingungen und Änderungen zur Prüfung durch einen anderen Komponenten-Agenten ist bei verschiedenen Interaktionen erforderlich: Bei der Prüfung der Auswirkungen einer Änderung der Konfiguration einer Komponenteninstanz wird die Aktion agenten-intern durch den Änderungswächter aufgerufen. Im Rahmen der Prüfung einer Verbindung zwischen Komponenteninstanzen wird die Aktion durch einen Aspekt-Agenten in der Rolle des Verbindungsmanagers beauftragt. Im Weiterleitungsauftrag wird der betroffene Anschlusspunkt sowie der Auftraggeber der Weiterleitung mitgeteilt. Der Agent ermittelt – gegebenenfalls unter Berücksichtigung zuvor ermittelter Änderungen – die für den betreffenden Anschlusspunkt gültigen Verbindungsregeln. Die entsprechenden Konsistenzbedingungen und die zuvor ermittelten Änderungen werden zur Prüfung an den Empfänger-Agenten gesendet, ebenso die Information über den Auftraggeber. Abbildung 8.9 zeigt den Ablauf der Aktion.

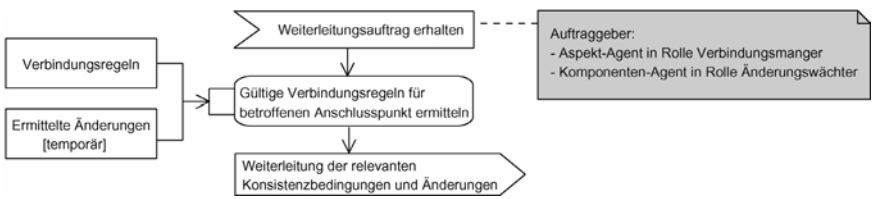


Abbildung 8.9: Aktion „Konsistenzbedingungen und Änderungen weiterleiten“

8.3.3.2 Externe Konsistenzbedingungen und Änderungen prüfen

Nachdem der Konfigurationsmanager eines Komponenten-Agenten die externen Konsistenzbedingungen und gegebenenfalls Änderungen vom Konfigurationsmanager eines anderen Komponenten-Agenten zur Prüfung erhalten hat, werden diese mit den lokalen Informationen und dem Wissen über die eigene Komponenteninstanz verknüpft, Wechselwirkungen geprüft, Inkonsistenzen identifiziert und – falls erforderlich – Änderungen der eigenen Konfiguration zur Sicherstellung der Konsistenz ermittelt. In der erhaltenen Nachricht ist auch die Information über den für die Prüfung relevanten Anschlusspunkt sowie über den Auftraggeber enthalten, an den die Prüfungsergebnisse weitergegeben werden sollen.

Zunächst werden aus der aktuellen Konfiguration der Komponenteninstanz die durch die erhaltenen, externen Konsistenzbedingungen betroffenen Eigenschaften ermittelt, verknüpft und Inkonsistenzen sowie die erforderlichen Änderungen ermittelt. Falls auch externe Änderungen übermittelt wurden, werden – unter Berücksichtigung der temporären internen Änderungen – alle gültigen Verbindungsregeln für den betroffenen Anschlusspunkt selektiert und die enthaltenen Konsistenzbedingungen mit den übermittelten Änderungen verknüpft. Für die dabei erkannten Inkonsistenzen werden ebenfalls die notwendigen Änderungen zur Sicherstellung der Konsistenz ermittelt. Alle ermittelten Änderungen werden temporär zwischengespeichert. Wurden im Rahmen der Prüfung keine Inkonsistenzen identifiziert und demnach auch keine Änderungen erforderlich, so kann bereits jetzt die Bestätigung der erfolgreichen Prüfung an den Auftraggeber gesendet werden. Andernfalls müssen die ermittelten Änderungen durch Anwendung der Parametrierregeln auf interne Wechselwirkungen überprüft werden. Werden dabei weitere Änderungen erforderlich, so muss nochmals geprüft werden, ob dadurch externe Konsistenzbedingungen verletzt werden oder zusätzlich eigene Verbindungsregeln Gültigkeit erlangen, die wiederum gegen die externen Änderungen zu prüfen sind. Die Prüfiteration endet, wenn alle erforderlichen Änderungen identifiziert wurden. Der Ablauf ist in Abbildung 8.10 dargestellt.

Werden in einem Prüfschritt Konflikte durch nicht erfüllbare Verbindungs- oder Parametrierregeln erkannt, so wird die Prüfung unterbrochen und eine entsprechende Konfliktmeldung (inklusive der Konfliktursache) an den Auftraggeber der Prüfung gesendet. Endet die Prüfiteration regulär, d.h. ohne erkannte Konflikte, so ist die Konsistenz der Komponenteninstanz bezüglich der übermittelten Konsistenzbedingungen und Änderungen durch die ermittelten internen Änderungen gewährleistet und es wird eine Bestätigung (mit ermittelten Änderungen) an den Auftraggeber gesendet. Die ermittelten internen Änderungen sind wiederum außerhalb der Komponenteninstanz auf Konsistenz zu überprüfen. Zur Ermittlung der Änderungsauswirkungen wird die entsprechende Aktion der Rolle des Änderungswächters angestoßen (siehe Kapitel 8.3.2.2).

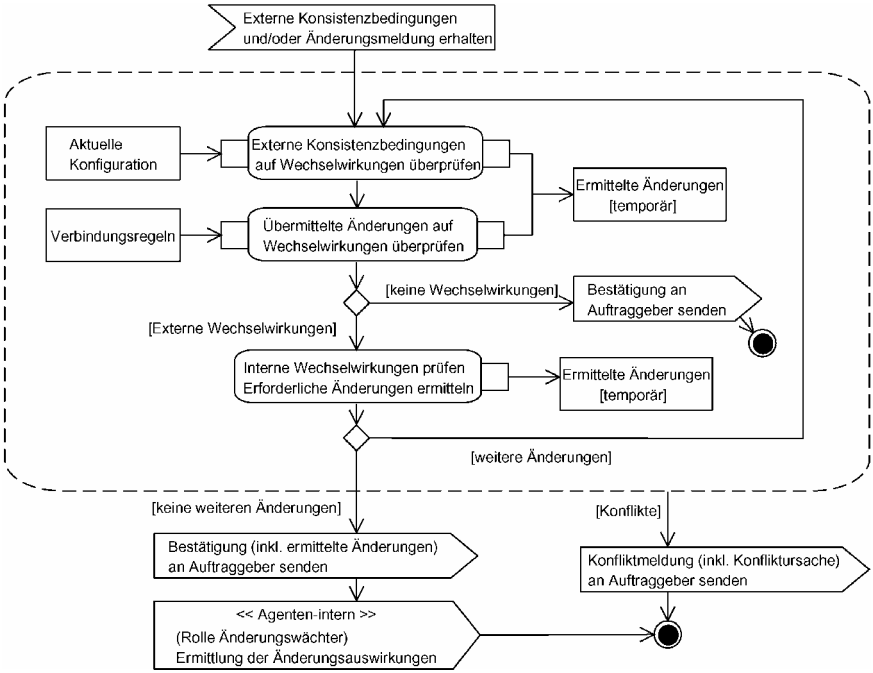


Abbildung 8.10: Aktion „Externe Konsistenzbedingungen und Änderungen prüfen“

8.3.3.3 Lokale Anpassungen umsetzen

Ebenso wie der Austausch und die Prüfung von Konsistenzbedingungen und Änderungen zwischen Komponenten-Agenten ist auch die Umsetzung von Anpassungen im Rahmen verschiedener Interaktionen möglich. Nach einer Verbindungsprüfung erfolgt der Anpassungsauftrag durch einen Aspekt-Agenten in der Rolle des Verbindungsmanagers, bei der Prüfung der Auswirkungen einer Änderung durch die Rolle des Änderungswächters eines Komponenten-Agenten. Entsprechend dem Anpassungsauftrag werden die zuvor ermittelten Änderungen in die Konfiguration der Komponenteninstanz übernommen oder verworfen sowie protokolliert. Abbildung 8.11 zeigt den Ablauf der Aktion.

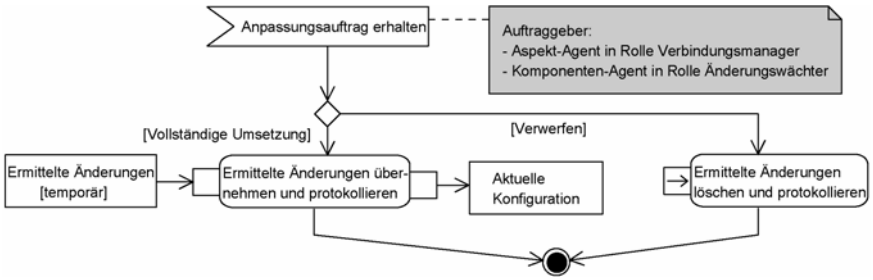


Abbildung 8.11: Aktion „Lokale Anpassungen umsetzen“

Zudem verfügt der Konfigurationsmanager noch über die Aktion *Verbindung erstellen oder löschen*, welche im Rahmen der Verbindungshandhabung benötigt wird (siehe Kapitel 8.4.2).

8.3.4 Aktionen der Rolle Baugruppenmanager

Aufgabe des Baugruppenmanagers ist die Handhabung und Anpassung ihrer internen Struktur. Bei der Anpassung von Eigenschaften einer Baugruppe wird die Aktion *Baugruppe intern anpassen* zur Auswahl und Einrichtung der passenden Varianten-Konfiguration ausgelöst. Mittels der spezifizierten Varianten-Konfigurationsregeln werden die Anpassungen der internen Struktur ermittelt und mit dem Ingenieur abgestimmt. Ihre Umsetzung erfolgt durch Instanziierung, Anpassung und Verbindung interner Komponenten. Bei der Verbindung von Baugruppen werden durch die beiden beteiligten Baugruppenmanager mittels der Aktion *Baugruppenverbindung umsetzen* die internen Komponenteninstanzen an den Eingangs- und Ausgangsanschlusspunkten der Baugruppen ermittelt und ihre Verbindung durch den Verbindungsmanager beauftragt.

8.4 Aktionen und Interaktionen von Aspekt-Agenten

8.4.1 Interaktion zur Verbindungsprüfung und -erstellung

Die Interaktion zur komponentenübergreifenden Verbindungsprüfung und -erstellung legt einen koordinierten Ablauf aller Agenten-Aktionen zur Verbindung zweier Komponenteninstanzen fest. Zunächst erfolgt die Schnittstellenprüfung für die betroffenen Anschlusspunkte der Komponenteninstanzen durch die Rolle des Schnittstellenprüfers des Bibliotheks-Agenten. Anschließend wird die wechselseitige Ermittlung und Prüfung von Wechselwirkungen zwischen den betroffenen Komponenteninstanzen eingeleitet. Dazu wird an den ersten Komponenten-Agenten der Auftrag zum Weiterleiten von Konsistenzbedingungen und Änderungen an den zweiten Komponenten-Agenten gesendet (vgl. Kapitel 8.3.3). Dessen Rückmeldung über das Prüfungsergebnis bestimmt den weiteren Verlauf der Verbindungsprüfung. Im Erfolgsfall erfolgt die umgekehrte Prüfung (gegebenenfalls unter Berücksichtigung bereits ermittelter Anpas-

sungen). Nach erfolgreich erfolgter wechselseitiger Prüfung werden die ermittelten Anpassungen zur Sicherstellung der Konsistenz zwischen den beiden Komponenteninstanzen dem Ingenieur präsentiert und mit ihm abgestimmt. Anschließend wird an die beteiligten Komponenten-Agenten der entsprechende Umsetzungsauftrag gesendet. Wurden für die Verbindung keine erforderlichen Änderungen ermittelt, kann dies ohne weitere Rücksprache mit dem Ingenieur erfolgen. Eine erfolgreich erstellte Verbindung wird durch die Komponenten-Agenten der betroffenen Komponenteninstanzen gespeichert. Abbildung 8.12 zeigt den Ablauf der Interaktion und die Aktionen der beteiligten Agenten.

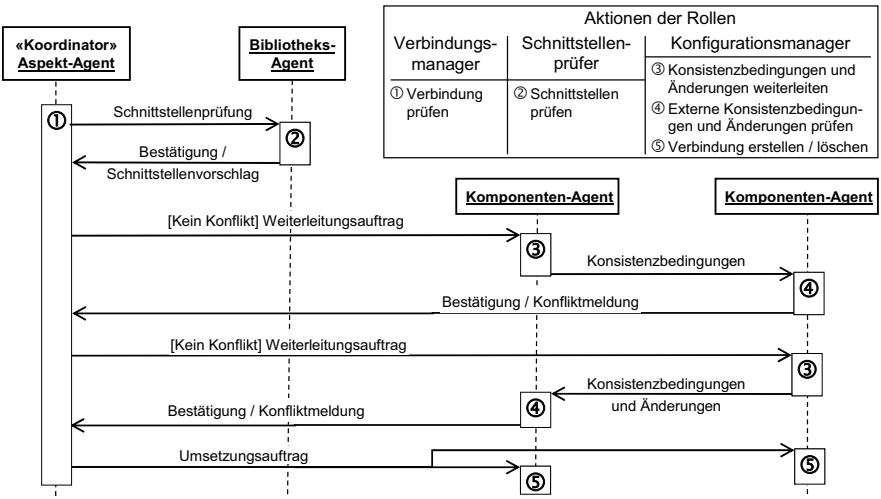


Abbildung 8.12: Interaktion zur Verbindungsprüfung und -erstellung

8.4.2 Aktionen der Rolle Verbindungsmanager

Aufgabe des Verbindungsmanagers ist das Verbinden von Komponenteninstanzen und die Sicherstellung ihrer Kombinierbarkeit durch Koordination der Verbindungsprüfung.

8.4.2.1 Verbindung prüfen

Auslöser der Aktion ist das Verbinden zweier Komponenteninstanzen durch den Ingenieur, das durch den Verbindungsmanager erkannt und als Auftrag wahrgenommen wird. Die Aktion kann auch vom Baugruppenmanager zur Verbindung der Komponenteninstanzen an den Baugruppen-Anschlusspunkten beauftragt werden. Der Verbindungsmanager koordiniert die Verbindungsprüfung durch die betroffenen Komponenten-Agenten und überwacht den Verbindungsstatus, bis die Verbindung erstellt oder verworfen wird. Abbildung 8.13 zeigt den Ablauf der Aktion. Zu Beginn wird die Schnittstellenprüfung beauftragt. Passen die Schnittstellen nicht zusammen,

wird der Konflikt zusammen mit einem Schnittstellenvorschlag an den Ingenieur gemeldet. Anschließend wird die wechselseitige Prüfung durch die Komponenten-Agenten veranlasst, ihre Rückmeldungen erfasst und dabei der Verbindungsstatus überwacht. Alle nach der vollständigen Prüfung ermittelten Anpassungen zur Sicherstellung der Konsistenz werden vom Verbindungsmanager dem Ingenieur präsentiert und mit ihm abgestimmt. Anschließend werden die beteiligten Komponenten-Agenten mit der Erstellung der Verbindung beauftragt oder über das Verwerfen der Anpassungen informiert.

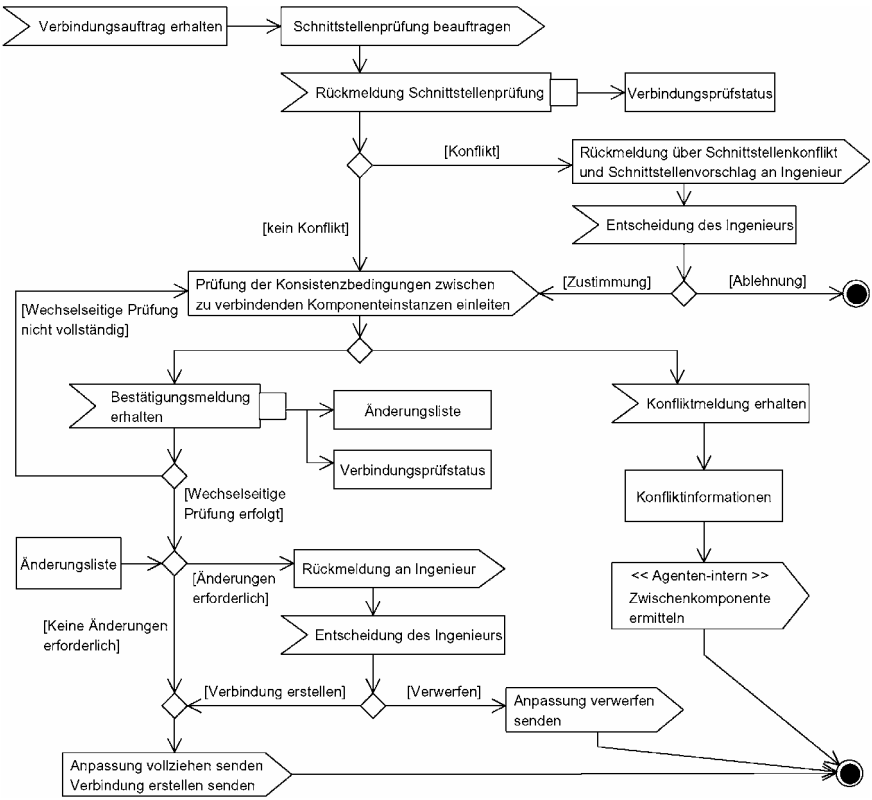


Abbildung 8.13: Aktion „Verbindung prüfen“

Im Konfliktfall durch nicht erfüllbare Verbindungs- oder Parametrierregeln zwischen den Komponenteninstanzen versucht der Verbindungsmanager, für die ermittelten Inkonsistenzen einen Lösungsvorschlag in Form einer Zwischenkomponente zu finden, welche eine indirekte Verbindung unter Wahrung der komponentenübergreifenden Konsistenz ermöglicht. Dies erfolgt durch die Aktion *Zwischenkomponente ermitteln.*

8.4.2.2 Zwischenkomponente ermitteln

Die Aktion zur Ermittlung einer Zwischenkomponente, welche eine Inkonsistenz zwischen zwei zu verbindenden Komponenteninstanzen kompensiert, besteht aus drei wesentlichen Schritten:

- *Passende Kandidaten finden*: Geeignete Zwischenkomponenten-Kandidaten werden durch die Interaktion *Zwischenkomponente anfordern* mit die Rolle Komponentensucher des Bibliotheks-Agenten ermittelt. Die Suchanforderung enthält die benötigten Schnittstellentypen sowie die Konfliktinformationen, sodass durch den Bibliotheksagenten eine Vorauswahl an passenden Kandidaten erstellt werden kann. Kandidaten sind entweder einzelne Komponenten oder Baugruppen, welche die passenden Schnittstellen aufweisen und die Verbindungsregeln, welche den Konflikt verursachen, erfüllen können.

- *Verbindung prüfen und erstellen*: Die Verbindbarkeit des Zwischenkomponentenkandidaten zu den jeweiligen Komponenteninstanzen wird mittels der Aktion *Verbindung prüfen* schrittweise überprüft (siehe Kapitel 8.4.2.1). Treten hierbei Konflikte auf, wird der Kandidat verworfen und temporäre Änderungen der Komponenteninstanzen werden zurückgesetzt.

- *Abstimmung und Umsetzung*: Mögliche Zwischenkomponenten sowie die jeweils notwendigen Anpassungen der beiden Komponenteninstanzen werden dem Ingenieur als Lösungsvorschlag übermittelt. Entsprechend der Wahl des Ingenieurs werden die Verbindungen erstellt.

Der detaillierte Ablauf dieser Aktionen wird in [Xu06] erläutert.

8.4.2.3 Verbindung entfernen

Das Entfernen einer Verbindung bedarf keiner weiteren Prüfung. Die Komponenten-Agenten der betroffenen Komponenteninstanzen werden beauftragt, die Verbindung zu löschen.

8.4.3 Aktionen der Rolle Instanzenselektierer

Für die Erstellung von gewerkspezifischen Teilmodellen oder für die fachspezifische Optimierung des Anlagenmodells müssen die existierenden Komponenteninstanzen, welche speziell für einen Aspekt von Bedeutung sind, erfasst und auf Basis ihrer Informationen ermittelt und selektiert werden. Diese Aufgabe wird durch die Rolle des Instanzenselektierers übernommen. Zur Erfüllung eines Auswahlauftrags wird die Aktion *Komponenteninstanzen auswählen* ausgeführt. Dabei werden zunächst die im Anlagenmodell existierenden Komponenten-Agenten ermittelt. Anschließend werden mittels der Interaktion *Informationsabfrage* die Informationen über die Eigenschaften der jeweiligen Komponenteninstanz erfasst, durch Anwendung fachspezifischer Auswahlregeln ausgewertet und die Auswahlliste der relevanten Komponenteninstanzen erstellt. Zudem kann die Auswahlliste unter Anwendung von Gruppierungs- oder Sortierregeln nach fachspezifischen Maßstäben geordnet werden Abbildung 8.14 zeigt den Ablauf der Aktion.

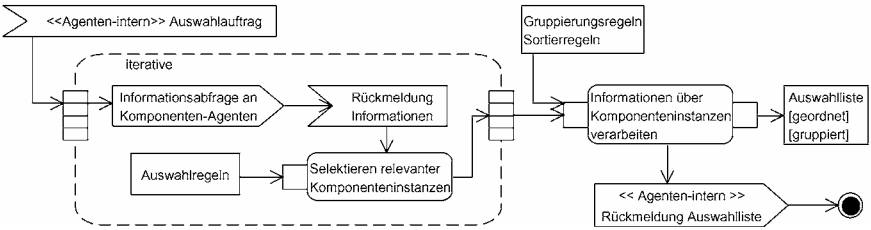


Abbildung 8.14: Aktion „Komponenteninstanzen auswählen“

8.4.4 Aktionen der Rolle Modellierer

Die Erstellung eines gewerkspezifischen Teilmodells aus den Informationen des Anlagenmodells oder die fachspezifische Anpassung von Komponenteninstanzen erfolgt durch die Rolle des Modellierers. Die wichtigste für den Benutzer sichtbare Aktion eines Aspekt-Agenten ist *Teilmodell bilden*. Nachdem diese durch den Ingenieur beauftragt wurde, erfolgt zunächst die Auswahl der relevanten Komponenteninstanzen des Anlagenmodells durch die Rolle Instanzenselektierer. Anschließend werden unter Anwendung der fachspezifischen Modellierungsregeln und der projektunabhängigen Gewerkinformationen die Modellinformationen des Teilmodells erstellt (siehe Abbildung 8.15). Aufgrund der Heterogenität der verschiedenen fachlichen Aspekte kann dabei der zusätzliche Einsatz von fachspezifischen Modellierungsalgorithmen erforderlich sein. Anschließend kann das Teilmodell vom Ingenieur überprüft und manuell nachbearbeitet oder ergänzt werden.

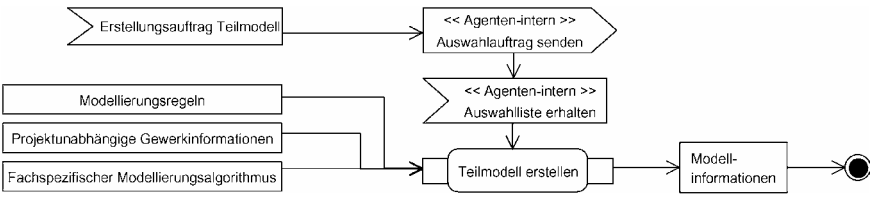


Abbildung 8.15: Aktion „Teilmodell bilden“

Die Aktion *Teilmodell prüfen* ist nach manuellen Änderungen der Modellinformationen eines Teilmodells erforderlich oder bei Änderungen im Anlagenmodell, die das Teilmodell beeinflussen. Die Änderungen werden durch Anwendung von gewerkspezifischen Modellierungsregeln, Informationen und Algorithmen überprüft, Inkonsistenzen ermittelt und mit dem Ingenieur abgestimmt. Anschließend erfolgt ihre Umsetzung.

Die Aktion *Komponenteninstanzen anpassen* wird bei der fachspezifischen Optimierung des Anlagenmodells benötigt. Die Konfiguration zuvor ausgewählter Komponenteninstanzen wird unter Anwendung fachspezifischer Optimierungsregeln durch Interaktion mit den Komponen-

ten-Agenten angepasst. Ebenso kann die Instanziierung zusätzlicher Komponenten und ihre Verbindung durch Beauftragung des Verbindungsmanagers erfolgen.

8.4.5 Aktionen der Rolle Teilmodellmanager

Die Rolle des Teilmodellmanagers ist für die Sicherstellung der Konsistenz eines Teilmodells verantwortlich. Hierfür ist es zunächst wichtig, nach der Bildung des Teilmodells die betroffenen Komponenten-Agenten über die entstandenen Abhängigkeiten zu informieren. Hierfür wird das Entwurfsmuster „Beobachter“ eingesetzt [GHJV95]: Durch die Aktion *Abhängigkeiten verwalten* und die Interaktion *Beobachteranmeldung* wird die Abhängigkeit in das Umgebungsmodell der betroffenen Komponenten-Agenten eingetragen, sodass im Falle einer Änderung an ihren Konfigurationen eine Benachrichtigung des Aspekt-Agenten erfolgt. In gleicher Weise erfolgt die Beobachteranmeldung beim Bibliotheks-Agenten, um im Falle der Instanziierung neuer Komponenten im Anlagenmodell informiert zu werden.

Der Teilmodellmanager erhält eine Änderungsbenachrichtigung von Komponenten-Agenten im Rahmen ihrer Interaktion zur Handhabung komponentenübergreifender Wechselwirkungen (siehe Kapitel 8.3.1 und 8.3.2.2). Ebenso werden manuelle Änderungen eines Teilmodells durch den Teilmodellmanager erfasst. Er führt die Aktion *Teilmodell aktualisieren* aus und veranlasst die Teilmodellprüfung. Entstehen dabei neue Abhängigkeiten zu Komponenteninstanzen oder werden Abhängigkeiten obsolet, wird dies den betroffenen Komponenten-Agenten mitgeteilt.

8.5 Aktionen und Interaktionen des Bibliotheks-Agenten

Die Unterstützung der Auswahl von Komponenten erfolgt durch den Bibliotheks-Agenten. In der Rolle des *Komponentenverwalters* werden durch die Aktion *Suchinformationen initialisieren* alle Komponenteninformationen aus der Komponentenbibliothek erfasst, für die rechnergestützte Suche aufbereitet sowie durch die Aktion *Suchinformationen aktualisieren* bei Ergänzungen der Komponentenbibliothek nachgeführt. In der Rolle des *Komponentensuchers* werden durch die Aktion *Komponente finden* ein Suchauftrag des Ingenieurs und die vorgegebenen Auswahlkriterien erfasst, übereinstimmende Komponenten oder Teillösungen ermittelt und dem Ingenieur zu Auswahl präsentiert. Nach erfolgter Auswahl folgt *Komponente instanziieren.* Suchaufträge zum Finden einer Komponente oder Teillösung werden auch durch die Rolle des Verbindungsmanagers mittels der Interaktion *Zwischenkomponente anfordern* erteilt, um bei einem Verbindungskonflikt zwischen zwei Komponenteninstanzen geeignete Zwischenkomponenten zu finden. Dabei werden auch die den Verbindungskonflikt verursachenden Konsistenzbedingungen übermittelt und in die Komponentensuche mit einbezogen, sodass durch den Bibliotheksagenten eine Vorauswahl an passenden Kandidaten zur Auflösung des Verbindungskonflikts ermittelt werden kann. Im Rahmen der Interaktion zur komponentenübergreifenden Verbindungsprüfung und -erstellung wird der Bibliotheks-Agent in der Rolle des Schnittstellenprü-

fers mit der Prüfung der Kombinierbarkeit zweier Komponenteninstanzen auf Schnittstellenebene beauftragt. Er führt die Aktion *Schnittstellen prüfen* aus, ermittelt Schnittstellenkonflikte und schlägt im Konfliktfall passende Schnittstellen zur Verbindung vor.

8.6 Spezifische Merkmale der Interaktionen der Agenten

Erfüllung der Randbedingung der Vollständigkeit

In den Kapiteln 8.3.1 und 8.4.1 wurde beschrieben, wie die Interaktionen zur Ermittlung komponentenübergreifender Wechselwirkungen und Änderungen in koordinierter Weise ablaufen: Bei der Ermittlung der Auswirkungen einer Änderung fungiert der auslösende Änderungswächter als Koordinator der Interaktionen, bei der Verbindungsprüfung der Verbindungsmanager. Er wird als Auftraggeber in allen Nachrichten, die im Rahmen der Interaktion von Agenten versendet werden, mitgeschickt (Nachrichtenattribut „Reply-to“). Durch die Rückmeldung aller Abhängigkeiten an den Koordinator kann die vollständige Erfassung und Koordination *aller* betroffenen Komponenten-Agenten erfolgen. Ebenso werden alle Ergebnisse der Prüfungen der jeweiligen Komponenten-Agenten an den Koordinator zurückgesendet. Ermittelte Anpassungen werden von den betroffenen Komponenten-Agenten so lange temporär vorgehalten, bis der Umsetzungsauftrag durch den Koordinator erfolgt. Auf diese Weise erfolgt der koordinierte Ablauf der Änderungsweitergabe, die vollständige Erfassung der betroffenen Komponenten-Agenten und ihrer Prüfungsergebnisse, die gesammelte Rückmeldung an den Ingenieur und die vollständige Umsetzung bzw. das Verwerfen aller erforderlichen Änderungen.

Durch das Koordinator-Prinzip ist auch die Einschränkung des Auswirkungsbereichs der Wechselwirkungsprüfung möglich, um die Komplexität der Zusammenhänge für den Ingenieur überschaubar zu halten. Dazu wird bei der Rückmeldung der potenziell betroffenen Agenten an den Koordinator ein Handshake-Mechanismus eingeführt: Der meldende Komponenten-Agent erwartet vor der Weiterleitung seiner Änderungen und Konsistenzbedingungen eine Bestätigung des Koordinators (in Abbildung 8.6 nicht dargestellt). Der Koordinator kann aufgrund der bisher erfolgten Rückmeldungen entscheiden, ob die Ermittlung der Änderungsauswirkungen an weitergehende Komponenten-Agenten erfolgen soll oder ob die gewünschte Reichweite bereits erreicht ist. Die maximale Reichweite kann für die verschiedenen Agenten-Interaktionen konfiguriert werden. So ist es beispielsweise sinnvoll, bei einer Verbindungsprüfung zunächst nur die direkten Wechselwirkungen zwischen den zu verbindenden Komponenteninstanzen zu betrachten, bevor weitergehende, indirekte Wechselwirkungen ermittelt werden.

Erfüllung der Randbedingung der Endlichkeit

Zusätzlich zum eingeführten Koordinator-Prinzip wird für jede ausgelöste Interaktion vom Koordinator eine eindeutige Konversations-ID vergeben, welche ebenfalls als Nachrichtenattribut in allen Nachrichten im Rahmen der Interaktion mitgesendet wird. Dadurch wird für jeden

Softwareagenten eine eindeutige Zuordnung erhaltener Nachrichten zu einer Interaktion möglich, und undefinierte Zustände des gesamten Agentensystems werden vermieden. Durch die Vorgabe einer maximalen Anzahl erlaubter temporärer Änderungen für die Eigenschaften von Komponenteninstanzen im Zuge einer einzelnen Interaktion können Endlosschleifen erkannt und als Konflikt an den Koordinator zurückgemeldet werden. Da der Konflikt nicht durch das Agentensystem behoben werden kann, ist dann ein manuelles Eingreifen des Ingenieurs erforderlich.

8.7 Interaktionen der Agenten mit dem Ingenieur

Ein wichtiges Ziel dieser Arbeit ist es, den Ingenieur bei der Komponentenhandhabung konstruktiv zu unterstützen und ihm Detailtätigkeiten abzunehmen, ohne seine Flexibilität einzuschränken. In diesem Kapitel wurde deutlich, dass die Softwareagenten nur dann mit dem Ingenieur in Interaktion treten, wenn sie ein tatsächliches Problem erkennen. Gleichzeitig unterbreiten sie ihm einen Lösungsvorschlag mit möglichen Anpassungen. Bei allen Agentenaktionen und -interaktionen sind Wechselwirkungsprüfung und Änderungsermittlung strikt von der Änderungsausführung auf Komponentenebene getrennt. So hat der Ingenieur die vollständige Kontrolle über die Handlungen der Agenten und entscheidet, ob und welche der vorgeschlagenen Anpassungen ausgeführt werden sollen. Er kann jederzeit manuelle Korrekturen vornehmen. Wenn nur Anpassungen in einem eingeschränkten Bereich des Anlagenmodells durchzuführen sind, kann er zudem eine erneute Prüfung weitergehender Wechselwirkungen veranlassen.

Die in diesem Kapitel vorgestellten Aktionen und Interaktionen der Softwareagenten ermöglichen die systematische Ermittlung und Handhabung von Komponenten-Wechselwirkungen. Der Ingenieur gibt durch seine Tätigkeiten den Handlungsrahmen für die Softwareagenten vor. Diese erkennen durch die situationsspezifische Verknüpfung von komponentenspezifischem und komponentenübergreifendem Wissen in Zusammenarbeit die Zusammenhänge und Abhängigkeiten im Anlagenmodell und unterbreiten dem Ingenieur konstruktive Lösungsvorschläge zu ihrer konsistenten Handhabung. Der koordinierte Ablauf der Agenten-Aktionen und Interaktionen gewährleistet die Nachvollziehbarkeit, die Vollständigkeit und die Endlichkeit der Unterstützungshandlungen. Zudem behält der Ingenieur stets die Kontrolle über die Umsetzung vorgeschlagener Lösungen und hat die Möglichkeit, manuelle Korrekturen vorzunehmen.

Um die Wirksamkeit der entwickelten Konzepte nachzuweisen, wird in den folgenden beiden Kapiteln die Realisierung des agentenorientierten Unterstützungssystems vorgestellt und seine Anwendung in verschiedenen typischen Szenarien beim Engineering betrachtet.

9 Realisierung des Unterstützungskonzepts

Nachdem in den vorangegangenen Kapiteln die verschiedenen Bestandteile des Konzepts des agentenunterstützten Engineerings erarbeitet wurden, steht in diesem Kapitel ihre Realisierung in einem Werkzeugsystem im Mittelpunkt der Betrachtungen. Dazu wird zunächst ein exemplarisches Komponentenmodell zur Beschreibung von Komponenteninformationen und -wissen und eine darauf aufbauende Komponentenbibliothek zur Ablage von Komponentenbeschreibungen präsentiert. Danach werden ein CAE-Werkzeug zur grafischen, komponentenbasierten Erstellung von Anlagenmodellen und das agentenorientierte Unterstützungssystem vorgestellt. Abschließend wird die Integration der beiden Werkzeuge erläutert.

9.1 Überblick über das Werkzeugsystem

Abbildung 9.1 zeigt die Gesamtarchitektur des Werkzeugsystems. CAE-Werkzeug und agentenorientiertes Unterstützungssystem werden als eigenständige Werkzeuge realisiert, um ihre unabhängige Verwendbarkeit zu gewährleisten. Beide greifen auf die Informationen der Komponentenbeschreibungen in einer Komponentenbibliothek zurück. Der Ingenieur arbeitet ausschließlich mit dem CAE-Werkzeug, das über eine Systemschnittstelle mit dem Unterstützungssystem verbunden ist, um die für die Unterstützung erforderlichen Informationen auszutauschen.

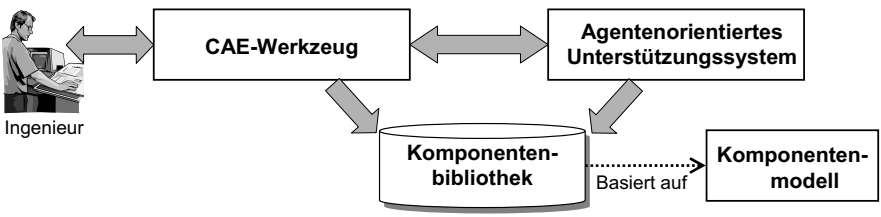


Abbildung 9.1: Gesamtarchitektur des Werkzeugsystems

9.2 Realisierung des exemplarischen Komponentenmodells

Für die praktische Anwendung des in Kapitel 7 vorgestellten Beschreibungsmodells für die integrierte Beschreibung von Informationen und Wissen über technische Komponenten wurde ein konkretes Komponentenmodell als Beschreibungssprache entwickelt, das die Vorgaben des Beschreibungsmodells umsetzt. Es wird als *Component Model Interchange* (CMI) bezeichnet. Als Beschreibungsmittel für das Komponentenmodell wurde XML (eXtensible Markup Language [W3C04]) gewählt, die sich durch Systemunabhängigkeit und einfache rechnergestützte Verarbeitbarkeit auszeichnet. Abbildung 9.2 zeigt einen Auszug aus der Komponentenbeschreibung

für die Komponente „Förderband“ im CMI-Format. In der ersten Zeile wird die Datei als XML-Datei identifiziert, und in der zweiten Zeile wird spezifiziert, dass die Komponentenbeschreibung der Komponente „Förderband“ dem CMI-Format entspricht. In Zeile 3-10 werden die Eigenschaften „Fördertyp“, „Materialfluss“ und „Signal-belegt“ der Komponente definiert. In Zeile 12-17 wird der Anschlusspunkt „Materialzufluss“ mit der Eigenschaft „Transportrichtung“ und der Verbindungsregel „Steuerung Materialfluss“ spezifiziert.

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <cmi:Component componentName="Foerderband">
3.      <Feature type="cmi:Property" name="Foerdertyp" type="String" value=„Stetigförderer"/>
4.      <Feature type="cmi:Parameter" name="Materialfluss" type="String" defaultValue="permanent">
5.          <ValueSet>
6.              <value> permanent </value>
7.              <value> manuell </value>
8.          </ValueSet>
9.      </Feature>
10.     <Feature type="cmi:Option" name="Signal-belegt"/>
11.     <Port type="cmi:PortIn" id="1" name="Materialzufluss" interface_type="Materialfluss">
12.         <Feature type="cmi:Parameter" name="Transportrichtung" type="String"  defaultValue="horizontal“ />
13.         <Rule name="Steuerung Materialfluss">
14.             <PreCondition factName="Materialfluss" relation="EQUALS" matchExp="permanent"/>
15.             <ExternalRequirement type="immediate" factName="Signal-belegt" matchExp="true"/>
16.         </Rule>
17.     </Port>
18. </cmi:Component>
```

Abbildung 9.2: Auszug aus einer Komponentenbeschreibung

Um die Anwendung des Komponentenmodells zu unterstützen, wurde ein grafischer Editor für die Erstellung von Komponentenbeschreibungen und Ablage in einer Komponentenbibliothek zur Bereitstellung für das Engineering realisiert.

9.3 Realisierung eines CAE-Werkzeugs

Zur Verwendung der Komponentenbeschreibungen im CMI-Format beim Engineering wurde ein CAE-Werkzeug entwickelt, das die komponentenbasierte Erstellung von Anlagenmodellen ermöglicht und die Tätigkeiten des Ingenieurs durch grafische Funktionen unterstützt [Krat05]. Das CAE-Werkzeug wurde mithilfe der Technologien Eclipse, Eclipse Modeling Framework (EMF) und Graphical Editor Framework (GEF) entwickelt, die eine umfassende Unterstützung bei der Entwicklung von Modellierungswerkzeugen bieten [GaBe03], [BSM+03], [MDG+04].

Abbildung 9.3 zeigt die grafische Benutzungsoberfläche des CAE-Werkzeugs, in der die Komponentenbibliothek (1) und das Anlagenmodell (2) dargestellt werden. Das Instanziieren von Komponenten erfolgt durch Ziehen und Ablegen (Drag’n’Drop), das Verbinden von Komponenteninstanzen durch Ziehen einer Verbindungslinie zwischen ihren Anschlusspunkten. Die Informationen der Komponenteninstanzen des Anlagenmodells werden durch Eigenschaftstabellen und Dialogfenster dargestellt (3) und dort vom Benutzer editiert.

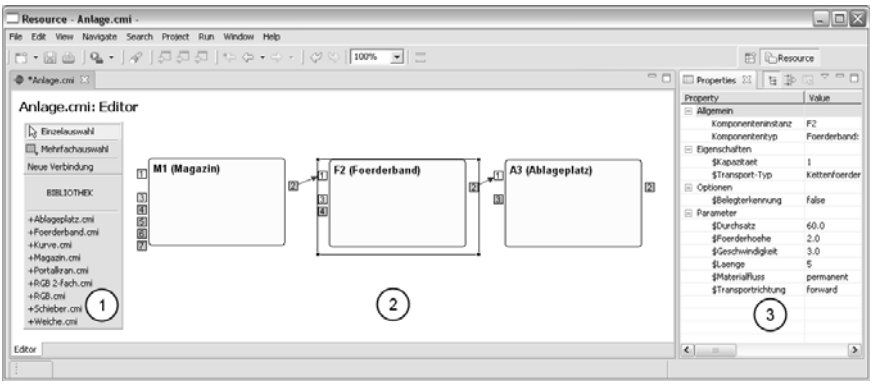


Abbildung 9.3: Benutzungsoberfläche des CAE-Werkzeugs

9.4 Realisierung des Agentensystems

Das agentenorientierte Unterstützungssystem wurde auf Basis der Entwicklungs- und Laufzeitumgebung JADE in der Programmiersprache Java realisiert [Arm05], [Yilm05]. Durch Verwendung von JADE können die verfügbaren Basisfunktionen für den Agentensystemaufbau umfassend genutzt werden, sodass sich die Implementierung auf die konzipierten Agententypen, ihre Aktionen und Interaktionen beschränkt. Folgende Agententypen wurden implementiert:

- Komponenten-Agent mit den Rollen Konfigurationsmanager und Änderungswächter
- Bibliotheks-Agent mit den Rollen Komponentenverwalter, -sucher und Schnittstellenprüfer
- Aspekt-Agent Anlagenmodell mit der Rolle Verbindungsmanager
- Aspekt-Agent Teilmodell Netzwerkplan mit den Rollen Instanzenselektierer, Modellierer und Teilmodellmanager

Jeder Agent verfügt über eine Wissensbasis zu Speicherung seines lokalen Wissens. Die Wissensbasis wird durch eine Wissensverarbeitungskomponente gekapselt, auf die er bei der Durchführung seiner jeweiligen Aktionen und Interaktionen zugreift [WaGö05].

Durch die vorhandenen Laufzeitkomponenten von JADE für Kommunikation, Agentenverwaltung und Diensteverwaltung werden alle Managementaufgaben des Agentensystems vollständig abgedeckt. Auch die in Kapitel 8.6 diskutierten Mechanismen zur Gewährleistung von Vollständigkeit und Endlichkeit werden von JADE bereits zur Verfügung gestellt [BCTR04]. Zudem verfügt JADE über spezielle Managementwerkzeuge (1), mit denen die Softwareagenten zur Laufzeit verwaltet, beobachtet und überwacht werden können (Abbildung 9.4 rechts).

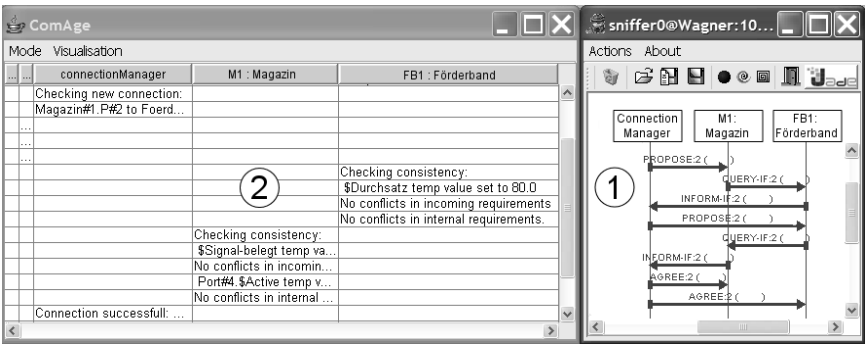


Abbildung 9.4: Management- und Beobachtungswerkzeuge des Agentensystems

Zusätzlich wurde eine Benutzungsoberfläche (2) entwickelt, in der der genaue Ablauf der Agentenaktionen protokolliert wird (Abbildung 9.4 links). Über die Benutzungsoberfläche können auch einzelne Basisaktivitäten des Ingenieurs bzw. ganze Aktivitätssequenzen simuliert und das Agentenverhalten getestet werden. Die genannten Werkzeuge sind jedoch nur für die Beobachtung und die Analyse des Verhaltens der Softwareagenten interessant und bleiben dem Ingenieur als Nutzer des Unterstützungssystems verborgen. Er wird mit dem agentenorientierten Unterstützungssystem nur mittelbar über das CAE-Werkzeug interagieren.

9.5 Integration des Agentensystems

Das agentenorientierte Unterstützungssystem wird als ein paralleles aktives Teilsystem zum verwendeten CAE-Werkzeug eingesetzt. Aus Sicht des Ingenieurs bleibt die gewohnte Arbeitsweise über die Benutzungsoberfläche des CAE-Werkzeugs bestehen. Abbildung 9.5 zeigt die Systeminteraktion zwischen CAE-Werkzeug und Unterstützungssystem: Die Softwareagenten greifen über eine externe Systemschnittstelle auf das CAE-Werkzeug zu, um Engineeringinformationen und die Aktivitäten des Ingenieurs zu erfassen. Daraus leiten sie ihre Unterstützungshandlungen ab und ermitteln situationsspezifisch die Wechselwirkungen im Anlagenmodell. Ebenso interagieren sie über die externe Systemschnittstelle und die CAE-Benutzungsoberfläche mit dem Ingenieur, um auf erkannte Konflikte und mögliche Lösungen hinzuweisen sowie diese durch Anpassung von Engineeringinformationen auszuführen. Manuelle Eingriffe des Ingenieurs werden von den Agenten fortlaufend erfasst und entsprechend berücksichtigt. Die Unterstützungsfunktionalität wird vom Ingenieur als "aktive Konstruktionshilfe" wahrgenommen. Die Systemschnittstelle zwischen dem CAE-Werkzeug und dem agentenorientierten Unterstützungssystem erfüllt somit folgende Aufgaben der Systeminteraktion:

- Übermittlung von Engineeringinformationen und Benutzeraktivitäten
- Anpassung von Engineeringinformationen durch die Softwareagenten
- Interaktion der Softwareagenten mit dem Ingenieur

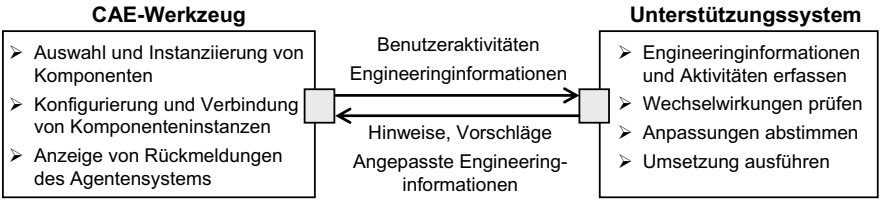


Abbildung 9.5: Systeminteraktion zwischen CAE-Werkzeug und Unterstützungssystem

Heutige CAE-Werkzeuge weisen entsprechende externe Systemschnittstellen auf, welche diese Systeminteraktionen ermöglichen [Denc03]. Auch für das in Abschnitt 9.3 präsentierte CAE-Werkzeug sowie für das Unterstützungssystem wurde eine entsprechende Schnittstelle realisiert [Kunz06]. Durch die Systeminteraktion über eine externe Systemschnittstelle werden eine einfache Integration und eine flexible Verwendung und des agentenorientierten Unterstützungssystems möglich. Es kann vom Ingenieur jederzeit zum CAE-Werkzeug zu- oder abgeschaltet werden. Zudem können beide Systeme unabhängig voneinander weiterentwickelt werden.

Durch die Unabhängigkeit des Unterstützungssystems kann es unverändert wiederverwendet und mit anderen CAE-Werkzeugen integriert werden. Dazu muss lediglich die entsprechende Systemschnittstelle aufseiten des Agentensystems ergänzt werden. Das Unterstützungssystem kann auch als eigenständiges Werkzeug benutzt werden. Hierzu verfügt es über eine kommandozeilenbasierte Benutzungsschnittstelle sowie über eine XML-basierte Import-/Exportschnittstelle für Anlagenmodelle [Wang06].

In diesem Kapitel wurde ein Werkzeugsystem vorgestellt, das dem Ingenieur die praktische Anwendung des Konzepts des agentenunterstützten Engineerings ermöglicht. Der Ingenieur muss sich weder mit der Benutzung des agentenorientierten Unterstützungssystems noch mit seinen internen Abläufen auseinander setzen. Er arbeitet normalerweise ausschließlich mit der grafischen Benutzungsoberfläche des CAE-Werkzeugs und erhält dabei situationsspezifisch die Hinweise und Lösungsvorschläge des Unterstützungssystems. Im nächsten Kapitel wird an einem Fallbeispiel aus der Automatisierungstechnik der praktische Einsatz des agentenunterstützten Engineerings gezeigt. Dazu wird der Anwendungsbereich der Fördertechnikanlagen gewählt, der sich sowohl durch domänentypische Komponenten als auch durch vielfältige technische Abhängigkeiten auszeichnet.

10 Anwendungsbeispiel für die Unterstützung

Um das vorgestellte Konzept des agentenunterstützten Engineerings an einem realen Anwendungsbeispiel der Automatisierungstechnik zu erproben, wurde die Domäne der Fördertechnikanlagen gewählt. Zunächst werden Komponentenbeschreibungen für Fördertechnikkomponenten bereitgestellt. Anschließend wird ein komponentenbasiertes Anlagenmodell für die Fördertechnikanlage „3-Achs-Portal“ erstellt, welche am Institut für Automatisierungs- und Softwaretechnik (IAS) als Demonstrationsmodell zur Verfügung steht. Dabei wird die Wirkungsweise des agentenorientierten Unterstützungssystems untersucht und bewertet.

10.1 Anwendungsbereich Fördertechnikanlagen

Fördertechnikanlagen beinhalten alle operativen Prozesse, bei denen Rohstoffe und Produkte gefördert oder als Stückgut umgeschlagen werden [HoLi06]. Sie werden aus einzelnen fördertechnischen Komponenten aufgebaut, z.B. Förderband, Kurvenförderer, Verteiler, Weiche, Drehtisch, Schieber, Ablageplatz, Magazin, Regalbediengerät, Portalkran und Hubstation.

Komponentenbeschreibungen für Fördertechnikkomponenten

Die Domäne Fördertechnikanlagen wurde in Industriekooperationen am praktischen Beispiel von Gepäckfördersystemen und Kommissioniersystemen untersucht. Es wurde festgestellt, dass das erforderliche Wissen über technische Abhängigkeiten und über gewerkspezifische Zusammenhänge ermittelt und formalisiert verfügbar gemacht werden kann. Durch Interviews mit Experten und durch Analyse von realen fördertechnischen Anlagen sowie von drei Demonstrationsanlagen am IAS wurde das Wissen über technische Abhängigkeiten von Fördertechnikkomponenten identifiziert, strukturiert und modelliert [Muba04] [Denc05], [Hild05], [Dris05]. Tabelle 10.1 zeigt für typische Komponenten die Anzahl der spezifizierten Eigenschaften, Anschlusspunkte, Verbindungs- und Parametrierregeln.

Tabelle 10.1: Beispiele für identifizierte Informationen und Wissen über Komponenten

		Förderband	**Magazin**	**Portalkran**	**Ablageplatz**
Anschlüsse	- Typ Materialfluss	2	2	5	2
	- Typ Signal	2	5	11	1
Eigenschaften	- komponentenintern	9	8	7	5
	- schnittstellenbezogen	2	4	5	4
Technische	- Verbindungsregeln	5	4	8	2
Abhängigkeiten	- Parametrierregeln	3	8	-	3

Einige Beispiele für technische Abhängigkeiten, die als Verbindungsregeln spezifiziert wurden:

- Förderband: Bei horizontalem Weitertransport wird ein permanenter Materialfluss gefordert.
- Magazin, Förderband, Ablage: Die Übereinstimmung der Förderhöhe muss gegeben sein.
- Förderband: Bei permanentem Materialfluss muss ein Übergabesignal erfolgen.
- Magazin: Ein permanenter Materialabtransport muss gewährleistet sein.
- Portalkran: Vor der Aufnahme muss der Stillstand des Transportguts gewährleistet sein.

Zur Bereitstellung von Informationen und Wissen wurden auf der Basis der gewonnenen Erkenntnisse Komponentenbeschreibungen erstellt und in der Komponentenbibliothek abgelegt.

10.2 Aufbau der Beispielanlage 3-Achs-Portal

Das 3-Achs-Portal ist eine stationäre Transporteinrichtung zur Weitergabe von Werkstücken aus einer Speichereinrichtung an eine Sortier- und Zwischenlagereinrichtung, wie sie z. B. in stark automatisierten Fertigungsbetrieben zum Einsatz kommt. Abbildung 10.1 zeigt die Gesamtansicht des Demonstrationsmodells. Das Modell besteht aus einem Werkstückmagazin (1), einem Förderband (2), Ablageplätzen (3), (5) und einem Portalkran (4) mit drei translatorischen Bewegungsachsen und einem elektromagnetischen Greifer, der in Z-Richtung fahrbar ist.

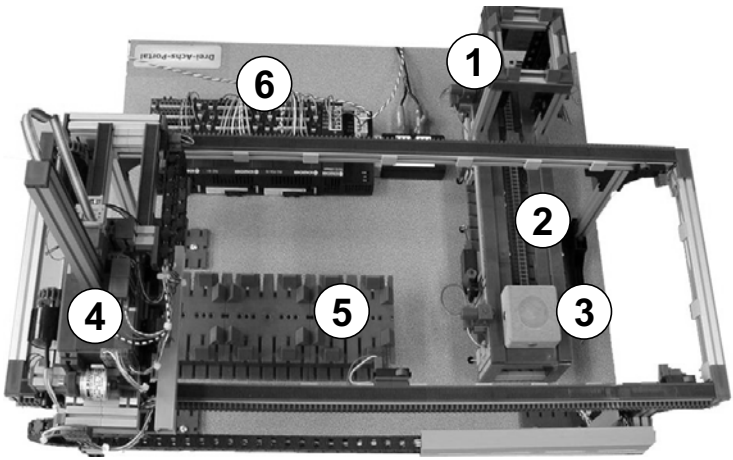


Abbildung 10.1: Gesamtansicht des Demonstrationsmodells 3-Achs-Portal

Abbildung 10.2 zeigt den schematischen Aufbau und den Materialfluss des 3-Achs-Portals. Codierte Werkstücke werden aus dem Magazin (1) einzeln ausgeworfen und durch das Förderband (2) über den Ablageplatz (3) dem Portalkran zugeführt. Beim Auswurf wird die Codierung der Werkstücke durch einen Sensor identifiziert. Aus dem Ablageplatz werden sie vom Portalkran (4) mittels des Greifers aufgenommen und zur Zwischenlagerung entsprechend ihrer Codierung

auf einem der Ablageplätze (5) sortiert abgelegt. Auf den Ablageplätzen befindliche Werkstücke können vom Portalkran geordnet entnommen und an eine nachfolgende Förderstrecke oder ein Transportsystem übergeben werden.

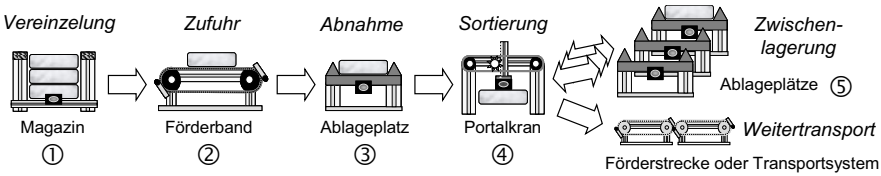


Abbildung 10.2: Schematischer Aufbau und Materialfluss des 3-Achs-Portals

Alle fördertechnischen Komponenten des 3-Achs-Portals besitzen analoge Signalanschlüsse zur Erfassung von Sensorsignalen und zur Ansteuerung der Aktoren. Die Signalanschlüsse sind über E/A-Module (6) an einen CAN-Feldbus (Controller Area Network [Etsc94]) angeschlossen. Über den CAN-Bus erfolgt die Steuerung des 3-Achs-Portals durch einen PC.

10.3 Agentenunterstütztes Engineering des 3-Achs-Portals

In den Anwendungsszenarien werden repräsentative Tätigkeiten im Rahmen des gesamten Engineerings der Demonstrationsanlage 3-Achs-Portal betrachtet: Durch Instanziierung, Konfiguration und Verbindung der einzelnen Komponenten ist das Anlagenmodell des 3-Achs-Portals zu entwickeln. Zudem ist ein gewerkspezifisches Teilmodell Netzwerkplan zu erstellen, das die Zuordnung der Signalanschlüsse zu den E/A-Modulen des CAN Busses beschreibt.

Ausgangssituation der Anwendungsszenarien

Mittels des CAE-Werkzeugs instanziiert der Ingenieur zunächst die benötigten Komponenten im Anlagenmodell. Dabei werden die in den Komponentenbeschreibungen spezifizierten Default-Werte der Komponenteneigenschaften in die Konfiguration der einzelnen Komponenteninstanzen übernommen. In Abbildung 10.3 oben sind die Komponenteninstanzen, ihre Anschlusspunkte vom Schnittstellentyp Materialfluss (durch ID gekennzeichnet) und wichtige Parameter dargestellt. Die Zuordnung von Eigenschaften zu Anschlusspunkten ist in Klammern angegeben, z.B. ist `Förderhöhe(P2)` die Förderhöhe der Komponente an ihrem Anschlusspunkt mit der ID 2.

Bei der Instanziierung werden die Informationen an das agentenorientierte Unterstützungssystem übermittelt und dort die entsprechenden Komponenten-Agenten erzeugt und initialisiert (Abbildung 10.3 unten). Die Agenten erfassen aus der Komponentenbibliothek das Wissen und die Informationen über die jeweiligen, von ihnen vertretenen Komponenteninstanzen. Zudem werden bei der Initialisierung des Agentensystems der Aspekt-Agent Anlagenmodell und der Bibliotheks-Agent erzeugt (Abbildung 10.3 unten).

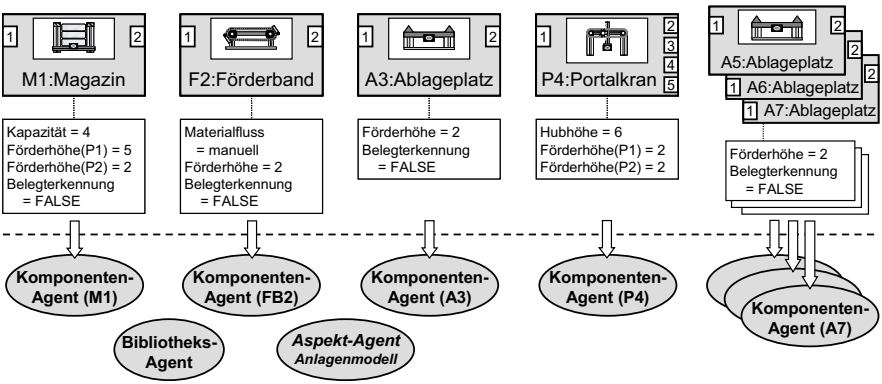


Abbildung 10.3: Anlagenmodell als Ausgangssituation der Anwendungsszenarien

10.3.1 Verbinden von Komponenten

Der Ingenieur möchte nun die instanziierten Komponenten entsprechend dem Materialfluss aus Abbildung 10.2 verbinden. Mittels der grafischen Funktion „Verbinden“ des CAE-Werkzeugs zieht er eine Verbindungslinie zwischen dem Anschlusspunkt 2 der Komponenteninstanz M1:Magazin und dem Anschlusspunkt 1 von F2:Förderband (vgl. Abbildung 10.5). Diese Aktivität wird vom CAE-Werkzeug an das Unterstützungssystem übermittelt und dort vom Aspekt-Agent Anlagenmodell als Verbindungsauftrag erfasst (0). In seiner Rolle als Verbindungsmanager koordiniert er die Verbindungsprüfung und überwacht den Verbindungsstatus, bis die Verbindung tatsächlich erstellt oder in Absprache mit dem Ingenieur verworfen wird. In Abbildung 10.4 sind zur Veranschaulichung die internen Abläufe des Agentensystems dargestellt.

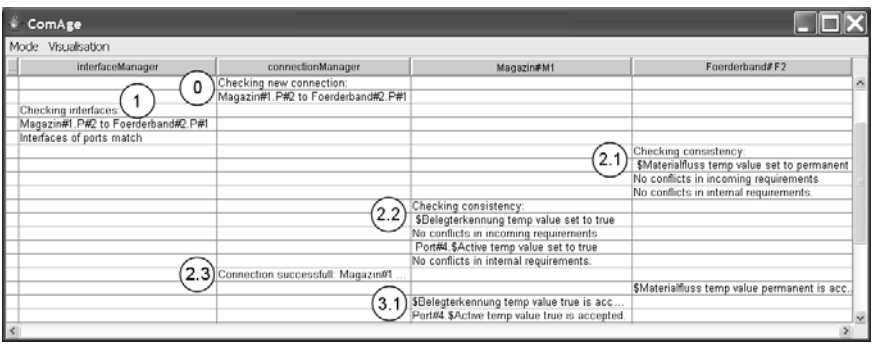


Abbildung 10.4: Interne Abläufe des Agentensystems bei Verbindungsprüfung

Schritt 1: Schnittstellenprüfung

Zunächst erfolgt die Schnittstellenprüfung für die betroffenen Anschlusspunkte der Komponenteninstanzen M1 und F2 durch die Rolle des Schnittstellenprüfers des Bibliotheks-Agenten. Da beide Anschlusspunkte den gleichen Schnittstellentyp aufweisen, findet anschließend die Ermittlung von Wechselwirkungen zwischen den zu verbindenden Komponenteninstanzen statt.

Schritt 2: Wechselseitige Prüfung zur Ermittlung der Wechselwirkungen

Der Verbindungsmanager beauftragt die Weiterleitung der Konsistenzbedingungen von M1 und ihre Prüfung durch F2. Der Komponenten-Agent von M1 ermittelt die gültigen Verbindungsregeln für den Anschlusspunkt 2, initialisiert sie mit den aktuellen Parameterwerten und sendet sie an den Komponenten-Agenten von F2:

$\texttt{Förderhöhe}_{\texttt{ext}}$ = Förderhöhe = 2
$\texttt{Materialfluss}_{\texttt{ext}}$ = permanent

Bedeutung: Die Förderhöhe der zu verbindenden Komponenteninstanz muss mit der Förderhöhe im Anschlusspunkt P2 übereinstimmen. Weiter muss stets ein permanenter Material(ab)fluss gewährleistet werden, um ein Verklemmen der Werkstücke im Magazin zu verhindern.

Der Komponenten-Agent von F2 prüft die erhaltenen Konsistenzbedingungen auf Wechselwirkungen mit der eigenen Konfiguration (2.1). Die Konsistenzbedingung `Förderhöhe = 2` ist durch die Konfiguration bereits erfüllt. Die Konsistenzbedingung `Materialfluss = permanent` stellt eine Inkonsistenz mit der aktuellen Konfiguration von F2 dar, da hier der Parameter `Materialfluss` den Wert `manuell` aufweist. Diese Inkonsistenz kann jedoch durch Änderung des Wertes auf `permanent` aufgelöst werden. Da auch keine internen Parametrierregeln von F2 im Widerspruch zu dieser Änderung stehen, wird die Änderung temporär zwischengespeichert und eine Bestätigungsmeldung an den Verbindungsmanager gesendet.

Anschließend erfolgt die entgegengesetzte Prüfung der Wechselwirkungen (2.2). Dabei ist unter Berücksichtigung der temporären Änderungen folgende Verbindungsregel von F2 gültig:

IF Materialfluss = permanent THEN $\texttt{Belegterkennung}_{\texttt{ext}}$ = TRUE

Wenn ein permanenter Materialfluss gewährleistet werden soll, muss die zuführende Komponente am Ausgang über einen Sensor zur Belegterkennung verfügen, um die Bewegung des Förderbandes auszulösen. Diese Konsistenzbedingung kann durch Setzen der Option `Belegterkennung` von M1 erfüllt werden. Zusätzlich wird dadurch eine interne Parametrierregel von M1 gültig, die als weitere Änderung die Aktivierung eines Signalanschlusses für das Sensorsignal der Belegterkennung bewirkt (in Abbildung 10.3 nicht dargestellt).

Die erfolgreiche Verbindungsprüfung wird vom Verbindungsmanager registriert (2.3). Da für die Umsetzung der Verbindung Änderungen der Konfigurationen der beiden Komponenteninstanzen M1 und F2 notwendig sind, ist eine Abstimmung mit dem Ingenieur erforderlich.

Schritt 3: Interaktion mit dem Ingenieur und Umsetzung der Anpassungen

Der Verbindungsmanager tritt über die Benutzungsoberfläche des CAE-Werkzeugs mit dem Ingenieur in Interaktion. In Abbildung 10.5 ist die Dialogmeldung des Verbindungsmanagers dargestellt. Sie besteht aus dem Hinweis, dass die Verbindung zu Wechselwirkungen zwischen Komponenteninstanzen führt und dem Vorschlag für geeignete Anpassungen zur Sicherstellung der Konsistenz. Zusätzlich werden die vorgeschlagenen Anpassungen in den betroffenen Komponenteninstanzen dargestellt. Der Ingenieur kann den Vorschlag annehmen oder ablehnen. Bei Zustimmung des Ingenieurs beauftragt der Verbindungsmanager die Komponenten-Agenten, die ermittelten Anpassungen von M1 und F2 umzusetzen (3.1) und die Verbindung zu speichern.

Abbildung 10.5: Ergebnis der Verbindungsprüfung mit Lösungsvorschlag

Abbildung 10.6 zeigt das Protokoll der Agenten-Interaktionen bei der Komponentenverbindung. Die Interaktionen bei der wechselseitigen Prüfung zur Ermittlung der Wechselwirkungen (2.1 und 2.2) und zur vollständigen Umsetzung der Anpassungen (3.1) werden deutlich erkennbar.

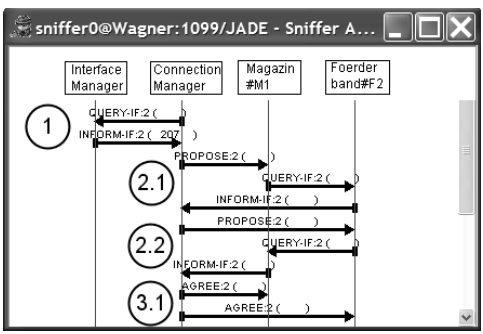


Abbildung 10.6: Agenten-Interaktionen bei Verbindungsprüfung und -umsetzung

Der Ingenieur setzt die Verbindung der Komponenteninstanzen fort, in deren Verlauf die Softwareagenten weitere Wechselwirkungen ermitteln, Inkonsistenzen identifizieren, dem Ingenieur geeignete Anpassungen vorschlagen und umsetzen. In Abbildung 10.7 ist der anschließende Zustand des Anlagenmodells dargestellt. Alle Parameteranpassungen sind fett hervorgehoben.

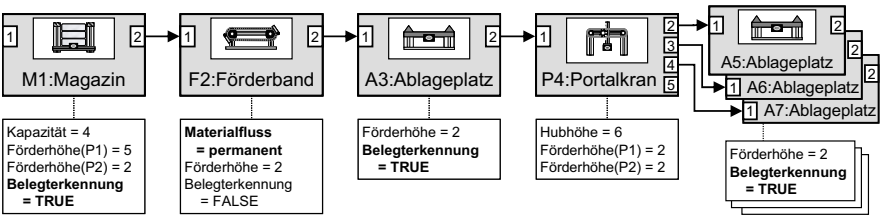


Abbildung 10.7: Anlagenmodell nach erfolgten Verbindungen und Anpassungen

10.3.2 Änderung der Konfiguration einer Komponente

Der Ingenieur möchte die Förderhöhe des Förderbandes F2 verringern, da hier zusätzlich ein manueller Arbeitsplatz zur Sichtprüfung vorgesehen werden soll. Durch grafische Auswahl der Komponenteninstanz F2 im CAE-Werkzeug und Eingabe des Wertes „1“ in der Eigenschaftstabelle ändert er den Parameter Förderhöhe. Diese Aktivität wird vom CAE-Werkzeug wiederum an das Unterstützungssystem übermittelt und dort vom Komponenten-Agent F2 als Änderungsauftrag erfasst. In seiner Rolle als Änderungswächter koordiniert er die Ermittlung der Änderungsauswirkungen und überwacht die Prüfungen und Änderungen aller Komponenten-Agenten, bis die Änderung in Absprache mit dem Ingenieur umgesetzt oder verworfen wird. In Abbildung 10.8 sind zur Veranschaulichung die internen Abläufe des Agentensystems dargestellt. In der Abbildung ist die Weiterleitung der Änderung, der Rückmeldung aller betroffenen Komponenten-Agenten und ihrer Prüfungsergebnisse sowie die Umsetzung aller Änderungen protokolliert.

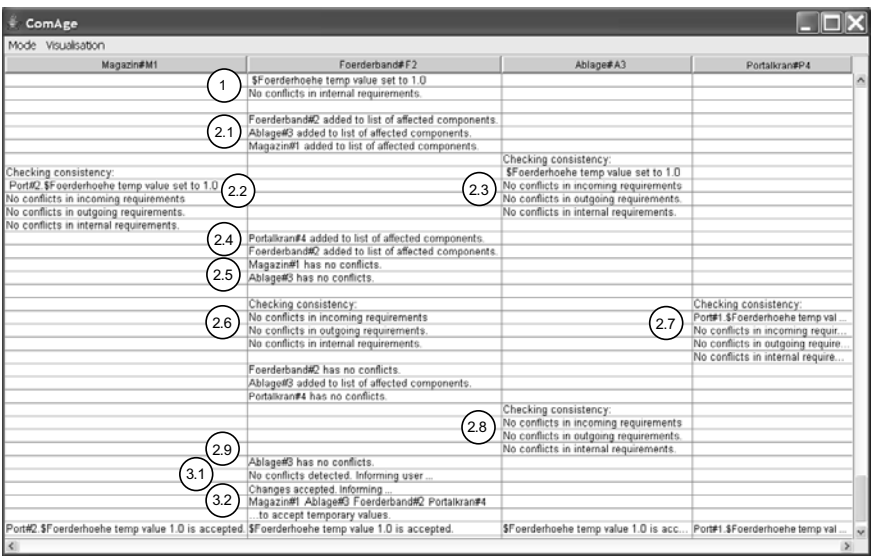


Abbildung 10.8: Interne Abläufe des Agentensystems bei Änderungsprüfung

Schritt 1: Parameteränderung komponentenintern prüfen

Zunächst erfolgt die interne Prüfung der Änderung für die Komponenteninstanz F2 durch die Rolle des Konsistenzprüfers. Da durch den Parameter Förderhöhe keine Parametrierregeln betroffen sind, sind keine weiteren Änderungen der Konfiguration erforderlich.

Schritt 2: Änderungsauswirkungen ermitteln und Rückmeldungen verwalten

Alle Komponenten des Anlagenmodells weisen Verbindungsregeln auf, welche die Übereinstimmung der Förderhöhe verbundener Komponenten mit der eigenen Förderhöhe fordern:

$\texttt{Förderhöhe}_{\texttt{ext}} = \texttt{Förderhöhe}$

Während die Komponenten Förderband und Ablageplatz jedoch nur eine Förderhöhe aufweisen, können die Komponenten Magazin und Portalkran über unterschiedliche Förderhöhen an ihren Eingänge und Ausgängen verfügen (der Parameter Förderhöhe ist jedem Anschlusspunkt vom Typ Materialfluss zugeordnet, vgl. Abbildung 10.7).

Die Ermittlung der Auswirkungen der Änderung der Konfiguration von F2 und die Koordination der Anpassungen aller betroffenen Komponenteninstanzen werden von der Rolle des Änderungswächters von Komponenten-Agent F2 durchgeführt. Der Änderungswächter legt eine Überwachungsliste zur Erfassung aller Rückmeldungen an und trägt die bekannten betroffenen Komponenten-Agenten F2, M1 und A3 ein (2.1). Die Änderung und die Konsistenzbedingungen für den jeweiligen Anschlusspunkt werden an die Konsistenzprüfer der verbundenen Komponenten-Agenten M1 und A3 weitergeleitet und dort auf Wechselwirkungen geprüft (2.2), (2.3).

Die Konsistenzbedingung `Förderhöhe = 1` führt zur Änderung der Konfigurationen der beiden Komponenteninstanzen: In M1 wird der Parameter `Förderhöhe` im Anschlusspunkt P2 angepasst, in A3 der komponenteninterne Parameter `Förderhöhe`. Durch diese Änderungen können wiederum indirekte Wechselwirkungen zu den mit M1 und A3 verbundenen Komponenteninstanzen entstehen. Von der Änderung in M1 kann nur F2 betroffen sein, von der Änderung in A3 können F2 und P4 betroffen sein. Die beiden Komponenten-Agenten von M1 und A3 melden die potenziell betroffenen weiteren Komponenteninstanzen an den Änderungswächter von F2 (2.4) sowie die Bestätigungsmeldung ihrer erfolgreichen Prüfung (2.5) zurück: Die Überwachungsliste wird entsprechend erweitert. Die Weitergabe der lokalen Änderung von M1 und A3 zusammen mit den jeweiligen Konsistenzbedingungen erfolgt entsprechend an die Konsistenzprüfer von F2 und P4 (2.6), (2.7). Der Konsistenzprüfer des Komponenten-Agenten von F2 stellt keine weiteren erforderlichen Änderungen fest, da die relevante Konsistenzbedingung `Förderhöhe = 1` bereits erfüllt ist, sodass hier keine weiteren indirekten Wechselwirkungen entstehen können. In P4 hingegen muss der Parameter `Förderhöhe` im Anschlusspunkt P1 angepasst werden. Von dieser Änderung ist lediglich der Komponenten-Agent A3 betroffen. Er wird informiert und prüft die Wechselwirkungen (2.8). Die Komponenteninstanzen A5, A6 und A7 sind nicht betroffen, da die Änderung im Anschlusspunkt P1 erfolgt, sie aber über ande-

re Anschlusspunkte (P2-P4) mit P4 verbunden sind. Daher werden sie nicht vom Komponenten-Agenten P4 informiert. Komponenten-Agent A3 sendet nach erfolgter Prüfung eine Bestätigungsmeldung an den Änderungswächter von F2, der die Überwachungsliste entsprechend aktualisiert (2.9). Nach Eintreffen aller Rückmeldungen erfolgt die Abstimmung der ermittelten zusätzlichen Änderungen mit dem Ingenieur.

Schritt 3: Interaktion mit dem Ingenieur und Umsetzung der Anpassungen

Der Änderungswächter tritt über die Benutzungsoberfläche des CAE-Werkzeugs mit dem Ingenieur in Interaktion (3.1). In Abbildung 10.9 ist die Dialogmeldung des Änderungswächters dargestellt. Sie besteht aus dem Hinweis, dass die Änderung der Förderhöhe von F2 zu Wechselwirkungen mit den Komponenteninstanzen M1, A3 und P4 führt und dem Vorschlag für geeignete Anpassungen zur Sicherstellung der Konsistenz. Zusätzlich werden die betroffenen Komponenteninstanzen hervorgehoben und die vorgeschlagenen Anpassungen jeweils dargestellt.

Abbildung 10.9: Ergebnis der Änderungsprüfung mit Lösungsvorschlag

Nach Zustimmung des Ingenieurs beauftragt der Änderungswächter von F2 die Konsistenzprüfer aller betroffenen Komponenten-Agenten, die ermittelten Anpassungen umzusetzen (3.2). In Abbildung 10.7 ist der anschließende Zustand des Anlagenmodells dargestellt. Alle Parameteranpassungen sind fett hervorgehoben.

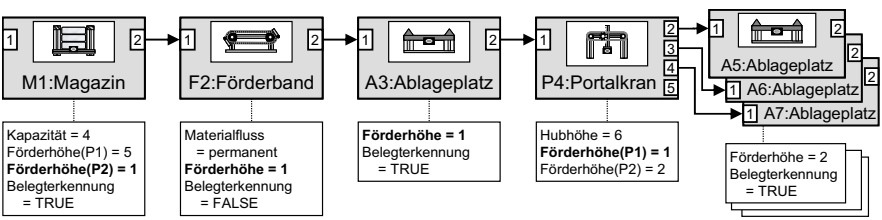


Abbildung 10.10: Anlagenmodell nach Parameteränderung und Agenten-Anpassungen

10.3.3 Erstellung eines gewerkspezifischen Teilmodells

Aus den Informationen des Anlagenmodells ist das gewerkspezifische Teilmodell Netzwerkplan zu erstellen, das die Zuordnung der Signalanschlüsse zu den E/A-Modulen des CAN Busses beschreibt. Diese Tätigkeit wird durch den Aspekt-Agenten Teilmodell Netzwerkplan unterstützt, der über eine fachspezifische Wissensbasis für die Erstellung von Netzwerkplänen verfügt [Yilm05]. Diese enthält u.a. folgende Regeln (siehe Anhang B):

- Alle aktivierten Anschlusspunkte der Komponenteninstanzen vom Schnittstellentyp "Signal" sind zu ermitteln und eine entsprechende interne Vernetzungsliste ist zu erstellen.
- Um möglichst kurze Signalleitungen zu erhalten, soll die räumliche Nähe der Komponenteninstanzen berücksichtigt und die erstellte Liste entsprechend geordnet werden.
- Ausgehend von der internen Vernetzungsliste ist der Netzwerkplan durch Zuordnung der Signalanschlüsse der Komponenteninstanzen zu den Feldbus-E/A-Modulen zu erstellen.
- Die verfügbaren Signale der E/A-Module sind möglichst vollständig zu belegen.
- Anschlüsse einer Komponenteninstanz sollen nicht auf mehrere E/A-Module verteilt sein.

Schritt 1: Informationen erfassen und Komponenteninstanzen selektieren

Der Ingenieur beauftragt den Aspekt-Agenten, das Teilmodell Netzwerkplan zu erstellen. In der Rolle Instanzenselektierer erfasst der Aspekt-Agent die Informationen aller im Anlagenmodell enthaltenen Komponenteninstanzen und selektiert mithilfe der Auswahlregeln alle relevanten Instanzen, die über aktivierte Anschlusspunkte vom Schnittstellentyp „Signal“ verfügen (1.1). Im vorliegenden Beispiel trifft dies auf alle Komponenteninstanzen des Anlagenmodells zu. Er gruppiert sie gemäß der Gruppierungsregel in einer internen Vernetzungsliste. Zusätzlich meldet der Aspekt-Agent in seiner Rolle als Teilmodellmanager sich selbst bei allen Komponenten-Agenten als abhängig an, um im Falle von Änderungen informiert zu werden (1.2). Abbildung 10.11 zeigt das Protokoll der Agenten-Interaktionen.

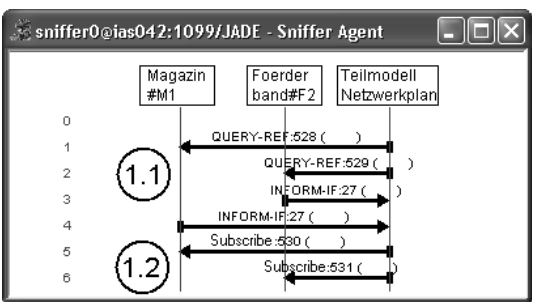


Abbildung 10.11: Agenten-Interaktionen bei Teilmodellerstellung (Ausschnitt)

Schritt 2: Teilmodell erstellen

Anschließend erfolgt ausgehend von den Informationen der internen Vernetzungsliste die Erstellung des Teilmodells Netzwerkplan durch die Rolle Modellierer. Dabei wird die benötigte Anzahl an CAN E/A-Modulen ermittelt und die Signalanschlüsse der Komponenteninstanzen werden entsprechend der Zuordnungsregeln aus der Wissensbasis den CAN E/A-Modulen zugeordnet. Im vorliegenden Beispiel stehen CAN E/A-Module mit je vier Eingangs- und vier Ausgangssignalen zur Verfügung. Nach der Erstellung kann sich der Ingenieur das Teilmodell Netzwerkplan als Excel-Tabelle darstellen lassen (siehe Abbildung 10.12).

E/A-Modul ID	Signal ID	I/0	Anschluss an	Port ID	Port Name
CAN#0	1	Input	M1:Magazin	port#4	Sensorsignal Belegtmelder
	2	Input	M1:Magazin	port#6	Sensorsignal Codierungssensor links
	3	Input	M1:Magazin	port#7	Sensorsignal Codierungssensor links
	4	Input	A3:Ablageplatz	port#3	Sensorsignal Belegtmelder
	5	Output	M1:Magazin	port#3	Steuersignal Motor Werkstückauswurf
	6	Output	F2:Foerderband	port#3	Steuersignal Motor
	7	Output	-	-	
	8	Output	-	-	
CAN#1	1	Input	P4:Portalkran	port#7	Sensorsignal Endschalter X-Achse
	2	Input	P4:Portalkran	port#9	Sensorsignal Endschalter Y-Achse
	3	Input	P4:Portalkran	port#11	Sensorsignal Endschalter Z-Achse
	4	Input	A5:Ablageplatz	port#3	Sensorsignal Belegtmelder
	5	Output	P4:Portalkran	port#6	Steuersignal fuer Motor X-Achse
	6	Output	P4:Portalkran	port#8	Steuersignal fuer Motor Y-Achse
	7	Output	P4:Portalkran	port#10	Steuersignal fuer Motor Y-Achse
	8	Output	P4:Portalkran	port#12	Steuersignal Greifer
CAN#2	1	Input	A6:Ablageplatz	port#3	Sensorsignal Belegtmelder
	2	Input	A7:Ablageplatz	port#3	Sensorsignal Belegtmelder
	3	Input	-	-	
	4	Input	-	-	
	5	Output	-	-	
	6	Output	-	-	
	7	Output	-	-	
	8	Output	-	-	

Abbildung 10.12: Erstelltes gewerkspezifisches Teilmodell Netzwerkplan

Im erstellten Netzwerkplan ist zu erkennen, wie die Zuordnungsregeln für CAN E/A-Module durch den Aspekt-Agenten berücksichtigt wurden: Alle Signalanschlüsse einer Komponenteninstanz wurden zusammen je einem E/A-Modul zugeordnet, die räumliche Nähe von verbundenen Komponenteninstanzen wurde berücksichtigt und die möglichst vollständige Ausnutzung der Signale der E/A-Module wurde umgesetzt.

10.4 Ergebnisse und Erfahrungen

Nachdem die Wirkungsweise des agentenunterstützten Engineerings anhand von Anwendungsszenarien betrachtet wurde, sollen in diesem Abschnitt die Ergebnisse und Erfahrungen bei der Verwendung des Unterstützungssystems zusammengefasst werden.

Zur Erstellung des Anlagenmodells der Fördertechnikanlage „3-Achs-Portal“ muss der Ingenieur 7 Komponenten instanziieren, 6 Verbindungen erstellen, 1 Parameteranpassung vornehmen und das Teilmodell Netzwerkplan erstellen. Bei jeder Tätigkeit wird er durch die Softwareagenten unterstützt. Die Ergebnisse der Unterstützung sind in Tabelle 10.2 zusammengefasst.

Tabelle 10.2: Ergebnisse der Unterstützung in den Anwendungsszenarien

Tätigkeiten	**Überprüfte Wechselwirkungen**	**Erkannte Inkonsistenzen**	**Ermittelte Anpassungen**	**Lösungsvorschläge**	**Anzahl Agenten-Interaktionen**	**Anzahl Nachrichten**
Komponenten M1-F2-A3-P4-A5-A6-A7 verbinden	32	13	13	6	13	69
Förderhöhe F2 anpassen	6	3	3	1	1	19
Netzwerkplan erstellen	-	-	-	1	2	21
Summe (14 Tätigkeiten)	38	16	16	8	16	109

Im Gegensatz zum manuellen Aufwand, der für die Prüfung und Handhabung von Wechselwirkungen und für die Erstellung gewerkspezifischer Teilmodelle erbracht werden muss, reduziert sich der Aufwand beim Engineering mit Agentenunterstützung deutlich. Den noch verbleibenden 14 manuellen Tätigkeiten stehen 38 Wechselwirkungsprüfungen und 16 Anpassungen zur Auflösung von ebenso vielen Inkonsistenzen gegenüber, die vom agentenorientierten Unterstützungssystem selbstständig durchgeführt werden. Dabei konnte festgestellt werden, dass alle Inkonsistenzen erkannt und geeignete, vollständige Lösungen ermittelt werden. Der Ingenieur muss lediglich noch über die 8 Lösungsvorschläge des Unterstützungssystems entscheiden. Die Erstellung des Teilmodells Netzwerkplan erfolgt vollkommen ohne manuellen Aufwand.

Die Anzahl der Agenten im Unterstützungssystem beträgt 10 (7 Komponenten-Agenten, 2 Aspekt-Agenten und 1 Bibliotheks-Agent). Insgesamt finden im Rahmen Unterstützungshandlungen 16 Agenten-Interaktionen statt, bei denen 109 Nachrichten ausgetauscht werden, wobei die Reaktion auf eine Tätigkeit ohne merkliche Verzögerung für den Anwender erfolgt.

In diesem Kapitel wurde am Beispiel des Engineerings einer Fördertechnikanlage die praktische Anwendung des agentenunterstützten Engineerings betrachtet. Dabei wurde deutlich, dass das in dieser Arbeit vorgeschlagene Konzept die Aufwendungen beim Engineering wirksam reduziert und eine effiziente Anwendung der Unterstützung ermöglicht, ohne zusätzliche Aufwendungen oder Überlegungen des Ingenieurs zu erfordern. Im nächsten Kapitel wird das Konzept des agentenunterstützten Engineerings abschließend zusammengefasst und bewertet. Dabei werden nochmals die wichtigsten Erkenntnisse dieser Arbeit dargelegt, die Vorteile und Grenzen des Konzepts diskutiert und ein Ausblick auf mögliche Fortsetzungen der Arbeit gegeben.

11 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde ein neues Konzept vorgestellt, das zur Unterstützung des Engineerings von Automatisierungsanlagen dient und eine einfachere und effizientere Durchführung der menschlichen Tätigkeiten bei der komponentenbasierten Erstellung von Anlagenmodellen ermöglicht. Das Konzept beinhaltet folgende wesentlichen Kernpunkte:

- Softwareagenten vertreten die Handlungsobjekte des Ingenieurs beim komponentenbasierten Engineering – Komponenten und gewerkspezifische Teilmodelle. Sie kapseln die relevanten Engineeringinformationen über diese Handlungsobjekte sowie das Wissen über technische Abhängigkeiten von Komponenten und über fachspezifische, komponentenübergreifende Zusammenhänge.
- Softwareagenten erfassen aktiv die menschlichen Tätigkeiten beim Engineering als Auftrag und ermitteln selbstständig entstehende Auswirkungen und Wechselwirkungen zwischen abhängigen Engineeringinformationen. Dabei agieren und interagieren sie permanent im Hintergrund der menschlichen Tätigkeiten.
- Softwareagenten interagieren mit dem Ingenieur durch tätigkeitsbezogene Hinweise und Lösungsvorschläge, um Entscheidungen über erforderliche Anpassungen von Engineeringinformationen zur Sicherstellung ihrer Konsistenz konstruktiv zu unterstützen.

Das vorgeschlagene Konzept wurde am eines Beispiel des Engineerings einer fördertechnischen Anlage in der Praxis evaluiert.

11.1 Bewertung des Konzepts

Das Konzept des agentenunterstützten Engineerings bewirkt eine Reduktion der Fehlermöglichkeiten bei der Komponentenintegration durch die selbstständige Überprüfung von Komponenteneigenschaften, den Abgleich von Schnittstellen und Konfigurationen und die Ermittlung von Änderungsauswirkungen sowie durch Vorschläge für geeignete Anpassungen von Konfigurationen und Verbindungen. Die Wiederverwendung von Baugruppen wird durch die selbstständige, geeignete Anpassung ihrer internen Struktur an wechselnde Anforderungen unterstützt. Die Fehlermöglichkeiten bei gewerkspezifischen Tätigkeiten werden durch die Unterstützung bei der Erstellung und Überprüfung gewerkspezifischer Teilmodelle sowie durch die selbstständige Ermittlung und Handhabung von Abhängigkeiten zwischen dem komponentenbasierten Anlagenmodell und gewerkspezifischen Teilmodellen reduziert.

Beitrag des Konzepts zur Steigerung der Effizienz im Engineering

Das vorgeschlagene Konzept bewirkt eine Entkopplung der Tätigkeiten zur Handhabung einzelner Komponenten von den Detailtätigkeiten zur Ermittlung bzw. Handhabung der Auswirkungen einer Tätigkeit auf andere Engineeringinformationen des Anlagenmodells oder gewerkspezifische Teilmodelle. Letztere werden aktiv von den Softwareagenten des agentenorientierten Unterstützungssystems übernommen und müssen somit nicht mehr vom Ingenieur selbst bei jeder einzelnen Tätigkeit berücksichtigt werden. Der Ingenieur kann sich auf den kreativen Problemlösungsprozess beim Engineering einer Automatisierungsanlage und seine Umsetzung durch Auswahl und Integration von technischen Komponenten zu einem Anlagenmodell konzentrieren. Lediglich wenn durch eine Tätigkeit tatsächlich Wechselwirkungen zwischen Engineeringinformationen entstehen, welche ihre Konsistenz beeinträchtigen, werden diese durch situationsbezogene Meldungen der Softwareagenten wieder in den menschlichen Problemlösungsprozess eingebracht. Gleichzeitig wird die Handhabung erkannter Wechselwirkungen durch Lösungsvorschläge für geeignete Anpassungen konstruktiv unterstützt. Der manuelle Aufwand für Detailtätigkeiten und zugehörige Überlegungen wird somit verringert und die Effizienz des Engineerings erhöht. Das in [LaGö99b] genannte Ziel der Rechnerunterstützung zur Entlastung von lästigen Routinearbeiten wird durch das Konzept erreicht.

Beitrag des Konzepts zur Steigerung der Qualität der Engineeringergebnisse

Neben der Eignung der vom Ingenieur erarbeiteten Lösung hinsichtlich der funktionalen und nichtfunktionalen Anforderungen an eine Automatisierungsanlage hängt die Qualität der Engineeringergebnisse wesentlich davon ab, ob alle technischen Abhängigkeiten der Automatisierungsanlage in den Engineeringinformationen beachtet wurden. Nur im Falle einer vollständigen technischen Konsistenz aller Engineeringinformationen ist die korrekte Funktionalität der Automatisierungsanlage gewährleistet. Dieser Aspekt wird durch das vorgeschlagene Konzept adressiert. Durch die Vorgehensweise der Softwareagenten bei ihren Aktionen und Interaktionen zur Ermittlung und Handhabung von Wechselwirkungen werden nicht nur die bei einer Tätigkeit entstehenden direkten Auswirkungen, sondern auch durch Anpassungen entstehende weitergehende, indirekte Wechselwirkungen betrachtet. Dadurch wird sichergestellt, dass *alle* bestehenden Abhängigkeiten zwischen Engineeringinformationen berücksichtigt und dem Ingenieur vermittelt werden. Das Übersehen von Zusammenhängen wird durch den aktiven Eingriff der Softwareagenten kompensiert und die in [LaGö99b] genannten weiteren Ziele der Rechnerunterstützung zur Vermeidung von Spezifikations- und Entwurfsfehlern und Beherrschung der Komplexität erreicht.

Anwendbarkeit für Ingenieure

Ein wesentlicher Vorteil des Konzepts ist die fortlaufende, selbstständige Anpassung der Softwareagenten an die Tätigkeiten des Ingenieurs und die dabei erstellten oder veränderten Engineeringinformationen. Die unterstützenden Handlungen fügen sich nahtlos in den durch den In-

genieur vorgegebenen Handlungsrahmen ein. Bei der Verwendung des agentenorientierten Unterstützungssystems muss der Ingenieur daher keine Einschränkungen der Flexibilität in seiner Vorgehensweise und hinsichtlich der zu verwendenden Komponenten in Kauf nehmen. Zudem wird er durch die begleitende Form der Problemlösung nicht in ein Korsett automatischer Korrekturen gezwungen, sondern kann Hinweise und Lösungsvorschläge der Softwareagenten annehmen, verwerfen oder durch manuelle Eingriffe an seine Vorstellungen anpassen. Manuelle Anpassungen sind für die Softwareagenten wiederum Anlass für erneute Prüfungen.

Ein weiterer wichtiger Vorteil des Konzepts ist die Tatsache, dass seine Anwendung keine zusätzlichen Expertenkenntnisse des Ingenieurs über Softwareagenten oder über wissensbasierte Konzepte erfordert. Lediglich wenige Domänenexperten, welche das Wissen über technische Abhängigkeiten von Komponenten und über fachspezifische komponentenübergreifende Zusammenhänge ermitteln und bereitstellen, müssen über Kenntnisse über regelbasierte Wissensrepräsentation verfügen. Die Abbildung der Handlungsobjekte des Ingenieurs beim komponentenbasierten Engineering auf die Struktur des Agentensystems und die Abbildung der Vorgehensweise des Ingenieurs bei der Wechselwirkungsermittlung und -handhabung auf die Abläufe der Aktionen und Interaktionen der Softwareagenten gewährleisten die Nachvollziehbarkeit von Hinweisen und Lösungsvorschlägen ohne weitere Kenntnisse der internen Abläufe des Unterstützungssystems. Durch die intuitive Nachvollziehbarkeit der unterstützenden Handlungen wird zudem das Vertrauen in ihre Korrektheit und die Anwenderakzeptanz erhöht.

Verträglichkeit mit bestehenden Komponentenmodellen und CAE-Systemen

Das vorgeschlagene Konzept wurde anhand eines exemplarischen Komponentenmodells und in Kombination mit einem exemplarischen CAE-Werkzeug evaluiert. Dabei wurde jedoch darauf geachtet, dass die zugrunde liegenden Konzepte nicht von den verwendeten Datenstrukturen und Softwaretechnologien abhängig sind. Das interne Informationsmodell der Softwareagenten entspricht den Anforderungen des in Kapitel 7 vorgestellten, anwendungsunabhängigen Strukturmetamodells. So wird gewährleistet, dass die Agenten alle für ihre Unterstützungsaktivitäten erforderlichen Informationen erfassen und verwerten können, unabhängig von der konkreten Syntax und Semantik des Komponentenmodells, das den Engineeringinformationen in der Komponentenbibliothek und im Anlagenmodell zugrunde liegt. Die flexible Interoperabilität unterschiedlicher Komponentenmodelle mit dem internen Informationsmodell der Softwareagenten kann durch Anwendung des Konzepts des adaptiven Informationsaustauschs hergestellt werden, welches in [Eber05] vorgestellt wird. Die Herstellung der Interoperabilität zwischen dem agentenorientierten Unterstützungssystem und CAE-Systemen auf Systemebene wird durch die Kopplung über externe Systemschnittstellen erleichtert. Zur Anbindung an ein weiteres CAE-System muss lediglich eine entsprechende Systemschnittstelle aufseiten des Agentensystems ergänzt werden. Durch diese Flexibilität kann das Konzept in bestehende Engineeringumgebungen integriert werden, ohne die Abkehr von bestehenden CAE-Systemen und Technologien vorauszusetzen.

11.2 Voraussetzungen und Grenzen

Das vorgeschlagene Konzept setzt voraus, dass das Wissen über die technischen Abhängigkeiten einzelner Komponenten innerhalb einer Anwendungsdomäne bei der Komponentenentwicklung erfasst und spezifiziert wird. Dies erfordert einen initialen, zusätzlichen Aufwand durch den Domänenexperten. Demgegenüber steht die Reduktion des Aufwandes vieler Ingenieure in vielen Engineeringprojekten durch den Einsatz des agentenorientierten Unterstützungssystems. Das Konzept bewirkt somit nicht nur eine Verlagerung des Aufwandes vom Engineering hin zur Komponentenentwicklung, sondern eine Reduzierung des Aufwandes bei Betrachtung des gesamten Entwicklungszyklus. Die Reduzierung wird durch Nutzung der Vorteile der Mehrfachverwendung von Informationen und Wissen und deren rechnergestützte Anwendung zur Unterstützung von Engineeringtätigkeiten durch Softwareagenten erzielt.

Während die Vorteile der Unterstützungsfunktionalität durch das agentenorientierte Unterstützungssystem unabhängig vom spezifischen Anwendungsfall gegeben sind, hängt der Nutzen der Mehrfachverwendung von der Anwendbarkeit des komponentenbasierten Ansatzes und damit vom Standardisierungspotenzial einer Anwendungsdomäne ab. So kommen die Vorteile des Konzepts besonders dann zum Tragen, wenn sich innerhalb einer Anwendungsdomäne abgrenzbare technische Komponenten identifizieren lassen, die in möglichst vielen Automatisierungsanlagen unverändert eingesetzt werden können. Ist eine derartige *Komponenten-Standardisierung* bereits weit fortgeschritten, wird der Aufwand für die Spezifikation technischer Abhängigkeiten durch die Betrachtung bereits bestehender Anwendungskontexte zusätzlich reduziert.

Ist eine sehr geringe Individualität und Lösungsvarianz verschiedener Automatisierungsanlagen innerhalb einer Anwendungsdomäne gegeben, weil diese aus überwiegend gleichartigen Strukturen und Komponentenkombinationen aufgebaut sind, so besteht ein Standardisierungspotenzial nicht nur hinsichtlich einzelner Komponenten, sondern hinsichtlich der Struktur der gesamten Anlage. In diesem Bereich kann eine weitergehende Effizienz- und Qualitätssteigerung des Engineerings erreicht werden, da die Automatisierungsanlagen nicht wiederkehrend aus einzelnen Komponenten oder Baugruppen neu aufgebaut werden. Stattdessen kommen bereits heute Framework-Ansätze zum Einsatz, die auf der Beschreibung einer vollständigen Referenzanlage und der Beschreibung der punktuellen Variationsmöglichkeiten basieren [Urba03]. Dadurch sind nicht nur alle wichtigen Entwurfsentscheidungen bereits vorweggenommen, sondern auch die möglichen Komponentenwechselwirkungen implizit berücksichtigt. Solange sich dabei die Variationsmöglichkeiten in einem überschaubaren Rahmen halten, ist ein solches Vorgehen auch weiterhin anzuraten.

11.3 Ausblick

Im Verlauf dieser Arbeit und auf Grundlage der gewonnenen Erfahrungen wurden einige interessante Punkte identifiziert, die Ansätze für eine Fortsetzung der Arbeit bieten:

- *Ingenieurgerechte Methodik zur Wissensidentifikation*: Im Rahmen von industriellen Projekten wurde die Verfügbarkeit des Wissens über technische Abhängigkeiten und gewerkspezifische Zusammenhänge untersucht [Muba04], [Denc05]. Dabei wurde festgestellt, dass dieses Wissen zwar verfügbar ist, aber in der Praxis seine Ermittlung auf der Erfahrung weniger Experten beruht. Ein Ansatzpunkt für die Fortsetzung der Arbeit ist daher die Erarbeitung einer ingenieurgerechten, allgemein anwendbaren Methodik zu systematischen Identifikation des benötigten Wissens. Ein vielversprechender Ansatz ist die Verwendung von so genannten „Engineering-Use-Cases“. Dabei werden auf der Basis von Experteninterviews und bestehender Dokumentation die Verwendungsfälle von Komponenten untersucht und die Voraussetzungen und Randbedingungen für ihren korrekten Einsatz explizit dokumentiert. Mit dieser Vorgehensweise wurden bei den oben genannten Projekten bereits positive Erfahrungen gesammelt. Zudem könnte der Erstellungsaufwand für entsprechende Komponentenbeschreibungen mit speziellen grafischen Editoren deutlich gesenkt werden. Insbesondere die Unterstützung bei der korrekten Formalisierung von Verbindungs- und Parametrierregeln würde hier eine wesentliche Vereinfachung bedeuten.

- *Verbesserung der Skalierbarkeit*: Da für das Engineering generell PCs eingesetzt werden, bezieht sich die Frage nach der Skalierbarkeit des agentenorientierten Unterstützungssystems für große Anlagenmodelle im Wesentlichen auf die Ressourceneffizienz der verwendeten Agentenplattform. Die mit der Anzahl der Softwareagenten steigende Anzahl paralleler Rechenprozesse wird durch die Agenten-Laufzeitumgebung verwaltet. Bei der Evaluierung des Prototyps konnte festgestellt werden, dass die verwendete Laufzeitumgebung von JADE nur für Anlagen mittlerer Größe eine hinreichende Rechengeschwindigkeit gewährleistet. Diese Erfahrung wird auch durch andere Studien bestätigt [CGK+05]. Seit Kurzem ist jedoch die Entwicklungs- und Laufzeitumgebung LS/TS der Fa. Whitstein verfügbar, die eine ressourceneffiziente Verwaltung von bis zu 100 000 Softwareagenten gewährleistet [LSTS]. Für große Anlagen, die aus bis zu 15 000 Komponenten bestehen können, muss das Unterstützungssystem auf einer solchen Agentenplattform implementiert werden.

- *Übertragung auf andere Anwendungsbereiche*: Das vorgeschlagene Konzept ist nicht nur für das Engineering von Automatisierungsanlagen gültig, sondern kann auch in anderen Anwendungsbereichen zur Unterstützung der komponentenbasierten Entwicklung eingesetzt werden. Die Anwendung des Konzepts ist besonders dann von Vorteil, wenn es sich um Anwendungsdomänen handelt, bei denen verfügbare Basiselemente ausgewählt, konfiguriert und zu einem Lösungsobjekt zusammengesetzt werden, welches bestimmte Anforderungen oder Randbedingungen erfüllen muss. Beispiele sind die komponentenbasierte Softwareentwicklung von eingebetteten Systemen [MaWa05] oder die rechnergestützte Konstruktion (CAD – Computer Aided Design).

Literaturverzeichnis

[AAF03] R. Alznauer, K. Auer, A. Fay: *Wiederverwendung von Automatisierungs-Informationen und -Lösungen*. In: atp – Automatisierungstechnische Praxis 45 (2003) H. 3, S. 31-35.

[AcLö05] R. Achatz, U. Löwen: *Industrieautomation*. In: Software Engineering eingebetteter Systeme: Grundlagen, Methodik, Anwendungen, P. Liggesmeyer, D. Rombach (Hrsg.), München: Elsevier, Spektrum Akademischer Verlag, 2005, S. 497-525.

[Arm05] P. Armoutsis: *Realisierung eines verteilten Algorithmus zur automatisierten Verknüpfung von Anlagenkomponenten durch Softwareagenten*. Diplomarbeit Nr. 2037, IAS, Universität Stuttgart, 2005.

[ART04] F. Anhäuser, H. Richert, H. Temmen: *Degussa PlantXML – integrierter Planungsprozess mit flexiblen Bausteinen.* In: atp – Automatisierungstechnische Praxis 46 (2004) H. 10, S. 62-72.

[Balz01] H. Balzert: *Lehrbuch der Software-Technik, Band 1: Software-Entwicklung*. 2.Aufl., Heidelberg: Spektrum Akademischer Verlag, 2001.

[Baue02] B. Bauer: *UML class diagrams revisited in the context of agent-based systems*. In:. Proceedings of the Second International Workshop on Agent-oriented Software Engineering (AOSE2001), M. Wooldridge, G. Weiß, P. Ciancarini (Hrsg.), Lecture Notes in Articial Intelligence, Vol. 2222. Berlin: Springer-Verlag, 2002.

[BCB+01] J. Barata, L. Camarinha-Matos, R. Boissier et. al.: *Integrated and Distributed Manufacturing, a Multi-Agent Perspective*. In: Proceedings of 3rd Workshop on European Scientific and Industrial Collaboration, Entschede, 2001, S. 145-156.

[BCTR04] F. Bellifemine, G. Caire, T. Trucco, G. Rimassa: *JADE Programmer's Guide*. University of Parma,. TILab SpA, 2004. http://jade.tilab.com/doc/

[Bech93] M. v. Bechtolsheim: *Agentensysteme: Verteiltes Problemlösen mit Expertensystemen*. Braunschweig: Vieweg-Verlag, 1993.

[Beuc05] D. Beuche: *Softwareproduktlinien-Entwicklung mit Merkmalmodellen*. In: OBJEKTSpektrum (2005), H. 6, Troisdorf: SIGS Verlag, S. 74-79.

[BeWe03] S. Becker, B. Westfechtel: *Integrationswerkzeuge für verfahrenstechnische Entwicklungsprozesse*. GMA-Fachtagung Engineering in der Prozessindustrie, Frankfurt In: VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (Hrsg.), VDI-Berichte Nr. 1684, Düsseldorf: VDI-Verlag, 2002, S. 103-112.

[Booc94] G. Booch: *Object-Oriented Analysis and Design with Applications*. Addison-Wesley, Reading, 1994.

[BSM+03] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, T. Grose: *Eclipse Modeling Framework*. 1. Aufl., Boston: Addison-Wesley. 2003.

[BUILDER] *AGENTBUILDER*
http://www.agentbuilder.com

[Buss98] S. Bussmann: *Autonome und kooperative Produktionssysteme*. In: Informationstechnik und Technische Informatik it+ti 40 (1998), H. 4, S. 34-39.

[CAEX04] *CAEX Metamodell.* Version 1.0.1.
http://www.plt.rwth-aachen.de/xml/

[CGK+05] K. Chmiel, M. Gawinecki, P. Kaszmarek, M. Szymczak, M. Prprzycki: *Efficiency of JADE agent platform*. JADE Workshop at International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005), Utrecht, 2005.

[CoPo02] M. Cossentino, C. Potts: *A case tool supported methodology for the design of multi-agent systems*. In: Proceedings of the 2002 International Conference on Software Engineering Research and Practice (SERP'02), Las Vegas, CSREA Press, 2002.

[CzEi00] K. Czarnecki, U. Eisenecker: *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, Reading, MA, 2000.

[Dann06] C. Dannegger: *Transportoptimierung in Echtzeit - Einsatz von Agentensystemen in der Logistik am Praxisbeispiel*. GMA-Fachtagung Automation - Erfolgsfaktor in allen Branchen, VDE Kongress, Aachen, 2006. In: VDE Fachtagungsberichte Band 2, Berlin: VDE-Verlag, 2006, S. 303-307.

[DaWi04] K. Dam, M. Winikoff: *Comparing agent-oriented methodologies*. In: Proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS 2003), Melbourne, Paolo Giorgini, Brian Henderson-Sellers, Michael Winikoff (Hrsg.), Lecture Notes in Computer Science Vol. 3030, 2004. S. 78 – 93.

[Denc03] K. Dencovski: *Agentenorientierte Modellierung von Engineeringprozessen am Beispiel des Werkzeugs Comos PT*. Studienarbeit Nr. 1914, IAS, Universität Stuttgart, 2003.

[Denc05] K. Dencovski: *Integration and Evaluation of Agent-based Engineering Concepts in a Real Engineering Environment*. Diplomarbeit Nr. 1999, IAS, Universität Stuttgart, 2005.

[Dete03] S. Deter: *Plug-and-Participate for Limited Devices in the Field of Industrial Automation*. Dissertation, Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, 2003

[Dris05] S. Drissi: *Entwicklung von Konzepten zur praktischen Formalisierung, Speicherung und Inferenz von Engineering-Know-how*. Diplomarbeit Nr. 2026, IAS, Universität Stuttgart, 2005.

[DrMu04] R. Drath, M. Fedai: *CAEX – ein neutrales Datenaustauschformat für Anlagendaten. Teil 1+2.* In: atp – Automatisierungstechnische Praxis 46 (2004), H. 2+3, S. 52-56, S. 22-27.

[Dujm02] S. Dujmovic: *Anwendungsentwicklung mit Komponentenframeworks in der Automatisierungstechnik*. Dissertation, IAS, Universität Stuttgart, 2002. IAS-Forschungsberichte, Bd. 1/2002.

[Eber05] S. Eberle: *Adaptive Internetanbindung von Feldbussystemen*. Dissertation, IAS, Universität Stuttgart, 2005. IAS-Forschungsberichte, Bd. 1/2005.

[EbGö04] S. Eberle, P. Göhner: *Softwareentwicklung für eingebettete Systeme mit strukturierten Komponenten. Teil 1+2*. In: atp – Automatisierungstechnische Praxis 46 (2004), H. 3+4, S. 41-52, S. 61-73.

[EbGö04] S. Eberle, P. Göhner: *Softwareentwicklung für eingebettete Systeme mit strukturierten Komponenten. Teil 1+2*. In: atp – Automatisierungstechnische Praxis 46 (2004), H. 3+4, S. 41-52, S. 61-73.

[EKC+01] R. Evans, P. Kearney, G. Caire et. al.: *Methodology for agent-oriented software engineering*. Technical Information Final version (Project p907, deliverable 3), European Institute for Research and Strategic Studies in Telecommunications (EURESCOM), September 2001.

http://www.upv.es/sma/teoria/metodologias/articulos/D3finalReviewed.pdf

[Eppl03] U. Epple: *Austausch von Anlagenplanungsdaten auf der Grundlage von Metamodellen*. In: atp – Automatisierungstechnische Praxis 45 (2003), H. 7, S. 61-70.

[Etsc94] K. Etschberger: *Controller-Area-Network: Grundlagen, Protokolle, Bausteine, Anwendungen*. München: Carl Hanser Verlag, 1994.

[FIPA02a] *FIPA Abstract Architecture Specification. Standard Nr.1, Version L.* Foundation for Intelligent Physical Agents (FIPA), 2004.
http://www.fipa.org

[FIPA02b] *FIPA ACL Message Structure Specification, Standard Nr.61, Version G.* Foundation for Intelligent Physical Agents (FIPA), 2002.
http://www.fipa.org

[FIPA04] *FIPA Agent Management Specification, Standard Nr. 23, Version K.* Foundation for Intelligent Physical Agents (FIPA), 2004.
http://www.fipa.org

[FIPAOS] *FIPA-OS Agent Toolkit*
http://sourceforge.net/projects/fipa-os

[Flei03] W. Fleisch: *Validierung komponentenbasierter Software für Echtzeitsysteme.* Dissertation, IAS, Universität Stuttgart, 2003. IAS-Forschungsberichte, Bd. 2/2002.

[Fodo02] G. Fodor: *Fault Detection, Isolation and Recovery (FDIR) in Distributed Agent Systems – A Domain Independent Approach*. In: Prodeedings of Net.Objectdays 2002, T. Dittmar et. al. (Hrsg.), tranSIT GmbH, 2002.

[FoHi02] P. Foyer, J. Higgins: *Concurrent Engineering: Making it work*. In: Engineering Management Journal 12 (2002), H. 5, S. 243-248.

[FrGr96] S. Franklin, A. Graesser: *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents*. In: Intelligent Agents III – Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages (ATAL'96), M. Wooldridge, N. Jennings (Hrsg.), Lecture Notes in Artificial Intelligence Vol. 1193, Berlin: Springer-Verlag, 1996, S. 21–36.

[Fuch00] K. Fuchs-Kittowski: *Wissens-Ko-Produktion – Organisationsinformatik: Verarbeitung, Verteilung und Entstehung von Informationen in kreativ-lernenden Organisationen*. In: Organisationsinformatik und Digitale Bibliothek in der Wissenschaft: Wissenschaftsforschung Jahrbuch 2000, K. Fuchs-Kittowski, H. Parthey, W. Umstätter, R. Wagner-Döbler (Hrsg.), Berlin: Gesellschaft für Wissenschaftsforschung, 2000, S. 9 – 88.

[GaBe03] E. Gamma, K. Beck: *Contributing to Eclipse, Principles, Patterns, and Plug-Ins*. 1. Aufl. Boston: Addison-Wesley. 2003.

[Gau05] J. Gausemeier: *Designing Tomorrow's Mechanical Engineering Products*. ICICT 2005, Cairo, Egypt, 2005.

[Gehm99] A. Gehmeyr: *Multiagentensysteme*. Java Spektrum (1999), H. 1, Troisdorf: SIGS Verlag, S. 26-34.

[GeSc96] A. Gerhardt, H. Schmied: *Externes Simultanes Engineering*. Springer-Verlag: Berlin - Heidelberg, 1996.

[GHJV95] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, Reading, MA, 1995.

[GMP02] F. Giunchiglia, J. Mylopoulos, A. Perini: *The Tropos software development methodology: Processes, Models and Diagrams*. In: Agent-Oriented Software Engineering III: Third International Workshop (AOSE 2002), F. Giunchiglia, J. Odell, G. Weiß (Hrsg.), Lecture Notes in Computer Science Vol. 2585, Berlin: Springer-Verlag, 2002. S. 162-173.

[Göhn98] P. Göhner: *Komponentenbasierte Entwicklung von Automatisierungssystemen.* VDE-GMA Kongress 1998, Ludwigsburg. In: VDI-Berichte Nr. 1397, Düsseldorf: VDI-Verlag, 1998, S. 513-521.

[Gunz03] M. Gunzert: *Komponentenbasierte Softwareentwicklung für sicherheitskritische eingebettete Systeme*. Dissertation, IAS, Universität Stuttgart, 2003. IAS-Forschungsberichte, Bd. 3/2003.

[GUW04] P. Göhner, P. de A. Urbano, T. Wagner: *Softwareagenten - Einführung und Überblick über eine alternative Art der Softwareentwicklung. Teil III: Agentensysteme in der Automatisierungstechnik: Aufbau, Struktur und Realisierung an einem Anwendungsbeispiel*. In: atp – Automatisierungstechnische Praxis 46 (2004), H. 2, S. 42-51.

[HaKi86] P. Harmon, D. King: *Expertensysteme in der Praxis - Perspektiven, Werkzeuge, Erfahrungen*. 2. Aufl., München: Oldenbourg Verlag, 1987.

[Happ04] M. Happacher: *Die Zeit der Agenten*. Editorial,Computer & Automation (2004), H. 3, S. 3.

[HeGi05] B. Henderson-Sellers, P. Giorgini (Hrsg.): *Agent-Oriented Methodologies*. Idea Group Publishing, 2005

[HeHe90] W. Herden, H.-W. Hein (Hrsg.): *Kurzlexikon Wissensbasierte Systeme*. München: Oldenbourg Verlag, 1990.

[Herz92] G. Herzwurm: *Möglichkeiten und Grenzen des Einsatzes wissensbasierter Systeme im Computer Aided Software Engineering (CASE)*. Dissertation, Lehrstuhl für Informatik, Universität Köln, 2002.

[Hild05] H. Hild: *Analysis and conception of a demonstration model for agent-based engineering*. Diplomarbeit Nr. 1983, IAS, Universität Stuttgart, 2005.

[HoLi06] M. ten Hompel, D. Liekenbrock: *Material Handling – Automatisiertes Handling von Stückgütern in der Prozessindustrie*. atp – Automatisierungstechnische Praxis 48 (2006), H. 2, S. 62-66.

[HoRi99] M. Son Hoang, P. Rieger: *Komponentenbasierte Automatisierungssoftware.* München: Carl Hanser Verlag, 1999.

[HWH04] B. Heintel, T. Wagner, G.Hohn: *Agentenorientiertes dezentrales Energie- und Funktionsmanagement für Kfz*. In: Begleittexte zum Entwicklerforum Kfz-Elektronik, Stuttgart, C. Grote, E. Elster (Hrsg.), WEKA Fachzeitschriften-Verlag, 2004, S. 171-180.

[IGG99] C. Iglesias, M. Garijo, J. Gonzalez: *A Survey of Agent-Oriented Methodologies*. In: Intelligent Agents V – Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98), J.P. Müller, M.P. Singh, A.S. Rao (Hrsg.) Lecture Notes in Artificial Intelligence Vol. 1555. Berlin: Springer-Verlag, 1999, S. 317-330.

[INNO03] Innotec GmbH, *Comos PT Handbuch*. HB701D10, Oktober 2003.

[Int03a] *Entwicklung des Automatisierungsweltmarktes für die Prozessindustrien bis 2010*. Studie, INTECHNO Consulting, 2003. http://www.intechnoconsulting.com

[Int03b] *Entwicklung des Weltmarktes für Anlagenbezogene Produkte und Dienstleistungen in den Prozessindustrien bis 2010.* Studie, INTECHNO Consulting, 2003. http://www.intechnoconsulting.com

[ISO10303] ISO 10303: *Industrial automation systems and integration - Product data representation and exchange.* International Organisation for Standardization, 1994.

[JACK] *JACK Intelligent Agents*. The Agent Oriented Software Group. http://www.agent-software.com.au

[Jack99] P. Jackson: *Introduction to Expert Systems*. 3. Aufl., Harlow, England: Addison Wesley Longman, 1999.

[JADE] *JADE (Java Agent DEvelopment Framework*)
http://jade.tilab.com/

[JATLITE] *JATLite (Java Agent Template, Lite)*
http://java.stanford.edu

[Jenn00] N. Jennings: *On agent-based software engineering*. Artificial Intelligence 117 (2000), Elsevier Press, S. 277-296.

[JoFo88] R. Johnson, B. Foote: *Designing Reusable Classe*s. In: Journal of Object-Oriented Programming, 1 (1998), H. 2, S. 22-35.

[Kend99] E. Kendall: *Role Modeling for Agent System Analysis, Design, and Implementation*. In: Proceedings of the first International Symposium on Agent Systems and Applications (ASA'99), Palm Springs, 1999. S. 204.

[Kläg93] R. Kläger: *Modellierung* von *Produktanforderungen* als Basis für Problemlösungsprozesse in intelligenten Konstruktionssystemen. Dissertation, Institut für Rechneranwendung in Planung und Konstruktion, Universität Karlsruhe, 1993.

[KlLü03] E. Klemm, A. Lüder: *Agentenbasierte Flexibilisierung der Produktion bei Verwendung von vorhandenen Steuerungssystemen*. In: atp – Automatisierungstechnische Praxis 45 (2003), H. 4, S. 66-75.

[KMC+03] B. Köppen-Seliger, T. Marcu, M. Capobianco, S. Gentil, M. Albert, S. Latzel: *MAGIC: Ein integriertes Konzept zum Diagnosemanagement und Operator Support*. In: atp - Automatisierungstechnische Praxis 45 (2003), H. 5, S. 79-83.

[KNS99] K. Kabitzsch, J. Naake, G. Stein: *Mobile Agents for Tele-Diagnosis of Automation Systems*. In: Proceedings of the IFAC Conference on Telematics Applications in Automation and Robotics (TA2001), Weingarten, K. Schilling, H. Roth (Hrsg.), Elsevier Press, 2001, S. 123-127.

[KoCa00] F. Kon, R. Campbell: *Dependence Management in Component-Based Distributed Systems.* In: IEEE Concurrency 8 (2000), H. 1, S. 26-36.

[Konr99] K. Konrad: *Konzeption eines agentenbasierten, aktiven Beratungssystems für die objektorientierte Entwicklung von Softwaresystemen bei Automatisierungsprojekten.* Dissertation, IAS, Universität Stuttgart, 1999.

[KQML93] *Specification of the KQML Agent-Communication Language, Draft Standard*. DARPA Knowledge Sharing Initiative External Interfaces Working Group, 1993.
http://www.cs.umbc.edu/kqml/

[Krat05] W. Kratou: *Werkzeug zur Unterstützung des komponentenbasierten Anlagenengineering mit Anbindung an ein Agentensystem*. Studienarbeit Nr. 2038, IAS, Universität Stuttgart, 2005.

[Krue02] C. Krueger: *Variation Management for Software Production Lines*. In: Proceedings of the 2nd International Software Product Line Conference, San Diego, California. 2002, S. 37-48.

[Kunz06] S. Kunz: *Integration eines Agentensystems in ein Engineeringwerkzeug*. Diplomarbeit Nr. 2090, IAS, Universität Stuttgart, 2006.

[LaGö99a] R. J. Lauber, P. Göhner: *Prozessautomatisierung I*. 3. Aufl., Berlin, Heidelberg: Springer-Verlag, 1999.

[LaGö99b] R. J. Lauber, P. Göhner: *Prozessautomatisierung II*, 1. Aufl., Berlin, Heidelberg: Springer-Verlag, 1999.

[Lanz99] M. Lanza: *Entwurf der Systemunterstützung des verteilten Engineering mit Axiomatic Design*. Dissertation, Institut für Werkzeugmaschinen und Betriebstechnik, Universität Karlsruhe, 1999. WBK-Forschungsberichte Bd. 95.

[Laub93] R. Lauber: *Artificial Intelligence Techniques in Real-Time Control Systems*. Proceedings of the 12th World IFAC Congress, Sidney, 1993.

[LBB+05] U. Löwen, R. Bertsch, B. Böhm, S. Prummer, T. Tetzner: *Systematisierung des Engineerings von Industrieanlagen.* In: atp – Automatisierungstechnische Praxis 47 (2005), H. 4, S. 54-61.

[LBP03] M. Luck, P. McBurney, C. Preist: *Agent Technology: Enabling Next Generation Computing – A Roadmap for Agent-Based Computing*. Version 1.0, AgentLink II, European Network of Excellence for Agent-Based Computing, 2003.
http://www.agentlink.org

[LEAP] *LEAP (Lightweight Extensible Agent Platform)*
http://leap.crm-paris.com

[Lehm04] E. Lehmann: *Grundlagen der Wissensverarbeitung und des Sprachverstehens*. Vorlesung, Institut für Intelligente Systeme, Universität Stuttgart, 2004.

[Levi05] P. Levi: *Multiagentensysteme in der Robotik*. 1. Workshop Agenten in der Automatisierungstechnik, IAS, Universtität Stuttgart, 2004.

[LGB+05] A. Liefeldt, G. Gutermuth, P. Beer, S. Basenach, R.Alznauer: *Effizientes Engineering – Begleitende Fortschrittskontrolle großer Projekte der Automatisierungstechnik*. In: atp – Automatisierungstechnische Praxis 47 (2005), H. 7, S. 60-63.

[Lin01] J. Lind: *Iterative software engineering for multiagent systems: The MASSIVE method*. Lecture Notes in Computer Science, Vol. 1994. Berlin: Springer-Verlag, 2001.

[Löwe02] U. Löwen: *Informationstechnologie als Enabler für effizientes Anlagenengineering.* In: Ringvorlesung Verfahren der Softwaretechnik, IAS, Universität Stuttgart, WS 2002/03.

[LSTS] *LS/TS (Living Systems Technology Suite).* Whitestein Technologies.
http://www.whitestein.com/pages/solutions/ls_ts.html

[Luce02] V. Ferreira de Lucena: *Flexible Web-based Management of Components for Industrial Automation. Dissertation*, IAS, Universität Stuttgart, 2002. IAS-Forschungsberichte, Bd. 4/2002.

[Lunz94] J. Lunze: *Künstliche Intelligenz für Ingenieure: Band 1 Methodische Grundlagen und Softwaretechnologie*. München: Oldenbourg Verlag, 1994.

[MaNa03] W. Marquardt, M. Nagel: *Arbeitsprozessorientierte Unterstützung verfahrenstechnischer Entwicklungsprozesse*. In. atp – Automatisierungstechnische Praxis 45 (2003), H. 3, S. 52-58.

[Mang02] E. Mangina: *Review of Software Products for Multi-Agent Systems*. Technical Report. AgentLink II, European Network of Excellence for Agent-Based Computing, 2002
http://www.agentlink.org

[MaWa05] M. Maurmaier, T. Wagner: *Development of embedded software systems with structured components and active composition support*. In: Proceedings of 3rd Workshop on Object-oriented Modeling of Embedded Real-Time Systems (OMER3), Paderborn, 2005.

[MDG+04] B. Moore, D. Dean, A. Gerber, G. Wagenknecht, P. Vanderheyden: *Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework*. IBM Redbook, 1. Aufl., International Business Machines Corporation, 2004.
www.ibm.com/redbooks

[MMYA02] H. Mili, A. Mili, S. Yacoub und E. Addy: *Reuse-Based Software Engineering: Techniques, Organization, and Controls*. Wiley, New York, NY, 2002.

[Muba04] H. Mubarak: *Agentenorientiertes Konzept für die Wiederverwendung von Know-how bei Engineeringprozessen*. Diplomarbeit Nr. 1935, IAS, Universität Stuttgart, 2004.

[Müll04] *W. Müller: Software-Agenten – Nutznießer von Ontologien als Wissensrepräsentation.* visIT (2004), H. 2, Fraunhofer Institut für Informations- und Datenverarbeitung, S. 12-13.

[OMG04] *Unified Modeling Language Specification V2.0.* Object Management Group (OMG), 2004
http://www.uml.org/

[Paru98] H. v. D. Parunak: *Practical and industrial applications of agent-based systems*. Environmental Research Institute of Michigan (ERIM), 1998.
http://www.cs.umbc.edu/agents/

[PaWi02] L. Padgham, M. Winiko: *Prometheus: A pragmatic methodology for engineering intelligent agents*. In: Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, Seattle, 2002, S. 97-108

[PBC01] H. v. D. Parunak, A. Baker, S. Clark: *The AARIA Agent Architecture: From Manufacturing Requirements to Agent-Based System Design*, In: Integrated Computer-Aided Engineering 8 (2001), H. 1, S. 45-58.

[PBL05] A. Pokahr, L. Braubach, W. Lamersdorf: *Agenten: Technologie für den Mainstream?* In: it – Information Technology 47 (2005), H. 5, S. 300-307.

[PDS03] Y. Penya, S. Detter, T. Sauter: *Plug-and-Participate Functionality for Agent-Enabled Flexible Automation*. In: Proceedings of 7th IEEE International Conference on Intelligent Engineering Systems, INES'03, 2003, S. 136ff.

[Perm90] G. Permantier: *Verfahren der Wissensrepräsentation bei der Entwicklung wissensbasierter Systeme für die Automatisierungstechnik*. Dissertation, IAS, Universität Stuttgart, 1990.

[PlWe99] R. Plösch, R. Weinreich: *An Agent-Based Environment for Remote Diagnosis, Supervision and Control*. In: Proceedings of the International Computer Science Conference (ICSC 99), L. Chi-Kwong Hui, D. Lun Lee (Hrsg.), Lecture Notes in Computer Science Vol. 1749, Berlin: Springer-Verlag, 1999, S. 385-392.

[Pöss06] Possel-Dölken, Frank: *Projektierbares Multiagentensystem für die Ablaufsteuerung in der flexibel automatisierten Fertigung.* Dissertation, Fraunhofer Institut für Produktionstechnologie, Rheinisch-Westfälische Technische Hochschule Aachen, 2006.

[PROF05] *EDDL Specification; Specification for PROFIBUS Device Description and Device Integration.* Vol. 2, PROFIBUS International, 2005. http://www.profibus.com

[Pupp91] F. Puppe: *Einführung in Expertensysteme*. 2. Aufl., Studienreihe Informatik, Berlin: Springer-Verlag, 1995.

[Raff05] W. Raffel: *Agentenbasierte Simulation als Verfeinerung der Diskreten-Ereignis-Simulation unter besonderer Berücksichtigung des Beispiels Fahrerloser Transportsysteme.*Dissertation, Institut für Informatik, Freie Universität Berlin, 2005

[RHA02] G. Rauprich, C. Haus, W. Ahrens: *PLT-CAE - Integration in gewerkeübergreifendes Engineering und Plant-Maintainance*, In: atp - Automatisierungstechnische Praxis 44 (2002), H. 2, S. 50-62.

[Rich03] H. Richter: *Internationales Concurrent Engineering: Anforderungen an die Engineering-Systeme aus Sicht des Anlagenbauers*. In: atp – Automatisierungstechnische Praxis 45 (2003), H. 3, S. 36-39.

[Rodi02] H.-J. Rodies: *Planungswerkzeuge aus Sicht des Anlagenbaus*. In: atp – Automatisierungstechnische Praxis 44 (2002), H. 1, S. 40-44.

[Rude98] S. Rude: *Wissensbasiertes Konstruieren.* Habilitationsschrift, Universität Karlsruhe, Shaker-Verlag, 1998.

[RuNo95] S. Russel, P. Norvig: *Artificial Intelligence: A Modern Approach*. New Jersey: Prentice Hall, 1995.

[ScFa05] T. Schmidberger, A. Fay: *Automatisiertes Engineering von Prozessleitsystem-Funktionen*. In: atp – Automatisierungstechnische Praxis 47 (2005), H. 2, S. 45-51.

[Sche02] A. Scheuermann: *Kosten an der Wurzel packen*. In: Chemie Technik 31 (2002), H. 7, S. 10-12.

[ScHu87] P. Schupp, C.T. Nguyen Huu: *Expertensystem-Praktikum*. Berlin: Springer-Verlag, 1987.

[SFB614] *SFB 614 - Selbstoptimierende Systeme des Maschinenbaus*. Selbstdarstellung.
www.sfb614.de

[SFDH06] T. Schmidberger, A. Fay, R. Drath, A. Horch: *Von Anlagenstrukturinformation automatisch zum Asset Management*. In: atp – Automatisierungstechnische Praxis 48 (2006), H. 6, S. 54-61.

[ShNo99] W. Shen, D.H. Norrie,: *Agent-Based Systems for Intelligent Manufacturing: A State of-the-Art Survey*. In: Knowledge and Information Systems, 1 (1999), H. 2, S. 129-156.

[Simo96] H. Simon: *The Sciences of the Artificial*, MIT Press, Cambridge, MA, 1996.

[SMBV03] I. Sora, F. Matthijs, Y. Berbers, P. Verbaeten: *Automatic Composition of Systems from Components with Anonymous Dependencies.* In: Kluwer International Series in Engineering and Computer Science, T. D'Hondt (Hrsg.), Vol. 732 (2003), S. 154-169.

[Stec03] T. Stecker: *Steuerungsentwurf für eine Produktionsanlage auf Basis von Agenten*. Studienarbeit Nr. 1908, IAS, Universität Stuttgart, 2003.

[StSh05] A. Sturm, O. Shehory: *A Framework for Evaluating Agent-Oriented Methodologies.* In: Proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS 2003), Melbourne, P. Giorgini, B. Henderson-Sellers, M. Winikoff (Hrsg.), Lecture Notes in Computer Science Vol. 3030, 2004. S. 94 – 109.

[SuBu02] K. Sundermeyer, S. Bussmann: *Einführung der Agententechnologie in einem produzierenden Unternehmen - Ein Erfahrungsbericht*. In: Wirtschaftsinformatik 43 (2001), H. 2, S. 135-142.

[Szyp98] Clemens Szyperski: *Component Software - Beyond Object-Oriented Programming.* Reading, MA: Addison-Wesley Longman, 1998.

[TöWo01] H. Tönshoff, P.-O. Woelk: *Flexible Process Planning and Production Control Using Co-operative Agent Systems*. In: International Conference on Competitive Manufacturing (COMA '01), Stellenbosch, 2001.

[Umst98] W. Umstätter: *Die Messung von Wissen*. Nachrichten für Dokumentation. 49 (1998), H. 4, Deutsche Gesellschaft für Informationswissenschaft und Informationspraxis (DGI), S. 221-224.

[Urba03] P. de A. Urbano: *Evaluation Teleperm XP. Final Report*. Interner Projektabschlussbericht, IAS, Universität Stuttgart, 2003.

[UWG03] P. de A. Urbano, T. Wagner, P. Göhner: *Softwareagenten - Einführung und Überblick über eine alternative Art der Softwareentwicklung. Teil II: Agentensysteme in der Automatisierungstechnik: Modellierung eines Anwendungsbeispiels*. In: atp - Automatisierungstechnische Praxis 45 (2003), H. 11, S. 57-65.

[UWG+04] P. de A. Urbano, T. Wagner, P. Göhner, U. Katzke, B. Vogel-Heuser: *Introducing Reliability and Real-Time Features in Flexible Agent Oriented Automation Systems*. In Proceedings of the 2nd IEEE International Conference on Industrial Informatics (INDIN 04), Berlin, R. Schoop, A. Colombo, R. Bernhardt, G. Schreck (Hrsg.), IEEE Industrial Electronics Society, 2004, S. 89-94.

[VDE00] *VDE-Studie 2000: Ingenieure der Elektro- und Informationstechnik*. Verband der Elektrotechnik Informationstechnik e.V. http://www.vde.de

[W3C04] *Extensible Markup Language (XML) Specification V1.0 (Third Edition)*, World Wide Web Consortium (W3C), XML Core Working Group, 2004 http://www.w3.org/XML

[W3C06] *XML Schema Language Specification, Draft Standard V1.1*, World Wide Web Consortium (W3C), XML Schema Working Group, 2006 http://www.w3.org/XML/Schema

[Wagn05] T. Wagner: *Agentenunterstütztes Engineering von Automatisierungsanlagen*. 7. GMA-Kongress 2005, Düsseldorf, In: VDI-Berichte Nr. 1883, VDI Verlag, 2005, S. 559-567.

[WaGö05] T. Wagner, P. Göhner: *Aufbau von Agentensystemen zur Unterstützung komponentenbasierter Entwicklungsprozesse*. In: it - information technology 47 (2005), H. 1, S. 5-12.

[Wang06] Wang, Ju: *XML-basierte Anbindung eines CAE-Editors an ein Agentensystem*. Diplomarbeit Nr. 2084, IAS, Universität Stuttgart, 2006.

[WaWa97] H. Wang, C. Wang: *Intelligent Agents in the Nuclear Industry*. IEEE Computer 30 (1997), H. 11, S. 28 -31.

[WaWe05] W. Wahlster, C. Weyrich: *Forschen für die Internet-Gesellschaft: Trends, Technologien, Anwendungen*. Ergebnisse des Symposiums 2005 des Feldafinger Kreises, 2005. http://www.feldafinger-kreis.de

[WDDD05] M. Weigt, Z. Djordjevic, O. Drumm, U. Döbrich: *Elektronisches Datenblatt für Softwarebausteine.* In: atp – Automatisierungstechnische Praxis 48 (2006) H. 1, S. 48-55.

[Weiß01] G. Weiß: *Agentenorientiertes Software Engineering*. In: Informatik Spektrum 24 (2001), H. 2, S. 98-101.

[Weiß02] G. Weiß: *Agent Orientation in Software Engineering*. Knowledge Engineering Review 16 (2002), H. 4, S. 349-373.

[WGU03] T. Wagner, P. Göhner, P. de A. Urbano: *Softwareagenten - Einführung und Überblick über eine alternative Art der Softwareentwicklung. Teil I: Agentenorientierte Softwareentwicklung*. In: atp - Automatisierungstechnische Praxis 45 (2003), H. 10, S. 48-57.

[Wiki06a] Wikipedia, die freie Enzyklopädie – *Wissen*. Wikimedia Foundation, 2006. http://de.wikipedia.org/wiki/Wissen

[Wiki06b] Wikipedia, die freie Enzyklopädie – *Implikation*. Wikimedia Foundation, 2006.
http://de.wikipedia.org/wiki/Implikation

[WJK00] M. Wooldridge, N. Jennings, D. Kinny: *The gaia methodology for agent-oriented analysis and design*. In: Autonomous Agents and Multi-Agent Systems 3 (2000), H. 3, S. 285-312.

[WoDe01] M. Wood, S. DeLoach: *An overview of the multiagent systems engineering methodology*. In: First International Workshop on Agent-Oriented Software Engineering (AOSE 2000), Ciancarini and Wooldridge (Hrsg.), Lecture Notes in Computer Science Vol. 1957, Berlin: Springer-Verlag, 2001. S. 207-222.

[WoJe95] M. Wooldridge, N. Jennings: *Agent Theories, Architectures, and Languages: A Survey*. Workshop on Agent Theories, Architectures, Amsterdam, 1994 & Languages In: Intelligent Agents, Wooldridge and Jennings (Hrsg.), Lecture Notes in Artificial Intelligence Vol. 890, Berlin: Springer-Verlag, 1995, S. 1-22.

[Wool97] M. Wooldridge: *Agent-based Software Engineering*. In: IEEE Proceedings on Software Engineering 144 (1997), H. 1, S. 26-37.

[Wu04] C. G. Wu: *Modeling Rule-Based Systems with EMF*. Eclipse Corner Articles, Eclipse Foundation, 2004.
www.eclipse.org

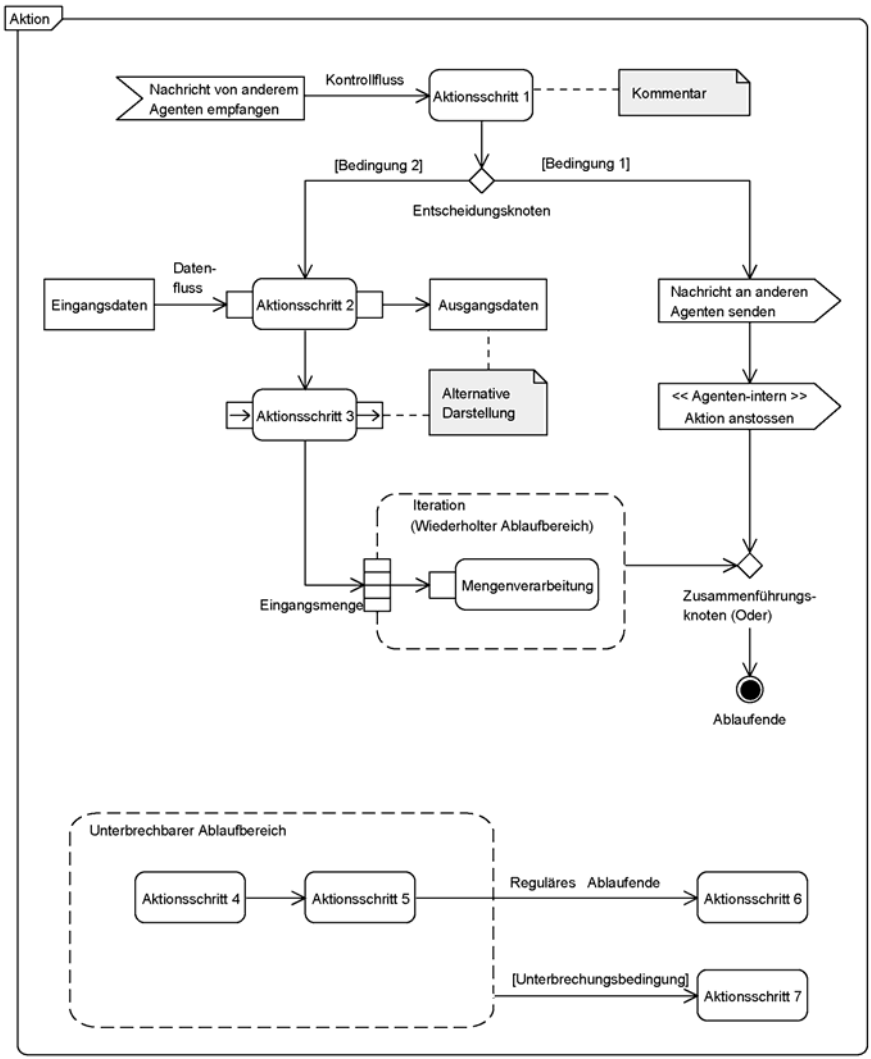
[Xu06] K. Xu: *Strategien zur automatischen Verbindung von Anlagenkomponenten durch Softwareagenten*. Diplomarbeit Nr. 2085, IAS, Universität Stuttgart, 2006.

[Yilm05] O. Yilmaz: *Konzeption und Realisierung von Softwareagenten zur Unterstützung des Anlagenengineerings.* Diplomarbeit Nr. 2048, IAS, Universität Stuttgart, 2005.

[Zeid01] C. Zeidler: *Industrial IT mit Komponententechnologien*. In: Ringvorlesung Verfahren der Softwaretechnik, IAS, Universität Stuttgart, WS 2000/01.

[ZEUS] *ZEUS Agent Toolkit.* British Telecom
http://labs.bt.com/projects/agents/zeus/

Anhang A Notationsübersicht Aktivitätsdiagramme



Anhang B Regeln des gewerkspezifischen Teilmodells Netzwerkplan

Regel 1 (Auswahlregel):

```
WENN  eine Komponenteninstanz Anschlusspunkte vom
      Schnittstellentyp „Signal“ besitzt,
UND   diese aktiviert sind,
DANN  füge die Anschlusspunkte der Vernetzungsliste hinzu.
```

Bedeutung: Alle zu vernetzenden Anschlusspunkte der Komponenteninstanzen sind zu ermitteln und eine entsprechende Vernetzungsliste zu erstellen.

Regel 2 (Gruppierungsregel):

```
WENN  zwei Komponenteninstanzen über ihre Anschlusspunkte vom
      Schnittstellentyp „Materialfluss“ verbunden sind,
DANN  ordne die Komponenteninstanzen in der Vernetzungsliste.
```

Bedeutung: Um möglichst kurze Signalleitungen zu erhalten, soll die räumliche Nähe der Komponenteninstanzen bei der Netzwerkplanung berücksichtigt werden.

Regelsatz 3 (allgemeine Zuordnungsregeln):

```
WENN  ein Anschlusspunkt in der Vernetzungsliste noch keinem
      E/A-Modul im Netzwerkplan zugeordnet ist,
DANN  erstelle eine Zuordnung.
WENN  noch kein E/A-Modul im Netzwerkplan enthalten ist,
ODER  kein E/A-Modul im Netzwerkplan noch freie Signale aufweist
DANN  füge ein neues E/A-Modul dem Netzwerkplan hinzu.
```

Bedeutung: Der Netzwerkplan wird durch Zuordnung der Signaleingänge und -ausgänge der Komponenteninstanzen an die Feldbus-E/A-Module erstellt. Dabei werden die verfügbaren Signale der E/A-Module möglichst vollständig belegt.

Regel4 (spezielle Zuordnungsregel):

```
WENN  ein Anschlusspunkt einer Komponenteninstanz bereits einem
      E/A-Modul zugeordnet ist,
DANN  ordne auch allen anderen in der Vernetzungsliste
      enthaltenen Anschlusspunkte dieser Komponenteninstanz
      demselben E/A-Modul zu.
```

Bedeutung: Die Anschlüsse einer Komponenteninstanz sollen nicht auf mehrere E/A-Module verteilt werden.

Lebenslauf

Persönliche Daten

22.11.1973	geboren in Gerabronn

Schulbildung

1980 – 1984	Grundschule, Blaufelden
1984 – 1990	Realschule, Blaufelden, Abschluss Mittlere Reife
1990 – 1993	Technisches Gymnasium, Crailsheim, Abschluss Abitur

Zivildienst

1993 – 1994	Betreuung von geistig behinderten Kindern und Jugendlichen, Heim Sonnenhof e.V., Schwäbisch Hall

Studium

1994 – 2001	Studium der Elektro- und Informationstechnik an der Universität Stuttgart, Studienmodell Informationstechnik
	Fachpraktikum und Nebentätigkeit als Werkstudent bei der Hewlett Packard GmbH, SW SDE Germany, Support Zentrum in Ratingen
06.03.2001	Abschluss als Diplom-Ingenieur

Berufstätigkeit

April 2001 – September 2006	Wissenschaftlicher Assistent am Institut für Automatisierungs- und Softwaretechnik der Universität Stuttgart
seit Januar 2007	Mitarbeiter der Siemens AG, Corporate Technology, Fachzentrum Systems Engineering in Erlangen