Kooperation in Multiagentensystemen

Verwendung der BDI-Konzepte in Gruppen

Diplomarbeit an der
Fakultät für Mathematik, Informatik und Naturwissenschaften der
Universität Hamburg im Department Informatik

Vorgelegt von Christian Poulter
mail@poulter.de
http://www.poulter.de

Dezember 2007

Betreuung:
Prof. Dr. Winfried Lamersdorf

Fachbereich Informatik
Arbeitsbereich Verteilte Systeme
und Informationssysteme
Universität Hamburg
Vogt-Kölln-Str. 30
D-22527 Hamburg

Dr. Daniel Moldt

Fachbereich Informatik
Arbeitsbereich Theoretische
Grundlagen der Informatik
Universität Hamburg
Vogt-Kölln-Str. 30
D- 22527 Hamburg

Inhaltsverzeichnis

1 Einleitung....1
1.1 Motivation....2
1.2 Zielsetzung....3
1.3 Vorgehensweise....4
1.4 Aufbau der Arbeit....4
2 Grundlagen....7
2.1 Agententechnologie....7
2.1.1 Agenten....7
2.1.2 Reaktive und deliberative Agenten....10
2.1.3 BDI-Agenten....10
2.1.4 Die Umgebung....13
2.1.5 Multiagentensysteme....13
2.1.6 Kommunikation....14
2.1.7 Agentenorientiere Softwareentwicklung....16
2.1.8 Zusammenfassung....18
2.2 Das Agentenframework Jadex....19
2.2.1 Die Jadex-Architektur....19
2.2.2 Explizite Darstellung von Zielen....21
2.2.3 Die Easy Deliberation Strategy....21
2.2.4 Jadex BDI-Interpreter....22
2.2.5 Sprache....23
2.2.6 Zusammenfassung....24
2.3 Gruppen....24
2.3.1 Gruppen, Teams und Organisationen....25
2.3.2 Bildung einer Gruppe....26
2.3.3 Beschreibung des Gruppenaufbaus....26
2.3.4 Rollen....27
2.3.5 Vorteile von Gruppenarbeit....29
2.3.6 Zusammenfassung....30
2.4 Koordination und Kooperation....30
2.4.1 Koordination in Multiagentensystemen....31
2.4.2 Formen der Koordination....32
2.4.3 Wettbewerb....33
2.4.4 Kooperation....34
2.4.5 Cooperative Distributed Problem Solving....36
2.4.6 Zusammenfassung....38
3 Koordinationstechniken....41

3.1 Joint Intentions....41
3.1.1 Das Modell von Cohen & Levesque....42
3.1.2 Rao, Georgeff und Sonnenberg....44
3.1.3 Kinny et al. (Planned Team Activity)....46
3.1.4 Jennings und Wooldridge....48
3.1.5 Zusammenfassung....49
3.2 AGR-Modell....50
3.2.1 Madkit....51
3.3 STEAM....51
3.3.1 Team Operatoren....52
3.3.2 Überwachung und Neuplanung....53
3.4 SimpleTeam in JACK....53
3.4.1 Rollen....54
3.4.2 Das Team-Konzept....54
3.4.3 Team-Pläne....55
3.4.4 Parallelität in Plänen....55
3.4.5 Teambeliefs....56
3.4.6 Automatisierte Teambildung....57
3.5 Weitere Techniken....57
3.5.1 Trader und Broker....57
3.5.2 Bekanntschaftsnetzwerke....58
3.5.3 Das Kontraktnetz....58
3.5.4 Gegenseitige Modellierung....59
3.5.5 Normen und soziale Gesetze....59
3.5.6 Emergente Aufgabenzuweisung....60
3.6 Zusammenfassung....60
4 Das BDI-Gruppenmodell....63
4.1 Funktionale Anforderungen....63
4.2 Vorstellung des BDI-Gruppenmodells....64
4.2.1 Gruppenziele....65
4.2.2 Gruppenpläne....65
4.2.3 Gruppenwissen....67
4.2.4 Kommunikationssystem....68
4.2.5 Rollen....69
4.2.6 Der Agent im BDI-Gruppenmodell....70
4.3 Die Gruppe als Teil einer anderen Gruppe....71
4.4 Das BDI-Gruppenmodell im Vergleich....71
4.5 Zusammenfassung....72
5 Ein Prototyp für JADEX....75
5.1 Die Gruppe....75
5.2 Rollen....76
5.3 Der GMS: Ein einfacher Matchmaker....77
5.4 Gruppenhandlungen und -Pläne....78
5.5 Gruppenwissen....83

5.6 GUIs für GMS und Gruppe....83
5.7 Das Kommunikationssystem....85
5.8 Einschränkungen....87
5.9 Zusammenfassung....88
6 Anwendungsbeispiele....89
6.1 Rechenaufgabe....89
6.1.1 Implementation....89
6.1.2 Zusammenfassung....92
6.2 GroupInGroup....92
6.2.1 Implementation....93
6.2.2 Zusammenfassung....94
6.3 BoxWorld....94
6.3.1 Implementation des Beispiels....96
6.3.2 Zusammenfassung....98
7 Zusammenfassung und Ausblick....101
7.1 Zusammenfassung....101
7.2 Ausblick....103

Literaturverzeichnis....105

Abbildungsverzeichnis

Abbildung 1: Das BDI-Modell....11
Abbildung 2: Architektur des Jadex-Interpreters....22
Abbildung 3: Verfahren der Kooperation und des Wettbewerbs....32
Abbildung 4: Schritte des CDPS-Prozesses....37
Abbildung 5: Das BDI-Gruppenmodell....64
Abbildung 6: Beispiel für ein Agent-Rolle-Gruppe Modell....70
Abbildung 7: Rollen und Positionen im BDI-Gruppenmodell....77
Abbildung 8: Gruppenbildung....78
Abbildung 9: Zusammenspiel von Gruppenzielen und Plänen....82
Abbildung 10: GUI des GMS....84
Abbildung 11: Positionen und Rollen....85
Abbildung 12: vorhandene Nachrichtentypen....86
Abbildung 13: Gruppen und Subgruppen....93
Abbildung 14: BoxWorld....95
Abbildung 15: Ziele des Träger-Agenten im BoxWorld-Beispiel....97
Abbildung 16: Ablauf der Gruppenhandlung in BoxWorld....98

1 Einleitung

Computersysteme und damit Soft- und Hardware werden mit der fortschreitenden Entwicklung immer komplexer. Um diese steigende Anforderung zu bewältigen, werden Systeme heute häufig in lose verbundene Einheiten aufgeteilt, die weitgehend unabhängig und durch einige wenige Interaktionen verknüpft sind. Es hat sich gezeigt, dass viele Probleme der realen Welt bereits in einer verteilten Form auftreten (Ferber 2001). Sie bestehen also aus einer Menge von Entitäten, die in einer bestimmten Umwelt handeln und miteinander interagieren. Wissen, Handlungsmöglichkeiten, Rechte und Pflichten kommen nur selten selbstständig an einer zentralen Stellen zusammen. Anstatt alle Ressourcen aufwendig zu integrieren, stellt die Idee der Problemlösung durch Kommunikation und Kooperation in einer Gruppe völlig neue Lösungsstrategien bereit.

Das Konzept eines *zentralen Programms* verliert hierdurch zunehmend an Bedeutung und weicht neuen Software-Engineering-Methoden (Pynadath et al. 2000). Die Agententechnologie wird in (Wooldridge 2002) als ein vielversprechender Ansatz betrachtet, um der steigenden Komplexität von Softwaresystemen zu begegnen. Das Konzept der Multiagentensysteme unterstützt die Sicherung wichtiger Eigenschaften wie Autonomie, Mobilität, Heterogenität, Offenheit und Intelligenz.

Sozialen Fähigkeiten eines Agenten spielen besonders in Multiagentensystemen eine entscheidende Rolle. Durch Interaktionen und Kooperation sind Agenten in der Lage, gemeinsam Lösungen zu erarbeiten oder Konfliktfälle zu lösen. Auch der Austausch von Wissen kann Vorteile mit sich bringen. Die beteiligten Agenten bilden durch ihre Beziehung zueinander Gruppen oder auf einer höheren Abstraktionsebene auch Organisationen. Der Agent muss dabei längst nicht mehr nur ein Computerprogramm sein, denn auch die natürliche Einbettung von Menschen innerhalb dieser Strukturen war bereits Thema mehrerer Arbeiten (z.B. in Pynadath u. Tambe 2003).

Die Verwendung einer gruppenorientierten Sichtweise ist im Bereich der Multiagentensysteme nicht neu und so wurden bereits unterschiedliche Modelle vorgestellt, die dem Entwickler bei der Spezifikation und Analyse von Softwareagenten unterstützen. Diese

Gruppenmodelle bieten die Möglichkeit, eine Ansicht einzunehmen, die von den Implementationsdetails eines einzelnen Agenten abstrahiert. Der Entwickler kann auf diese Weise das Zusammenspiel der Agenten auf einer höheren Ebene betrachten.

1.1 Motivation

Die Zusammenarbeit verschiedener Agenten in Gruppen wird als wichtiges Element von Multiagentensystemen betrachtet (Tambe 1997a, S. 83f.). Teams, Gruppen und Organisationen stellen natürliche Erweiterungen des Agentenparadigmas dar, wobei diese Konzepte auf einer semiautomatischen Ebene die Aktionen mehrerer Agenten koordinieren sollen. Dieser Ansatz bietet viele Vorteile wie beispielsweise den der Überwindung lokaler Einschränkungen, natürliche Repräsentation verteilter Probleme, Wiederverwendbarkeit von Komponenten, Erweiterung der Fähigkeiten des Gesamtsystems oder die Beschleunigung der Arbeit (Jennings 1993a). Ziel dieser Technologien ist es, Aktionen zu koordinieren oder auch bereits verteilt vorliegendes Expertenwissen bzw. Fähigkeiten nutzbar zu machen (Nwana et al. 1996). Auch die bereits erwähnte Reduzierung von Komplexität durch eine abstraktere Sichtweise macht diese Technologie interessant.

Entsprechende Techniken bringen aber auch einige Schwierigkeiten mit sich. So verfügen Agenten meist nur über eine unvollständige Weltsicht und können daher nicht alle Informationen über die Gruppenmitglieder besitzen. Auch der Umgang in Fehlersituationen (z.B. Ausfall eines Gruppenmitglieds) stellt eine Herausforderung dar. Die in der Arbeit (Kinny et al. 1994, S. 228) formulierten Fragen geben einen ersten Eindruck der Problemfelder:

1. **Gruppenbildung**: Wie kann eine passende Gruppe gefunden oder gegründet werden?
2. **Synchronisation**: Wie sollen Wissen und Zustände über gemeinsame Ziele bzw. Pläne synchronisiert werden?
3. **Rollenverteilung**: Wie sollen Teilaufgaben eines Gesamtplans an Mitglieder der Gruppe verteilt werden. Wofür ist das einzelne Mitglied verantwortlich?
4. **Rollensynchronisation**: Wie wird sichergestellt, dass die Teile des aktuellen Plans von den unterschiedlichen Mitgliedern in der richtigen Reihenfolge ausgeführt werden?

5. **Planauswahl**: Welche Pläne sollen ausgewählt werden, wenn Alternativen vorhanden sind?
6. **Fehlerfall**: Wie soll die Gruppe sich im Fehlerfall verhalten?

Kommunikation ist ein wichtiges Mittel zur Vermeidung fehlgeleiteter Kooperation. Häufig werden Agenten allerdings nur mit einem starren Koordinationsmechanismus versehen, der sich auf einen konkreten Anwendungsfall bezieht. Dies führt dazu, dass die Agenten mit unbekannten Situationen kaum oder gar nicht umgehen können und auch die Wiederverwendung erschwert wird.

1.2 Zielsetzung

Die Forschung hat in den letzten 20 Jahren viele theoretische Modelle entwickelt, unter anderem die Joint Intention Theorie (Cohen u. Levesque 1991) oder das AGR-Modell (Ferber et al. 2004). Davon haben einige den Weg zu einer Implementationen gefunden, wie beispielsweise die SimpleTeams-Umsetzung (Hodgson et al. 1999) in JACK, AGR in Madkit (Gutknecht u. Ferber 2001) oder das STEAM-System (Tambe 1997b). Ziel dieser Arbeit ist, zunächst verschiedene theoretische und praktische Ansätze zu untersuchen und ihre Eigenschaften herauszuarbeiten. Auf Grundlage der Ergebnisse wird ein Modell entwickelt und in das Agentenframework Jadex (Pokahr 2007), (Braubach 2007), (Jadex 2007a) integriert. Die in Jadex vorhandenen BDI-Konzepte sollen dabei möglichst auch auf Gruppenebene zur Verfügung stehen.

Durch das neue Modell soll der Agentenentwickler Konzepte und Werkzeuge in die Hand bekommen, welche es erlauben, Multiagentensysteme aus einer Gruppenansicht zu entwerfen und so die Komplexität der zu betrachtenden Software zu reduzieren. Ein weiterer Vorteil ist, dass eine bestimmte Menge von Agenten als abgeschlossene Einheit betrachtet werden kann. Dies kann in der Entwicklungsphase den Aufwand für Analyse und Spezifikation und während des Betriebs den Kommunikationsaufwand reduzieren.

1.3 Vorgehensweise

Um die im vorangegangenen Abschnitt vorgestellten Ziele zu erreichen, werden als Erstes die Kerngedanken erörtert, existierende Modelle untersucht und dann ein geeignetes Modell vorgeschlagen. Der Grundlagenteil beginnt mit einer Auseinandersetzung über Agenten, Multiagentensysteme und Kommunikation zwischen Agenten, da diese Themen die Basis für das Ziel dieser Arbeit darstellen. Im weiteren Verlauf werden dann Koordination, Kooperation und Gruppen im Allgemeinen untersucht und mit der Vorstellung von theoretischen Formalismen wie der Joint Intention Theorie oder dem AGR-Modell vertieft. Die Analyse endet mit einer Untersuchung der Systeme Madkit, STEAM und SimpleTeams, für die jeweils eine Implementation verfügbar ist.

Die gewonnenen Erkenntnisse werden anschließend dazu verwendet, ein neues Modell – das BDI-Gruppenmodell – vorzuschlagen. Der nächste Schritt besteht dann in der Realisierung einer prototypischen Implementation des Modells für das Jadex-Agentenframework. Zum Schluss der Arbeit werden zwei Beispielanwendungen vorgestellt, welche vom BDI-Gruppenmodell Gebrauch machen. Die letzten drei Schritte – d.h. Modellentwurf, Implementierung und die Beispiele – stehen in einem engen Zusammenhang, da in mehreren Iterationen die einzelnen Teilaufgaben einander beeinflussen und so das Gesamtergebnis verbessern.

1.4 Aufbau der Arbeit

Nach dieser Einleitung beschäftigt sich das zweite Kapitel zunächst mit den Grundlagen der Agententechnologie. Es wird dabei insbesondere auf die BDI-Agenten eingegangen, um dann mit dem Framework Jadex eine Agentenplattform vorzustellen. Zudem wird der Begriff der *Gruppe* sowie *Koordination* und *Kooperation* näher untersucht. Das nächsten Kapitel vertieft dies durch die Vorstellung theoretischer Modelle und praktischer Implementationen, wie etwa die *Joint Intentions* Theorie, *AGR*, *STEAM* und den *SimpleTeams*-Ansatz. Zusätzlich wird ein Ausblick auf weitere Verfahren gegeben. Im vierten und fünften Kapitel folgt die Beschreibung des BDI-Gruppenmodell und dessen Implementation für das Agentenframework Jadex. Anschließend werden

im sechsten Kapitel zwei Anwendungsbeispiele präsentiert, die das BDI-Gruppenmodell verwenden. Den Abschluss dieser Arbeit bildet eine Zusammenfassung und der Ausblick im siebten Kapitel.

2 Grundlagen

In diesem Kapitel werden zunächst die Grundlagen der gestellten Aufgabe erarbeitet. Es beginnt mit einer allgemeinen Betrachtung der Agententechnologie und der Präsentation des Jadex-Agentenframeworks als Vertreter des BDI-Konzepts. Dann wird der erste Schritt in Richtung gemeinsamen Handelns durch eine Vorstellung des von der Psychologie geprägten Themas der Gruppe gemacht. Den Abschluss bildet die Untersuchung von Koordination und Kooperation in Multiagentensystemen, worauf die nächsten Kapitel durch eine vertiefende Untersuchung aufbauen.

2.1 Agententechnologie

Dieser Abschnitt beschäftigt sich zunächst mit der Vorstellung der Agententechnologie und der Frage, was Agenten von anderer Software unterscheidet. Im weiteren Verlauf werden die BDI-Agenten vorgestellt und deren mentale Eigenschaften erläutert. Agenten kommen meist nicht allein vor, sondern bilden Multiagentensysteme. Diese sind auch Grundlage der restlichen Arbeit, da eine gruppenorientierte Sichtweise das Vorhandensein mehrere Agenten sinnvoll erscheinen lässt. In einem Multiagentensystem ermöglicht Kommunikation den Austausch von Informationen oder die Lösung von Konflikten. Aus diesem Grund ist auch in Gruppen Kommunikation ein wichtiger Aspekt, der am Ende dieses Kapitels kurz behandelt wird.

2.1.1 Agenten

Das Agentenkonzept ist aus der Verteilten Künstlichen Intelligenz hervorgegangen, womit die Anfänge bis zur Mitte des letzten Jahrhunderts zurückgehen (s. geschichtlicher Überblick in Wooldridge 2002 oder Ferber 2001). In der agentenorientierten Softwareentwicklung werden Systeme aus der Sicht einzelner Agenten entworfen. Diese werden als abgrenzbare Einheiten aufgefasst, welche Aufgaben autonom lösen und gegebenenfalls mit anderen Agenten interagieren (Wooldridge

2002). Im weiteren Verlauf wird versucht, den Agentenbegriff anhand mehrerer Definitionen genauer zu fassen. Trotzdem konnte er bisher nicht eindeutig bestimmt werden und so gibt es eine ganze Reihe von Definitionen, die häufig von der jeweiligen Anwendungsdomäne geprägt sind.

Müller (1996) beschreibt den Agenten als eine Black-Box, die Informationen aus einer Umgebung aufnimmt, diese intern verarbeitet und eine Ausgabe erzeugt, welche die Umgebung beeinflussen kann. Die Verarbeitung wird nicht von einer externen Einheit beeinflusst, was eine autonome Handlungsweise erlaubt. Müller wirft dann allerdings die Frage auf, ob eine so weite Definition überhaupt sinnvoll bzw. gewollt ist, da von ihr ein breites Feld von Systemen erfasst wird.

Wooldridge und Jennings (1995) unterscheiden auf Basis von anderen Arbeiten zwischen einer schwachen und starken Definition. Nach der schwachen Definition ist ein Hard- oder Softwaresystem ein Agent, wenn es die folgenden Eigenschaften besitzt:

1. **Autonomie**: Der Agent verrichtet seine Arbeit ohne direkten Einfluss von außen und trifft so selbstständig Entscheidungen über seine Aktivitäten. Weiß und Jackob (2005) gehen noch weiter und deuten an, dass diese Entscheidungen auch schwierig sein können, also regelmäßig eine umfangreiche Wissensverarbeitung erfordern.
2. **Soziale Fähigkeiten**: Die Agenten sind in der Lage, untereinander oder mit dem Menschen zu kommunizieren. Auf diesem Weg soll kooperiert und die Lösung von Konflikten erreicht werden.
3. **Reaktivität**: Der Agent kann seine Umwelt wahrnehmen und in akzeptabler Zeit auf Veränderungen reagieren.
4. **Proaktivität**: Der Agent entwickelt zielgerichtetes Verhalten ohne auf Reize aus seiner Umgebung zu reagieren.

So entsteht eine einfache Definition, wenn ein herkömmlicher Softwareprozess die vier oben genannten Kriterien erfüllt:

> *A simple way of conceptualising an agent is thus as a kind of UNIX-like software process, that exhibits the properties listed above. (Wooldridge u. Jennings 1995, S. 117)*

Die starke Definition verlangt zusätzlich, dass der Agent auch mentale Zustände besitzt. Diese werden in (Weiß u. Jakob 2005, S. 5) in drei Arten unterteilt:

1. **informationsbezogen** (Wissen, Vermutungen)
2. **konativ** (Intentionen, Pläne, Verpflichtungen)
3. **affektiv** (Ziele, Präferenzen, Wünsche)

Neben diesen als *mentale* bzw. *deliberativen* Agenten bezeichneten Klasse, unterscheidet man noch *reaktive* und *hybride* Agenten. Reaktive Agenten besitzen keine abstrakte Schlussfolgerungskomponente, sondern reagieren auf Eingaben durch die Abarbeitung einfacher Routinen mit dem Vorteil, dass eine wesentlich schnellere Antwortzeit gewährleistet ist. In hybriden Agenten versucht man beide Systeme zu vereinen. Häufig werden Agenten noch unter weiteren Gesichtspunkten betrachtet. Hierzu zählen beispielsweise Mobilität, Emotionen oder Lernfähigkeit (siehe Weiß u. Jakob 2005).

Wooldridge (2002) adaptiert die im vorangegangenen Absatz erläuterte Definition. Er betrachtet nun die Autonomie als eine zentrale Eigenschaft und kommt so zu der folgenden Definition:

> *An Agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives. (Wooldridge 2002, S. 15)*

Der Agent ist in der Lage, Informationen über seine Umgebung wahrzunehmen und diese durch Handlungen zu modifizieren. Häufig ist es dem Agenten aber nicht möglich, eine vollständige Kontrolle auszuüben, da er nur über eine bestimmte Menge von Aktionen verfügt, die auch nicht immer anwendbar sein können. Zudem sollte der Agent darauf vorbereitet werden, mit Fehlern umzugehen, da in der Regel nicht garantiert ist, dass eine Aktion erfolgreich verläuft. Eines der Hauptprobleme besteht darin, dass ein Agent entscheiden muss, welche Aktionen er im Hinblick auf seine Ziele am Besten auswählen sollte, um ein optimales Ergebnis zu erreichen.

Wooldridge versucht auch eine Antwort auf die Frage nach intelligenten Agenten zu finden und betrachtet dabei die oben eingeführten Attribute *Reaktivität, Proaktivität* sowie *soziale Fähigkeiten* als notwendige Anforderungen.

2.1.2 Reaktive und deliberative Agenten

Man kann Agenten anhand ihres inneren Aufbaus in reaktive und deliberative Systeme unterteilen. Wesentliches Merkmal ist das Verfahren, welches die Entscheidungen zu bestimmten Handlungen trifft.

Ein reaktiver Agent empfängt über Sensoren Informationen, die in Modulen mit spezieller Kompetenz bearbeitet werden und hierdurch entsprechende Aktionen auslösen. Jedes Modul ist für einen kleinen, nicht-komplexen Aufgabenbereich zuständig. Kommunikation zwischen einzelnen Modulen und der Umwelt ist möglich, basiert aber meinst nicht auf einer höheren Sprache. Eine schnelle Reaktion auf bekannte Probleme und Robustheit (z.B. bei Ausfall eines Moduls) sind ein wesentlicher Vorteil reaktiver Agenten. Unvorhergesehene Aufgaben können aber Schwierigkeiten bereiten oder zu unerwarteten Ergebnissen führen.

Deliberative Agenten hingegen durchlaufen regelmäßige Zyklen, die aus Wahrnehmen, Schlussfolgern und Handeln bestehen. Dabei werden Informationen aus der Umgebung aufgenommen und verarbeitet. Die Handlungen leiten sich aus dieser Wissensverarbeitung ab. Sowohl die Transformation der realen Welt in eine geeignete interne Darstellung als auch die Suche nach Entscheidungen machen die Agenten sehr vielseitig, aber auch langsam. Manche Szenarien erfordern jedoch schnelles Handeln, um rechtzeitig auf Änderungen der Umwelt reagieren zu können. Dies führte zur Entwicklung hybrider Systeme, welche sowohl reaktive als auch deliberative Verfahren in einem Agenten vereinen.

2.1.3 BDI-Agenten

Wie bereits erwähnt, gibt es eine Vielzahl von Modellen, die den internen Aufbau oder den Prozess der Entscheidungsfindung beschreiben. Da die BDI-Agenten als Basis der weiteren Arbeit dienen und das BDI-Konzept auch Einfluss auf die Gestaltung des Gruppenmodells nimmt, soll diese bestimmte Klasse von Agenten in diesem Abschnitt näher vorgestellt werden.

Das BDI-Modell wurde in (Bratman 1987) als eine Theorie über den menschlichen Entscheidungsprozess entwickelt. So soll jeder Mensch über Wissen *(Beliefs)*, Ziele *(Desires)* und Absichten *(Intentions)* verfügen, die seine Entscheidungen zu bestimmten Aktionen steuern. Andere Einflüsse, wie Emotionen oder Erkenntnisse, werden aber nicht berücksichtigt. In der Agententechnologie wurde dies aufgegriffen und

führte zur Entwicklung der BDI-Agenten. Die Idee besteht darin, den Zustand des Agenten in Form von mentalen Haltungen zu beschreiben mit deren Hilfe der Agent seine Handlungen bestimmt (s. Abbildung 1, angelehnt an (Wooldridge 2002), S. 83). Das Modell besteht aus den drei Grundkomponenten: Wissen *(**B**eliefs)*, Wünsche *(**D**esires)*, Intentionen *(**I**ntentions)*.

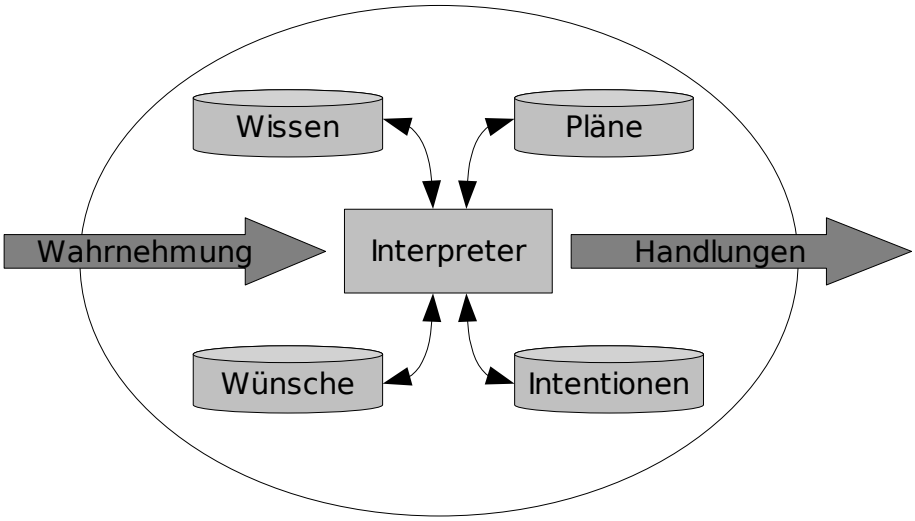

Abbildung 1: Das BDI-Modell

Beliefs sind das Wissen, das ein Agent über den aktuellen Weltzustand besitzt. Zudem können auch zukünftige Erwartungen enthalten sein. In der Regel beeinflusst das Wissen die Entscheidung, welche Ziele aktiv verfolgt und welche Pläne ausgewählt werden und umgekehrt wird das Wissen von der Wahrnehmung und den Entscheidungen des Agenten verändert. Die Weltzustände, die ein Agenten erreichen möchte, werden mit Hilfe von Wünschen (*Desires*) dargestellt. Dabei sind auch unerfüllbare oder sich widersprechende Wünsche erlaubt. Ziele sind dabei eine konsistente Untermenge aller Wünsche, sie widersprechen sich also nicht. Häufig wird auch gefordert, dass der Agent glaubt, er wäre in der Lage, das Ziel zu erfüllen.

Intentionen hingegen sind eine verbindliche Festlegung des Agenten auf die Verfolgung bestimmter Ziele, wobei eine Intention normalerweise nur unter besonderen Umständen wieder aufgegeben werden kann (z.B. bei einem endgültigem Fehlschlag). Dies ist notwendig, da die Ressourcen eines Agenten beschränkt sind und er somit nicht laufend alle Ziele gegeneinander abwägen kann (Jennings 1993b). Sie unterstützen zudem den Planungs- und Auswahlprozess (s.u. *Practical Reasoning*), da sie relevante Informationen über Erfüllbarkeit und Fortschritt enthalten.

Damit Intentionen nicht ständig übernommen oder verworfen werden, müssen diese eine gewisse Stabilität aufweisen. Jennings (1993a) schlägt hierfür die Verwendung von Konventionen (*conventions)* vor, die beschreiben, unter welchen Umständen Intentionen aufgegeben werden können.

Practical Reasoning

Practical Reasoning ist ein Verfahren, mit dem der Agent versucht, Entscheidungen zu treffen. Es ist darauf ausgerichtet, im Ergebnis zu einer Handlung des Agenten zu führen. Dies unterscheidet *Practical Reasoning* auch vom *Theoretical Reasoning*, in dem nur Wissenszustände verändert werden. Der Prozess des *Practical Reasoning* besteht aus mindestens zwei Schritten (Wooldridge 2002, S.66):

1. **Deliberation**: Welche Ziele sollen erreicht werden?
2. **Means-End Reasoning**: Wie soll ein Ziel erreicht werden?

Während der Deliberationsphase werden zunächst aus der Menge möglicher Wünsche diejenigen herausgefiltert, die im Moment verfolgt werden sollen bzw. können. Das Ergebnis ist eine Menge von Intentionen mit bestimmten Eigenschaften. Der wohl wichtigste Aspekt ist, dass Intentionen persistent sind, d.h. sie werden nur noch unter besonderen Umständen wieder aufgegeben. Darüber hinaus stoßen Intention den Planungsprozess an, mit dem der Agent versucht, einen Lösungsweg zu erarbeiten. Weiterhin beeinflusst die getroffene Auswahl sowohl den zukünftigen Deliberationsprozess als auch die Wissensbasis des Agenten. Dies liegt vor allem daran, dass keine Intentionen gebildet werden dürfen, die der bestehenden Menge widersprechen.

Der auch als Planung bezeichnete Prozess des *Means-End Reasoning* bedeutet herauszufinden, wie die während der Deliberation übernommenen Intentionen erreicht werden können. Aus der Eingabe „Intention/Ziel, aktueller Weltzustand, Menge der möglichen Aktionen“ wird durch den Prozess ein Plan erstellt, welcher eine Folge von Aktionen besitzt, die das gewünschte Ziel erfüllen könnten.

Procedural Reasoning Systems

Georgeff und Lansky entwickelten mit PRS (beschrieben in Georgeff u. Lansky 1987) eine verbreitete Architektur, die Repräsentationen für Wissen, Wünsche, Intentionen und Plänen besitzt (s. auch Abbildung 1). Die Ausgestaltung der einzelnen mentalen Haltungen wird hier nicht

vorgeschrieben. Pläne können abstrakt sein und vom Agenten während der Ausführung konkretisiert werden. Für diese Architektur wurde mit (Rao u. Georgeff 1995) eine Implementation vorgeschlagen. Diese besteht im Wesentlichen aus einem Interpreter, der in Form einer Endlosschleife die einzelnen Schritte des *Means-End Reasonings* durchführt. Wie in anderen Implementationen musste auch hier das ursprüngliche BDI-Modell vereinfacht werden, um die darin enthaltenen Funktionen berechenbar zu machen. Beispielsweise werden weder Wünsche noch Intentionen explizit dargestellt (Rao u. Georgeff 1995, S. 315f.).

2.1.4 Die Umgebung

Bisher wurde auf die internen Merkmale eines Agenten eingegangen und eine spezielle Klasse der Agenten vorgestellt. Der Agent wird aber regelmäßig in einer Umgebung eingebettet, die wahrgenommen und/oder verändert werden kann. Wooldridge (2002) diskutiert verschiedenen Kriterien, nach denen die Umgebung klassifiziert werden kann:

- **Zugang**: Der Agent hat häufig nur eine eingeschränkte Sicht auf die Umgebung, er kann also nicht jede Information sofort und vollständig erlangen.
- (**Nicht-**) **Determinismus**: Jede Aktion hat (nicht) immer einen garantierten wiederholbaren Effekt.
- **Statisch:** Die Umgebung verändert sich nur aufgrund von Aktionen des Agenten. Oder **Dynamisch**: Auch andere Prozesse können die Umgebung verändern.
- **Diskret** oder **Kontinuierlich**: Die Umgebung kann nur eine *endliche* oder eine *unendliche* Anzahl von Zuständen einnehmen.

2.1.5 Multiagentensysteme

Während der einzelne Agent sich in einer Umwelt bewegt, die keine anderen Agenten enthält, sind Agenten in Multiagentensystemen in der Lage, mit anderen Agenten zu kommunizieren und zu kooperieren.

Die Autoren von (d'Inverno u. Luck 2004), beschreiben ein Multiagentensystem als ein verteiltes System, in dem einzelne abgeschlossene Einheiten unabhängig voneinander Probleme lösen, aber trotzdem ein Ganzes bilden. Grundsätzlich gibt es keine vordefinierten Strukturen oder Interaktionen. Da aber auch kein globales Ziel vorhanden

ist, müssen die Agenten ihre Aktivitäten durch Kommunikation koordinieren, um doppelte Arbeit bzw. Konflikte zu vermeiden oder die Fähigkeiten anderer Agenten zu nutzen.

> *„... a Multi-Agent System (MAS) is a macro-system comprimising multiple agents, each of which is considered as a micro-system. MASs result therefore from the organization of multiple agents within an environment". (Schumacher 2001, S. 14)*

Ein Multiagentensystem ergibt sich demnach aus dem Zusammenfügen von Einzelagenten in einer Umgebung. Häufig verfügt jeder Agent nur über eine eingeschränkte Sicht, es stehen ihm also nicht alle Informationen über Umgebung und anderen Agenten zur Verfügung. Folglich gibt es auch keine globale Kontrolle, sondern jeder Agent hat seinen eigenen Zustand, der sich nur durch das Verhalten des Agenten ändert.

Franklin und Graesser (1997) unterscheiden zwischen Multiagentensystemen, die aus einer Ansammlung von einzelnen Agenten bestehen und einem hierarchischen Baum von Subagenten, die einen einzelnen Agenten in Schichten aufteilt. Außerdem weisen sie darauf hin, dass nicht alle Agenten in ein System von Subagenten zerlegt werden können. Vielmehr komme es darauf an, wie die Sensoren und Aktoren in den Schichten verteilt sind. In einem horizontalen System befinden sich diese auf jeder Schicht und ermöglichen die Zerlegung in ein Multiagentensystem, wohingegen ein vertikales System mit Sensoren in der untersten und Aktoren in der obersten Schicht kaum Ansatzpunkte bietet.

2.1.6 Kommunikation

Die Kommunikation ist ein zentrales Thema in der Informatik. Auch in den gerade vorgestellten Multiagentensystemen spielt sie eine wichtige Rolle, da die Agenten nur so Informationen austauschen oder gemeinsam Probleme lösen können. Allerdings muss man sich einen wichtigen Unterschied zur klassischen Kommunikation klar machen. Da Agenten autonom handelnde Einheiten sind, kann Kommunikation nicht als einfacher Methodenaufruf aufgefasst werden, der die Werte einer anderen Entität verändert. Sie ist vielmehr eine einzelne oder eine Folge von Aktionen, die versucht den anderen Agenten in seinem Wissen zu beeinflussen.

In der Agentenwelt haben sich die gebräuchlichen Sprachen KQML (Finin 1993, Finin u. Labrou 1997) und FIPA-ACL (FIPA 2002) etabliert, die hier aber nicht im Detail erläutert werden sollen. Sie basieren im Wesentlichem auf der Sprechakttheorie.

Nachrichtenübermittlung

In dieser Form der Kommunikation werden einzelne Nachrichten direkt zwischen den Agenten ausgetauscht. Dies bedeutet, dass der Absender die Informationen über ein Medium an den Empfänger sendet. Unbeteiligte Agenten erfahren hiervon nichts und können auch nicht auf die Inhalte zugreifen (siehe auch Ferber 2001, S. 336ff.).

Blackboard-Systeme

Ein Blackboard-System (Corkill 1991) beinhaltet einen für alle Agenten zugänglichen, gemeinsamen Arbeitsbereich. Jeder Agent kann hier Informationen ablegen oder auslesen. Mit Hilfe von Filtern kann der einzelne Agent die für ihn relevanten Daten erkennen und muss nicht alle neuen Informationen übertragen.

Um Zugang zu erhalten, muss sich jeder Agent bei einer zentralen Stelle registrieren, die das Blackboard verwaltet. Zwischen den Agenten findet meist keine eigenständige Kommunikation, d.h. Nachrichtenübermittlung, statt. Durch regelmäßiges Auslesen des Blackboards kann der Agent Aufschluss über die Arbeit anderer Agenten erhalten, wodurch er diese Informationen in seinen eigenen Prozess einbinden kann. Es ist aber auch möglich, dass die Agenten bei neuen Informationen vom Blackboard benachrichtigt werden. In diesem Fall müssen die Agenten beim Blackboard hinterlegen, an welcher Art von Ereignissen sie interessiert sind (Huhns u. Stephens 1999).

Blackboardsysteme stellen somit einen flexiblen Ansatz zur Kooperation bereit, da sie unabhängig vom gewählten Kooperationsmodell sind. Auch die Anforderungen an die Agenten und das System sind gering. Allerdings können Blackboardsysteme mit steigender Anzahl von Agenten zu einem Flaschenhals werden, weil sie der einzige Weg sind, über den die Agenten kommunizieren können.

Verhandlungen

Verhandlungen sind ein Hilfsmittel, um Konsens über einen bestimmten mentalen Zustand zu erzielen:

> *„…negotiation is the communication process of a group of agents in order to reach a mutually accepted agreement on some matter.“ (Bussmann und Müller 1992); zitiert nach (Nwana et al. 1996 S. 83)*

Es wurden verschiedene Techniken entwickelt, um Verhandlungen zwischen Agenten effizient zu gestalten (Nwana et al. 1996):

1. Verfahren zur Repräsentation und Aktualisierung von Wissensmodellen,
2. Schlussfolgern über das Wissen anderer Agenten und
3. Beeinflussung von Wissen bzw. Zielen anderer Agenten.

Dies führte zur Entwicklung einer breiten Palette an Verhandlungsverfahren. Grob lassen sich diese in folgende Kategorien einteilen:

1. spieltheoretische Ansätze,
2. planbasierte Techniken und
3. vom Menschen inspirierte und andere AI-Techniken.

Da sich diese Arbeit nur am Rande mit Verhandlungsverfahren beschäftigt, wird für eine detaillierte Beschreibung auf die jeweilige Literatur verwiesen (z.B. Nwana et al. 1996 oder Wooldridge 2002).

2.1.7 Agentenorientiere Softwareentwicklung

Die bisher vorgestellten Konzepte und Technologien haben zu einer neuen Ausprägung im Bereich der Softwareentwicklung geführt. Wie in der Einleitung ausgeführt, werden Agenten und besonders Multiagentensysteme als eine der Zukunftstechnologien betrachtet. Dieser Abschnitt soll aufzeigen, wo der Unterschied zur bekannten Objektorientierung liegt, welches die Vorteile sind und wann sich der Einsatz von Agenten lohnt.

Auch wenn Agenten und Objekte in der Softwareentwicklung Gemeinsamkeiten besitzen, können beide Konzepte nicht gleichgestellt werden. Es gibt verschiedene Merkmale, in denen sie sich fundamental unterscheiden (Wooldridge 2002):

1. Ein Agent unterstützt als wesentliches Merkmal das Autonomiekonzept. Er entscheidet also selbstständig, ob er Aktionen, die von anderen Agenten angefragt werden, ausführen will. Ein Objekt hingegen hat keinen Einfluss darauf, wann eine Methode aufgerufen wird.
2. Agenten sind zu unterschiedlichem Verhalten *(reaktiv, proaktiv, sozial)* fähig.
3. Jeder Agent besitzt mindestens eine eigene, von den anderen Agenten abgetrennte Ausführungskontrolle.

Auch Objekte können diese Eigenschaften implementieren: so könnte beispielsweise eine Methode durchaus vor der Ausführung der eigentlichen Funktionalität anhand eines internen Zustands prüfen, ob eine Ausführung gewollt ist. Aber die genannten Merkmale sind nicht grundsätzlich im objektorientierten Modell vorgesehen.

Was sind also die wesentlichen Vorteile, die das Agentenparadigma im Gegensatz zu anderen Entwicklungsmethoden mit sich bringt? Dieser Frage sind Weiß und Jakob (2005) nachgegangen und kamen auf Grundlage von (Jennings 1999), (Jennings 2000) und (Booch 1994) zu dem Ergebnis, dass die agentenorientierte Softwareentwicklung die folgenden Techniken auf natürliche Weise unterstützt:

1. **Dekomposition**: Zerlegung des Gesamtproblems in kleinere, übersichtlichere Teile, die im Wesentlichen unabhängig voneinander sein sollen.
2. **Abstraktion**: Prozess, in dem ein Modell entwickelt wird, das nur über die wesentlichen Eigenschaften verfügt und Details ausblendet.
3. **Organisation**: Struktur, welche die Beziehungen zwischen einzelnen Komponenten identifiziert und festlegt. An dieser Stelle spielen Interaktionen eine wichtige Rolle.
4. **Wiederverwendung**: ist die Möglichkeit erarbeitete Ergebnisse leicht in neuen Projekten wiederzuverwenden.

Die aufgeführten Techniken erlauben, auch komplexe Softwaresysteme zu erstellen und zu warten. Zusätzlich wird es möglich, diese als menschenähnliche Organisationen zu betrachten (Weiß u. Jakob 2005), was eine intuitive Betrachtungsweise und die Nutzung von organisationstheoretischen Konzepten (z.B. aus Daft 1992) ermöglicht. Zudem werden durch die agenten- und organisationsorientierte

Systemansicht weitere Abstraktionsebenen einführt, welche die Entwicklung komplexer Anwendungen erleichtern, da die Details tiefer liegender Schichten ausgeblendet werden.

Abschließend stellt sich somit die Frage, wann es sinnvoll ist, Agenten in der Softwareentwicklung einzusetzen. Einen ersten Anhaltspunkt bietet hier die Arbeit (Braubach et al. 2006, S. 504):

- Es gibt eine (auch im laufenden Betrieb) wechselnde Anzahl von Komponenten.
- Eine externe Kontrolle ist entweder nicht möglich oder unerwünscht.
- Koordination innerhalb des Systems findet durch komplexe Kommunikationsbeziehungen statt.

Es folgt, dass vor allem die Anwendungsdomäne die einzusetzende Technik (d.h. die Methoden oder die Plattform) und damit die zu erstellende Anwendung bestimmt.

2.1.8 Zusammenfassung

Es scheint schwer zu fallen, für den Agentenbegriff eine präzise Definition festzulegen. Neben den vorgestellten Versuchen gibt es in der Literatur viele weitere Ansätze und Taxonomien, die versuchen zu beschreiben, was einen Agenten eigentlich ausmacht und von anderer Software unterscheidet. Je nach Anwendungsdomäne gibt es verschiedene Attribute, die wichtig erscheinen und von ihren jeweiligen Vertretern gefordert werden. Trotzdem kann man feststellen, dass bestimmte Vorstellungen und Begriffe, wie der der Autonomie, immer wieder auftauchen. So könnte man die von Jennings identifizierten Attribute nutzen, um eine interessante Menge von Agenten zu identifizieren.

Mit Blick auf das Thema der Arbeit wurde die Klasse der BDI-Agenten und die damit verbundenen Konzepte Wissen, Wünsche und Absichten vorgestellt. Diese Elemente bestimmen im *Practical Reasoning* genannten Prozess die Handlungen, die der Agent unternimmt, um seine Ziele zu erreichen. Diese Ideen werden später wieder aufgegriffen und im entwickelten BDI-Gruppenmodell zum Tragen kommen.

Der nächste Schritt führte zu den Multiagentensystemen, die nicht nur einen, sondern eine Vielzahl unterschiedlicher Agenten enthalten können. Diese kommunizieren und interagieren miteinander, um ihre Aufgaben

zu bewältigen. Es existiert bereits eine Vielzahl von Verfahren, welche die Agenten zur Koordination ihrer Aktivitäten anwenden können. Auch im Bereich der Kommunikation sind verschiedene Techniken darunter das Blackboard entwickelt worden.

Die Vorstellung der Agententechnologie endete mit einer Untersuchung, wie diese Konzepte in der Softwareentwicklung eingesetzt werden können. Im Gegensatz zur Objektorientierung sind verschiedene Konzepte (wie das der Autonomie) fest im Paradigma enthalten. Zudem wird der Softwareentwicklungsprozess mit Blick auf die zunehmende Komplexität durch die natürliche Unterstützung der Schlüsselelemente Dekomposition, Abstraktion, Organisation und Wiederverwendung erleichtert.

2.2 Das Agentenframework Jadex

In diesem Kapitel soll das für die Implementation vorgesehene Agentenframework Jadex vorgestellt werden. Es wurde an der Universität Hamburg im Rahmen der Dissertationen (Braubach 2007) und (Pokahr 2007) entwickelt und erlaubt die Konstruktion reaktiver als auch zielgesteuerter BDI-Agenten. Das System ermöglicht durch einen hybriden Sprachansatz den Einsatz bekannter Techniken wie XML und objektorientierter Programmierung in Java. Zudem stellt Jadex eine Laufzeitumgebung bereit, in der entsprechende Agenten ausgeführt werden können.

2.2.1 Die Jadex-Architektur

Sowohl die rein reaktive als auch proaktive Agentenentwicklung ist mit Jadex möglich und für den Einsatz mehrerer Agenten in einem Multiagentensystem werden entsprechende Unterstützungen angeboten. Die interne Verarbeitung von Zielen und Events ist stark an das oben bereits eingeführte PRS angelehnt, besitzt aber im Detail Unterschiede bei der Darstellung der BDI-Konzepte.

Beliefs

Jadex bedient sich einer objektorientierten Form zur Darstellung der *Beliefs* (Pokahr et al. 2005d). Die Wissensbasis besteht aus benannten Fakten oder Faktmengen, welche ausgelesen und manipuliert werden können. Anfragen oder Operationen werden in einer beschreibenden,

satzorientierten Art dargestellt, welche sich stark an die OQL (Object Query Language) anlehnt. Das Wissen ist aber nicht nur eine passive Ansammlung von Daten, sondern es können Änderungen über Events mitgeteilt und so Seiteneffekte im Agenten ausgelöst werden.

Desires und Goals

Desires repräsentieren die Wünsche des Agenten und beeinflussen so die Art seiner Aktionen. *Goals* sind momentane konkrete Instanzen der *Desires* und ergeben sich damit aus den Wünschen. Ein Auswahlprozess (*Goal-Deliberation*) wählt aus allen *Goals* eine konsistente Menge aus, die der Agent aktuell verfolgen soll (Pokahr et al. 2005d). Für jedes *Goal* führt der Agent vordefinierte Pläne aus bis es erreicht oder nicht mehr erfüllbar ist. Der Agent ist hierzu aber nicht gezwungen, so dass es auch *Goals* ohne aktuell ausgeführte Pläne geben kann. Dieses Goalkonzept ermöglicht es dem Agenten, Pläne zu wiederholen oder bei einem Fehler andere Pläne zu versuchen. Auch das proaktive Verhalten lässt sich hieraus ableiten.

Jadex erlaubt die Strukturierung mit Hilfe von *Toplevel*- und *Subgoals*. Der Agentenentwickler ist so in der Lage, Methoden wie die Dekomposition anzuwenden und ein umfassendes Ziel in mehrere Teilziele aufzubrechen. Diese bilden in der Regel eine Hierarchie bestehend aus Zielen und Plänen, welche von unten nach oben immer abstrakter werden.

Pläne

Pläne stellen die Möglichkeiten dar, mit denen ein Agent seine Ziele erreichen oder auf Ereignisse reagieren kann (Pokahr et al. 2005d). Sie müssen aber nicht nur atomare Aktionen enthalten, sondern können in Jadex auch Subgoals erzeugen.

Jeder Plan besteht aus einem *Header* und einem *Body*. Im *Header* werden verschiedene Parameter und die drei Bedingungen *Creation-, Context*- und *Drop*-Condition spezifiziert. Diese geben an, unter welchen Umständen ein Plan ausgeführt, unterbrochen und verworfen werden soll. Im *Body* wird eine Abfolge von Handlungen definiert. Dies können beispielsweise Nachrichtenversand, Manipulationen der Wissensbasis oder die Erzeugung von Subgoals beinhalten. Der Agentenentwickler kann für diese Aktionen auf eine spezielle API zurückgreifen. Andere Aktionen, wie der Zugriff auf externe Klassen oder eine Datenbank, sind

aber auch möglich, müssen jedoch vom Agentenentwickler selbst implementiert werden. Dieser Aufbau der Pläne erlaubt die Wiederverwendung durch den Entwickler in anderen Agenten.

2.2.2 Explizite Darstellung von Zielen

Im Gegensatz zu anderen Plattformen werden Ziele in Jadex explizit und nicht als Event dargestellt (Pokahr et al. 2005a). Der Deliberationsprozess weiß so jederzeit, welche *Goals* vorhanden sind und welche Eigenschaften diese besitzen. Jedes Ziel verfügt über verschiedene Bedingungen, die den Zustand des *Goals* während der Laufzeit beeinflussen.

Ziele befindet sich immer in einem von drei Zuständen: *Option, Active* oder *Suspended.* Der Zustand *Active* bedeutet, dass der Agent das Ziel aktiv verfolgt. Wenn ein Ziel sich im Zustand *Option* befindet, könnte der Agent es zwar verfolgen, tut es aber nicht, da es sich im Widerspruch mit einem anderen Ziel befindet. Zuletzt gibt der *Suspended*-Zustand an, dass das Ziel nicht verfolgt werden kann, da die *Context*-Bedingung nicht erfüllt ist – es fehlt z.B. eine Information in der Wissensbasis.

2.2.3 Die Easy Deliberation Strategy

Die *Easy Deliberation Strategy* (Pokahr et al. 2005a) ist ein Verfahren mit dessen Hilfe der Agent bestimmt, welche *Goals* aktiv verfolgt und welche als *Option* zurückgehalten werden. Dies ist notwendig, da der Agent auch über mehrere sich widersprechende Ziele verfügen kann. Verschiedene Faktoren beeinflussen die Entscheidungen, die das Verfahren trifft, wobei aber nur Informationen über *Goals* verwertet werden. Der Agentenentwickler spezifiziert in der Agentendefinition,

1. wie viele *Goals* eines Typs gleichzeitig aktiv sein dürfen und
2. welche Beziehungen zwischen *Goals* bestehen.

Diese Beziehungen geben an, in welchem Verhältnis sich die einzelnen Ziele zueinander befinden. Mit dieser Technik kann beschrieben werden, ob sich zwei Ziele gegenseitig unterstützen (positive Beziehung) oder ausschließen (negative Beziehung). Das Verfahren besitzt allerdings auch einige Einschränkungen, die sich durch die Tatsache ergeben, dass nur negative Beziehungen berücksichtigt werden. So kann unter anderem nicht angegeben werden, dass das Erreichen zweier *Goals* wichtiger ist,

als ein anderes widersprüchliches Ziel. Auf der Ebene der Pläne werden Konflikte hingegen überhaupt nicht erkannt, so dass Ressourcenzugriffe gegebenenfalls eigenständig abgesichert werden müssen.

2.2.4 Jadex BDI-Interpreter

Mit den Metaaktionen wurde der klassische Interpreterzyklus (Rao u. Georgeff 1995) in kleine Einzelaktionen aufgebrochen, wobei die angeführten Operationen als Ausgangspunkt dienen (Pokahr et al. 2005b). Anstatt genereller Methoden bezieht sich jede Metaaktion nur auf einen Teilaspekt, wie der Ausführung eines Planschrittes. Dies erlaubt, jede Aktion nur bei Bedarf und in beliebiger Reihenfolge auszuführen.

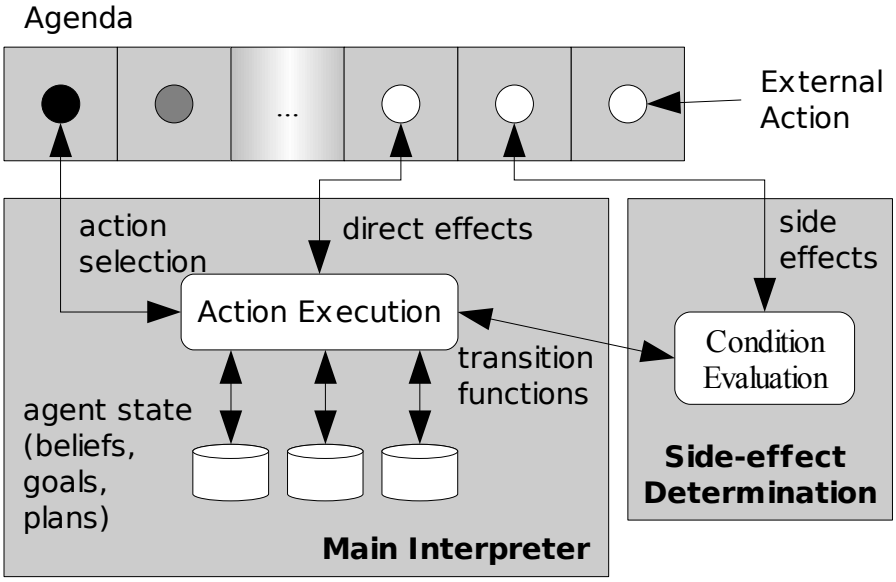

Abbildung 2: Architektur des Jadex-Interpreters

Der neuartige Interpreter (Pokahr et al. 2005a und Pokahr et al. 2005b) besteht aus drei Kernkomponenten (s. Abbildung 2, entnommen aus Pokahr et al. 2005a). In der Agenda werden alle noch ausstehenden Metaaktionen abgelegt. Der Interpreter nimmt sich immer die erste Aktion und führt diese aus. Bei der Ausführung einer Metaaktion können als direkter Effekt neue Metaaktionen erzeugt und in die Agenda eingefügt werden. Durch Änderungen der Wissensbasis über die *Context-* bzw. *Drop*-Bedingung können aber auch Seiteneffekte entstehen. Die dritte Komponente *Side-effect Determination* sorgt daher

durch Überwachung der genannten Bedingungen für das Einfügen von entsprechenden Metaaktionen. Als weitere Quelle neuer Metaaktionen kommen externe Einflüsse, wie der Nachrichtenempfang, in Betracht.

2.2.5 Sprache

Jadex definiert keine neue Agentensprache, sondern setzt auf vorhandene Techniken wie Java und XML (Pokahr et al. 2005c). Ein Agent wird mit Hilfe eines *Agent Definition File* (ADF) und Javaklassen programmiert. In der ADF werden alle *Beliefs*, *Goals*, *Plan-Header* und *Initialwerte* in XML eingetragen. Die Javaklassen hingegen dienen dem Entwickler zur Programmierung des Planbodys.

Jede ADF enthält zunächst allgemeine Definitionen wie den Namen und die Beschreibung des Agenten. Zusätzlich können Laufzeitoptionen, Imports, Expressions und Properties spezifiziert werden. An dieser Stelle soll lediglich ein kurzer Überblick gegeben und nicht weiter auf die Details eingegangen werden. Diese sind bei Bedarf im *Jadex Userguide* (Jadex 2007a) und dem *Tutorial* (Jadex 2007b) nachzulesen.

Wie bereits erwähnt wird das Wissen in Form von Javaobjekten dargestellt. Diese besitzen neben weiteren Eigenschaften vor allem einen Namen, einen Typ und gegebenenfalls einen Initialwert. Ziele besitzt eine *Creation-, Context-* und *Drop*-Bedingung, welche als boolescher Ausdruck anzugeben ist. Es können weitere Parameter zur Informationsübergabe und Flags gesetzt werden, welche die Eigenschaften des Ziels näher festlegen. Für jeden Plan wird innerhalb der ADFs ein Header spezifiziert, der *Name*, *Trigger*, *Pre-* und *Context*-Bedingung enthält. Zusätzlich können weitere Parameter übergeben werden, die die speziellen Eigenschaften eines Plans festlegen. Der Planbody muss dagegen in Java geschrieben werden. Diese Klasse erbt aus der Plan-Klasse des Jadex-Packages und implementiert verschiedene Methoden:

- **body()**: enthält das Planverhalten
- **passed()**: wird bei Erfolg aufgerufen
- **failed()**: wird bei unbehandelten Exceptions aufgerufen
- **aborted()**: wird bei vorzeitigem Planabbruch aufgerufen

2.2.6 Zusammenfassung

Jadex ist ein umfangreiches Framework für die Entwicklung und Ausführung von BDI-Agenten. Einfache Programmierung und die Weiterentwicklung der klassischen BDI-Komponenten stehen im Mittelpunkt, während versucht wurde, möglichst viele Konzepte aus der Reasoning- und der Middleware-Welt zu vereinen. Die Laufzeitumgebung und verschiedenen Tools (Jadex 2007c) erlauben eine einfache Nutzung der entwickelten Agenten.

2.3 Gruppen

Dieser Abschnitt macht den ersten Schritt von der Betrachtung des einzelnen Agenten hin zur Untersuchung, wie mehrere Agenten in Gruppen ihre Aktivitäten abstimmen. Hierfür wird zuerst der Begriff der Gruppe betrachtet und auf die Agentenwelt übertragen. Es werden die Vorteile von Gruppenarbeit und die Formation einer Gruppe untersucht, um dann das Konzept der Rolle als wichtiges Beschreibungselement vorzustellen.

Ähnlich wie für den Agentenbegriff bietet die Psychologie eine Fülle von Definitionen zum Begriff 'Gruppe'. In (Sader 1998) stellt der Autor neben weiteren diese sehr allgemein gehaltene Definition vor:

> *Wenn zwei oder mehr Personen in irgendeiner Beziehung zueinander stehen, bilden sie eine Gruppe. (Lindgren 1973, S. 347)*

Um eine Gruppe näher zu charakterisieren, könnten verschiedene Eigenschaften der Beteiligten betrachtet werden. So erleben sich die Mitglieder einer Gruppe als zusammengehörig oder verfolgen gemeinsame Ziele. Häufig entwickeln sie auch Formen der Aufgabenteilung, lassen sich in Rollen unterteilen oder führen untereinander viele Interaktionen durch.

In der Welt der Agenten lassen sich verschiedene Definitionen finden. Odell et al. (2003) formulieren diese für Agenten wie folgt:

> *A group is a set of two or more agents that are related via their role assignments, where these relationships must form a connected graph within the group. (Odell et al. 2003, S. 29)*

Kernelement sind hier Rollen, welche die Agenten zueinander in Beziehung setzen. Dabei bilden die vorhandenen Rollen einen Graphen. Alle über Kanten miteinander verbundenen Rollen und damit die daran hängenden Agenten bilden eine Gruppe.

Häufig haben Gruppen die Möglichkeit, nach außen als einzelne Einheit aufzutreten. Dies bedeutet, dass externe Entitäten die Gruppe als ein Ganzes auffassen und sich meist auch nicht für die interne Struktur interessieren. In der Welt der Agenten kann dies auch als eine agentifizierte Gruppe (Odell et al. 2005) bezeichnet werden. Im Wesentlichen bedeutet dies, dass die Gruppe über alle Merkmale verfügt, die auch ein Agent besitzt, wie etwa die Möglichkeit des Nachrichtenversands. Ein Außenstehender kann dann mit der gesamten Gruppe genauso kommunizieren, als wenn es nur ein einzelner Agent wäre.

Eine weitere interessante Fähigkeit besteht darin, dass die Gruppe wieder selbst als Teilnehmer einer anderen Gruppe auftreten kann. Dies kann beispielsweise nützlich sein, wenn kein Agent allein den Anforderungen einer Gruppe gerecht wird, so dass nur durch Zusammenwirken einer Teilgruppe die gewünschte Rolle in einer anderen Gruppe erfüllt werden kann. Die Folge ist eine hierarchische Struktur von Gruppen, Teilgruppen und Agenten, die von oben nach unten eine immer konkreter werdende Dekomposition zeigt.

2.3.1 Gruppen, Teams und Organisationen

Wie eben eingeführt besteht eine Gruppe also aus Entitäten, die zueinander in Beziehung stehen. In der Psychologie wird die Größe einer Gruppe als ein wesentliches Merkmal betrachtet (Sader 1998). Auch wenn sich keine optimale Gruppengröße festmachen lässt, wird eine Gruppe häufig als ein überschaubares Gebilde wahrgenommen.

Ein Team dagegen ist eine besondere Form der Gruppe. Nach (Wiendieck 1994) gibt es in einem Team einen besonderen Zusammenhalt, der von einem ausgeprägten Gemeinschaftsgeist unterstützt wird. Häufig verfolgt das Team als Gruppe von bis zu 12 Personen ein fest definiertes gemeinsames Ziel. Die Motivation kommt dabei von innen, es wird also von allen Teilnehmern nach besten Kräften auf das Ziel zugearbeitet.

Der Begriff der Organisation kann zwei unterschiedliche Bedeutungen in sich tragen (Coutinho et al. 2005). Zunächst kann eine Organisation eine Einheit mit eigener Identität darstellen, die durch eine oder mehrere

Gruppen repräsentiert wird (Ferber et al. 2004). Im zweiten Fall bezeichnet eine Organisation die in einer Gruppe explizit oder implizit vorhandenen Strukturen, Normen oder Verhaltensweisen, die die Mitglieder zu einem bestimmten Handeln veranlasst.

Zusammenfassend lässt sich festhalten, dass Teams besondere Gruppen sind, in denen ein enger Zusammenhalt herrscht. Organisationen hingegen sind größere, teils anonyme Gruppen mit eigener Identität oder formal definierte Gebilde, die die Handlungen von Gruppen bzw. deren Mitgliedern bestimmen.

2.3.2 Bildung einer Gruppe

Bevor Teilnehmer in einer Gruppe zusammenwirken können, muss die Gruppe zunächst erstellt werden. In dynamischen Umgebungen ist die Zusammensetzung einer Gruppe aber nicht immer fest vorgegeben. Es ist daher notwendig, geeignete Verfahren anzuwenden, um die gewünschte Gruppe zu bilden. Das Problem basiert im Wesentlichen darauf, ausreichend Informationen über die anderen Agenten und deren Fähigkeiten zu erhalten. In (Vanzin und Barber 2006) werden unterschiedliche Verfahren erläutert. Da auf einige dieser Techniken im weiteren Verlauf der Arbeit (s. Kapitel 3.5, Weitere Techniken) zurückgegriffen wird, soll an dieser Stelle nur ein kurzer Überblick gegeben werden:

- Broadcasting, z.B. mit dem Kontraktnetz (Smith u. Davis 1988),
- Modellierung der Umgebung, d.h. Wiederverwendung von Informationen aus früheren Kontakten,
- zentrale Verzeichnisse und
- dezentraler Informationsaustausch, z.B. Distributed Matchmaking (s. Ogston und Vassiliadis 2001).

2.3.3 Beschreibung des Gruppenaufbaus

Die vorangegangenen Abschnitte deuten an, dass sowohl Teilnehmer als auch Struktur einer Gruppe nicht immer im Voraus festgelegt sind. Trotzdem sind Mittel zur Gruppenbeschreibung erforderlich, die bei der Bildung und Verwaltung unterstützen. In der Agentenwelt wird üblicherweise eine *agenten-* oder *organisationszentrierte* Sichtweise eingenommen (Hübner et al. 2002). Im ersten Fall stehen die Agenten im Mittelpunkt und gemeinsame Handlungen ergeben sich aus Mechanismen, die innerhalb der einzelnen Agenten spezifiziert sind. Der

zweite Ansatz betrachtet eine Organisation bereits als gegeben und es wird erwartet, dass sich die Agenten nach den enthaltenen Vorgaben richten.

Die Autoren schlagen für die organisationszentrierte Sicht eine Unterteilung in folgende Klassen vor, die man auch bei der Betrachtung von Gruppen einsetzen könnte:

1. **funktional**: beschreibt die Organisationen durch die Spezifikation von Plänen, Vorgaben für die Aufgabenverteilung, Koordinationsverfahren zur Ausführung globaler Pläne und Überwachungsmechanismen, um den Fortschritt und die Qualität zu kontrollieren.
2. **strukturell**: diese Modelle beschäftigen sich mit dem eher statischen Aspekt der Struktur, welcher häufig durch Rollen, Beziehungen zwischen Rollen, Rechte und Pflichten aus Rollen und Gruppen von Rollen beschrieben wird. Das globale Ziel wird durch Erfüllung der jeweiligen Rollen von Agenten erreicht.
3. **deontische** Relation: Die funktionale und strukturelle Dimensionen können meist unabhängig von einander definiert werden. Diese dritte Dimension erlaubt es, Beziehungen zwischen der *strukturellen* und *funktionalen* Sicht festzulegen.

Durch die Definition sowohl der *funktionalen* als auf der *strukturellen* Dimension kann ein Multiagentensystem effizienter arbeiten. Dies liegt daran, dass beim Fehlen der *funktionalen* Dimension, immer passende Pläne vorhanden sein müssen, unbekannte Probleme also nicht gelöst werden können. Wenn keine Struktur definiert wurde, muss hingegen jedes Mal ein neuer globaler Plan entwickelt werden, wenn die Agenten zusammenarbeiten wollen.

2.3.4 Rollen

In der strukturellen Definition tauchte bereits der Begriff der Rolle zur Beschreibung von Gruppen auf. Hier soll dieses Konzept, welches im Bereich der Multiagentensysteme häufig zum Einsatz kommt, vertieft werden. Meist geht es darum, das Verhalten eines Agenten auf abstrakte Weise zu beschreiben.

> *A role is a class that defines a normative behavioral repertoire of an agent. (Odell et al. 2003, S. 29)*

Eine Rolle enthält verschiedene Regeln, Wissen und Fähigkeiten, die ein bestimmtes Verhaltensmuster charakterisieren, wobei eine Rolle auch aus mehreren anderen Rollen zusammengesetzt werden könnte. Sie werden zwar unabhängig von Gruppen definiert, müssen aber innerhalb einer solchen zum Einsatz kommen. Nach (Cabri et al. 2003) bewirken Rollen ein bestimmtes, vorhersagbares Verhalten und enthalten die für eine Gruppe wichtigen Verhaltensmuster. Ein Agent, der eine bestimmte Rolle einnimmt, sollte somit auf eine festgelegt Menge von Ereignissen reagieren können (Cabri et al. 2004).

Ähnlich betrachten Steegmans et al. (2004) Rollen als einen Teil der Beschreibung eines Agenten, welche die Komponenten enthält, die mit einem bestimmten Verhalten in Beziehung stehen. Ein Agent, welcher nur eine Rolle einnehmen kann, wird vollständig von dieser beschrieben, ansonsten wird der Agent durch die Menge aller Rollen definiert, die der Agent einnehmen kann. Die Autoren unterscheiden bei der Zusammensetzung von Rollen zunächst zwischen Zuweisung, d.h. welche Rollen ein Agent grundsätzlich ausüben könnte, und Auswahl, die bestimmt, wann die zugewiesenen Rollen zum Einsatz kommen. Je nachdem, ob die Zuweisung und Auswahl zur Entwicklungszeit oder während der Laufzeit durchgeführt wird, ergeben sich drei Möglichkeiten:

	Zuweisung	Auswahl
statisch	Entwicklung	Entwicklung
adaptiv	Entwicklung	Laufzeit
dynamisch	Laufzeit	Laufzeit

Rollen fördern damit sowohl die Wiederverwendbarkeit als auch die Anpassbarkeit von Software, da Rollen eine abstrakte Spezifikation über die Fähigkeiten eines Agenten enthalten (Cabri et al. 2004, S. 246). Steegmans et al. (2004) vertreten daher die Ansicht, dass Rollen ein Schlüsselelement bei der Entwicklung von Multiagentensystemen sind.

Neben der vorangegangenen Sichtweise, kann eine Rolle auch als Träger von Rechten und Pflichten aufgefasst werden (Mao und Yu 2004). Der Agent erhält durch die Übernahme einer Rolle die Möglichkeit, bestimmte Ressourcen zu nutzen. Andererseits verpflichtet er sich, die mit der Rolle verbundenen Anforderungen (z.B. Lösung bestimmter Aufgaben oder Reaktion auf spezifische Ereignisse) zu erfüllen.

2.3.5 Vorteile von Gruppenarbeit

Zum Abschluss dieses Abschnitts soll der Frage nachgegangen werden, was der Nutzen bei der Zusammenarbeit in Gruppen bzw. Verwendung des Gruppenkonzepts beim Systementwurf sein kann. Denn trotz erhöhtem Verwaltungsaufwand und dem Problem, geeignete Teilnehmer zu finden, lassen sich sowohl für das Individuum in der Gruppe als auch für Systementwickler verschiedene Vorteile ausmachen:

- Der Kommunikationsaufwand kann sich vereinfachen, da es nicht mehr notwendig ist, das gesamte System einzubeziehen. Beispielsweise genügt es, Nachrichten nur an die Gruppenmitglieder und nicht allen Teilnehmern des Systems zu senden. Auch kann der Gruppenverwalter als Ansprechpartner dienen, so dass externe Agenten auf diesen zurückgreifen können, ohne Kenntnis vom internen Aufbau der Gruppe zu benötigen.
- Die Gruppe ermöglicht es, eine Sichtweise einzunehmen, die von einzelnen Aktionen eines Individuums abstrahiert. Die Teilnehmer müssen somit nicht mehr alle Details der anderen kennen und können sich auf ihre Aufgabe konzentrieren.
- Verschiedene Mechanismen ermöglichen die Förderung der Sicherheit. Wenn z.B. nur innerhalb der Gruppe kommuniziert wird, ist es Außenstehenden nicht möglich, an sensible Informationen zu gelangen.
- Durch den Zusammenschluss in einer Gruppe können die Teilnehmer als Einheit betrachtet und angesprochen werden. Dies ist häufig in militärischen Szenarien anzutreffen, in denen beispielsweise auf einer höheren Ebene nur noch selten von einzelnen Soldaten sondern von Einheiten, Verbänden und ähnlichem gesprochen wird.
- Durch die Gruppe können den angeschlossenen Teilnehmern die Fähigkeiten, Rechte und Möglichkeiten der anderen zugänglich gemacht werden. So sind auch sie Aufgaben gewachsen, zu denen ein Einzelner nicht in der Lage gewesen wäre.
- Innerhalb einer Gruppe können Aufgaben in Teilprobleme zerlegt und verteilt werden. Dies ermöglicht meist eine schnelleres oder präziseres Erreichen des gewünschten Ziels.

Diese Aufzählung zeigt, dass gemeinsames Handeln in dynamischen Umgebungen zu signifikanten Vorteilen führen kann und daher auch in Softwaresystemen genutzt werden sollte.

2.3.6 Zusammenfassung

Gruppen sind ein mächtiges Konzept, das erlaubt, Entitäten mit gemeinsamen Eigenschaften (z.B. Zielen) zusammenzufassen. Struktur und Teilnehmer sind dabei nicht immer im voraus bekannt, was die Gruppe zu einem flexiblen Konstrukt in dynamischen Umgebungen macht, aber auch entsprechende Verfahren bei der Bildung und Verwaltung von Gruppen erfordert.

Neben der strukturellen Beschreibung kann die Gruppe auch aus funktionaler Sicht untersucht werden, wobei Beziehungen zwischen beiden Dimensionen möglich sind. Rollen sind dabei ein Instrument, um die Struktur und Aufgabenverteilung innerhalb der Gruppe zu beschreiben. Sie ermöglichen, von Details der Implementation zu abstrahieren und bewirken ein bestimmtes Verhalten des Rollennehmers.

2.4 Koordination und Kooperation

Koordination und Kooperation sind essentielle Bestandteile eines Multiagentensystems. Erst diese Verfahren ermöglichen das Zusammenspiel mehrerer Agenten innerhalb einer Gruppe. In diesem Kapitel sollen beiden Begriffe näher erläutert und deren Relevanz für meine Arbeit aufgezeigt werden.

> *„Coordination is managing dependencies between activities.“ (Malone und Crowston 1994, S. 90)*

Diese Definition drückt aus, dass Koordination sich mit der Abstimmung von Aktivitäten beschäftigt. Sie fordert aber auch, dass zwischen diesen Handlungen Beziehungen bestehen. Wie die Autoren treffend formulieren, ist nur dort Koordination notwendig, wo auch wechselseitige Abhängigkeiten vorhanden sind. In (Ferber 2001) und auch (Nwana et al. 1996) wird dies präzisiert:

1. Wissen, Fähigkeiten oder Rechte sind auf unterschiedliche Entitäten verteilt und sollen gemeinsam genutzt werden.
2. Die verfügbaren Ressourcen sind begrenzt und können nicht von allen Entitäten gleichzeitig genutzt werden.
3. Der Aufwand zur Lösung eines Problems soll optimiert werden.
4. Erhöhung der Effizienz (z.B. durch parallele Handlungen).

5. Es bestehen wechselseitige Abhängigkeiten zwischen Aktionen oder Zielen der einzelnen Entitäten.
6. Vermeidung von Konflikten, doppelter Bearbeitung und Ähnlichem, da jede Entität meist nur eine eingeschränkte Weltsicht besitzt.

In jedem Koordinationsvorgang sind dabei drei Schlüsselelemente enthalten (Schumacher 2001, S. 34). Dazu zählen neben den an der Koordination *beteiligten Entitäten* und einem *Medium,* über das die Koordination abgewickelt wird, auch entsprechende *Regeln,* die beschreiben, wie das Koordinationsverfahren im Detail abläuft. Insgesamt kommt Ossowski (1999) zu dem Ergebnis, dass Koordination vor allem dann eingesetzt wird, wenn im System eine höhere Komplexität erreicht werden soll. Häufig weisen die koordinierten Einheiten auch eine bestimmte Struktur (z.B. Hierarchie) auf oder sollen in sich als ein Ganzes aufgefasst werden.

2.4.1 Koordination in Multiagentensystemen

Da Agenten in sich abgeschlossene Einheiten bilden, die zu autonomem Handeln fähig sind, ist es sinnvoll, bei Bedarf die von den Agenten ausgeführten Aktionen zu koordinieren. Um die Wiederverwendbarkeit eines Agenten zu erhöhen, sollten möglichst flexible Koordinationsverfahren eingesetzt werden. Von Martial (1990) hat eine Taxonomie entwickelt, welche die Beziehungen zwischen Plänen verschiedener Agenten untersucht.

Eine negative Beziehung bedeutet, dass beide Pläne nicht wie beabsichtigt ausgeführt werden können. Dies kann entweder einen Konflikt um eine physikalische Ressource bedeuten oder die Ziele schließen sich gegenseitig logisch aus. Im Gegensatz dazu profitieren die Agenten bei der Verbindung von Plänen mit positiver Beziehung. Hier kann zwischen expliziter Anforderung von Hilfe bei der Planausführung und impliziten Effekten unterschieden werden. Wenn sich zwei Agenten beispielsweise unabhängig zu den gleichen Handlungen entscheiden, genügt es, wenn ein Agent diese auch tatsächlich ausführt. Um die Leistung des Gesamtsystems zu erhöhen, sollte ein Agent daher in der Lage sein, entsprechende Situationen zu erkennen, um die sich ergebenden Vorteile zu nutzen.

2.4.2 Formen der Koordination

Auf Basis der in (Huhns u. Stephens 1999) vorgestellten Taxonomie (s. Seite 83, Abbildung 2.1) kann die Koordination zwischen Agenten zunächst in kooperative und kompetitive Verfahren (Braubach 2007) unterteilt werden (s. Abbildung 3). Ähnlich bezeichnen Nwana et al. (1996) sowie Malone und Crowston (1994) Kooperation und Wettbewerb als Formen der Koordination.

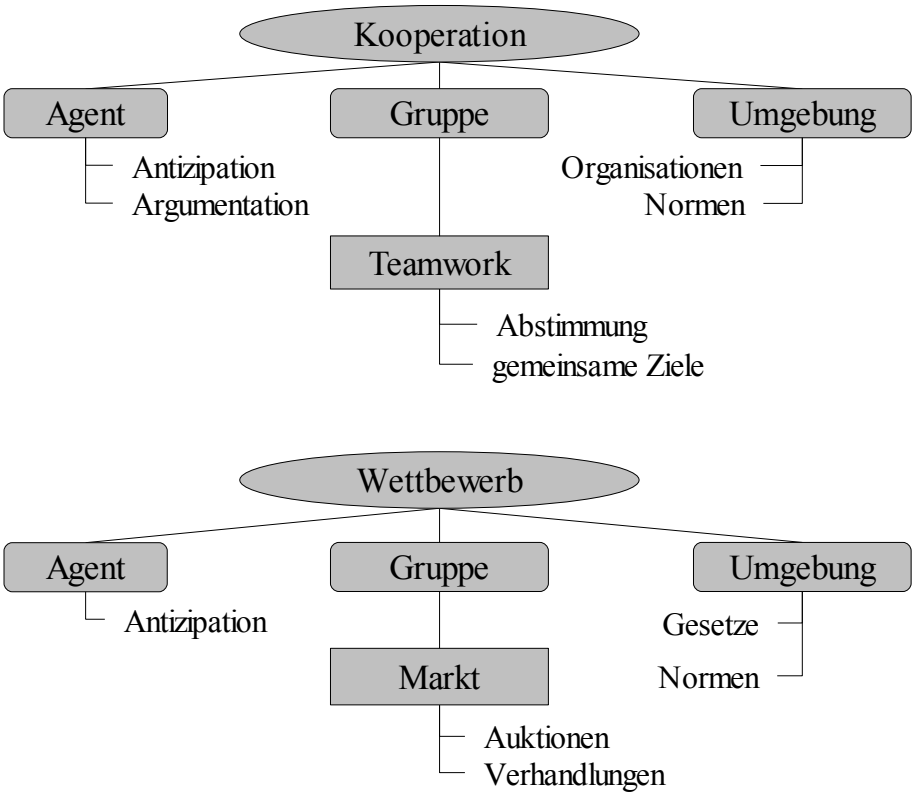

Abbildung 3: Verfahren der Kooperation und des Wettbewerbs

Ferber (2001) stellt drei Kriterien vor, deren Untersuchung die Einordnung von Verfahren erleichtern:

1. Inwieweit decken sich die Ziele einzelner Agenten oder stehen sie im Widerspruch zueinander?
2. Sind die Ressourcen im System begrenzt und können Konflikte um deren Nutzung entstehen?
3. Sind Agenten in der Lage, die ihnen gestellten Aufgaben selbständig zu lösen oder sind sie auf Hilfe durch andere Agenten angewiesen?

Widersprüche oder Konflikte um Ressourcen sind meist Anzeichen kompetitiver Verfahren, wohingegen die gemeinsame Lösung von Aufgaben kooperative Techniken kennzeichnet.

2.4.3 Wettbewerb

Kompetitiven Verfahren ist gemein, dass sich die beteiligten Agenten in einer Form von Wettbewerb befinden. Neben begrenzten Ressourcen ist auch das eigennützige Handeln ein Indiz für eine Wettbewerbssituation. Die meisten Marktmechanismen beruhen dabei auf dem Prinzip von Angebot und Nachfrage (z.B. bei Versteigerung einer Ressource).

Die Aufgaben werden ohne explizite Planung an die Agenten verteilt, wobei die Mechanismen meist Protokolle definieren, in denen genau festgelegt ist, wie die Entscheidung gebildet wird. Zwischen den Teilproblemen sollten möglichst wenig Beziehungen bestehen, um Konflikte zwischen Aktionen einzelner Agenten und damit weiteren Koordinationsbedarf während der Ausführung zu minimieren. Typische Beispiele sind Auktionen oder Vermittlungs- bzw. Verhandlungsstrategien.

Auktionen

Auktionen sind eine einfache Form, um den Zugriff auf Ressourcen zu steuern. Auch einzelne Aufgaben können versteigert werden. Wie in der Realität wird der Gegenstand zu einem bestimmten Preis angeboten. Je nach Auktionstyp können die Agenten den Preis erhöhen (englische Auktion) oder das Angebot annehmen, wenn der Preis ausreichend gefallen ist (holländische Auktion). Für komplexe Interaktionsbeziehungen oder das Aushandeln umfangreicher inhaltlicher Werte sind Auktionen allerdings weniger gut geeignet (Braubach 2007).

Verhandlungen

In einer Verhandlung geht es darum, zwischen den beteiligten Agenten einen Konsens zu erzielen (Malone und Crowston 1994). Häufig werden dabei mehrere Runden durchlaufen, in denen die Agenten ihre Angebote verändern oder auf Veränderungen reagieren. Das *Kontraktnetzprotokoll* ist ein weitverbreitetes Beispiel, welches die dynamische Zuweisung zu den aktuell besten Möglichkeiten erlaubt. Das Verfahren selbst wird u.a. in (Smith u. Davis 1988) beschrieben.

Gesetze und Normen

Gesetze und Normen (Wooldridge 2002) schreiben bestimmte Verhaltensweisen vor. Der Unterschied besteht darin, dass die Überschreitung eines Gesetzes in der Regel mit empfindlichen Strafen belegt wird, wohingegen eine Normüberschreitung meist nur geächtet wird. Ein typisches Beispiel für Norm ist das Anstellen an das hintere Ende einer Schlange. Gesetze sind den meisten hingegen aus dem Straßenverkehr bekannt, wo beispielsweise die Missachtung zu teuren Strafen führt. In beiden Fällen wird deutlich, dass durch die Vorschriften, der Zugriff auf eine begrenzte Ressource koordiniert werden soll.

2.4.4 Kooperation

Ähnlich wie für den Begriff der Koordination ist in der Literatur eine Vielzahl von Definitionen für Kooperation zu finden.

> *„An agent, A, is said to cooperate with another agent, B, if they are both autonomous, and A has adopted the goal of B.“ (d'Inverno u. Luck 2004, S. 58)*

Ein wichtiges Element der Kooperation ist demnach, dass ein Agent selbstständig ein Ziel eines anderen Agenten übernimmt und verfolgt. Wenn beide miteinander kooperieren ergibt sich ein gemeinsam verfolgtes Ziel, wodurch die beteiligten Agenten in Beziehung gesetzt werden. Nwana et al. (1996) merken zu dem an, dass Kooperation zwar die Koordination mehrerer Agenten zum Ziel hat, koordiniertes Handeln aber nicht zwangsläufig die Folge von Kooperation ist. Beispielsweise kann die Koordination aufgrund von Fehleinschätzungen trotz vorliegender Kooperationsbemühungen misslingen.

Abstimmung

Um eine Aufgabe zu lösen, können Agenten ihre Aktionen direkt aufeinander abstimmen. Ein typisches Beispiel ist die *Multiagentenplanung*, bei der ein gemeinsamer Plan entwickelt und abgearbeitet wird. Es lassen sich dabei drei Varianten unterscheiden, wie der Planungsprozess ablaufen kann (Durfee 2001):

1. **Zentrale Planung verteilter Pläne**: Eine zentrale Einheit erstellt einen Plan und verteilt diesen auf die Agenten.

2. **Verteilte Planung eines Gesamtplans**: Eine Gruppe von Agenten erzeugt gemeinsam einen Plan, der dann auf die Agenten aufgeteilt wird. Die ausführenden Agenten müssen nicht unbedingt der planenden Gruppe angehören.
3. **Verteilte Planung für verteilte Pläne**: Durch Kooperation entwickelt jeder Agent langsam einen individuellen Plan, wobei die einzelnen Handlungen koordiniert werden.

Koordination in Multiagentenplänen

Da die Koordination von Aktionen mit der reinen Erstellung eines Plans nicht zwangsläufig abgeschlossen ist, unterscheidet Durfee (2001) drei Zeitpunkte, die hierfür in Frage kommen:

1. **Koordination nach der Planung**: Während der Ausführung des verteilten Plans werden die einzelnen Aktionen von den Beteiligten koordiniert. Jeder Agent überwacht den Erfolg seiner Handlungen und teilt den anderen auftretende Probleme mit. Dies ermöglicht die nachträgliche Anpassung der Pläne an neue Gegebenheiten.
2. **Koordination vor der Planung**: Durch organisatorische Strukturen, Normen oder Gesetze wird im Voraus festgelegt, wie die Agenten Handlungen auszuführen haben. Es werden somit Richtlinien geschaffen, die von den Agenten bei der Ausführung von Plänen berücksichtigt werden müssen, ohne dass es einer zusätzlichen nachträglichen Koordination bedarf.
3. **Laufende Planung während der Ausführung**: Der Plan wird dynamisch zur Laufzeit erstellt, was bedeutet, dass Agenten Planung, Koordination und Handlungen nebenläufig oder abwechselnd durchführen.

Gemeinsame Ziele

Das Konzept *Gemeinsame Ziele* orientiert sich an der menschlichen Zusammenarbeit und greift daher auf mentale Eigenschaften zurück, wie sie in dem oben bereits eingeführten BDI-Modell verwendet werden. Allerdings beziehen sich hier die Ziele und Intentionen nicht nur auf den Agenten selbst, sondern auf die gesamte Gruppe, d.h. die Agenten verpflichten sich zur gemeinsamen Erfüllung eines Ziels. Da diese Form

der Kooperation für den weiteren Verlauf der Arbeit wesentliche Aspekte beinhaltet, wird hier nicht weiter darauf eingegangen, sondern die Details werden im nachfolgenden Kapitel 3.1 (Joint Intentions) besprochen.

Rollenmodelle und Organisationen

Wenn ein Agent eine bestimmte Rolle einnimmt, werden ihm damit bestimmte Rechte aber auch Pflichten zugeteilt. Es wird dabei erwartet, dass der Agent den vordefinierten Richtlinien folgt und sich entsprechend verhält. Eine Organisation legt eine vordefinierte Umgebung fest, die die Rollen, Aufgaben und Interaktionen der Agenten bestimmen. Unter anderem lassen sich mit Prioritätsregeln, Zugriffe auf beschränkte Ressourcen regeln oder klassische Master-/Slave-Systeme aufbauen, in der ein Agent Aufgaben an die ihm unterstellten Agenten verteilt.

Das im Kapitel 3.2 vorgestellte *AGR*-Modell (Ferber et al. 2004) erlaubt auf einfache Weise die Strukturierung von Agenten in Gruppen mittels Rollen erlaubt. Diese können wiederum in Organisationen zusammengefasst werden und damit das Verhalten beeinflussen.

2.4.5 Cooperative Distributed Problem Solving

> *„CDPS studies how a loosely-coupled network of problem solvers can work together to solve problems that are beyond their individual capabilities.“ (Durfee et al. 1989, S. 63)*

CDPS beschäftigt sich mit der Frage, wie mehrere unabhängige Einheiten zusammenarbeiten können, um Aufgaben zu lösen, die über die Fähigkeiten eines einzelnen Agenten hinausgehen. Es erfordert Verfahren, welche die Möglichkeit bieten, unter Berücksichtigung von Fähigkeiten und Ressourcen den Agenten Aufträge zu erteilen. Zerlegung und Verteilung stehen in einem engen Zusammenhang, da die Zuweisung von Aufgaben einfacher ist, wenn die Teilaufgaben Bezug zu den Fähigkeiten oder Ressourcen des jeweiligen Agenten haben.

Die verteilte Problemlösung setzt zunächst die Existenz einer Aufgabe und bestimmte Erwartungen über die Art der Lösung voraus (Durfee 2001). Zudem muss für die Agenten ein Anreiz bestehen, an der Arbeit teilzunehmen (*Kohärenz*). Eine weitere Voraussetzung ist, dass die Agenten wissen, wie die Gruppe am besten zusammenarbeitet (*Kompetenz*). Neben der tatsächlichen Zerlegung des Problems und der darauf folgenden Zusammenführung der Ergebnisse spielt die Optimierung der Verfahren eine wichtige Rolle. Außerdem sollten die

eingesetzten Techniken so ausgerichtet sein, dass die Aktivitäten der Agenten sinnvoll koordiniert werden, da es sonst zu Behinderungen oder doppelter Arbeit kommen kann.

Abbildung 4: Schritte des CDPS-Prozesses

Wie Abbildung 4 zeigt, kann der CDPS-Prozess im Allgemeinen als ein Drei-Stufen-Prozess aufgefasst werden (Smith u. Davis 1988). Daneben stellt Wooldridge (2002) auch einen Modell mit vier Phasen vor. Dieses wird in Kapitel 3.1.4 (Jennings und Wooldridge) näher erläutert.

Bevor ein Problem verteilt werden kann, muss es in entsprechende Teile zerlegt werden. Dies erfolgt meist hierarchisch, wobei die unterschiedlichen Ebenen verschiedene Abstraktionsstufen darstellen. Eine wichtige Frage dabei ist, wie weit das Problem zerlegt werden soll bzw. kann und wer welches Teilproblem erhält. Dies ist besonders im Hinblick auf die Fähigkeiten der Agenten relevant, damit jeder Agent genau die Aufgaben erhält, die er am besten lösen kann. In (Ferber 2001) wird allerdings festgestellt, dass es bis heute kein System zur vollautomatischen Zerlegung von beliebigen Problemen gibt. Daher muss der Systementwickler bereits im Voraus entsprechende Vorkehrungen treffen, wie die in (Bond u. Gasser 1988) vorgestellten Verfahren.

Im nächsten Schritt wird versucht, für jedes Teilproblem einen passenden Agenten zu finden. Auch hierfür stellen Bond und Gasser (1988) eine Reihe von Verfahren (z.B. Marktähnliche Techniken, Multiagentenplanung oder Organisationen) vor. Die eigentliche Lösung des Teilproblems ist Aufgabe des jeweiligen Agenten. Er kann sein Teilproblem aber auch weiter zerlegen und verteilen. Oftmals ist es sinnvoll, wenn die Agenten auch weiterhin in Kontakt stehen und Informationen austauschen. Dies spiegelt sich in den weiter unten vorgestellten *Task*- & *Result-Sharing*-Strategien wieder.

Der letzte Schritt besteht üblicherweise darin, die gewonnenen Teilerkenntnisse in eine Gesamtlösung zu integrieren. An dieser Stelle können Probleme entstehen, wenn sich die Teilergebnisse widersprechen.

Task- und Resultsharing

In einem CDPS-Prozess sind häufig die Strategien des *Task Sharing* oder *Result Sharings* zu beobachten (Wooldridge 2002). In einigen Systemen werden aber auch die Vorteile beider Strategien vereint. Beim *Tasksharing* besteht die Hauptaufgaben in der Zerlegung des gegebenen Problems in Teilstücke und deren Zuweisung an entsprechende Agenten. Diese können dann entweder die jeweiligen Teilaufgaben selbst lösen, oder aber sie zerlegen sie in weitere Teile, welche sie dann anderen Agenten zuweisen. Zum Schluss werden die Ergebnisse gesammelt und zu einer Gesamtlösung zusammengeführt.

Bei der Technik des *Resultsharings* wird das Problem von allen Agenten gleichzeitig bearbeitet, wobei die Agenten versuchen, sich schrittweise durch Lösen kleiner Probleme dem Gesamtergebnis anzunähern. Die Gruppe kann in folgender Weise von dem Verfahren profitieren (Durfee 2001):

1. größeres **Vertrauen** in ein richtiges Ergebnis, da ggf. mehrere Agenten die gleiche Lösung erarbeiten,
2. Erhöhung der **Vollständigkeit** eines Ergebnisses,
3. **Präzision** bedeutet Verfeinerung des eigenen Ergebnisses durch Lösungen anderer Agenten,
4. **Beschleunigung** der Problemlösung, da beispielsweise bereits erarbeitete Lösungen frühzeitig erkannt werden.

Weil die Übertragung zu vieler Teilinformationen das Netz oder einzelne Agenten überlasten könnte, sollten entsprechende Verfahren dafür sorgen, dass nur relevante Daten gesendet werden. Dies kann *proaktiv* (ein Agent weiß, dass ein anderer Agent sich mit einer bestimmten Aufgabe befasst) oder *reaktiv* (durch Auslösen vorgegebener Bedingungen) geschehen.

2.4.6 Zusammenfassung

Koordination von Handlungen und Kooperation zur Problemlösung sind wesentliche Elemente bei der Zusammenarbeit in Gruppen. Je nach Problemstellung und Teilnehmern gibt es verschiedene Formen, die sich im Allgemeinen in kooperative und kompetitive Verfahren unterteilen

lassen. Innerhalb dieser Klassen haben sich viele unterschiedliche Techniken (wie gemeinsame Ziele, die Multiagentenplanung, Auktionen und weitere) entwickelt. Der *CDPS-Prozess* beschreibt in einem Drei-Stufen-Prozess, wie die verteilte Problemlösung durch Kooperation ablaufen kann.

3 Koordinationstechniken

In diesem Kapitel werden verschiedene Koordinationsmechanismen im Detail vorgestellt. Dabei wird besonders auf die *Joint Intention*-Theorie eingegangen, da sie ähnlich wie das BDI-Modell auf mentalen Konzepten beruht. Im Anschluss daran wird mit *AGR* ein organisationsbezogener Ansatz und die zugehörige Implementation *Madkit* präsentiert. Als nächstes werden mit *STEAM* und *SimpleTeams* zwei Ansätze untersucht, für die jeweils entsprechende Implementationen realisiert wurden. Zum Schluss wird das Kapitel mit kurzen Erläuterungen zu Marktmechanismen und weitere Techniken abgerundet.

3.1 Joint Intentions

Gemeinsame Ziele und Intentionen sind wichtige Instrumente bei der Einbeziehung von BDI-Konzepten in das Teamwork von Agenten. Die Agenten einer Gruppe besitzen die Verpflichtung ein bestimmtes Ziel gemeinsam zu erreichen. Diese gemeinschaftlichen, mentalen Zustände unterscheiden *Joint Intentions* auch von parallelen Zielen.

Nach (Wooldridge 2002) bieten *Joint Intentions* eine solide und vorhersehbare Grundlage für soziale Interaktionen, sind aber dennoch flexibel genug, um sich Änderungen anzupassen. Der wesentlichen Unterschied zu den oben eingeführten individuellen Intentionen besteht darin, dass jeder Agent auch gegenüber den anderen eine Form von Verantwortung besitzt (Levesque et al. 1990). Daraus entwickelte Jennings *Commitments und Conventions*, die einerseits die Verpflichtung und andererseits die Überwachung des Zustands einer gemeinsamen Intention beinhalten. Bei der Modellierung von *Joint Intentions* sollten die folgenden Eigenschaften berücksichtigt werden (Bratman 1992):

1. *Commitment to joint activity*: Alle Teilnehmer verfügen über entsprechende Verpflichtungen zur gemeinsamen Handlung. Dies bedeutet, jeder besitzt den Willen teilzunehmen und seine Aufgaben wahrzunehmen.

2. *Mutual Responsivness*: Die Agenten sind gegenseitig an den Zielen und Intentionen der anderen interessiert, wobei jeder versucht, mit Rücksicht auf die anderen zu handeln.
3. *Commitment to mutual support*: Die Teilnehmer sehen sich verpflichtet, die anderen zu unterstützen und über den Fortschritt ihrer Handlungen zu informieren. Analog erwarten sie entsprechende Unterstützung bzw. Informationen.

In den folgenden Unterabschnitten werden mehrere Ansätze zur Bildung von *Joint Intentions* vorgestellt.

3.1.1 Das Modell von Cohen & Levesque

In (Cohen u. Levesque 1991) werden gemeinschaftliche Handlungen als Aktionen eingeführt, die von Individuen durchgeführt werden. Allerdings verfügen alle Beteiligten über gemeinsame mentale Eigenschaften.

> *„This paper ... characterizes joint intentions as shared commitments to perform an action (typically composite) while the group is in a certain shared mental state." (Levesque et al. 1990, S. 94)*

In (Cohen u. Levesque 1991) wird zudem dargestellt, wie man die aus ihrer früheren Arbeit (Cohen u. Levesque 1990) aufgestellten Definitionen für Ziele und Intentionen generalisieren kann. Die Autoren stellen dabei fest, dass es nicht genügt, gemeinsame Verpflichtungen als individuelle Verpflichtung mit einer Gruppe als Agent zu definieren (Levesque et al. 1990). Dies liegt daran, dass die Agenten innerhalb der Gruppe nicht immer über das gleiche Wissen verfügen, so dass daher unterschiedliche Ansichten über den Zustand eines Ziels existieren können.

Ein weiteres Problem liegt darin, dass ein Ziel nicht einfach verworfen werden darf, wenn ein Agent glaubt, es wäre nicht mehr zu erreichen. Vielmehr muss die Gruppe das Ziel gemeinsam aufgeben, was aber nur dann möglich ist, wenn alle Agenten die entsprechenden Umstände kennen. Daher sollte ein Agent, der erkannt hat, dass ein bestimmtes Ziel unerfüllbar geworden ist, dieses nicht vollständig verwerfen, sondern ein neues Ziel bekommen, dessen Inhalt die Bekanntmachung des Umstand in der Gruppe ist. Hierfür wird das *Weak Achievement Goal (WAG)* eingeführt (Cohen u. Levesque 1991, S. 498[1]):

1 Die Seitenangabe bezieht sich auf die gebundene Zeitschriftensammlung der Universität Hamburg (Signatur Z116,25).

Definition: Ein Agent α aus der Gruppe γ besitzt ein *WAG* für ein Ziel ∂ mit der Motivation ω, wenn

1. α ein reguläres Ziel für ∂ besitzt (d.h. α glaubt, ∂ ist derzeit unerfüllt und will es erfüllen) und
2. α glaubt, dass ∂ erfüllt wurde, irrelevant ist oder sogar nie erfüllt werden kann. Der Agent α verfügt aber über das Ziel, dass der Zustand von ∂ allen Agenten gemeinsam bekannt werden soll.

Wenn ein Agent ein *WAG* für ein Ziel ∂ besitzt, bedeutet dies, dass der Agent ein reguläres Ziel ∂ hat und gleichzeitig versucht, die neuen Informationen über das Ziel ∂ bekannt zu geben. Im nächsten Schritt wird mit Hilfe des *WAG* das *Joint Persistent Goal (JPG)* eingeführt, welches eine gemeinsame Verpflichtung aller beteiligten Agenten über ein bestimmtes Ziel darstellt (Cohen u. Levesque 1991, S. 499).

Definition: Eine Gruppe γ von Agenten verfügt demnach über ein *JPG* ∂ mit einer Motivation ω, wenn

1. alle Agenten aus γ gemeinsam glauben, dass das Ziel ∂ noch nicht erfüllt ist,
2. alle Agenten aus γ gemeinsam glauben, dass jeder Agent die Erfüllung des Ziels ∂ beabsichtigt (also jeder Agent über ein Ziel für ∂ verfügt) und
3. alle Agenten aus γ gemeinsam glauben, dass alle mindestens ein *WAG* vom Ziel ∂ haben, bis sie gemeinsam wissen, dass ∂ erfüllt wurde, unerreichbar oder irrelevant geworden ist.

Grundsätzlich verpflichtet ein *JPG* jedes Gruppenmitglied dazu, das Ziel und ggf. ein *WAG* des Ziel zu behalten, bis allen Agenten bekannt ist, dass das Ziel erfüllt wurde, unerreichbar oder irrelevant geworden ist. Diese Verpflichtung ist das Schlüsselelement eines *JPG*, da es sicherstellt, dass alle Gruppenmitglieder den Zustand des Ziels kennen. Ähnlich wie eine individuelle Intention eine Verpflichtung zu bewusstem Handeln darstellt, können nun *Joint Intentions* als eine gemeinschaftliche Verpflichtung zu einer kollektiven Aktion in einem gemeinsamen mentalen Zustand aufgefasst werden.

Definition: Eine Gruppe von Agenten besitzt eine *Joint Intention* bezüglich einer Bedingung β für eine Aktion α, wenn

1. alle Agenten ein *JPG* bezüglich der Bedingung β besitzen, die Aktion α gemeinsam ausgeführt zu haben und

2. die Agenten während der gesamten Ausführung alle glauben, dass sie die Aktion gemeinsam erledigen.

Zum Abschluss des Artikels zeigen Cohen und Levesque (1991), dass sich aus einer *Joint Intention* auch individuelle Intentionen ergeben, ein Agent also durch eine *Joint Intention* auch die notwendigen eigenen Intentionen erlangt. Da dieses Modell auf den Arbeiten zu individuellen Intentions aufbaut, besitzt es aber auch die damit verbundenen Probleme (Details in Singh 1992). Ebenso werden viele wichtige Eigenschaften, wie beispielsweise die Bildung einer Gruppe, Planerzeugung oder Synchronisation, nicht behandelt (Haddadi 1995).

3.1.2 Rao, Georgeff und Sonnenberg

Das Modell von Rao, Georgeff und Sonnenberg (Rao et al. 1992) führt zwei unterschiedliche Arten von Agenten ein. Neben dem herkömmlichen, individuellen Agenten gibt es auch den *Social Agent*. Dieser besteht aus einer Menge individueller Agenten oder anderen *Social Agents*, was eine hierarchische Notation ermöglicht. Die Agenten benutzen *Social Plan Structures*, um bestimmte Ziele zu erreichen. Sie enthalten neben dem Plankörper mit den tatsächlichen Anweisungen eine Vorbedingung, die angibt, unter welchen Umständen der Plan ausgeführt werden kann. Weiterhin ist es wichtig zu wissen, wer (x) über einen Plan (p) verfügt und was dieser Plan leistet (π). Dies wird durch die Relation *has-plan(x p* π*)* ausgedrückt.

Gemeinsames Wissen ist in diesem Modell dasjenige Wissen, welches jeder in der Gruppe kennt und zudem glaubt, dass alle anderen auch über dieses Wissen verfügen (Rao et al. 1992). Wenn ein Agent ein *Social Agent* ist, bedeutet dies, dass auch alle Mitglieder der Gruppe über dieses Wissen verfügen. Auf ähnliche Weise werden gemeinsame Ziele und Intentionen definiert.

In kooperativen Szenarien dient die *Means-End Analyse* dazu, ein gemeinsames Ziel auszusuchen und dann einen entsprechenden *Social Plan* auszuwählen. Im Ergebnis sollen alle Agenten beabsichtigen, den ausgewählten Plan als *Joint Intention* auszuführen. Dieser Prozess wird durch Kommunikation und Synchronisation unterstützt. Wenn ein Agent einen Plan besitzt und die Aktionen des Plankörper ausführen möchte, verfügt der Agent auch über ein Ziel bzgl. des Plankörpers und er glaubt, dass die zugehörige Vorbedingung erfüllt ist. Diese Mindestanforderung

genügt aber nicht, um den Agenten zur Bildung von Intentionen anzuregen. Daher wird eine stärkere Definition vorgestellt (Rao et al. 1992, S. 71).

Definition: Wenn ein individueller Agent,

1. einen Plan besitzt,
2. das Ziel bzgl. des Plankörpers übernommen hat,
3. glaubt, dass die Vorbedingung wahr ist und
4. die *Deliberationsfunktion* den Plan ausgewählt hat,

dann besitzt der Agent auch die Intention, den Inhalt des Plankörpers auszuführen.

Die Deliberationsfunktion beschreibt, ob der Agent sich entschieden hat, ein Ziel mit einem bestimmten Plan zu erreichen. Diese für einen individuellen Agenten geltenden Regeln können entsprechend auf Agentengruppen ausgeweitet werden.

Definition: Ein *Social Agent* hat eine *Joint Intention*, den Inhalt des Plans auszuführen, wenn er

1. einen *Social Plan* besitzt,
2. ein gemeinsames Ziel bzgl. des Planinhaltes übernommen hat und
3. gemeinsam (sofern er aus mehreren Agenten besteht) glaubt, dass die Vorbedingung des Plans wahr ist.

Allerdings kritisiert Haddadi (1995), dass in diesem Fall keine Deliberationsfunktion einbezogen wird und es demnach entweder nur einen *Social Plan* geben darf oder dass die individuelle Deliberationsfunktion bei jedem Agenten den gleichen Plan auswählt. Auch gibt es keine Mechanismen, um über den Erfolg einer einmal gebildeten *Joint Intention* zu reflektieren. Die vorgestellte Theorie erfüllt die ersten beiden der oben vorgestellten Eigenschaften (Rao et al. 1992, S. 68ff.):

1. *Commitment to joint activity*: Die Theorie erfüllt diese Eigenschaft, wobei allerdings die Bereitschaft zur Teilnahme eines Agenten nicht berücksichtigt wird. Es kann also auch dann zu einer *Joint Intention* kommen, wenn ein Agent zwar im Sinne der gemeinsamen Intention handelt, dies aber von sich aus gar nicht will.
2. *Mutual Responsivness*: wird vollständig erfüllt.

3. *Commitment to mutual support*: Diese Eigenschaft wird nicht unterstützt, wobei die Autoren aber argumentieren, dass dies auch nicht notwendig sei, da entsprechende Kommunikation eine Folge einer *Joint Intention* sei. Dies habe den Vorteil, dass die Art der Kommunikation nicht festgelegt werde, und so offen für verschiedene organisatorische Strukturen sei.

3.1.3 Kinny et al. (Planned Team Activity)

Mit Planned Team Activity (Kinny et al. 1994) bezeichnen die Autoren eine Mengen von kooperierenden Agenten. Die Arbeit baut auf die in (Rao et al. 1992) beschriebenen Konzepte auf und beschäftigt sich speziell mit den bei der Ausführung von Plänen auftretenden Problemen:

1. Wie kann ein passendes Team für eine bestimmte Aufgabe gefunden werden (Teambildung)?
2. Synchronisation der übernommenen gemeinsamen Ziele/Intentionen (Mind-Set Synchronisation).
3. Wie wird die Verantwortung für einen bestimmten Teil des Plans an die Mitglieder der Gruppe zugewiesen (Role Assignment)?
4. Wie wird die zeitliche Beziehungen zwischen der Ausführung von Teilplänen verwaltet (Role Synchronisation)?
5. Wie soll zwischen alternativen Plänen ausgewählt werden (Plan Selection)?
6. Wie sollen sich die Agenten verhalten, wenn ein Plan fehlschlägt?

Das Modell geht davon aus, dass alle Pläne und Fähigkeiten der Agenten im Voraus bekannt sind, was es ermöglicht, bereits vor der Ausführung festzustellen, welche Ziele von einer Gruppe möglicherweise erreicht werden können. Teambildung, Planauswahl und Rollenzuweisung werden durch Wissen um die Fähigkeiten und Pläne eines Agenten bzw. Teams erreicht. Das Modell geht ähnlich wie die Ursprungsarbeit von einer rekursiven Notation der Gruppen aus. Jede Gruppe verfügt über eine Sammlung von Plänen, was aber nicht bedeutet, dass diese Pläne beabsichtigt oder ausgeführt werden.

Definition: Ein Plan wiederum besteht aus

1. einem Körper, der eine Abfolge von Aktionen enthält,
2. einer Vorbedingung, die anzeigt unter welchen Umständen ein Plan zur Ausführung gebracht werden kann und

3. dem Zweck in Form eines Ziels, das nach erfolgreicher Ausführung als erreicht gilt.

Auf unterster Ebene enthält ein Plan primitive Aktionen, wobei jede Gruppe über eine bestimmte Menge von primitiven Aktionen oder auch Fähigkeiten verfügt (*skills*). Dabei kann ein individueller Agent über unterschiedliche Fähigkeiten verfügen und eine Gruppe kann Fähigkeiten besitzen, die keines der Mitglieder besitzt. Diese Darstellung erlaubt die Betrachtung auf unterschiedlichen Abstraktionsebenen.

Die Rolle eines individuellen Agenten in der Gruppe ist in diesem Modell der Teil eines Plans, der von dem Mitglied ausgeführt wird. Nur das Mitglied, welches eine Rolle übernimmt, muss dabei den genauen Inhalt des Teilplans kennen. Die rekursive Gruppenstruktur muss zunächst aufgefaltet werden, um alle Individuen zu finden. Diese Agenten werden mit Bezug auf ihre Rolle auch als *Role-Player* bezeichnet und der von ihnen ausgeführte Teilplan als *Role-Plan*. Führen alle *Role-Player* ihre *Role-Plans* gleichzeitig aus, werden insgesamt wieder die Handlungen des Plans vollzogen.

Definition: Eine Gruppe von Agenten besitzt bei gegebener Rollenverteilung eine *Joint Intention* bezüglich eines Plans, wenn

1. jeder *Role-Player* eine *Joint Intention* bezüglich des Plans besitzt,
2. jeder *Role-Player* glaubt, dass die *Joint Intention* von allen Mitgliedern gehalten wird, und
3. jeder *Role-Player* glaubt, dass der gemeinsame Plan ausgeführt wird, wenn alle *Role-Player* ihre *Role-Plans* ausführen.

Das vorgestellte Verfahren deckt alle in (Bratman 1992) genannten Eigenschaften für die Modellierung von Joint Intentions ab (Kinny et al. 1994, S. 253f.):

1. *Commitment to joint activity*: Dies kann direkt aus der Definition von *Joint Intentions* abgeleitet werden.
2. *Mutual Responsivness*: Da der Zweck eines gemeinsamen Plans die Erfüllung eines gemeinschaftlichen Ziels ist, besitzt ein Agent, welcher eine *Joint Intention* übernommen hat, auch ein gemeinschaftliches Ziel.
3. *Commitment to mutual support*: Die Transformation eines Plans in die einzelnen *Role-Plans* enthält explizit die Verpflichtung zur Kommunikation.

Allerdings kritisiert Haddadi (1995), dass das Modell verschiedene Voraussetzungen enthält, welche die Bildung von *Joint Intentions* übermäßig vereinfachen. Er kommt zu dem Ergebnis, dass das Modell aufgrund seiner Ausrichtung auf Architektur- und Designfragen zu eingeschränkt für eine allgemeine Theorie über gemeinsames Handeln ist.

3.1.4 Jennings und Wooldridge

Das von Jennings und Wooldridge (1994) vorgestellte Modell geht auf die Arbeiten von (Levesque et al. 1990), (Rao et al. 1992) und (Kinny et al. 1994) zurück. Zunächst werden *Commitments* und *Conventions* als zentrale Elemente eingeführt (Jennings 1993a). *Commitments* sind Verpflichtungen eines Agenten zu einer bestimmten Handlungsfolge und entsprechen den aus anderen Modellen bekannten Intentionen. *Joint Commitments* besitzen zusätzlich die Eigenschaft sich auf eine Menge von Agenten zu beziehen.

Es wird erwartet, dass eine Verpflichtung stabil ist und sich nicht laufend ohne Grund ändert. Um aber intelligentes Handeln zu ermöglichen, muss der Sinn einer Verpflichtung überprüft werden können. *Conventions* enthalten hierfür geeignete Regeln, welche die Umstände beschreiben, in denen ein Überdenken gerechtfertigt ist. Zudem enthalten sie entsprechende Bedingungen, wann eine Verpflichtung beibehalten, korrigiert oder verworfen werden kann. Das Schwierige ist, eine ausgewogenes Verhältnis zwischen häufigem und seltenem Überdenken zu finden, da sowohl ein entsprechender Prozess, aber auch überflüssige Verpflichtungen, immer Ressourcen belegen. *Joint Commitments* werden wie folgt definiert (Wooldridge u. Jennings 1997, S. 410).

Definition: Eine Gruppe γ ist gemeinsam einem Ziel ∂ mit einer Motivation ω, einer Vorbedingung pre und einer *Convention c* verpflichtet, wenn

1. anfangs die Vorbedingung wahr ist und
2. bis die Abbruchbedingung wahr wird, jeder Agent aus γ entweder
   a) ein Ziel ∂ besitzt oder
   b) glaubt, dass die Bedingung einer Regel aus *c* wahr geworden ist und er das zugehörige Ziel übernommen hat.

Wenn eine Gruppe γ eine *Joint Intention* besitzt, eine kooperative Handlung α auszuführen, bedeutet dies, dass ein *Joint Commitment* vorhanden ist, in dem γ glaubt, α wird als nächstes geschehen und dann geschieht α. Die individuelle Intention eines Agenten wird analog als ein Sonderfall der *Joint Intention* behandelt.

Jennings und Wooldridge (1994) stellen außerdem ein Vier-Stufen-Verfahren vor, der das Modell in den *CDPS-Prozess* einordnet:

1. **Erkennung**: Die Kooperation beginnt damit, dass ein Agent mögliches Potenzial an dieser entdeckt, ein Problem nicht allein, sondern in einer Gruppe zu lösen. Das kann einen Mangel an Ressourcen oder Kompetenz zur Ursache haben, aber auch einfach die Erkenntnis, dass die Lösung schneller oder besser erreicht werden kann.
2. **Gruppenbildung**: In diesem Schritt sucht der Agent nach geeigneter Unterstützung und formt eine Gruppe. Wenn dies erfolgreich ist, steht am Ende dieser Stufe eine Gruppe mit einem entsprechenden *Joint Commitment.*
3. **Planentwicklung**: Die Agenten verhandeln über einen Plan, mit dem das gesteckte Ziel erreicht werden soll.
4. **Ausführung**: Der vereinbarte Plan wird umgesetzt, wobei die Agenten in engem Kontakt bleiben. Diese Beziehungen werden durch die festgelegten *Conventions* bestimmt.

3.1.5 Zusammenfassung

Joint Intentions bilden die Beziehungen zwischen den beteiligten Agenten und enthalten die gemeinsame Verpflichtung, ein bestimmtes Ziel als Einheit zu verfolgen. Während dieses Prozesses befinden sich die Mitglieder der Gruppe in einem gemeinsamen, mentalen Zustand. Wichtig ist, dass *Joint Intentions* mehr als die Summe einzelner Intentionen sind, da sich für jedes Mitglied der Gruppe auch immer eine Verpflichtung gegenüber den anderen ergibt.

Ein zentrales Thema ist daher die Frage, wann, wie oft und in welcher Form gebildete *Joint Intentions* überprüft und verworfen werden sollten. Ziele können nicht wie auf der Agentenebene üblich ohne weiteres aufgegeben werden, wenn der Agent sie als erfüllt oder unerreichbar betrachtet. Er muss vielmehr sein Wissen um den neuen Zustand den anderen Agenten bekannt machen.

Die Gruppe scheint vielfach wie eine eigene Einheit zu wirken, die über Ziele, Intentionen und Wissen verfügt. Dies wird beispielsweise in Modellen deutlich, in denen rekursive Strukturen möglich sind, eine Gruppe also wieder Mitglied einer übergeordneten Gruppe sein kann. Die Theorien um *Joint Intentions* basieren allerdings auf komplexen Logiken, die meist nur schwer oder gar nicht in praktische Systeme überführt werden können, da diese nur mit begrenzter Rechen- und Speicherkapazität ausführbar sind. Trotzdem helfen diese Modelle, wichtige Strukturen und Mechanismen zu erkennen.

3.2 AGR-Modell

Das *AGR*-Modell (Ferber et al. 2004) ist die Weiterentwicklung des *AALAADIN*-Modells (Ferber u. Gutknecht 1998). Es ermöglicht die Strukturierung von Agentengruppen mit Hilfe der Basiskonzepte ***A**gent,* ***G**ruppe* und ***R**olle.* Der *Agent* ist eine aktive, kommunizierende Einheit, welche Rollen in Gruppen übernimmt. Ein wichtiger Aspekt des *AGR*-Modells ist, dass an den Typ des Agenten keine Anforderungen gestellt werden, es sich also um einen einfachen, reaktiven oder komplexen, mentalen Agenten handeln kann.

Die *Gruppe* besteht aus einer Menge von Agenten, die bestimmte gemeinsame Eigenschaften teilen. Sie ist außerdem der Kontext für eine Sammlung von verschiedenen Verhaltensmustern. Ein Agent darf nur innerhalb derselben Gruppe mit anderen Agenten kommunizieren, aber er kann auch Mitglied mehrerer Gruppen sein, was die Definition einer Organisationsstruktur erlaubt. Mithilfe von Gruppen ist es möglich, die Organisation zu partitionieren.

Eine *Rolle* ist eine abstrakte Beschreibung der funktionalen Position, die ein Agent innerhalb der Gruppe ausübt. Jeder Agent muss mindestens eine Rolle spielen, kann aber auch mehrere Rollen einnehmen. Auf Organisationsebene können Zwänge, wie der der Abhängigkeit oder Zugehörigkeit, festgelegt werden.

Ferber et al. (2004) stellen weiterhin mehrere grafische Notationen vor, welche die visuelle Darstellung von Strukturen im erläuterten Modell erlauben. Mit Interaktionsdiagrammen können zusätzlich Kommunikationsprotokolle zwischen den Rollen spezifiziert werden.

3.2.1 Madkit

Madkit (Gutknecht u. Ferber 2001) ist eine modulare multiagentenfähige Plattform, die das vorgestellte AGR-Modell beinhaltet. Das System besteht aus einer Reihe von Java-Packages, die einen Microkernel implementieren. Dieser enthält alle für den Betrieb der Plattform notwendigen Schlüsselfunktionen, wie die Verwaltung der Agenten, Gruppen und Rollen. Auch die Verarbeitung von lokalen Nachrichten ist Aufgabe des Kernels. Zudem bietet er zwei Schnittstellen, die es Agenten erlauben, die Funktionalität des Kernels zu erweitern (z.B. Monitoring). Die meisten Dienste (z.B. nicht-lokaler Nachrichtenversand) sind als eigenständiger Agent implementiert, welche durch Rollen in bestimmten Gruppen dargestellt werden.

3.3 STEAM

STEAM (Shell for TEAMwork, u.a. beschrieben in Tambe 1997b) ist eines der ersten implementierten Gruppenmodelle, welches auf Joint Intentions aufbaut und enthält zusätzlich einige Elemente der *SharedPlans*-Theorie. Das System überwacht den Fortschritt der gemeinsamen Aufgaben durch Beobachtung der expliziten Ziele und Pläne. Außerdem ist es zur dynamischen Reorganisation des Teams im Fehlerfall fähig. Der Planungsprozess innerhalb eines Agenten wird durch *Joint Intentions* unterstützt. Mit Hilfe von *Joint Commitments* können zudem bestehende *Joint Intentions* überwacht und verwaltet werden.

Um die Kooperation der Agenten weiter zu optimieren, sind verschiedene Elemente der *SharedPlans*-Theorie in das Modell eingeflossen (Tambe 1997a, S. 93):

1. Die Teammitglieder sollen einen gemeinsamen Lösungsweg verfolgen und nicht alternative Pläne aktivieren, die zwar das lokale Problem lösen, aber zu Widersprüchen im Gesamtplan führen könnten.
2. Ähnlich wie in der *SharedPlans*-Theorie müssen die Agenten keine detaillierten Kenntnisse über die Planschritte anderer kennen. Es genügt, den Zustand der *Joint Intention* zu überwachen.

3. Falls die *Joint Intention* fehlschlägt, weil dem Team ein Mitglied fehlt, verhält sich STEAM ähnlich wie die *SharedPlans*-Theorie, in dem versucht wird, durch entsprechende Handlungen der Agenten eine Lösung zu finden.
4. STEAM implementiert ein hybrides Kommunikationsverfahren, das sowohl Aspekte der *Joint Intentions* und der *SharedPlans*-Theorie vereint.

3.3.1 Team Operatoren

Für jede gemeinschaftliche Aktivität muss in *STEAM* eine geeignete Gruppe gebildet werden und die Mitglieder fassen dann eine entsprechende *Joint Intention.* Schlüsselelement ist der sogenannte Team Operator, welcher die gemeinsame Intention explizit beschreibt. Ähnlich wie auf der individuellen Ebene können Team Operatoren Vor-, Kontext- und Terminierungsbedingungen enthalten. Falls ein Agent feststellt, dass er lokales Wissen besitzt, welches zu diesen Bedingungen passt, versendet er dieses an alle Teammitglieder, um beispielsweise eine Beendigung des Team Operators zu erreichen.

STEAM legt erst zur Laufzeit fest, ob ein Operator an einen einzelnen Agenten oder an ein Team gebunden wird. Die *Joint Intention* entsteht dabei durch Auswahl des gleichen Operators von allen Agenten des Teams. Dieses System ermöglicht die Bildung von Hierarchien, die aus individuellen und gemeinsamen Intentionen bestehen.

Jeder Team Operator kann aber nur dann ausgeführt werden, wenn alle Agenten des Teams über eine entsprechende *Joint Intention* verfügen. Diese wird durch ein spezielles Protokoll *establish commitments* erreicht (Details in (Tambe 1997a), S. 97). Das Protokoll verhindert, dass Mitglieder bereits vorzeitig mit der Arbeit beginnen. Ferner wird der interne Zustand synchronisiert, d.h. es werden alle relevanten Informationen ausgetauscht. Falls zwischen den Agenten Uneinigkeit herrscht, hat der Teamleiter die Entscheidungsgewalt, da *STEAM* keine erweiterten Verhandlungsprotokolle enthält. Nach Ablauf des Protokolls besteht eine *Joint Intention* bezüglich eines Team Operators, die nur durch Änderung des Teamzustands wieder aufgehoben werden kann. So ist gewährleistet, dass alle Agenten sich auf das Bestehen der Intention verlassen können und kein Agent die Arbeit vorzeitig beendet.

3.3.2 Überwachung und Neuplanung

Durch Überwachung der Team Operatoren kann *STEAM* automatisch feststellen, welchen Fortschritt die Abarbeitung macht. Es können geeignete Bedingungen festgelegt werden, die anzeigen, ob ein Ziel erreicht wurde, unerreichbar oder irrelevant geworden ist. In *STEAM* sind Rollen eine abstrakte Beschreibung der Aufgaben, die ein Agent innerhalb des Teams wahrnimmt. Die Rolle bestimmt die Operatoren, die vom Rollennehmer zur Erfüllung des Team Operators erfolgreich ausgeführt werden müssen. Dabei ist es möglich, verschiedene Beziehungen zwischen den Rollen zu spezifizieren. Weil die Auswertung von der Anwendungsdomäne abhängt, bietet *STEAM* nur einige allgemeine Mechanismen, zum Beispiel wenn alle Mitglieder ihre Rolle nicht ausführen können.

Wenn *STEAM* durch Anwendung dieser Mechanismen feststellt, dass ein Team Operator unerreichbar geworden ist, wird eine spezielle Routine *repair* aufgerufen, um die Neuplanung anzustoßen. Da *repair* auch ein Team Operator ist, müssen allerdings alle Agenten mit seiner Ausführung einverstanden sein. Je nachdem aus welchem Kontext *repair* aufgerufen wurde, werden unterschiedliche Aktionen ausgeführt. Bei einem domänenspezifischen Problem muss eine entsprechende domänenabhängige Fehlerbehebung durchgeführt werden. Falls aber ein Agent seine Rolle nicht ausfüllen kann, versucht das System, einen geeigneten Ersatz zu finden. Weil nicht immer eine Behebung der Probleme möglich ist, kann *repair* auch selbst fehlschlagen.

3.4 SimpleTeam in JACK

SimpleTeams ist eine Erweiterung der *JACK Agent System und* wird u. a. in (Hodgson et al. 1999) beschrieben. Das Modell basiert auf der Idee, dass jedes Team als autonome Einheit existiert und das Verhalten der Mitglieder steuert. Jedes Team verfügt über den BDI-Agenten entsprechende Konzepte wie *Teamplans* oder *Teambeliefs* und eigene Möglichkeiten der Schlussfolgerung. Das System unterstützt bei der Koordination gemeinsamer Aktivitäten ohne die Autonomie des einzelnen Agenten einzugrenzen.

3.4.1 Rollen

SimpleTeams erlaubt beliebige Gruppenstrukturen wie Hierarchien oder flache Gruppen. Bei der Definition eines Teams sind Rollen ein zentrales Mittel, da hierdurch die Aufgaben der einzelnen Mitglieder klassifiziert werden. Jede Rolle beschreibt auf abstraktem Weg, welche Fähigkeiten ein Agent bzw. eine Gruppe erfüllt oder besitzen muss.

```
1  role GroundSurveillance extends Role {
2        #handles event Observe obs;
3        #posts event Evacuate;
4  }
```

Beispiel 1: Rollendefinition in JACK-Simpleteams

Wie Beispiel 1 zeigt wird eine Rolle wird mit Hilfe von Ereignissen (*events*) beschrieben. Sie werden von Agenten (oder Teams) gespielt und in einem Team eingenommen. Es kann festgelegt werden, welche Ereignisse vom Rollennehmer behandelt (*#handles*) werden müssen und welche dieser auslösen (*#posts*) kann.

3.4.2 Das Team-Konzept

Teams werden als abgeschlossene Einheiten betrachtet, welche bestimmte Aufgaben erfüllen und Rollen in anderen Teams einnehmen können. Jedes Team setzt sich aus Agenten und/oder Subteams zusammen, welche die vom Team vorgeschriebenen Rollen erfüllen können.

```
1  team Company extends Team {
2      #performs role CompanyRole;
3
4      // min of 3 PlatoonRole fillers required.
5      // No upper limit.
6      #requires role PlatoonRole platoons(3,0);
7
8      // exactly 1 Commander filler required
9      #requires role CommandRole command(1,1);
10
11      // 0 or more ScoutRole fillers required
12      #requires role ScoutRole scout(0,0);
13         :
14         :
15  }
```

Beispiel 2: Teamdefinition in JACK-Simpleteams

Zentrales Element einer Teamdefinition sind Rollen. Mit ihnen wird beschrieben, welche Aufgaben vom Team durchgeführt werden können (*#perform role*) und welche Rollen hierzu benötigt werden (*#requires role*). Durch Attribute können Anforderungen (z.B. die Anzahl) an den zukünftigen Rollennehmer festgelegt werden. Während der Bildung des Teams werden die Rollen von geeigneten Agenten aufgefüllt.

3.4.3 Team-Pläne

Ein Team-Plan ist ähnlich wie der aus dem BDI-Modell bekannte Plan eine festgelegte Abfolge von Aktionen und beschreibt das koordinierte Verhalten des Teams. Im Wesentlichen bestehen Team-Pläne aus der Delegation von Teilzielen an die Mitglieder. Jeder Agent erhält so ein Subziel, das er autonom bearbeiten kann. Das *JACK*-Plan Konzept wurde zu diesem Zweck um mehrere Konstrukte erweitert:

- Die Bedingung, welche anzeigt, ob der Plan anwendbar ist, wurde in Hinsicht auf die Rollen erweitert. Der Team-Plan soll nur dann ausgeführt werden, wenn eine geeignete Rolle aktiv ist.
- Im Plankörper können Ziele mit Hilfe der Funktion *@team_achieve(goal)* erzeugt und an das entsprechende Mitglied weitergegeben werden.
- Die Spezifikation von parallelen Aktivitäten ist möglich (s. u.).

3.4.4 Parallelität in Plänen

Mit dem *@parallel* Konstrukt (s. Beispiel 3) kann die Ausführung paralleler Aktivitäten spezifiziert werden.

```
1  @parallel (...) {
2     @team_achieve(A.move("left"));
3     @team_achieve(B.move("right"));
4  }
5
6  @parallel (...) {
7     @team_achieve(A.attack("center"));
8     @team_achieve(B.attack("center"));
9  }
```

Beispiel 3: Parallelität in JACK-Simpleteams

SimpleTeams unterstützt mehrere Varianten, wie parallele Aktionen ausgeführt und ausgewertet werden. Dies ist notwendig, um festzustellen, ob ein entsprechender Block erfolgreich ausgeführt wurde.

Eine Erfolgsbedingung zeigt an, ob der erste, der letzte, einer oder alle Schritte erfolgreich sein müssen. Durch eine Terminierungsbedingung kann spezifiziert werden, wann die Abarbeitung eines Blocks beendet werden soll. Dabei werden unbearbeitete oder laufende Aktivitäten abgebrochen und gelten als fehlgeschlagen. Bei einer *Notification Exception* werden alle Aktivitäten vom Block entkoppelt und mit der gegebenen *Exception* informiert, so dass innerhalb jeder Aktivität individuell entschieden werden kann, wie mit der *Exception* umgegangen werden soll. Der Block selbst gilt als sofort beendet.

Das Ergebnis eines *@parallel*-Block wird berechnet, in dem zu Beginn jede Aktivität einen Satz mit lokalen Kopien der Variablen erhält. Wenn eine Aktivität fertig geworden ist, wird versucht, diese Variablen des Blocks mit denen der Aktivität zu unifizieren. Falls dies fehlschlägt, weil eine frühere Aktivität eine Variable bereits an einen anderen Wert gebunden hat, gilt die aktuelle Aktivität als fehlgeschlagen. Das Gesamtergebnis besteht aus allen erfolgreichen Unifikationen.

3.4.5 Teambeliefs

Gruppenwissen wird im *SimpleTeam*-Modell nur am Rande behandelt. Die Agenten verfügen über kein gemeinsames Wissen (*mutual beliefs*), sondern das System stellt einen Ort zum Austausch von Informationen bereit. Damit steht ein Mechanismus zur Verfügung, der bei der Verbreitung von Informationen unterstützt. Man kann spezifizieren, unter welchen Umständen welches Wissen verbreitet (*Source Data Definition*) und wie die eingehenden Informationen verarbeitet werden sollen (*Target Data Definition*).

Die Zusammenführung der *JACK-Beliefsets* kann entweder mit einer *Synthesizing Belief Connection Definition* oder einer *Inheriting Belief Connection Definition* beschrieben werden. Ersteres führt alle Beliefsets der Agenten bzw. Subteams zu Teamwissen zusammen (von unten nach oben), während letzteres bestimmtes Teamwissen in die Agenten und Subteams abbildet (von oben nach unten). *SimpleTeams* erlaubt so den Gebrauch von gemeinsamem Wissen, wobei das Verfahren in gewissen Zügen einem Blackboard ähnelt.

3.4.6 Automatisierte Teambildung

Der *SimpleTeams*-Ansatz bietet zwar eine gewisse Unterstützung bei der automatischen Zusammensetzung eines Teams, allerdings wird dieses Thema in der Modellbeschreibung nur kurz angeschnitten. Es wird lediglich beschrieben, dass jedes Team während der Abarbeitung zwei Phasen durchläuft:

1. In der Initialphase wird das Team gebildet, also die durch Rollen beschriebenen Platzhalter aufgefüllt.
2. Danach wechselt das Team in die Operationsphase und erfüllt seine eigentliche Aufgabe.

Die Implementation enthält zumindest einen JACK-Agenten, der wie ein Brooker oder Matchmaker die Übersicht über gebildete Teams besitzt und entsprechend bei der Vermittlung hilft. Der Entwickler kann zudem für die Teambildung geeignete Pläne schreiben und dadurch den Vorgang individuell anpassen.

3.5 Weitere Techniken

An dieser Stelle werden weitere Verfahren aufgezeigt, die im Bereich der Koordination entwickelt wurden. Aufgrund der Vielfalt wird diese Vorstellung allerdings keine erschöpfende Liste aller denkbaren Techniken enthalten.

3.5.1 Trader und Broker

Trader und *Broker* ermöglichen die Verteilung von bereits vorhandenen Teilproblemen an geeignete Agenten. Dies ist normalerweise nach der Zerlegung der Fall, kann aber auch während der Bearbeitung eines Teilproblems notwendig werden. *Trader* und *Broker* unterstützen bei dieser Aufgabe und stellen die notwendigen Kontakte zwischen den Agenten her. Allerdings muss der *Trader* bzw. *Broker* allen anderen Agenten bekannt gemacht werden (z.B. durch eine standardisierte Adresse).

Der auch als *Matchmaker* bezeichnete *Trader* verwaltet eine Datenbank, in der sich Agenten als Dienstanbieter registrieren können. Der Anbieter übermittelt zusätzliche, beschreibende Informationen,

damit der *Trader* Aufgaben möglichst passend zuordnen kann. Bei der Aufnahme in die Datenbank verpflichtet sich der Agent gleichzeitig, gestellte Aufgaben auch zu erfüllen.

Agenten, die eine Aufgabe zu vergeben haben, können sich an den *Trader* wenden, um eine Liste mit geeigneten Agenten zu erhalten. Als *Requester* wird der Agent bezeichnet, der eine Aufgabe oder Anfrage an einen anderen Agenten richten möchte. Der Server ist der Agent, welcher die Aufgabe übernehmen will. In den tatsächlichen Kontakt zwischen Diensterbringer und -nehmer ist der *Trader* aber nicht verwickelt. Der *Broker* hat eine ähnliche Funktionsweise. Er unterscheidet sich vom *Trader* aber darin, dass er das gestellte Problem selbst weiterleitet. Der anfragende Agenten muss sich so nicht selbst um einen Kontakt bemühen, sondern übergibt das Problem einfach dem *Broker*.

3.5.2 Bekanntschaftsnetzwerke

In Bekanntschaftsnetzwerken ist jeder Agent sein eigener Trader und führt somit Informationen über die Fähigkeiten anderer Agenten. Dabei ist es nicht erforderlich, dass jeder Agent alles über alle anderen Agenten weiß, da Anfragen ggf. über mehrere Knoten laufen können. Trotzdem ist die Verwaltung entsprechender Daten nicht ganz einfach, da Agenten auch nachträglich in die Gesellschaft eintreten oder diese verlassen können. Eine ausführliche Beschreibung dieses Verfahrens findet sich in (Ferber 2001).

3.5.3 Das Kontraktnetz

Ein *Kontraktnetz* (Smith u. Davis 1988) besteht aus verschiedenen Agenten, die sich in einem marktähnlichen Verfahren für die Lösung von ausgeschriebenen Teilproblemen bewerben können. Ziel ist es die Teilprobleme möglichst effizient an die Agenten zu verteilen, die im Moment am besten zur Lösung in der Lage sind.

Nachdem ein Problem in seine Teilaufgaben zerlegt wurde, beginnt die Arbeit des *Kontraktnetzes*. Der Manager, ein spezieller Knoten im *Kontraktnetz*, versucht, die Teilaufgaben an Agenten zu vergeben. Hierzu schreibt er diese öffentlich aus. Jeder Agent kann die laufenden Ausschreibungen einsehen und anhand des vorhandenen Wissens und verfügbarer Ressourcen bewerten. Falls der Agent die Aufgabe übernehmen möchte, übermittelt er dem Manager eine entsprechende Nachricht. Aus der Menge der eingegangenen Gebote wählt der Manager

nach Ablauf der Bewerbungsfrist einen geeigneten Knoten aus und überträgt ihm die Aufgabe. Sobald der Agent die Lösung erarbeitet hat, meldet er diese an den Manager zurück.

3.5.4 Gegenseitige Modellierung

In (Genesereth et al. 1988) wird die Idee aufgeworfen, die eigenen Handlungen mit Hilfe von Modellen anderer Agenten zu koordinieren. Jeder Agent verwaltet ein Modell über das Wissen und die Ziele der anderen, was es erlaubt, die Aktionen anderer vorauszuahnen und Konflikte auf diese Weise zu vermeiden. Häufig werden spieltheoretische Ansätzen bei der Suche nach einer möglichst rationale Handlungen benutzt. Dieses Verfahren kann die Anzahl der Kommunikationsvorgänge reduzieren, erfordert aber zusätzlichen Aufwand bei der Modellierung anderer Agenten.

3.5.5 Normen und soziale Gesetze

Normen sind erlernte Muster, die das erwartete Verhalten anderer Agenten widerspiegeln. Die Individuen befolgen von der Gesellschaft vorgelebte Leitlinien. Gesetze sind sehr ähnlich, werden aber regelmäßig im Zusammenhang mit einer Autorität verwendet. Dies bedeutet, dass die Leitlinien sich nicht aus der Gesellschaft ergeben, sondern vorgeschrieben werden. Gesetze werden in der Regel überwacht und führen bei Übertretung zur Bestrafung.

Die Leitlinien spielen eine entscheidende Rolle, da sie dem Agenten ein Muster vorgeben, anhand dessen er seine Aktionen planen kann. In vielen Fällen erleichtern sie auch den Entscheidungsprozess, da sie häufig bestimmte Vorgaben machen (z.B. Bestimmung einer Reihenfolge durch vordefinierte Prioritäten). Es gibt zwei Möglichkeiten, wie Normen oder Gesetze in ein System gelangen können (Wooldridge 2002, S. 213):

1. **Offline Design**: Die Regeln werden im Voraus fest in die Agenten einprogrammiert.
2. **Emergenz**: Die Regeln entwickeln sich langsam aus häufigem Gruppenverhalten.

Während das Erste regelmäßig leichter zu implementieren ist, bietet die zweite Variante mehr Flexibilität, wie beispielsweise Änderungen im laufenden Betrieb.

3.5.6 Emergente Aufgabenzuweisung

In (Ferber 2001) wird Emergenz als ein Mechanismus zur Aufgabenverteilung vorgestellt. Durch in die Umgebung eingebracht Signale (z.B. Laute oder Pheromone) werden je nach den Fähigkeiten eines Agenten bestimmte Verhaltensmuster ausgelöst. Diese Technik eignet sich vor allem für reaktive Systeme, da weder direkte Kommunikation noch besondere Entscheidungs- bzw. Planungsprozesse erforderlich sind.

3.6 Zusammenfassung

In diesem Kapitel wurden auf theoretischer Ebene einige Modelle und Verfahren beschrieben, die den Systementwickler bei der Verwendung von Koordinationstechniken unterstützen sollen. Sowohl die *Joint Intentions*-Theorie als auch das *AGR*-Modell bieten für diese Arbeit interessante Konzepte. Neben den hier vorgestellten Modellen gibt es aber noch eine Vielzahl weiterer Ansätze, wie etwa *TOP* in (Pynadath et al. 1999), *CAST* in (Yen et al. 2001) oder *Machinetta* in (Scerri et al. 2004). Oft ist zu bemerken, dass die Modelle mit bestimmten Zielsetzungen und für konkrete Anwendungen entwickelt wurden.

Die Theorien lassen sich auch in implementierten Modellen wiederfinden, so dass in diesem Kapitel auch die Ansätze *STEAM*, *SimpleTeams* und *Madkit* untersucht werden. Die durch die Betrachtung der theoretischen und praktischen Modelle gewonnenen Erkenntnisse sollen in den folgenden Kapiteln bei der Entwicklung des BDI-Gruppenmodells helfen. Vorhandene Konzepte, wie das der Rollen, sollen aufgegriffen und umgesetzt werden.

In der nachstehenden Tabelle werden die in diesem Kapitel besprochenen Verfahren (außer den weiteren Ansätzen) unter den für diese Arbeit wichtigen Gesichtspunkten gegenübergestellt. Der SimpleTeams-Ansatz enthält durch seine starke Anlehnung an das BDI-Konzept viele interessante Ideen und wird daher für den weiteren Verlauf als Vorbild dienen. Aber auch das AGR-Modell bietet mit dem Gruppen-Rollen-Modell ein Konzept, welches später wieder aufgegriffen werden soll.

	Joint Intentions	AGR	STEAM	SimpleTeams
Kooperations-verfahren	gemeinsame Ziele	Rollen	gemeinsame Ziele	Ziele & Pläne
Gruppenbildung	durch Errichtung einer Joint Intention	über Ereignisse, z.B. in Squenz-diagrammen	vordefiniert	semi-automatische Bildung auf Basis der Rollen
Gruppenwissen	Mutual Beliefs	-	separater Teamstate	Teambeliefs, ähnlich Blackboard
Verwaltung der Gruppe	verteilt über alle Agenten	durch Plattform, Agent kann aber Repräsentant sein	Teamleiter	mit einer BDI-Komponente
Sicherheit	-	Kommunikation nur in Gruppe, Betritt erfordert Berechtigung	Reduzierung der Kommunikation	-
Fehlerfall	Rückmeldung aufgrund der gemeinsamen Verpflichtung	-	dynamische Reorganisation des Teams	Verhalten kann in Teampläne festgelegt werden

4 Das BDI-Gruppenmodell

Während die bisherigen Kapitel sowohl in die Grundlagen als auch in bestehende Theorien und praktische Umsetzungen eingeführt haben, soll nun das BDI-Gruppenmodell vorgestellt werden. Im ersten Schritt werden einige wünschenswerte Anforderungen formuliert, die das Modell erfüllen soll, um dann die Details des Modells zu erläutern. Im Anschluss wird auf die Frage nach Teilgruppen (d.h. einer Gruppe innerhalb einer anderen Gruppe) eingegangen. Zum Ende soll das entworfene BDI-Gruppenmodell in Bezug zu den oben vorgestellten Modellen gesetzt werden.

4.1 Funktionale Anforderungen

Die in den zwei vorangegangenen Kapitel vorgestellten Modelle geben bereits eine Fülle von Anregungen, die Einfluss auf das BDI-Gruppenmodell genommen haben. Daher ist es sinnvoll, zunächst wesentliche Aspekt herauszugreifen und diese als Anforderungen an das neue Modell zu betrachten.

1. Durch ein Rollenmodell soll die Struktur einer Gruppe beschrieben werden können.
2. Agenten können nach offenen Rollen suchen oder selbst anbieten.
3. Agenten können eigene Gruppen gründen.
4. Gruppen sollen offene Rollen ausschreiben.
5. Gruppen sollen in der Lage sein, wiederum Rollen in anderen Gruppen zu erfüllen.
6. Jede Gruppe soll eine verwaltende Einheit besitzen, die alle Belange einer Gruppe koordiniert und als Ansprechpartner für externe Agenten bzw. Gruppen dient.
7. Die Gruppe muss in die Lage versetzt werden, intern Aufgaben zu verteilen, Teillösungen und Wissen auszutauschen sowie eine Endlösung zu erstellen.

Mit Bezug zu (Kinny et al. 1994) soll auch das BDI-Gruppenmodell über gemeinsames Wissen, gemeinsame Ziele und Gruppenpläne verfügen, mit deren Hilfe die gemeinsamen Ziele erfüllt werden können.

4.2 Vorstellung des BDI-Gruppenmodells

In Anlehnung an die Struktur eines BDI-Agenten besitzt auch das Gruppenmodell entsprechende mentale Komponenten. Die Abbildung 5 zeigt die Kernelemente und deren Beziehungen im BDI-Gruppenmodell.

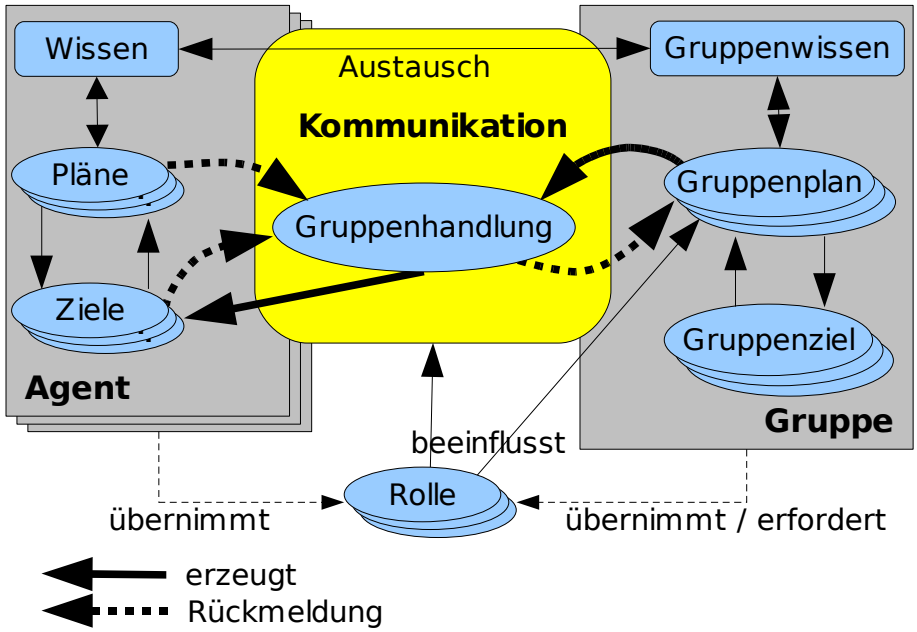

Abbildung 5: Das BDI-Gruppenmodell

Die Darstellung ist in vier Bereiche unterteilt, die die wesentlichen Merkmale enthalten. Auf der linken Seite ist der Agent mit seinen mentalen Eigenschaften *Wissen*, *Ziele* und *Pläne* zu finden. Die rechte Seite deutet die Gruppe an, die über dem BDI-Agenten ähnliche Konstrukte *Gruppenwissen*, *Gruppenziel* und *Gruppenplan* verfügt. Mit Hilfe eines *Kommunikationssystems* (in der Mitte) werden die Elemente der Gruppe mit denen des Agenten verbunden. Eine *Gruppenhandlung* beschreibt hier die Teilaufgabe, welche ein Agent zu bearbeiten hat. Das oben bereits eingeführte Konzept der *Rolle* wird auch im BDI-

Gruppenmodell verwendet und ist im unteren Bereich dargestellt. In den nun folgenden Abschnitten werden die einzelnen Komponenten im Detail und deren Beziehungen untereinander vorgestellt.

4.2.1 Gruppenziele

Gruppenziele sind die treibende Kraft einer Gruppe. Ähnlich wie auf der Ebene der Agenten sind sie Auslöser für die Handlungen der Gruppe. Im BDI-Gruppenmodell kann jede Gruppe ein oder mehrere Ziele besitzen, die dann zur Ausführung vordefinierter Gruppenpläne führen. Um weitere Abstraktionsebenen zu ermöglichen, kann ein Gruppenplan auch neue Gruppenziele erzeugen.

Dieser Aufbau erfordert einen geeigneten Mechanismus zur Auswahl von aktiven Zielen und auszuführenden Plänen, die sowohl das Means-End Reasoning als auch die Deliberation auf Gruppenebene ausführen. Um das Modell nicht einzuschränken, wird an dieser Stelle kein Algorithmus vorgestellt, denkbar wäre jedoch die Verwendung der oben eingeführten *Easy Deliberation Strategy*. Aber auch die Implementierung eines erweiterten Verfahrens, das nicht nur auf Gruppenebene, sondern auch die Intentionen der Gruppenmitglieder berücksichtigt, wäre interessant. Im Folgenden wird davon ausgegangen, dass jede Implementation ein geeignetes Verfahren zur Auswahl von Zielen und Plänen bereitstellt. Das nächste Kapitel zeigt außerdem, wie die *Easy Deliberation Strategy* verwendet werden kann.

4.2.2 Gruppenpläne

Gruppenpläne sind die Mittel, mit denen eine Gruppe die gesteckten Ziele erreichen kann. Wie im vorausgehenden Absatz angedeutet, muss ein Verfahren zur Auswahl existieren, da der Agent nicht wahllos sondern zielgerichtet passende Pläne zur Ausführung bringen soll. Wesentlicher Teil eines Plans ist der Plankörper. Dieser enthält die einzelnen Aktionen, welche in einer vorgeschriebenen Reihenfolge bei Ausführung des Plans ausgelöst werden. Da sich Gruppenpläne vor allem mit der Zuweisung von Aufgaben an die Gruppenmitglieder und die Verarbeitung der Ergebnisse beschäftigen, bieten sie entsprechende Erweiterungen.

Gruppenbildungsprozess

Mit Hilfe dieser Funktionen kann der Prozess zur Bildung der Gruppe überwacht und gesteuert werden. Hierzu zählt die Möglichkeit zu prüfen, ob die Gruppe vollständig ist, das Aktivieren und Deaktivieren von Rollen oder das Entfernen von Gruppenmitgliedern.

Zuteilung von Aufgaben

Jeder Gruppenplan beschreibt, wie das Gruppenziel zerlegt werden soll. Daher ist es notwendig, die entsprechenden Teile an geeignete Agenten weiterzugeben. Rollen erlauben es dabei, Agenten zu finden, die die gewünschten Anforderungen erfüllen. Unter Gruppenhandlung wird hier der Teil einer Gesamtaufgabe verstanden, der einem Agenten zugewiesen werden soll. Dieser wird dann geeignete lokale Ziele erzeugen, um die gewünschten Handlungen zu vollziehen. Mit Blick auf die *Joint Intention Theorie* wird vom Agenten aber auch die Überwachung und Rückmeldung bezüglich der Gruppenhandlung gefordert.

Überwachung und Verarbeitung der Ergebnisse

Jeder Agent überwacht den Fortschritt seiner Problemlösung und meldet die Ergebnisse der Gruppe. Im Wesentlichen bedeutet dies, dass der Agent den Zustand der Gruppenhandlung in geeigneter Form aktualisiert. Für einen BDI-Agenten lässt sich dies mit einfachen Mitteln bewerkstelligen, da im Grunde nur der Zustand des zugehörigen Ziels im Agenten beobachtet werden muss. Die Gruppe ist dann durch Überwachung dieses Zustands in der Lage, Informationen über den Lösungsfortschritt aber auch über Fehler zu erhalten. Auch das Ergebnis einer Aufgabe kann auf diesem Weg übertragen werden, was den Zugriff der Gruppe auf diese Daten ermöglicht. Sie kann dann aus den einzelnen Rückmeldungen eine Gesamtlösung konstruieren.

Fehlerfall

Die Gruppe kann auf unterschiedliche Weise reagieren, sobald das Fehlschlagen einer Gruppenhandlung erkannt wird (da beispielsweise der Agent ein zugehöriges, lokales Ziel nicht erreichen konnte). Im einfachsten Fall gilt auch der gesamte Gruppenplan als fehlgeschlagen, was dazu führt, dass die übrigen Agenten ihre bisherige Arbeit ebenfalls einstellen. Eine weitere Möglichkeit besteht darin, dass der Gruppenplan nicht sofort abgebrochen wird, sondern zunächst versucht wird, einen

anderen Agenten zu finden, der die bisher fehlgeschlagene Gruppenhandlung doch noch erledigen kann. Gelingt dies aber nicht, schlägt auch in dieser Variante der Gruppenplan fehl.

Falls weitere Gruppenpläne für ein Gruppenziel vorhanden sind, könnte der Deliberationsalgorithmus gegebenenfalls noch weitere Pläne ausprobieren. Dies bedeutet dann aber auch, immer wieder von neuem mit der Arbeit zu beginnen. Sollten keine weiteren Gruppenpläne zur Verfügung stehen, ist ähnlich wie im BDI-Modell auch das Gruppenziel nicht mehr erreichbar und wird verworfen.

Parallelität

Die Gruppenpläne des BDI-Gruppenmodells unterstützen die parallele Ausführung von Gruppenhandlungen und damit die gleichzeitige Bearbeitung von Teilproblemen durch verschiedene oder sogar den gleichen Agenten. Durch beliebige Anordnung von Zuweisungs- und Warte-Funktionen im Plankörper kann der Systementwickler den gewünschten Ablauf flexibel festlegen. Dabei wird nicht in die Autonomie des jeweiligen Agenten eingegriffen, d.h. der Agent entscheidet weiterhin selbstständig, wann er die zugewiesenen Aufgaben erledigt. Dies wird dadurch erreicht, dass die lokal erzeugten Ziele grundsätzlich keinen Vorrang besitzen und somit vom Agenten nicht zwangsweise als Intention übernommen werden.

Im Gegensatz zum SimpleTeams-Ansatz wird die Gesamtlösung aber nicht automatisch berechnet. Stattdessen stehen alle Teilergebnisse in getrennten Namensräumen, und zwar den Gruppenhandlungen, zur Verfügung. Durch entsprechende Funktionen im Körper des Gruppenplans können die Lösungen dann abgerufen werden und zu dem Gesamtergebnis des Gruppenziels zusammengefügt werden.

4.2.3 Gruppenwissen

Unter gemeinsamem Wissen werden im BDI-Gruppenmodell Informationen verstanden, die von allen Mitgliedern geteilt werden. Es kann im Rahmen der Ziel und Planverarbeitung auf Gruppenebene, aber auch wie gewohnt von den Agenten verwendet und modifiziert werden. Jedes *Belief* wird hierzu in allen Agenten und der Gruppe vorgehalten und muss bei Änderungen in geeigneter Form synchronisiert werden. Das BDI-Gruppenmodell sieht vor, dass der Agent bei einer Änderung das neue Wissen dem Gruppenverwalter übermittelt und dieser die Informationen dann an alle Mitglieder weitergibt.

Wie dieser Wissenstransfer im Detail abläuft, kann auf unterschiedliche Art geschehen. Im einfachsten Fall wird die neue Information von allen Gruppenmitgliedern direkt übernommen. Es wäre aber auch denkbar, erst eine Zustimmung jedes einzelnen zu fordern, bevor der neue Wert aktualisiert wird. Für den Agentenentwickler wäre auch eine Unterstützung der *ACID*-Eigenschaften hilfreich, da so Probleme wie Dirty-Read oder Lost-Update (s. (Ritter 2003), Kapitel 8) vermieden werden.

Da je nach Implementation und Anwendungsfeld unterschiedliche Formen des Wissensaustausches von Interesse sein können, wird an dieser Stelle kein festes Verfahren definiert. Wichtig ist festzuhalten, dass im BDI-Gruppenmodell ein Raum vorhanden ist, in dem die Gruppenmitglieder auf standardisiertem Weg gemeinsames Wissen verwalten können. Mit der Implementation im nächsten Kapitel wird ein einfaches Verfahren vorgestellt, das den Austausch von gemeinsamem Wissen transparent durchführt.

4.2.4 Kommunikationssystem

Das Verbindungsstück zwischen den Agenten und der Gruppe ist das Kommunikationssystem. Da es im BDI-Gruppenmodell für jede Gruppe einen Gruppenverwalter gibt, besitzt das System eine sternförmige Topologie. Dies bedeutet, dass jeder Agent nur mit dem Gruppenverwalter kommuniziert, welcher Informationen gegebenenfalls weiterreicht. Anders als im *AGR*-Modell ist es den Agenten allerdings nicht untersagt, im Rahmen anderer Protokolle und über die Grenzen der Gruppe hinweg auch direkt miteinander zu kommunizieren.

Das Kommunikationssystem stellt den für die Gruppenprozesse notwendigen Austausch von Informationen zwischen den Agenten und der Gruppe sicher. Die entsprechenden Mechanismen werden vor dem Systementwickler verborgen, der sich daher nicht mit den Details der Kommunikation beschäftigen muss. Zur Erfüllung dieser Aufgabe werden von dem System mehrere Aufgaben übernommen:

1. Bereitstellung eines transparenten Mechanismus zum Austausch und zur Nutzung gemeinsamen Wissens.
2. Weiterleitung der in den Gruppenplänen vorgenommenen Zuweisung von Zielen (Gruppenhandlungen) durch Erzeugung entsprechender lokaler Ziele auf Agentenebene.
3. Überwachung der lokalen Zielausführung und Rückmeldung der Ergebnisse bzw. von Zustandsinformationen.

Durch seinen Aufbau muss das Kommunikationssystem also sowohl auf der Gruppen als auch auf Agentenseite eingefügt werden. Es bildet damit die Schnittstelle zwischen den Agenten und der Gruppe.

4.2.5 Rollen

Wie in den vorherigen Kapiteln mehrfach festgestellt, kann die Struktur einer Gruppe mit Hilfe von Rollen beschrieben werden. Auch im BDI-Gruppenmodell soll hiervon Gebrauch gemacht werden und der allgemeine Aufbau einer Gruppe sowie die Aufgabenverteilung in der Gruppe mit Rollen festgelegt werden. Zunächst ist festzustellen, dass Rollen sowohl im Agenten als auch in der Gruppe benötigt werden und somit nicht der einen oder anderen Seite zugeordnet werden können. Wichtig ist auch, dass alle Agenten und Gruppen unter einer Rolle das gleiche verstehen sollten und daher Rollenbezeichner nicht für unterschiedlichen Formen verwenden dürfen. Das BDI-Gruppenmodell sieht daher einen Container vor, in dem alle dem System bekannten Rollen enthalten sind.

Jeder Agent verfügt über eine Liste mit Bezeichnern für Rollen, die er prinzipiell einnehmen könnten. Selbstverständlich sollte der Agent über geeignete Fertigkeiten verfügen, da es wenig sinnvoll ist, Fähigkeiten zu propagieren, die im Bedarfsfall nicht vorhanden sind. Zur Laufzeit kann der Agent dynamisch entscheiden, ob er im Moment tatsächlich bereit ist, eine Rolle zu übernehmen. Zum Erreichen des Gruppenziels ist eine Menge von Teilnehmern erforderlich, die bestimmte Rollen übernehmen. Eine Gruppe benötigt daher zunächst Akteure, die die jeweiligen Rollen erfüllen. Da aber die Gruppe als Ganzes auch Handlungen ausführt, ist sie ebenfalls imstande, die Anforderungen an eine Rolle zu erfüllen. Sie kann also sowohl Akteure für bestimmte Rollen brauchen als auch selbst Akteur einer anderen Gruppe sein, wo sie eine Rolle inne hat.

Die Elemente Agent, Rolle und Gruppe geben eine strukturelle Beschreibung der zwischen ihnen bestehenden Zusammenhänge. Diese Struktur ist bereits im Voraus bekannt und erfordert nicht die Teilnahme konkreter Agenten an der Gruppe. So wird es möglich, auf abstraktem Weg die Problemlösung innerhalb der Gruppe festzulegen. Im BDI-Gruppenmodell werden die Rollen sowohl bei der Gruppenbildung als auch bei der Zuweisung von Aufgaben verwendet.

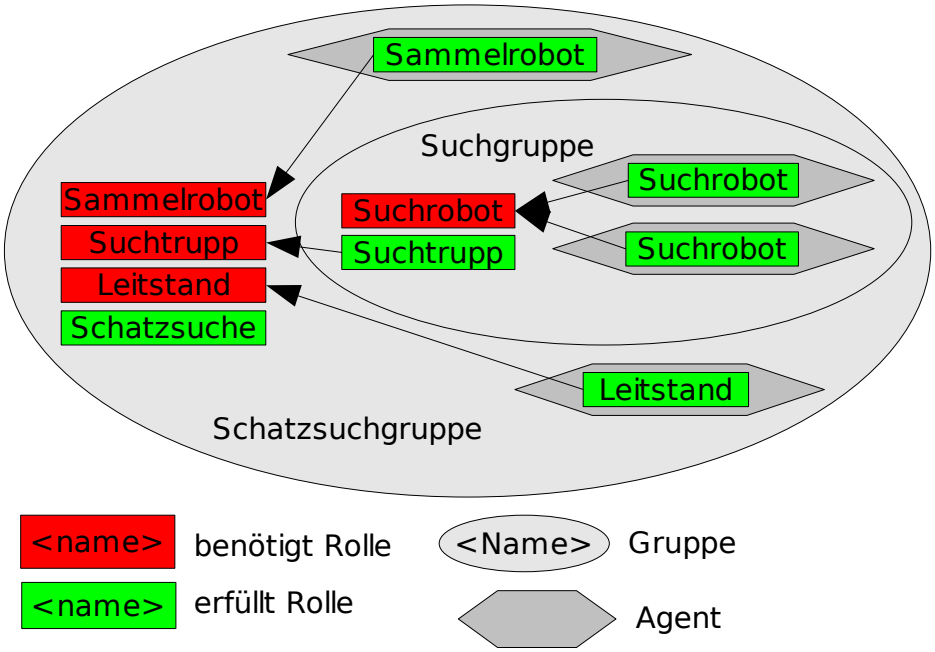

Abbildung 6: Beispiel für ein Agent-Rolle-Gruppe Modell

Die Abbildung 6 veranschaulicht ein derartiges Agent-Rolle-Gruppe Modell. Die Gruppe *Schatzsuchgruppe* erfüllt die Rolle *Schatzsuche*, in dem sie auf Entitäten zurückgreift, die die Rollen *Sammelrobot*, *Suchtrupp* oder *Leitstand* erfüllen können. Ähnlich bilden mehrere Agenten, welche die Rolle *Suchroboter* erfüllen eine *Suchgruppe*, die in der Lage ist, die Rolle *Suchtrupp* zu erfüllen.

4.2.6 Der Agent im BDI-Gruppenmodell

Da das BDI-Gruppenmodell mit mentalen Konzepten arbeitet, sind diese eine Mindestanforderung an die beteiligten Agenten. Ein Agent kann nur dann an der Gruppe nur teilnehmen, wenn er in der Lage ist, mit Zielen, Plänen und Wissen umzugehen. Dieser Umstand schränkt das Modell in seiner Allgemeingültigkeit ein, da es einige Agentenklassen ausschließt. Andererseits lassen sich auf diesem Wege die BDI-Konzepte auch auf Gruppen anwenden.

Im BDI-Gruppenmodell werden keine Änderungen des zugrunde liegenden BDI-Modells erforderlich. Jeder Agent muss lediglich um das Kommunikationssystem erweitert werden, da dieses die Schnittstelle zu den Mechanismen der Gruppe bietet. Das System initiiert und überwacht die Ausführung der zugeteilten Aufgaben.

4.3 Die Gruppe als Teil einer anderen Gruppe

Durch den Aufbau des BDI-Gruppenmodells kann eine Gruppe als Akteur in einer anderen Gruppe auftreten. In Abbildung 6 wird dies bereits schematisch dargestellt, indem die *Suchgruppe* Teil der *Schatzsuchgruppe* ist. So wird eine komplexe, hierarchische Struktur von Gruppen und Agenten ermöglicht, die ebenfalls durch das oben eingeführte Gruppen-Rollenmodell beschrieben werden kann. Vorteil ist die von unten nach oben gerichtete Abstraktion über mehrere Ebenen. Auch besteht die Möglichkeit, einzelne Gruppen je nach Bedarf in höheren Gruppen zusammenzufassen und diese damit wiederzuverwenden.

Wenn eine Gruppe α Teil einer Gruppe β ist, sieht das BDI-Gruppenmodell vor, dass der Gruppenverwalter der Gruppe α, als Ansprechpartner für die Gruppe β dient. Von der Gruppe β zugewiesene Aufgaben, werden zu Gruppenzielen der Gruppe α. Gruppenwissen wird über die verschiedenen Ebenen der Hierarchie verteilt, als wäre es Bestandteil jedes Agenten, der zu einer der Gruppen der Hierarchie gehört.

4.4 Das BDI-Gruppenmodell im Vergleich

Das BDI-Gruppenmodell kann nicht als konkrete Umsetzung einer der vorgestellten Joint Intentions-Theorien betrachtet werden. Dennoch wurde der Kerngedanke, dass die Teilnehmer einer Gruppe ein gemeinsames Ziel besitzen und sich gegenseitig über den Fortschritt ihrer Arbeit informieren, übernommen. Auch die Idee des *Social Agent* aus den Arbeiten von Rao, Georgeff und Sonnenberg (Rao et al. 1992) hat dazu beigetragen, dass im BDI-Gruppenmodell auch Gruppen Teil einer anderen Gruppe sein können.

Dem *AGR*-Modell wurde vor allem die strukturelle Modellierung mit den Elementen Agent, Rolle und Gruppe entnommen. Die von Ferber eingeführten Diagramme (z.B. das Cheeseboard, s. (Ferber et al. 2004), S. 222) wären auch zur strukturellen Gruppenbeschreibung im BDI-Gruppenmodell geeignet. Es wurde allerdings darauf verzichtet,

Kommunikation nur innerhalb einer Gruppe zuzulassen. Ein Agent im BDI-Gruppenmodell kann somit auch mit Agenten kommunizieren, die sich nicht in der gleichen Gruppe befinden.

Wesentlichen Einfluss hat der *SimpleTeam*-Ansatz genommen. Neben der Beschreibung von Teams, die im BDI-Gruppenmodell als Gruppe auftreten, wurde auch die Idee des Teamplans übernommen. Die Zuweisung von Teilaufgaben im BDI-Gruppenmodell sind vom *@team_achieve*-Konstrukt inspiriert. Das System zum Austausch von Beliefs ist hingegen nicht in der bestehenden Form übernommen worden, sondern strukturell vereinfacht.

Wie in vielen Modellen wird das Problem der Gruppenbildung auch im BDI-Gruppenmodell nicht behandelt. Durch Gruppenbeschreibung und Rollen gibt es aber Ansatzpunkte, die in einer Implementation für eine automatische Gruppenbildung genutzt werden können. Wie dies aussehen könnte wird im nächsten Kapitel gezeigt.

4.5 Zusammenfassung

Das BDI-Gruppenmodell bietet einen generischen Ansatz, um die Handlungen mehrerer Agenten zu koordinieren. Mit Hilfe von Gruppenzielen und -plänen kann der Entwickler die Abläufe ohne Berücksichtigung von Kommunikationsaspekten beschreiben und muss auch nicht die internen Details der jeweiligen Agenten kennen. So wird es möglich, auf einer abstrakten Ebene das Zusammenspiel der Agenten in einer Gruppe zu spezifizieren. Zudem können die bekannten BDI-Konzepte auch auf Ebene der Gruppe angewandt werden.

Im Modell wurden die Form der Kommunikation und die Ausarbeitung eines Deliberationsverfahrens bewusst offen gelassen. Diese müssen damit erst bei der Implementierung des Modells entwickelt werden und schränken das Modell nicht auf eine bestimmte Architektur ein. Im nächsten Kapitel wird auf diese beiden Aspekte näher eingegangen.

Wie aber bereits erwähnt, greift das Modell auf mentale Konzepte zurück, so dass die beteiligten Agenten diese verstehen müssen. Es nützt wenig, einem reaktiven Agenten Zielvorgaben zu machen, da er nicht über eine geeignete Planungskomponente verfügt. Das entwickelte BDI-Gruppenmodell ist daher auf die Welt der BDI-Agenten begrenzt. In dieser kann es aber in beliebige Architekturen eingefügt werden. Wenn

für die Kommunikation ein standardisiertes Protokoll (z.B. FIPA) genutzt wird, wäre sogar die Bildung von Gruppen über unterschiedliche Plattformen denkbar.

5 Ein Prototyp für JADEX

Dieses Kapitel beschreibt die Umsetzung des im vorangegangenen Kapitel vorgestellten BDI-Gruppenmodells im *JADEX*-Agentenframework. Im Allgemeinen kann das Modell aber in allen Frameworks zum Einsatz kommen, die auf mentalen Agenten basieren und die entsprechenden BDI-Konzepte umsetzen. Die folgenden Abschnitte werden jeweils die Implementation der einzelnen Elemente des Modells vorstellen und auf deren Verwendung in Agentensystemen eingehen. In Anlehnung an das *ADF* (*Agent Definition File*) gibt es im BDI-Gruppenmodell auch ein *GDF* (*Group Definition File*), in welchem analog zum Agenten eine Gruppe spezifiziert wird.

5.1 Die Gruppe

Jede Gruppe wird von einem erweiterten Agenten repräsentiert. Dies bedeutet, dass die Gruppe im Jadex-System, wie ein einzelner Agent auftritt und auch über die entsprechenden Eigenschaften verfügt. Dieser spezielle Agent wird im weiteren Verlauf als Gruppenverwalter bezeichnet. Die Ziele des Gruppenverwalters entsprechen den Zielen der Gruppe. So können die existierenden Mechanismen zur Deliberation (in diesem Fall die *Easy Deliberation Strategy*) ohne Änderung übernommen werden.

Die Spezifikation der Gruppe erfolgt in einem XML-Dokument, welches stark an das *ADF* angelehnt ist. Zusätzlich stehen zur Beschreibung der Gruppeneigenschaften die folgenden Elemente bereit:

- *GroupActions* (Gruppenhandlungen)
- *GroupPlans* (Gruppenpläne)
- *GroupBeliefs* (Gruppenwissen)
- *Roles* (Rollen)

Durch Modell- und Paketname erhält die Gruppenspezifikation wie ein Agent einen eindeutigen Namen, auf den zur Laufzeit zurückgegriffen werden kann.

5.2 Rollen

Rollen werden über einen im ganzen System eindeutigen Namen angesprochen. In der *ADF* bzw. *GDF* wird festgelegt, welche Rollen vom Agenten bzw. der Gruppe erfüllt oder benötigt werden. Zusätzlich kann der Entwickler bei benötigten Rollen Kardinalitäten festlegen (z.B. es werden 5 Suchroboter benötigt). Beispiel 4 zeigt die Rollendefinition einer Gruppe, welche ist in der Lage ist, die Rolle *Aufraeumer* zu erfüllen. Sie benötigt hierfür aber einen bis maximal fünf Agenten, die die Rolle *Traeger* in der Gruppe erfüllen.

```
1  <roles>
2
3    <role name="Tr" rolename="Traeger"
4          mode="require" min="2" max="5" />
5
6    <role name="Tf" rolename="Aufraeumer"
7          mode="fulfil"/>
8
9  </roles>
```

Beispiel 4: Rollendefinition

Intern wird für jede Rolle eine oder mehrere Positionen geschaffen, die eindeutig über Paket- und Rollennamen sowie einer laufenden Nummer angesprochen werden. Zwischen erfüllenden und benötigten Positionen können die Agenten bzw. Gruppen Verbindungen herstellen (s. Abbildung 7). Für den Agenten bedeutet dies, dass er sich verpflichtet, die über diese Verbindung eingehenden und zur Rolle der Position passenden Aufgaben zu erledigen. Die Gruppe kann sich im Gegenzug darauf verlassen, dass der verbundene Agent die gestellten Aufgaben annimmt und bestmöglich bearbeitet.

Zur Laufzeit können Agent und Gruppe selbstständig entscheiden, welche Rollen gerade aktiv sein sollen. Dies führt dazu, dass auch die Anzahl der vorhandenen Positionen dynamisch angepasst werden müssen. Zusätzlich können im Rahmen der Kardinalitäten jederzeit neue Positionen erzeugt oder entfernt werden. Allerdings darf eine belegte Position nicht ohne weiteres entfernt werden, d.h. sie kann erst gelöscht werden, wenn die zugehörige Verbindung aufgegeben wurde. Ähnlich sollte eine Rolle nur dann deaktiviert werden, wenn keine verbundenen Positionen mehr vorhanden sind.

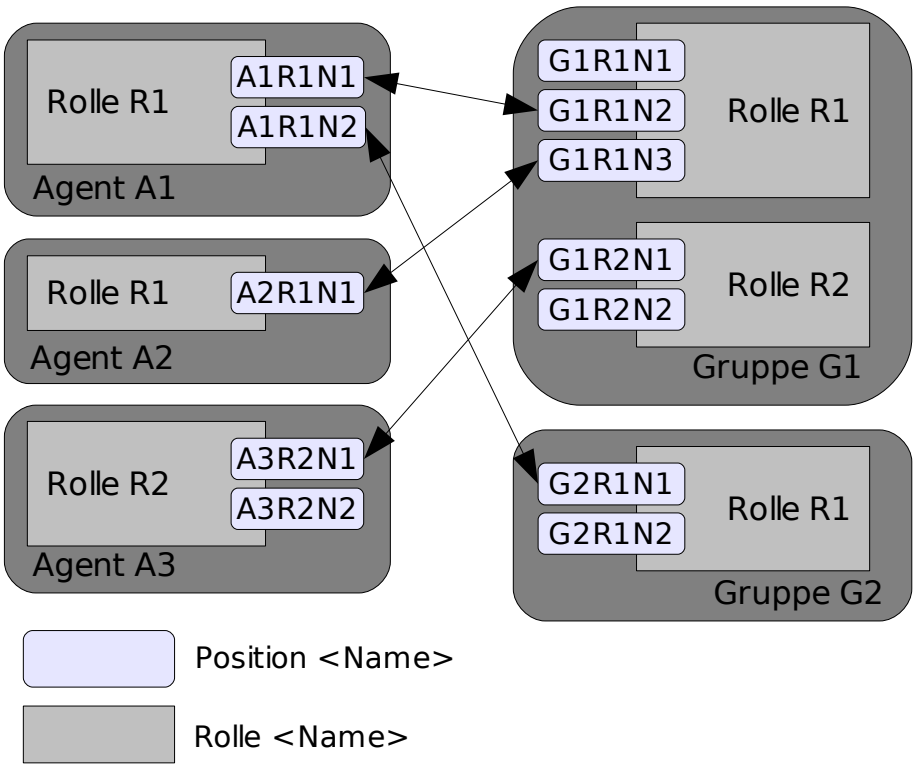

Abbildung 7: Rollen und Positionen im BDI-Gruppenmodell

5.3 Der GMS: Ein einfacher Matchmaker

Da Agenten und Gruppen sich nicht von selbst finden, war es notwendig, einen Mechanismus zu implementieren, der die im vorherigen Abschnitt angesprochenen Verbindungen zwischen den Positionen vermittelt. Hierfür wurde ein einfacher *Matchmaker* entwickelt, bei dem sowohl Agenten als auch Gruppen ihre Positionen anmelden. Dieser löst das im BDI-Modell noch offen gelassene Problem der Gruppenbildung.

Der als regulärer Jadex-Agent implementierte *GMS* (*Group Matchmaker Service*) führt eine Datenbank, die alle angemeldeten Positionen von Gruppen und Agenten enthält. Sobald geeignete Übereinstimmungen festgestellt werden, meldet der *Matchmaker* dies dem Agenten, der eine Rolle übernehmen möchte. Außerdem werden die jeweiligen Positionen als vergeben markiert, um eine doppelte Zuweisung zu verhindern.

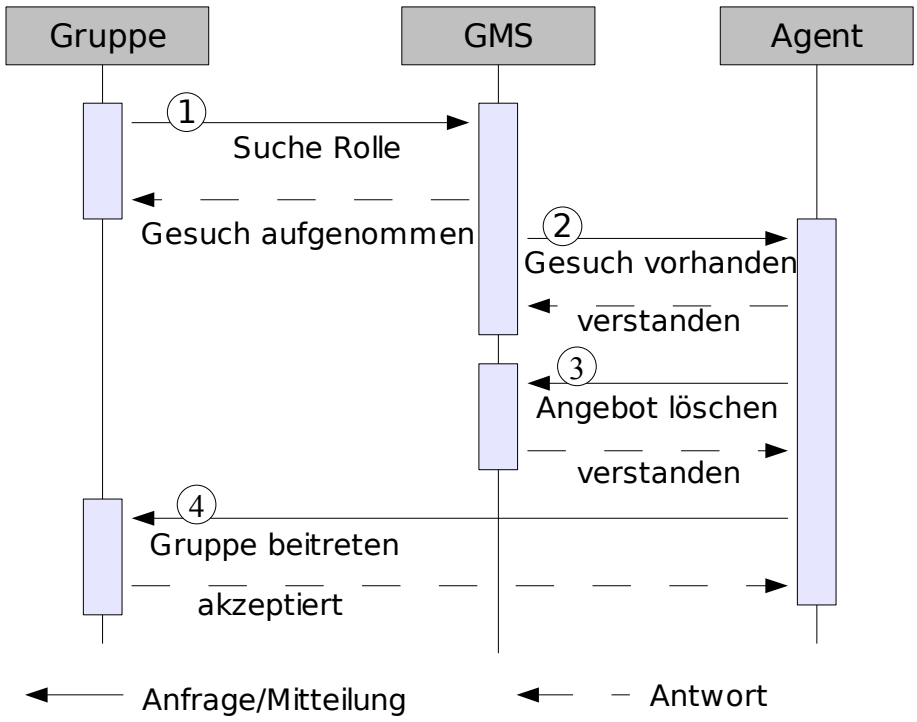

Abbildung 8: Gruppenbildung

Abbildung 8 zeigt den grundsätzlichen Ablauf bei der Bildung einer Gruppe unter Zuhilfenahme des *GMS*. Im ersten Schritt (1) teilt die Gruppe dem *GMS* mit, dass ein Agent mit einer bestimmten Rolle gesucht wird. Bei Übereinstimmung meldet der *GMS* dies dem jeweiligen Agenten (2), welcher wiederum sein Angebot aus der Datenbank des *GMS* entfernt (3) und dann versucht, der Gruppe beizutreten (4).

5.4 Gruppenhandlungen und -Pläne

Gruppenhandlungen bilden eine Schnittstelle zwischen den Gruppen und Agenten. Sie werden innerhalb von Gruppenplänen erzeugt und dann an einzelne Agenten delegiert, wobei jede Gruppenhandlung zur Ausführung lokaler Ziele innerhalb eines Agenten führt. Ähnlich wie für Ziele und Pläne kann der Entwickler bestimmte Muster von Gruppenhandlungen definieren. Beispiel 5 zeigt eine solche Definition, in der jede Handlung mit einem Namen und einer geeigneten Rolle versehen wird. Zusätzlich ist die Angabe von Parametern möglich.

```
1  <groupactions>
2
3    <groupaction name="gehezu"
4                 rolename="ta_Traeger_r">
5      <parameter name="position" class="Koordinate" />
6    </groupaction>
7
8    <groupaction name="heben"
9                 rolename="ta_Traeger_r">
10     <parameter name="kiste" class="Integer" />
11   </groupaction>
12
13 </groupactions>
```

Beispiel 5: Gruppenhandlungen, die von Agenten durchgeführt werden sollen

Das eigentliche Ziel der Gruppe wird durch herkömmliche *Jadex-Goals* im Gruppenverwalter dargestellt. Um diese zu erfüllen kommen entweder reguläre Pläne oder Gruppenpläne zum Einsatz. Means-End-Reasoning und Deliberation erfolgen im Prototyp für die Gruppe mit den gleichen Verfahren wie auf Agentenebene. Der Gruppenverwalter kann also entscheiden, ob er das Ziel selbst lösen will (Pläne) oder Teilaufgaben an andere Agenten versendet (Gruppenpläne).

```
14 <groupplan name="KisteWegBringenGroupPlan">
15   <parameter name="kiste" class="Integer">
16     <goalmapping ref="KisteWegBringen.kiste"/>
17   </parameter>
18
19    <body>new KisteWegBringenGroupPlan()</body>
20
21    <trigger>
22      <goal ref="KisteWegBringen"/>
23   </trigger>
24
25   <precondition>
26     $beliefbase.env.getKistenGewicht($goal.kiste)>1
27   </precondition>
28 </groupplan>
```

Beispiel 6: Gruppenplan

Auf ähnliche Weise werden Gruppenpläne im *GDF* eingeführt. Es stehen dabei alle von einfachen Plänen bekannten Elemente zur Verfügung (s. Beispiel 6). Der Körper eines Gruppenplans unterscheidet sich von individuellen Plänen durch zusätzliche Konstrukte, die die Nutzung des Gruppenmanagement erlauben. Neben Funktionen zum Zugriff auf die

Rollen und Positionen kann der Status des Gruppenbildungsprozesses abgefragt werden. Weiterhin können Aufgaben an einzelne Teilnehmer der Gruppe delegiert und deren jeweiliges Ergebnis abgefragt werden.

```
1   public void body() {
2
3     Irole r1 = getRolebase().getRole("ta_Traeger_f");
4     r1.setActive(false);
5
6     IGroupAction gg =
7                 createGroupAction("achievegehezu");
8     gg.getParameter("position").setValue(position);
9     GroupActionFilter ggf = dispatchGroupAction(gg);
10
11    waitForGroupAction(ggf);
12
13    IRole r2 = getRolebase().getRole("ta_Traeger_r");
14    releaseAgents(r2);
15
16  }
```

Beispiel 7: Beispielanweisung eine Gruppenplans

In Beispiel 7 ist der Körper eines Gruppenplans skizziert. In Zeile 3 wird der Status einer Rolle modifiziert. Der nächste Block (Zeile 6-9) erzeugt eine neue Gruppenhandlung, setzt den Wert eines Parameters und bringt diese dann zur Ausführung. Die Anweisung in Zeile 11 lässt den Gruppenplan warten, bis das vorher gestartete Ziel abgearbeitet wurde. Die letzten beiden Anweisungen in Zeile 13 und 14 lösen die Gruppe auf, indem alle Agenten, die eine bestimmte Rolle in der Gruppe gespielt haben, entfernt werden.

In der folgenden Übersicht sind alle wichtigen Methoden, die ein Gruppenplan bereithält, mit einer kurzen Erklärung zusammengefasst:

`public IPositionbase getPositionbase();`

liefert eine Referenz auf den Container, der alle vorhandenen Positionen des Agenten bzw. der Gruppe enthält

`public IPosition getPosition(String name);`

ermittelt eine Position anhand ihres Namens

`public IRolebase getRolebase();`

liefert eine Referenz auf den Container, der alle Rollen des Agenten bzw. der Gruppe enthält

```
public void waitUntilGroupFormation();
```

wartet, bis die Gruppe vollständig ist

```
public boolean isGroupComplete();
```

prüft, ob die Gruppe vollständig ist

```
public IGroupActionbase getGroupActionbase();
```

liefert eine Referenz auf den Container, der alle bekannten Gruppenhandlungen enthält

```
public IGroupAction createGroupAction(String type);
```

erzeugt eine neue Gruppenhandlung vom Typ *type*

```
public GroupActionFilter dispatchGroupAction(IGroupAction
                                             action);
```

löst die Verarbeitung einer bestimmten Gruppenhandlung aus. Das System sucht selbstständig nach einem geeigneten Agenten, der die Aufgabe übernehmen könnte.

```
public GroupActionFilter dispatchGroupAction(IGroupAction
                                action, IPosition position);
```

löst die Verarbeitung einer bestimmten Gruppenhandlung aus. Der ausführende Agent wird nicht automatisch ermittelt.

```
public void waitForGroupAction(GroupActionFilter filter);
```

wartet, bis eine bestimmte Gruppenhandlung abgeschlossen wurde

```
public void dispatchGroupActionAndWait(IGroupAction
                                       action);
```

löst die Verarbeitung einer bestimmten Gruppenhandlung aus und wartet bis zu dessen Erfüllung

```
public List getUsedPositions();
```

liefert alle Positionen (d.h. Agenten), die derzeit mit der Bearbeitung eines Teilproblems beschäftigt sind

```
public void releaseAgents(IRole role);
```

löst alle Verbindungen, die zwischen den Positionen einer Rolle bestehen auf. Entspricht dem Entfernen derjenigen Agenten, die eine bestimmte Rolle in der Gruppe spielen.

Alle Agenten, die sich zur Erfüllung einer Rolle verpflichtet haben, erhalten über das Kommunikationssystem die Anweisung zur Erfüllung von Teilzielen in Form von *GroupActions*. Diese lösen entweder direkt oder durch ein Mapping (den *GoalMappings*) lokale Ziele aus, die vom Agenten zu erfüllen sind. Ein Mechanismus überwacht den Zustand der Abarbeitung und meldet den Erfolg bzw. Fehlschlag an die Gruppe.

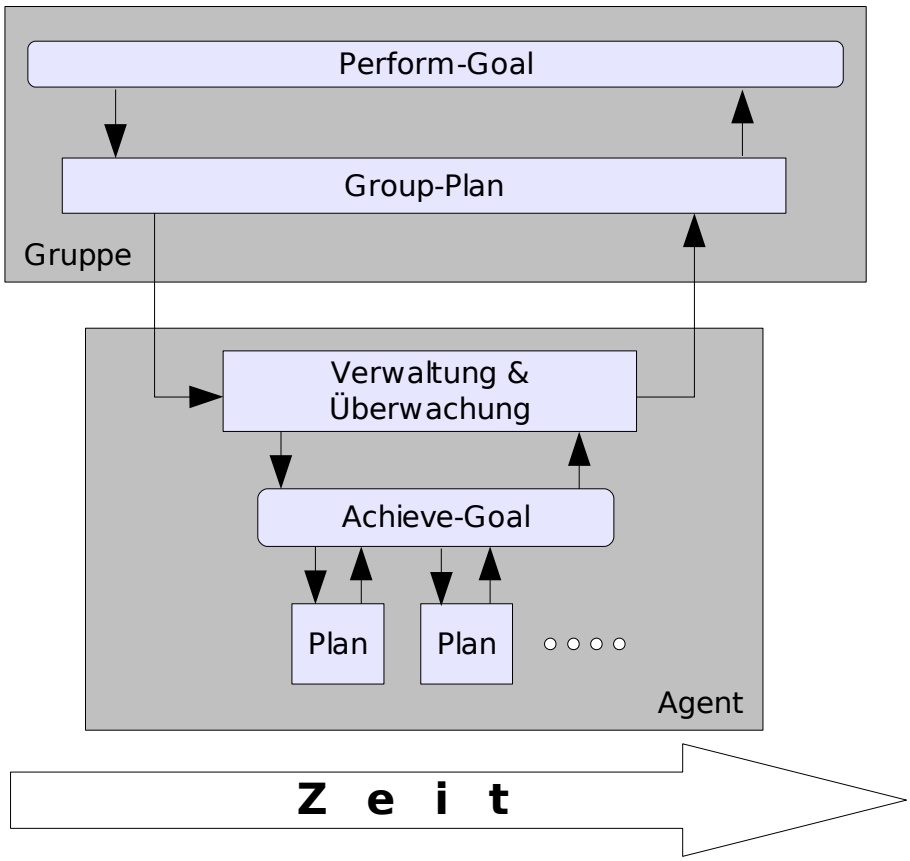

Abbildung 9: Zusammenspiel von Gruppenzielen und Plänen

Die Abbildung 9 skizziert den Ablauf bei der Bearbeitung von Gruppenzielen und Plänen. Im Beispiel verfügt die Gruppe über ein *Perform-Goal*, welches zur Ausführung eines *GroupPlans* führt. Dieser soll die Abarbeitung eines Ziels auf Agentenebene auslösen. Hierfür werden die benötigten Informationen an eine Steuerungseinheit im Agenten übermittelt, die ein geeignetes Ziel (hier *Achieve-Goal*) auswählt. Das lokale Ziel wird nach den üblichen Regeln des BDI-Modells bearbeitet, aber zusätzlich von der Steuerungseinheit überwacht. Sobald das Ziel erfolgreich (oder fehlgeschlagen) ist, wird das Ergebnis

an die Gruppe zurückgeschickt. Natürlich kann die Gruppe auch mehrere Ziele gleichzeitig verfolgen und verschiedene Gruppenpläne ausführen, an denen wiederum mehrere Agenten beteiligt sind.

5.5 Gruppenwissen

Gruppenwissen wird in der Implementation mit Hilfe des vorhandenen Jadexbeliefsystems umgesetzt. Dazu wurde das *Belief*-Element in *GroupBelief* umbenannt und um das Attribut *position* (Zeile 2 in Beispiel 8) erweitert. Da jede aktive Position eine Verbindung zu einer Gruppe besitzt, kann genau bestimmt werden, welches *Belief* zu welcher Gruppe gehören soll. Wenn sich nun der Wert eines *Beliefs* ändert, wird dieser an die entsprechende Gruppe gesendet. Von dort aus werden alle Mitglieder der Gruppe über den neuen Wert informiert.

```
1  <beliefs>
2    <groupbelief name="pos" class="long" position="a">
3      <fact>100</fact>
4    </groupbelief>
5  </beliefs>
```

Beispiel 8: Gruppenwissen mit GroupBeliefs

Neues Wissen wird auf diese Weise automatisch in der gesamten Gruppe verteilt. Ansonsten verhält sich das *GroupBelief* ebenso, wie ein herkömmliches *Belief*, d.h. es kann mit den vorhandenen Jadex-Funktionen ausgelesen bzw. verändert werden. Natürlich führen Änderungen eines *GroupBeliefs* auch zur Auslösung von Bedingungen innerhalb des Agenten sowie zur Erfüllung von Erfolgs- bzw. Abbruchbedingungen in Plänen und Zielen.

5.6 GUIs für GMS und Gruppe

Die Implementation enthält derzeit zwei graphische Benutzerinterfaces, die den Datenbestand des *GMS (Group Matchmaker Service)* und den Zustand der Rollen und Positionen in einem Agenten bzw. einer Gruppe visualisieren. Diese GUIs wurden vor dem Hintergrund entworfen, den Systementwickler beim Testen einer Multiagentenanwendung zu

unterstützen. Durch die angezeigten Informationen können Rückschlüssen auf den aktuellen Zustand des Agenten bzw. der Gruppe und deren Einbindung in laufende Gruppenprozesse gezogen werden.

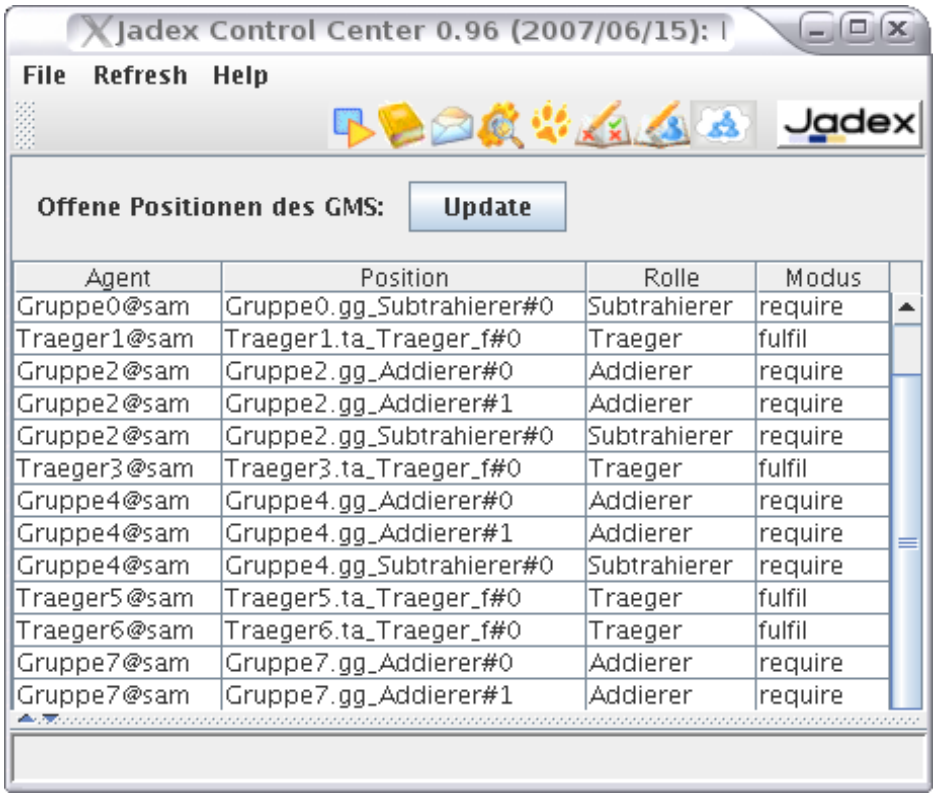

Abbildung 10: GUI des GMS

Die Abbildung 10 zeigt die GUI des *GMS*, welche als Plugin für das *Jadex-Controlcenter* (*JCC*) entwickelt wurde. In der Tabelle werden in der ersten Spalte alle Agenten und Gruppen aufgelistet, die Positionen angemeldet haben. Die zweite Spalte zeigt den genauen Namen der Position und in der nächsten Spalte befindet sich die zugehörige Rolle. Die letzte Spalte gibt an, ob die Position besetzt werden soll (*fulfil* = die zur Position gehörigen Rolle soll erfüllt werden, *require* = eine Gruppe sucht einen Agenten oder eine andere Gruppe, die die Position besetzen kann).

Die in Abbildung 11 beispielhaft dargestellte GUI kann durch einen speziellen Plan von einem Agenten oder einer Gruppe gestartet werden. Auf der linken Seite werden die Verbindungen mit anderen Agenten bzw. Gruppen aufgelistet. In der Mitte befinden sich alle vorhandenen Positionen. Wenn eine Position ausgewählt wird, werden auf der rechten Seite weitere Details wie die aktuelle Verbindung (*assoc*) oder

zugehörige Verarbeitungsschritte (*activeGoals*, *activeGoalMappings*) angezeigt. Der Knopf „(De-)Aktivieren“ erlaubt, eine Rolle manuell zu aktivieren bzw. zu deaktivieren.

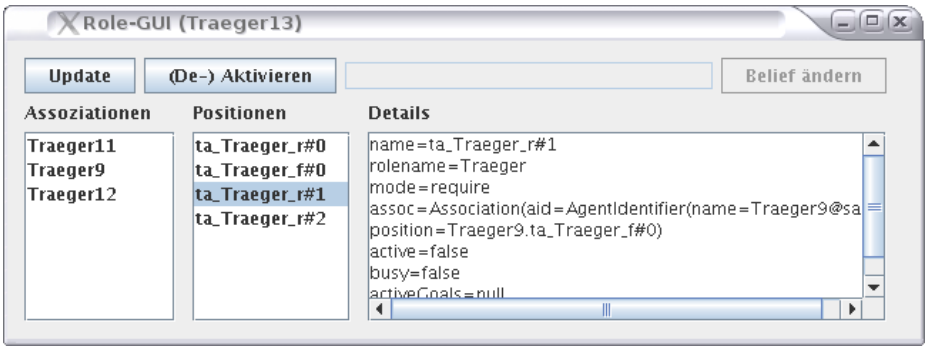

Abbildung 11: Positionen und Rollen

5.7 Das Kommunikationssystem

Das Kommunikationssystem bildet wie auch im BDI-Gruppenmodell die Schnittstelle zwischen den einzelnen Agenten und der Gruppe. Immer wenn Informationen von der Gruppe zum Agenten oder umgekehrt übertragen werden müssen, kommt dieses System zum Einsatz. Auch der Austausch von Daten mit dem *GMS* wird hierüber ausgeführt.

Für die unterschiedlichen Inhalte wurden eine Reihen von Klassen entworfen, die die jeweiligen Informationen aufnehmen. In jedem Kommunikationsvorgang wird ein neues Objekt der gewünschten Klasse erzeugt und mit den entsprechenden Daten gefüllt. Der Versand dieser Nachrichten erfolgt mit den vom *Jadex-Framework* bereitgestellten Mechanismen des Nachrichtenversands.

Jeder Kommunikationsvorgang besteht immer aus zwei Nachrichten. Die erste enthält ein Objekt, welches von der Klasse *AGAction* erbt. Die zur Verfügung stehenden Klassen und deren Bedeutung sind in der Abbildung 12 aufgelistet, wobei die Implementation bei Bedarf leicht um weitere Nachrichtentypen erweitert werden kann. Der Empfänger quittiert den Erhalt mit einer weiteren Nachricht, die immer eine Klasse *AGActionDone* enthält. Aus dieser kann der Absender erkennen, ob die Nachricht erfolgreich angekommen ist und verarbeitet wurde.

DeRegisterAgentInGroup	A→G	Der Agent möchte eine Position aus einer Gruppe anmelden oder entfernen.
DeRegisterPositionAtGms	A→M	Ein Agent möchte eine Position beim *GMS* an- bzw. abmelden.
DispatchGroupActionAt-Agent	G→A	Die Gruppe möchte eine neue Gruppenhandlung im Agenten auslösen.
GetEntriesFromGms	*→M	Fordert eine Liste aller beim *GMS* gemeldeten Positionen an.
InformAgentAboutGroup	M→A	Informiert einen Agenten, dass eine Gruppe eine geeignete Position offen hat.
InformAgentThatGroup-PlanFailed	G→A	Teilt einem Agenten mit, dass der Gruppenplan gescheitert ist. Der Agent kann die noch laufenden Ziele abbrechen.
InformGroupAbout-DispatchedAction	A→G	Informiert die Gruppe über den Ausgang der Bearbeitung einer Handlung (Erfolg / Fehlschlag).
RequestAgentToLeaveGroup	G→A	Teilt einem Agenten mit, dass er die Gruppe verlassen soll. Der Agent sollte mit einer *DeRegisterAgentInGroup* Nachricht reagieren.
SendNewFactToAgent	G→A	Die Gruppe sendet einen neuen Wert für ein *Belief*.
SendNewFactToGroup	A→G	Ein Agent sendet einen neuen Wert für ein *Belief* an die Gruppe. Diese sollte den neuen Wert mit Hilfe von *SendNewFactToAgent* weiter verteilen.
A = Agent, G = Gruppe, M = Matchmaker (GMS)		

Abbildung 12: vorhandene Nachrichtentypen

5.8 Einschränkungen

Der entworfene Prototyp enthält eine Reihe von Einschränkungen, auf die an dieser Stelle kurz eingegangen werden soll, da sie ein guter Ausgangspunkt für weitere vertiefende Untersuchungen bieten und gegebenenfalls den Rahmen neuer Arbeiten bilden könnten.

Deliberation auf Gruppenebene

Da lediglich die in *Jadex* vorhandenen Deliberationsmechanismen zur Anwendung kommen, wird bei der Entscheidung, welche Gruppenziele aktuell verfolgt werden sollen, keine Rücksicht auf die Intentionen der einzelnen Mitglieder genommen. Zwar kann über das Gruppenwissen bedingt Einfluss auf das Verfahren genommen werden, trotzdem wäre es wünschenswert, entweder die *Easy Deliberation Strategy* entsprechend zu erweitern oder ein neues Verfahren zu entwerfen.

Das Kommunikationssystem könnte genutzt werden, um im Bedarfsfall Informationen über den aktuellen Zustand der einzelnen Agenten zu erlangen. Dieses Wissen sollte dann in den Entscheidungsprozess einfließen, um beispielsweise Konflikte zwischen individuellen Zielen und Gruppenzielen zu erkennen.

Rollen

Im Gegensatz zum BDI-Gruppenmodell wird kein gemeinsamer Container für alle im System vorhandenen Rollen eingesetzt. Es wird daher vorausgesetzt, dass der Systementwickler beim Entwurf keine Rollennamen doppelt belegt, da dies von der Implementation nicht als Fehler erkannt werden kann.

Matchmaker

Der implementierte *Matchmaker* arbeitet nach dem Zufallsprinzip. Dies bedeutet, dass keine weiteren Attribute – wie Auslastung, Ausführungsqualität oder Vertrauen der einzelne Agenten – einfließen, sondern einfach die nächsten, passenden Einträge (d.h. Positionen) verbunden werden. Um die Leistung des Systems zu optimieren, wäre es sinnvoll, bei der Vergabe weitere Parameter zu berücksichtigen.

GroupBeliefs

In ihrer derzeitigen Form unterstützen *GroupBeliefs* keine transaktionalen Aspekte. Gleichzeitige Lese- bzw. Schreibzugriffe können daher zu Problemen führen und müssen vom Agentenentwickler selbst berücksichtigt werden. Um die Nutzung von *GroupBeliefs* zu vereinfachen, sollten geeignete Mechanismen (z.B. Sperren) eingeführt werden.

5.9 Zusammenfassung

Die vorgestellte Implementation setzt alle wichtigen Kernaspekte des im vorangegangenen Kapitel vorgestellten BDI-Gruppenmodells um und erweitert durch seine vollständige Integration das Jadex-Agentenframework um generische Koordinations- und Gruppenmechanismen. Es bietet mit Gruppenplänen die Möglichkeit, Ziele zu zerlegen und Teilaufgaben an Agenten zu übertragen. Auf Ebene der Agenten wurde ein Mechanismus implementiert, der die Abarbeitung von Gruppenzielen überwacht und die Gruppe über Erfolg bzw. Misserfolg unterrichtet.

Allerdings muss das System bis zu einem möglichen Einsatz in produktiven Umgebungen an vielen Stellen optimiert und ausgiebig getestet werden. Die im vorangegangenen Abschnitt angesprochenen Aspekte könnten in weiteren Arbeiten vertieft werden, um die Leistung des Gesamtsystems zu verbessern.

6 Anwendungsbeispiele

Der Entwurf des BDI-Gruppenmodells wurde von der Betrachtung zweier Anwendungsszenarien begleitet. Beim ersten handelt es sich um die Lösung einer einfachen Rechnenaufgabe durch mehrere Agenten. Das zweite Szenario ist wesentlich komplexer und fordert ein Zusammenspiel von Agenten bei der Bewältigung einer Transportaufgabe. In den folgenden Abschnitten werden beide Szenarien vorgestellt, wobei deutlich wird, wie man das BDI-Gruppenmodell verwendet.

6.1 Rechenaufgabe

In diesem Beispiel werden die Agenten mit der Lösung einer einfachen Rechenaufgabe konfrontiert. Dabei ist jeder Agent zuständig für eine bestimmte Rechenoperation (Addition, Subtraktion). Entsprechend übernimmt der jeweilige Agent die Rolle des Addierers oder Subtrahierers. Der Gruppenplan beschreibt in diesem Fall, wie die Gesamtaufgabe in für die Experten nützliche Teile zerlegt und die Ergebnisse später wieder zusammengesetzt werden sollen. Auch wenn die Idee an sich einfach ist und auch ohne weiteres von einem Agenten gelöst werden könnte, bietet dieses Szenario Ansatzpunkte, um über Rollen, Gruppenbildung und Gruppenziele zu reflektieren.

6.1.1 Implementation

Die Realisierung besteht aus zwei Agenten, wobei der eine zur Addition und der andere zur Subtraktion fähig ist. Beide Agenten verfügen über geeignete Ziele (*gg_addiere* bzw. *subtrahiere*) und Pläne (*AddierePlan* bzw. *SubtrahierePlan*), um gestellte Rechenaufgaben lösen zu können. Der für die Subtraktion zuständige Agent zeigt außerdem die Verwendung von *GoalMappings* (s. Beispiel 9), in dem die Gruppenhandlung *gg_subtrahiere* auf das lokale Ziel *subtrahiere* mitsamt den benötigten Parametern abgebildet wird.

```
1  <goalmapping name="gg_subtrahiere">
2
3      <parameter name="value1" class="Integer">
4          <goalmapping ref="subtrahiere.value1"/>
5      </parameter>
6
7      <parameter name="value2" class="Integer">
8          <goalmapping ref="subtrahiere.value2"/>
9      </parameter>
10
11      <parameter name="result" class="Integer">
12          <goalmapping ref="subtrahiere.result"/>
13     </parameter>
14
15     <trigger>
16         <goal ref="subtrahiere"/>
17     </trigger>
18
19  </goalmapping>
```

Beispiel 9: Goalmappings erlauben die Abbildung auf lokale Ziele

Nach dem Start registrieren sich die Agenten beim *GMS* mit ihrer jeweiligen Rolle und bieten damit ihre Dienste als Addierer bzw. Subtrahierer an. Interessierte Gruppen können auf diesem Weg geeignete Experten zum Lösen von komplexen Aufgaben finden. Da die Agenten sonst keine weiteren Ziele besitzen, warten sie bis der *GMS* ihnen eine geeignete Gruppe zuweist.

In diesem Beispiel wird von einer im Voraus festgelegten Aufgabe mit variablen Werten ((v1 + v2) – (v3 + v4)) ausgegangen. Die Gruppendefinition besteht daher im Wesentlichen aus einem *Achieve-Goal* und einem zugehörigen Gruppenplan (s. Beispiel 10), welche beide nach dem Start einer Gruppe proaktiv ausgelöst werden. Um auch den Gebrauch von parallelen Handlungen zu demonstrieren, sollen die beiden Additionen von zwei unterschiedlichen Agenten zur gleichen Zeit ausgeführt werden. Daher werden zur Lösung der Aufgabe neben einem Subtrahierer insgesamt zwei Addierer benötigt.

```
1   waitUntilGroupFormation();
2
3   IGroupAction a1 = createGroupAction("ga_addiere");
4   a1.getParameter("value1").setValue(v1);
5   a1.getParameter("value2").setValue(v2);
6   GroupActionFilter gafa1 = dispatchGroupAction(a1);
7
8   IGroupAction a2 = createGroupAction("ga_addiere");
9   a2.getParameter("value1").setValue(v3);
10  a2.getParameter("value2").setValue(v4);
11  GroupActionFilter gafa2 = dispatchGroupAction(a2);
12
13  waitForGroupAction(gafa1);
14  waitForGroupAction(gafa2);
15
16  Integer result1;
17  Integer result2;
18
19  result1 = a1.getParameter("result").getValue();
20  result2 = a2.getParameter("result").getValue();
21
22  IGroupAction s = createGroupAction("ga_subtra");
23  s.getParameter("value1").setValue(result1);
24  s.getParameter("value2").setValue(result2);
25  dispatchGroupActionAndWait(s);
26
27  System.out.println(
28        s.getParameter("result").getValue());
```

Beispiel 10: Gruppenplan zur Lösung der Rechenaufgabe

Der erste Schritt des Gruppenplans besteht darin, zu warten, bis die Gruppe vollständig ist. Die Formation der Gruppe erfolgt automatisch mit Hilfe des *GMS*, d.h. das System erkennt durch die in der Gruppenspezifikation festgelegten Rollen selbstständig, wieviele Agenten in welchen Rollen erforderlich sind und stellt eine entsprechende Anfrage an den *GMS*. Dieser wiederum prüft, ob ihm geeignete Agenten bekannt sind und informiert diese über die neue Gruppe. Die Gruppe selbst wartet aufgrund der Anweisung in Zeile 1 (s. Beispiel 10), bis alle benötigten Agenten der Gruppe beigetreten sind.

In den Zeilen 3-6 wird eine neue Gruppenhandlung vom Muster *gg_addiere* erzeugt und mit den Werten der Aufgabe gefüllt. Durch das Muster ist dem System bekannt, dass die Handlung von einem Agenten durchgeführt werden muss, der die Rolle *Addierer* erfüllen kann. Die konkrete Zuweisung an einen Agenten erfolgt daher automatisch während der Laufzeit und muss vom Entwickler nicht weiter bedacht werden.

Die Zeilen 8-11 enthalten den analogen Fall für die zweite Addition. Da die Additionen in diesem Beispiel parallel ausgeführt werden sollen, sind hierfür zwei Agenten erforderlich. Ist nur ein Agent gewünscht, könnte zwischen den Zeilen 6 und 8 eine Warteanweisung eingefügt werden. Das System würde dann die Ergebnisse der ersten Rechnung abwarten und dann die zweite Aufgabe an den selben Agenten vergeben, da dieser wieder frei geworden ist.

Die Subtraktion kann nur nach Erhalt der Additionsergebnisse durchgeführt werden. Daher muss im Gruppenplan in den Zeilen 13 und 14 auf die Lösung der beiden Agenten gewartet werden. Zum Schluss wird in den Zeilen 22-25 ähnlich wie bei der Addition die Subtraktion durch Erzeugen einer geeigneten Gruppenhandlung angestoßen und dann auf das Ergebnis gewartet.

6.1.2 Zusammenfassung

Das erläuterte Beispiel zeigt auf einfache Weise die Verwendung von Rollen, Bildung einer Gruppe und Zuweisung von Teilaufgaben. Der Entwickler kann beim Entwurf eines Multiagentensystems auf abstrahierende Elemente zurückgreifen. Hier muss ihm beispielsweise nicht bekannt sein, wie die Rechenoperationen im Detail ausgeführt werden, es genügt zu wissen, dass es Agenten gibt, die über die entsprechenden Fähigkeiten verfügen. Auch die Aspekte der Kommunikation werden vollständig verborgen, so dass es nicht notwendig ist, eigene Protokolle für den Austausch der Anweisungen bzw. Daten zu entwerfen.

6.2 GroupInGroup

Mit diesem Beispiel soll die hierarchische Strukturierung von Gruppen in Teilgruppen bzw. einzelne Agenten demonstriert werden. Hierfür wurde das im vorangegangenen Kapitel 6.1 (Rechenaufgabe) vorgestellte Multiagentensystem erweitert. Neu ist vor allem die Gruppe *Addiere4Gruppe*, welche eine Addition von vier anstatt bisher zwei Werten erlaubt. Dabei realisiert die Gruppe die Addition aber nicht selbst, sondern greift auf den Additionsexperten für zwei Werte zurück. Insgesamt ergibt sich die in Abbildung 13 dargestellte Hierarchie.

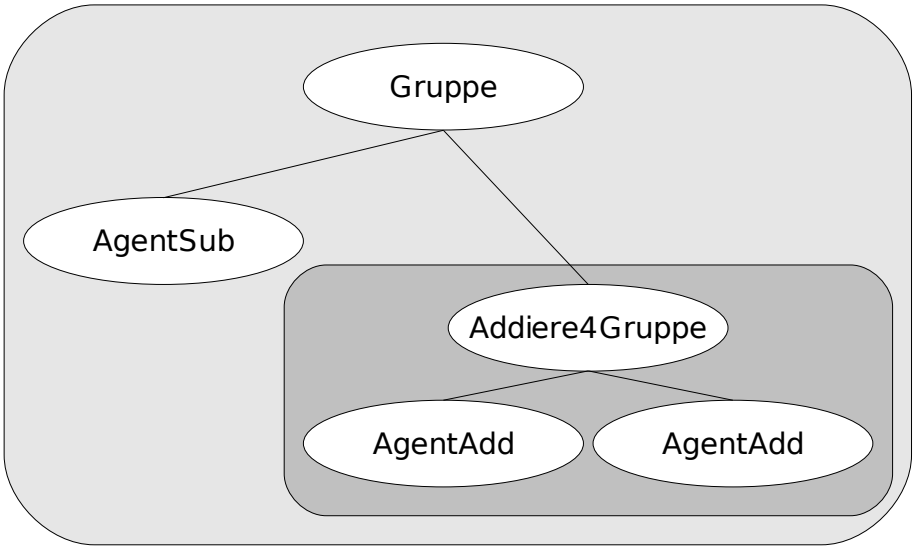

Abbildung 13: Gruppen und Subgruppen

6.2.1 Implementation

Die Implementation dieses Beispiels hat große Ähnlichkeit mit dem Vorangegangenen. So konnten die Additions- und Subtraktionsagenten ohne Änderungen wiederverwendet werden. Auch der Gruppenplan zur Lösung der Gesamtaufgabe musste nur die neue Aufgabenstellung *((v1 + v2 + v3 + 4) – (v5 – v6))* und die Verwendung der neuen Rolle *Addierer4* angepasst werden. Der Unterschied besteht hier lediglich in der Anzahl der zu übergebenen Parameter. Neu ist die Spezifikation der Gruppe *Addiere4Gruppe* und ein zugehöriger *Addiere4GroupPlan*, wobei der Plan aber auf die gleichen Konstrukte wie die im vorherigen Abschnitt eingeführten zurückgreift.

Der Ablauf beginnt damit, dass die Gruppe die gestellte Aufgabe in die zwei Teile (v1 + v2 + v3 + v4) und (v5 – v6) zerlegt. Der erste Teil wird an die Gruppe *Addiere4Gruppe* weitergeleitet und der zweite an einen Subtrahierer. Aus Sicht der obersten Gruppe spielt es keine Rolle, dass es sich bei *Addiere4Gruppe* um eine Gruppe handelt, wichtig ist vielmehr, dass diese Einheit in der Lage ist, die Rolle *Addierer4* zu erfüllen und damit die Addition von vier Werten ermöglicht.

Die Gruppe *Addiere4Gruppe* versucht das gestellte Problem durch Zerlegung in drei Teilschritte, d.h. Addition der ersten beiden, der letzten beiden Werte und deren Ergebnisse, zu lösen. Hierfür wird, wie bereits im ersten Beispiel, auf Experten für die Grundrechenarten zurückgegriffen.

6.2.2 Zusammenfassung

Dieses zweite Beispiel zeigt, wie eine hierarchische Strukturierung von Agenten und Gruppen ermöglicht wird. Auf den höheren Ebenen können Gruppen sowie Gruppenpläne noch sehr abstrakt sein. Nach unten gehend werden diese zwar immer detailreicher, enthalten aber jeweils nur noch einzelne Teilaspekte der vormals komplexen Aufgabe. Daneben lässt sich an diesem Beispiel auch die Wiederverwendung von Softwareteilen erkennen. Dadurch, dass die Agenten ihre konkrete Realisierung hinter Rollen verbergen, ist es möglich, große Teile des aus dem vorangegangenem Beispiel stammenden Programmcodes weiter zu verwenden. Die neue Gruppe konnte einfach auf die bereits existierenden Agenten zurückgreifen, ohne über deren Implementationsdetails Kenntnis zu haben.

6.3 BoxWorld

Im BoxWorld-Szenario geht es darum, ein etwas aufwändigeres, kooperatives Problem zu lösen, welches im Folgenden beschrieben wird. Das Beispiel wurde vor allem zum Testen der Implementation des BDI-Gruppenmodells genutzt, zeigt aber auch das mögliche Potential von Zusammenarbeit in der Gruppe auf. Die Problemlösung soll dabei möglichst autonom ablaufen, die Agenten verfügen also über keine im Voraus bekannte, umfassende Weltsicht. Auch haben sie keinen Einfluss auf die Handlungen anderer Agenten oder Zugriff auf deren Informationen. Dies ermöglicht die Verwendung in unterschiedlichsten Einsatzgebieten mit wechselnder Anzahl von Agenten.

Kern des Beispiels ist ein zweidimensionales Feld, in dem sich die Agenten frei bewegen können. In dieses Feld befinden sich Kisten, die unterschiedliche Gewichte haben können (hier nur ganzzahlige Werte). Die Aufgabe der Agenten besteht darin, die Kisten[2] (hellbraun) in ein speziell gekennzeichnetes Feld (gelb) zu tragen (s. Abbildung 14). Eine blaue Färbung der Agenten bedeutet, dass der jeweilige Agent keine Kiste trägt. Rot hingegen heißt, dass eine Kiste angehoben bzw. getragen wird. Kisten mit dem Gewichts eins können von einem einzelnen Agenten ohne weitere Hilfe transportiert werden. Um aber schwerere

2 Das Gewicht steht in der Visualisierung immer links oben neben der Kiste. Zusätzlich kann daneben der Name des Agenten auftauchen, der sich um den Transport der jeweiligen Kiste kümmert.

Kisten ins Ziel zu bringen, ist der Agent auf Unterstützung von anderen angewiesen. Zu diesem Zweck gründet der Agent eine Gruppe, deren Aufgabe darin besteht, die Kiste in den gelben Bereich zu tragen.

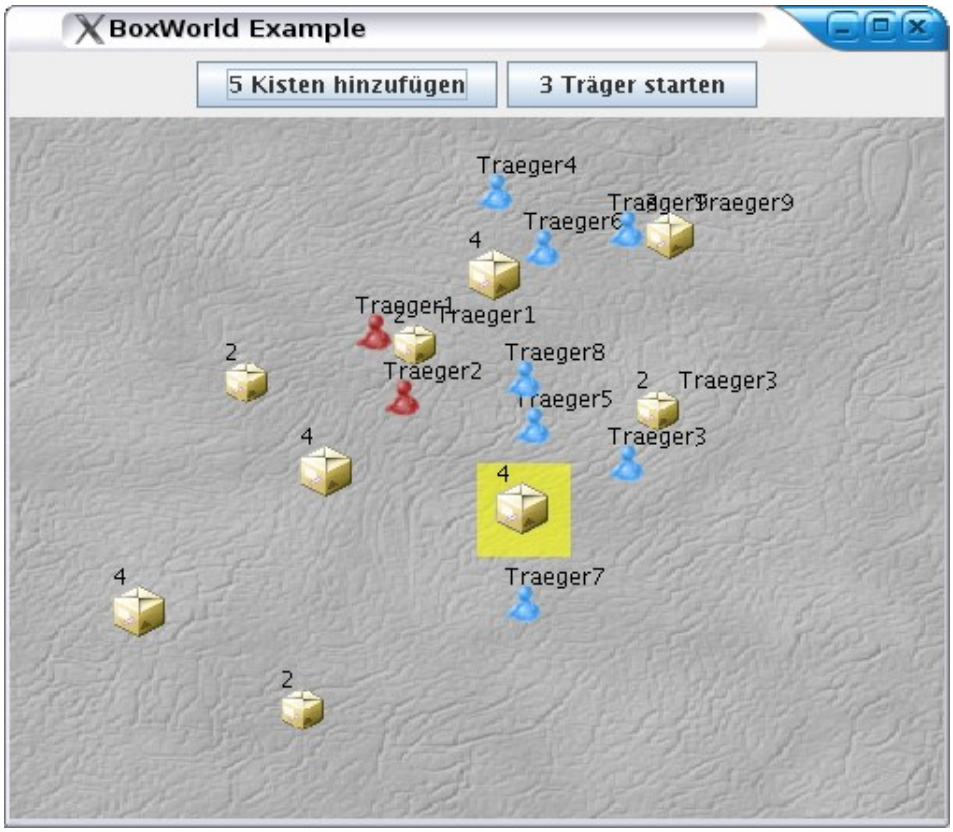

Abbildung 14: BoxWorld

Dieses Beispiel deckt alle Phasen einer kooperativen Handlung ab (s. 2.4.5 Cooperative Distributed Problem Solving). Zunächst muss erkannt werden, dass die Lösung der Transportaufgabe nur durch Kooperation möglich ist und eine entsprechende Gruppe gebildet werden. In der nächsten Phase müssen die Teilnehmer der Gruppe ihre Handlungen in geeigneter Weise koordinieren. Dies bedeutet, dass Anheben, Tragen und Absetzen von Kisten gemeinsam und mit ausreichender Anzahl von Agenten ausgeführt werden muss. Nur der letzte Schritt (d.h. die Zusammenführung der Teillösungen) ist in diesem Szenario nicht stark ausgeprägt. Er besteht hier lediglich in der gemeinsamen, erfolgreichen Feststellung, dass die getragene Kiste sich im Zielbereich befindet.

6.3.1 Implementation des Beispiels

Die Implementation kommt mit nur einem Agententyp aus, der auch gleichzeitig die Gruppendefinition enthält. Dieser modelliert die im Szenario benötigten *Träger*. Ein zweiter Agent (*Beobachter*) wurde zur Visualisierung des Feldes, der Agenten sowie Kisten entworfen.

Die Umgebung

Die Umgebung besteht aus einer Klasse, die alle relevanten Informationen des Feldes – wie die Positionen der Agenten und Kisten – enthält. Sie verfügt über verschiedene Operationen, die die Wahrnehmung und Modifikation der Umgebung ermöglichen. Jeder *Träger*-Agenten besitzt eine Referenz auf die Umgebung und kann so seine Position ändern oder eine Liste aller Kisten in seiner Nähe erhalten. Der Agent ist aber nicht in der Lage die Umgebung insgesamt wahrzunehmen. Der *Beobachter*-Agenten hingegen kann die Umgebungen vollständig einsehen und stellt die Informationen in einer GUI graphisch dar (s. auch Screenshot Abbildung 14).

Der Träger

Der aus Sicht der Kooperation interessante Teil befindet sich in den *Träger*-Agenten, die aus einer ganzen Reihe von möglichen Zielen und Plänen zusammengesetzt sind. Dieser Agententyp beinhaltet sowohl die Spezifikation des individuellen Verhaltens als auch die Aspekte der Gruppe. So kann jeder einzelne *Träger*-Agent zunächst versuchen, seine Aufgaben allein zu lösen und bei Bedarf aus sich heraus eine Gruppe gründen. In dieser ist er dann gleichzeitig Verwalter als auch Teilnehmer – er steuert damit neben seinem eigenem Verhalten auch die Handlungen der anderen.

Jeder *Träger* beginnt mit einem *performsuche-Goal*, was den Agenten zufällig durch die Umgebung laufen lässt, um eine neue Kiste zu finden. Für jedes der aufgeführten Goals (s. Abbildung 15) existieren geeignete Pläne, um möglichst das Ziel zu erfüllen. Beispielsweise erzeugt ein *achievegehezu-Goal* solange die notwendigen *achieveschrittmachen-Goals*, bis sich der Agent am Zielpunkt befindet. Bei jedem Schritt wird in der Umgebung nach nahe gelegenen Kisten gesucht. Wenn der Agent eine Kiste gefunden hat, wird ein Ziel vom Typ *KisteWegBringen* erzeugt. Dieses Ziel kann je nach Gewicht der Kiste einen von zwei Plänen zur Ausführung bringen. Bei einem Gewicht von eins wird der *KisteWegBringenPlan* ausgeführt, da keine weitere Hilfe notwendig ist.

Andernfalls kommt der *KisteWegBringenGroupPlan* zum Einsatz. Interessant ist das Verhalten bei einem größeren Gewicht (d.h. bei Ausführung des Gruppenplans), da nur dort die Bildung einer Gruppe und die Koordination der Teilnehmer nötig wird.

performsuche	Perform-Goal	Der Agent sucht in der Umgebung nach neuen Kisten, die von keinem anderen Agenten bearbeitet werden.
achievegehezu	Achieve-Goal	Der Agent geht zu einer bestimmten X/Y-Koordinate.
achieveschrittmachen	Achieve-Goal	Ist erfüllt, wenn der Agent einen Schritt in eine bestimmt Richtung gemacht hat.
achievekisteheben	Achieve-Goal	Lässt den Agenten eine Kisten anheben.
achievekisteabsetzen	Achieve-Goal	Lässt den Agenten eine Kiste absetzen.
KisteWegBringen	Achieve-Goal	Der Agent soll eine bestimmte Kisten in den Zielbereich tragen.

Abbildung 15: Ziele des Träger-Agenten im BoxWorld-Beispiel

Das Gruppenverhalten ist im *KisteWegBringenGroupPlan* festgelegt. Da während der Suche (performsuche) jeder Träger gleichzeitig beim *GMS* gemeldet ist, um gegebenenfalls andere Agenten zu helfen, beginnt der Gruppenplan damit, die eigene Registrierung zu entfernen. Nun will der Agent nicht mehr selbst unterstützen, sondern unterstützt werden. Hierfür fordert er über den *GMS* ausreichend viele Helfer an. Der nächste Schritt besteht darin, alle Teilnehmer zur Kiste zu schicken und zu warten, bis so viele Agenten angekommen sind, wie für das Gewicht nötig sind (Gewicht 2 erfordert 2 Agenten, Gewicht 3 entsprechend 3 Agenten usw.). Wenn die erforderliche Anzahl eingetroffen ist, wird die Kiste gemeinsam angehoben, zum Zielbereich getragen und dort wieder abgesetzt. Alle diese Aktionen werden parallel ausgeführt, d.h. alle beteiligten Agenten erhalten die entsprechenden Befehle gleichzeitig. Zuletzt wird die Gruppe wieder aufgelöst, so dass sich die Agenten entweder wieder auf die Suche begeben oder andere Gruppe unterstützen.

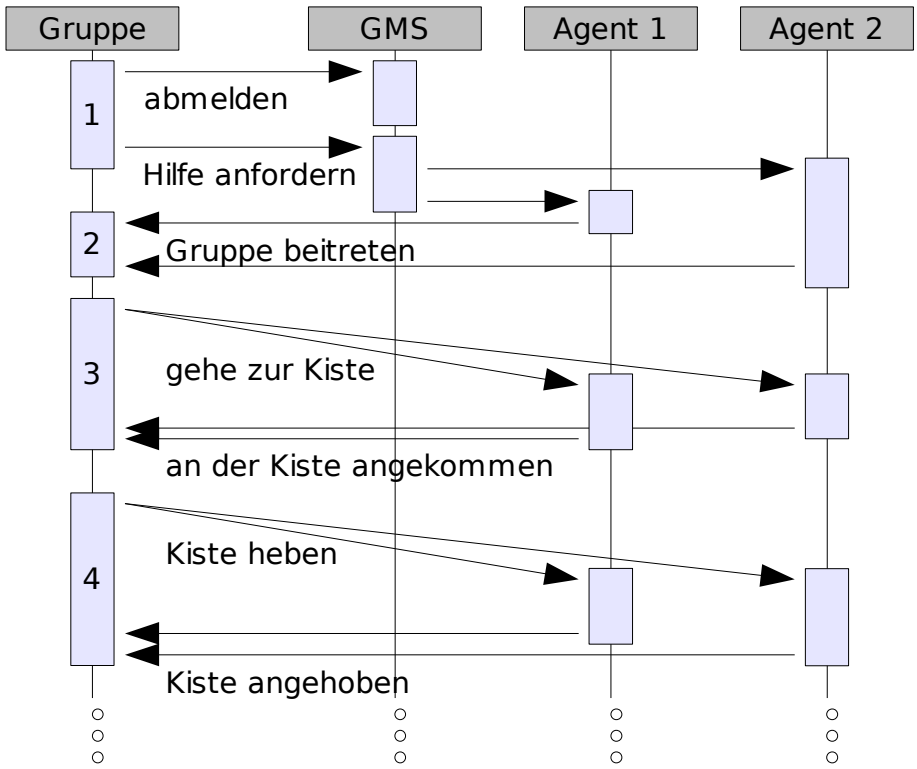

Abbildung 16: Ablauf der Gruppenhandlung in BoxWorld

Die Abbildung 16 zeigt den Anfang des Koordinationsablaufes für den Transport einer Kiste. Im ersten Schritt wird die unterstützende Rolle (*fulfil*) des Agenten, der die Kiste entdeckt hat, beim *GMS* abgemeldet, um dann die nötige Hilfe anzufordern. Der *GMS* leitet dies an geeignete Agenten weiter, welche sich dann in der Gruppe anmelden. Der nächste Schritt beinhaltet die Zuweisung von Zielen an die Agenten, hier die Aufforderung, sich zu einer bestimmten Kiste zu begeben. Danach wird gewartet, bis die Agenten diese Aufgabe erfüllt haben und sich mit Erfolg zurückmelden. Alle weitere Schritte (Kiste heben, transportieren und wieder absetzen) laufen nach dem gleichen Schema ab.

6.3.2 Zusammenfassung

Mit dem BoxWorld-Beispiel wurde die Entwicklung eines kooperativen Szenarios ausprobiert. Es zeigt anschaulich, wie die Zusammenarbeit innerhalb einer Gruppe, sowie die notwendigen Prozesse der Gruppenbildung ablaufen. Es hat sich gezeigt, dass bei der Entwicklung des Gruppenplans kein Detailwissen über einen Agenten erforderlich ist.

Aus Sicht der Gruppe spielt es beispielsweise keine Rolle, wie der Agent seinen Weg zur Kiste findet oder welche Verfahren der Agent zum Heben einer Kiste einsetzt.

Der implementierte Gruppenplan fällt sehr kompakt aus, da die für das Gruppenverhalten unwesentliche Details ausgeblendet werden. Der Systementwickler muss sich auch in diesem Beispiel nicht mit dem Problem der Kommunikation oder der Gruppenbildung auseinandersetzen, da diese Aufgaben vom BDI-Gruppenmodell automatisch übernommen werden. Er muss lediglich den statischen Aufbau einer Gruppe spezifizieren und festlegen, wann ein Agent welche Rolle übernehmen kann.

Da das Boxworld-Beispiel beim Einsatz vieler Agenten und Kisten stark an Komplexität gewinnt, stellt es eine besondere Herausforderung an die Implementation dar. Aus diesem Grund wurde das Beispiel in einem iterativen Prozess dazu verwendet, sowohl das Modell als auch die Implementation zu testen und zu überarbeiten.

7 Zusammenfassung und Ausblick

Diese Arbeit wurde mit dem Ziel begonnen, ein Gruppenmodell unter Berücksichtigung des BDI-Konzepts im agentenorientierten Umfeld zu entwerfen. Mit Blick auf einen möglichen Einsatz im Agentenframework *Jadex* wurde ein auf Zielen und Plänen basierender Ansatz ausgewählt. Die strukturelle Gruppenbeschreibung ist durch eine Erweiterung dieses Ansatzes um das Rollenkonzept möglich geworden.

7.1 Zusammenfassung

Zu Beginn der Arbeit wurde zunächst versucht, Zugang zur Agententechnologie zu schaffen. Dabei ging es um die Analyse dessen, was Agenten eigentlich sind, wie sie sich in Multiagentensysteme zusammenfügen lassen und untereinander kommunizieren. Mit Blick auf das Thema der Arbeit stellen die BDI-Agenten eine besonders interessante Teilmenge dar, weshalb die darin enthaltenen Konzepte zusätzlich vertieft wurden. Den Abschluss dieses ersten Teils bildete eine kurze Vorstellung des Agentenframework *Jadex*, da es die Basis des im weiteren Verlauf entwickelten Prototyps ist.

Im darauf folgenden Abschnitt wurden dann die Grundlagen für das entwickelte BDI-Gruppenmodell gelegt und Begriffe wie der der *Gruppe* als auch Vorteile gemeinsamen Arbeitens thematisiert. Neben Mechanismen zur Gruppenbildung wurde auch auf das Konzept der Rolle eingegangen, welches die Beschreibung von Gruppenstrukturen erlaubt. Vor dem Hintergrund, Handlungen von Agenten innerhalb einer Gruppe zu koordinieren, wurden dann sowohl Koordination als auch Kooperation untersucht. Nach allgemeiner Betrachtung dieses Themenbereichs wurden verschiedene Koordinationsformen wie *Gemeinsame Ziele* oder *Organisationen* angesprochen, um einen Überblick zu erhalten. Den Abschluss bildete die Vorstellung des *CDPS-Prozess* (*Cooperative Distributed Problem Solving*), da dieser Einfluss auf das BDI-Gruppenmodell genommen hat.

Die folgenden beiden Kapitel widmeten sich der Betrachtung von theoretischen und praktischen Formen der Koordination. Besonders die *Joint Intentions*-Theorie wurde im Detail aus dem Blickwinkel unterschiedlicher Autoren dargestellt. Diese Theorie beinhaltet den Aspekt *Gemeinsame Ziele* mit der Verpflichtung, sich gegenseitig über den Fortgang der Gruppenarbeit zu informieren. Anschließend folgte die Präsentation des *AGR*-Modell als Vertreter einer organisationszentrierten Ansicht. Neben den aus der Praxis kommenden Implementationen *Madkit* und *STEAM* wurde dann besonders auf den *SimpleTeams*-Ansatz eingegangen, da diese Systeme das BDI-Gruppenmodell wesentlich inspirierten. Die Untersuchung der angesprochenen Systeme diente dem Zweck, beim Entwurf des BDI-Gruppenmodells auf eine breite Palette von Konzepten und Ideen zurückgreifen zu können.

Nach der Erarbeitung der Grundlagen und der Betrachtung theoretischer bzw. praktischer Systeme folgte die Vorstellung des entwickelten BDI-Gruppenmodells. Ähnlich wie auf Ebene der BDI-Agenten kann hier mit Hilfe von Zielen und Plänen das Gruppenverhalten spezifiziert werden, wobei Rollen bei der Beschreibung der Gruppenstruktur unterstützen. Auch die Verwendung von Gruppenwissen ist vorgesehen. Zwar schränken die mentalen Eigenschaften die Verwendbarkeit auf entsprechende Architekturen ein, erlauben dafür aber die Verwendung der BDI-Konzepte auch auf Gruppenebene.

Die Kernaspekte des BDI-Gruppenmodells wurden schließlich in einer prototypischen Implementation umgesetzt, die im darauf folgenden Kapitel präsentiert wurde. Die in *Jadex* bereits vorhandene *Easy Deliberation Strategy* konnte dabei zur Steuerung des Gruppenverhaltens eingesetzt werden. Darüber hinaus wurde ein schlichter *Matchmaker* entworfen, der die automatisierte Bildung von Gruppen auf Basis der Gruppen-Rollen-Struktur unterstützt. Weiterhin wurde der Bezug zwischen dem BDI-Gruppenmodell und den in den vorangegangenen Kapiteln erarbeiteten Koordinations- bzw. Kooperationstechniken hergestellt.

Zum Abschluss wurden ein einfaches Beispiel und ein aufwändigeres Szenario beschrieben. Das Beispiel in Form einer Rechenaufgabe verdeutlichte, wie der *CDPS*-Prozess durch Zusammenspiel mehrerer Experten in einer Gruppe aussehen könnte. Im BoxWorld-Szenario hingegen ging es um die kooperative Lösung eines Transportproblemes,

welches sich dadurch auszeichnete, dass die Handlungen aller beteiligten Agenten koordiniert werden mussten. Beide Beispiele dokumentieren die Verwendung des BDI-Gruppenmodells.

Mit der vorliegenden Diplomarbeit wurde ausgehend von einer eingehenden Analyse des Problemfelds durch Untersuchung von Multiagentensystemen, Gruppen, Koordination und Kooperation sowie der Betrachtung existierender Systeme ein Gruppenmodell entwickelt, welches die BDI-Konzepte auch auf Ebene der Gruppe nutzbar macht. Ein Rollenmodell hilft bei der Spezifikation von Gruppenstrukturen und ermöglicht die automatisierte Gruppenbildung. Mit dem Prototyp und den beiden Anwendungsbeispiele konnte der praktische Einsatz des BDI-Gruppenmodells demonstriert werden.

7.2 Ausblick

Dieser letzte Abschnitt soll mögliche und sinnvolle Erweiterungen bzw. Vertiefungen des BDI-Gruppenmodells aufzeigen. Diese könnten die Leistungsfähigkeit des Systems optimieren oder dem Entwickler die Arbeit beim Entwurf von Agentensystemen erleichtern.

Da auf Gruppenebene mit Zielen gearbeitet wird, ist auch hier ein entsprechender Deliberationsmechnanismus erforderlich, wobei diese Arbeit die Verwendung der bereits vorhandenen *Easy Deliberation Strategy* vorschlägt. Dieses Verfahren wurde allerdings für den Gebrauch auf Agentenebene entwickelt und ignoriert daher die Intentionen der Gruppenmitglieder. Um die Interessen der einzelnen Agenten besser zu berücksichtigen, wäre die Adaption oder gar der Neuentwurf eines geeigneten Verfahrens wünschenswert.

Eine weitere Einschränkung des Prototyps bildet der Wissensaustausch zwischen den Agenten. Das Modell lässt zwar die konkrete Implementation des Verfahrens offen, jedoch könnte das implementierte Verfahren wesentlich erweitert werden. Zu empfehlen wäre vor allem die Integration einer transaktionalen Unterstützung. Interessant dürfte aber auch der Entwurf eines Systems zur Bildung von gemeinsamem Wissen sein, in dem ein neuer Wert nicht blind von anderen übernommen wird, sondern von der Gruppe erst beschlossen werden muss.

Darüber hinaus kann die Entwicklung eines Werkzeugs zur visuellen Bearbeitung der Gruppen-Rollen-Agenten-Beziehungen eine erhebliche Erleichterung darstellen. Der Systementwickler wäre so in der Lage, die

Gruppenstruktur in übersichtlicher Form zu spezifizieren, was sowohl die Arbeitszeit als auch die Fehleranfälligkeit reduziert. Gegebenenfalls wäre es sogar möglich, die Gruppenpläne aus einer graphischen Notation heraus zu generieren. Zumindest ist eine Untersuchung dieser Fragestellung im Rahmen der Entwicklung eines Modellierungswerkzeugs interessant.

Literaturverzeichnis

[Bond u. Gasser 1988]
A. H. Bond, L. Gasser: *A survey of distributed artificial intelligence*. In: A. H. Bond, L. Gasser: Readings in Distributed Artificial Intelligence, San Mateo: Morgan Kaufmann Publishers, Inc., 1988, S. 3-36

[Booch 1994]
G. Booch: *Objektorientierte Analyse und Design*. Bonn: Addison-Wesley, 1994

[Bratman 1987]
M. E. Bratman: *Intention, Plans, and practical reason*. Cambridge, Massachusetts and London, England: Harvard University Press, 1987

[Bratman 1992]
M. E. Bratman: *Shared Cooperative Activity*. In: The Philosophical Review 101 (2), 1992, S. 327-341

[Braubach 2007]
L. Braubach: *Architekturen und Methoden zur Entwicklung verteilter agentenorientierter Softwaresysteme*. Universität Hamburg, Deutschland, 2007

[Braubach et al. 2006]
L. Braubach, A. Pokahr und W. Lamersdorf: *Tools and Standards*. In: S. Kirn, O. Herzog, P. Lockemann und O. Spaniol: Multiagent Systems. Intelligent Applications and Flexible Solutions, Berlin, Heidelberg, New York: Springer, 2006, S. 503-530

[Bussmann und Müller 1992]
S. Bussmann, J. Müller: *A Negotiation Framework for Co-operating Agents*. In: S. M. Deen: Proc CKBS-SIG, Keele: Dake Centre, 1992, S. 1-17

[Cabri et al. 2003]
G. Cabri, L. Leonardi, F. Zambonelli: *BRAIN: a Framework for Flexible Role-based Interactions in Multiagent Systems*. In: R. Meersman, Z. Tari, D. C. Schmidt: On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, Berlin, Heidelberg, New York: Springer, 2003, S. 145-161

[Cabri et al. 2004]
G. Cabri, L. Ferrari, F. Zambonelli: *Role-Based Approaches for Engineering Interactions in Large-Scale MAS*. In: C. Lucena, A. Garcia, A. Romanovsky, J. Castro, P. S. C. Alencar: Software Engineering for Multi-Agent Systems II, Berlin, Heidelberg, New York: Springer, 2004, S. 243-263

[Cohen u. Levesque 1990]
P. R. Cohen, H. J. Levesque: *Intention is Choice with Commitment*. In: Artificial Intelligence 42 (2/3), 1990, S. 213-261

[Cohen u. Levesque 1991]
P. R. Cohen, H. J. Levesque: *Teamwork*. In: Noûs 25 (4), 1991, S. 487-512

[Corkill 1991]
D. Corkill: *Blackboard systems*. In: AI Expert 6 (9), 1991, S. 40-47

[Coutinho et al. 2005]
L. Coutinho, J. Sichman, O. Boissier: *Modeling organization in MAS: a comparison of models*. In: Proceedings of the 1st International Workshop on Software Engineering for Agent-Oriented Systems (SEAS ’05), 2005

[d'Inverno u. Luck 2004]
M. d'Inverno, M. Luck: *Understanding Agent Systems*. Berlin, Heidelberg, New York: Springer, 2004

[Daft 1992]
R. L. Daft: *Organization theory and design*. Saint Paul, New York, Los Angeles, San Francisco: West Publishing Company, 1992

[Durfee 2001]
E. H. Durfee: *Distributed Problem Solving and Planning*. In: M. Luck, V. Marík, O. Stepánková, R. Trappl: Multi-Agent Systems and Applications, Berlin, Heidelberg, New York: Springer, 2001, S. 118-149

[Durfee et al. 1989]
E. H. Durfee, V. R. Lesser, D. D. Corkill: *Trends in Cooperative Distributed Problem Solving*. In: IEEE Transactions on Knowledge and Data Engineering 1 (1), 1989, S. 63-83

[Ferber 2001]
J. Ferber: *Multiagentensysteme - Eine Einführung in die Verteilte Künstliche Intelligenz*. München: Addison Wesley, 1999

[Ferber et al. 2004]
J. Ferber, O. Gutknecht, F. Michel: *From Agents to Organizations: an Organizational View of MAS*. In: P. Giorgini, J. Müller, J. Odell: Agent-Oriented Software Engineering IV, Berlin, Heidelberg, New York: Springer, 2004, S. 214-230

[Ferber u. Gutknecht 1998]
J. Ferber, O. Gutknecht: *A meta-model for the analysis and design of organizations in multi-agent systems*. In: Proceedings of the 3rd International Conference on Multi Agent Systems, 1998, S. 128-135

[Finin 1993]
T. Finin: *Draft Specification of the KQML Agent-Communication Language*, 1993. Im Internet unter (Abrufdatum 7.5.2007) http://www.cs.umbc.edu/kqml/

[Finin u. Labrou 1997]
T. Finin, Y. Labrou: *A Proposal for a new KQML Specification*, 1997. Im Internet unter (Abrufdatum 4.11.2007) http://www.flacp.fujitsulabs.com/~yannis/publications/tr9703.pdf

[FIPA 2002]
Foundation for Intelligent Physical Agents (FIPA): *FIPA Standards*, 2002. Im Internet unter (Abrufdatum 4.11.2007) http://www.fipa.org/repository/standardspecs.html

[Franklin u. Graesser 1997]
S. Franklin, A. Graesser: *Is it an Agent, or Just a Program? A Taxonomy for Autonomous Agents*. In: J. P. Müller, M. J. Wooldridge, N. R. Jennings: Intelligent Agents III, Berlin, Heidelberg, New York: Springer, 1997, S. 21-36

[Genesereth et al. 1988]
M. R. Genesereth, M. L. Ginsberg, J. S. Rosenschein: *Cooperation without communication*. In: A. H. Bond, L. Gasser: Distributed Artificial Intelligence, San Mateo: Morgan Kaufmann Publishers Inc., 1988, S. 220-226

[Georgeff u. Lansky 1987]
M. P. Georgeff, A. L. Lansky: *Reactive reasoning and planning*. In: K. Forbus, H. E. Shrobe: Proceedings AAAI-87 sixth National Conference on Artificial Intelligence, July 13-17, Cambridge, Mass: MIT Press, 1987, S. 677-682

[Gutknecht u. Ferber 2001]
O. Gutknecht, J. Ferber: *The MADKIT Agent Platform Architecture*. In: T. Wagner, O. F. Rana: International Workshop on Infrastructure for Multi-Agent Systems: Infrastructure for Agents, Multi-Agent Systems and Scalable Multi-Agent Systems, Berlin, Heidelberg, New York: Springer, 2001, S. 48-55

[Haddadi 1995]
A. Haddadi: *Communication and Cooperation in Agent Systems*. Berlin, Heidelberg, New York: Springer, 1995

[Hodgson et al. 1999]
A. Hodgson, R. Rönnquist, P. Busetta: *Specification of Coordinated Agent Behavior (The SimpleTeam Approach)*, 1999. Im Internet unter (Abrufdatum 4.11.2007) http://www.agentoriented.com/shared/resources/tech-reports/tr9905.pdf

[Hübner et al. 2002]
J. F. Hübner, J. S. Sichman, O. Boissier: *A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems*. In: G. Bittencourt, G. L. Ramalho: Advances in Artificial Intelligence, Berlin, Heidelberg, New York: Springer, 2002, S. 118-128

[Huhns u. Stephens 1999]
M. N. Huhns, L. M. Stephens: *Multiagent Systems and Societies of Agents*. In: G. Weiss: Multiagent Systems: A Modern Approach to Distributed Artifcial Intelligence, Cambridge, MA, USA: The MIT Press, 1999, S. 80-120

[Jadex 2007a]
A. Pokahr, L. Braubach: *Jadex Userguide*, 2007. Im Internet unter (Abrufdatum 4.11.2007) http://vsis-www.informatik.uni-hamburg.de/projects/jadex/jadex-0.96x/userguide/userguide.pdf

[Jadex 2007b]
A. Pokahr, L. Braubach: *Jadex Tutorial*, 2007. Im Internet unter (Abrufdatum 4.11.2007) http://vsis-www.informatik.uni-hamburg.de/projects/jadex/jadex-0.96x/tutorial/tutorial.pdf

[Jadex 2007c]
A. Pokahr, L. Braubach: *Jadex Toolguide*, 2007. Im Internet unter (Abrufdatum 4.11.2007) http://vsis-www.informatik.uni-hamburg.de/projects/jadex/jadex-0.96x/toolguide/toolguide.pdf

[Jennings 1993a]
N. R. Jennings: *Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems*. In: The Knowledge Engineering Review 8 (3), 1993, S. 223-250

[Jennings 1993b]
N. R. Jennings: *Specification and Implementation of a Belief-Desire-Joint-Intention Architecture For Collaborative Problem Solving*. In: Intl. Journal of Intelligent and Cooperative Information Systems 2 (3), 1993, S. 289-318

[Jennings 1999]
N. R. Jennings: *Agent-Oriented Software Engineering*. In: F. J. Garijo, M. Boman: Multi-Agent System Engineering, Berlin, Heidelberg, New York: Springer, 1999, S. 1-7

[Jennings 2000]
N. R. Jennings: *On agent-based software engineering*. In: Artificial Intelligence 117 (2), 2000, S. 277-296

[Kinny et al. 1994]
D. Kinny, M. Ljungberg, A. S. Rao, E. Sonenberg, G. Tidhar, E. Werner: *Planned Team Activity*. In: C. Castelfranchi, E. Werner: Artificial Social Systems - Selected Papers from the Fourth European Workshop on Modelling Autonomous Agents in a Multi-Agent World MAAMAW-92, Berlin, Heidelberg, New York: Springer, 1994, S. 227-258

[Levesque et al. 1990]
H. Levesque, P. Cohen, J. Nunes: *On Acting Together*. In: American Association for Artificial Intelligence: Proceedings of the Eighth National Conference on Artificial Intelligence AAAI-90, Boston, Massachusetts: AAAI Press, 1990, S. 94-99

[Lindgren 1973]
H. C. Lindgren: *Einführung in die Sozialpsychologie*. Weinheim, Basel: Beltz Verlag, 1973

[Malone und Crowston 1994]
T. W. Malone, K. Crowston: *The Interdisciplinary Study of Coordination*. In: ACM Computing Surveys 26 (1), 1994, S. 87-119

[Mao und Yu 2004]
X. Mao, E. Yu: *Organizational and Social Concepts in Agent Oriented Software Engineering*. In: J. J. Odell, P. Giorgini, J. Müller: Agent-Oriented Software Engineering, Fifth International Workshop (AOSE), Berlin, Heidelberg, New York: Springer, 2004, S. 1-15

[Müller 1996]
J. P. Müller: *The Design of Intelligent Agents: A Layered Approach*. Berlin, Heidelberg, New York: Springer, 1996

[Nwana et al. 1996]
H. S. Nwana, L. C. Lee, N. R. Jennings: *Co-ordination in Software Agent Systems*. In: The British Telecom Technical Journal 14 (4), 1996, S. 79-88

[Odell et al. 2003]
J. J. Odell, H. Van Dyke Parunak, Mitchell Fleischer: *The Role of Roles in Designing Effective Agent Organizations*. In: A. Garcia, C. Lucena, F. Zambonelli, A. Omicini, J. Castro: Software Engineering for Large-Scale Multi-Agent Systems, Berlin, Heidelberg, New York: Springer, 2003, S. 27-38

[Odell et al. 2005]
J. Odell, M. Nodine, R. Levy: *A Metamodel for Agents, Roles, and Groups*. In: J. Odell, P. Giorgini, J. Müller: Agent-oriented software engineering V, Berlin, Heidelberg, New York: Springer, 2005, S. 78-92

[Ogston und Vassiliadis 2001]
E. Ogston, S. Vassiliadis: *Local Distributed Agent Matchmaking*. In: C. Batini, F. Giunchiglia, P. Giorgini, M. Mecella: Cooperative Information Systems, Berlin, Heidelberg, New York: Springer, 2001, S. 67-79

[Ossowski 1999]
S. Ossowski: *Co-ordination in Artificial Agent Societies*. Berlin, Heidelberg, New York: Springer, 1999

[Pokahr 2007]
A. Pokahr: *Programmiersprachen und Werkzeuge zur Entwicklung verteilter agentenorientierter Softwaresysteme*. Universität Hamburg, Deutschland, 2007

[Pokahr et al. 2005a]
A. Pokahr, L, Braubach, W. Lamersdorf: *A Goal Deliberation Strategy for BDI Agent Systems*. In: T. Eymann, F. Klügl, W. Lamersdorf, M. Klusch, M. Huhns: Third German conference on Multi-Agent System TEchnologieS (MATES-2005), Berlin, Heidelberg, New York: Springer, 2005, S. 82-93

[Pokahr et al. 2005b]
A. Pokahr, L, Braubach, W. Lamersdorf: *A Flexible BDI Architecture Supporting Extensibility*. In: A. Skowron, J. P. Barthes, L. Jain, R. Sun, P. Morizet-Mahoudeaux, J. Liu, N. Zhong: The 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-2005), Washington, DC, USA: IEEE Computer Society, 2005, S. 379-385

[Pokahr et al. 2005c]
A. Pokahr, L, Braubach, W. Lamersdorf: *Jadex: A BDI-Agent System Combining Middleware and Reasoning*. In: R. Unland, M. Klusch, M. Calisti: Software Agent-Based Applications, Platforms and Development Kits, Heidelberg: Birkhäuser Basel, 2005, S. 143-168

[Pokahr et al. 2005d]
A. Pokahr, L, Braubach, W. Lamersdorf: *Jadex: A BDI Reasoning Engine*. In: R. H. Bordini, M. Dastani, J. Dix, A. E. F. Seghrouchini: Multi-Agent Programming, Berlin, Heidelberg, New York: Springer, 2005, S. 149-174

[Pynadath et al. 1999]
D.V. Pynadath, M. Tambe, N. Chauvat, L. Cavedon: *Toward Team-Oriented Programming*. In: N. R. Jennings, Y. Lespérance: Intelligent Agents VI, Berlin, Heidelberg, New York: Springer, 1999, S. 233-247

[Pynadath et al. 2000]
D.V. Pynadath, M. Tambe, N. Chauvat, L. Cavedon: *Toward team-oriented programming*. In: N. R. Jennings, Y. Lespérance: Intelligent Agents VI: Agent Theories, Architectures and Languages (ATAL'99), Berlin, Heidelberg, New York: Springer, 2000, S. 233-247

[Pynadath u. Tambe 2003]
D. V. Pynadath, M. Tambe: *An automated teamwork infrastructure for heterogeneous software agents and humans*. In: Autonomous Agents and Multi-Agent Systems 7 (1-2), 2003, S. 71-100

[Rao et al. 1992]
A. S. Rao, M. P. Georgeff, E. A. Sonnenberg: *Social Plans: A Priliminary Report*. In: E. Werner, Y. Demazeau: Decentralized A.I. 3, Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-91), Amsterdam, Netherland: Elsevier Science B.V., 1992, S. 57-76

[Rao u. Georgeff 1995]
A. S. Rao, M. P. Georgeff: *BDI Agents: from theory to practice*. In: V. Lesser, L. Gasser: Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95), San Francisco, CA, USA: The MIT Press: Cambridge, MA, USA, 1995, S. 312-319

[Ritter 2003]
N. Ritter: *Datenbanken und Informationssysteme*, 2003. Im Internet unter (Abrufdatum 4.11.2007) http://vsis-www.informatik.uni-hamburg.de/teaching/ss-03/dis/

[Sader 1998]
Manfred Sader: *Psychologie der Gruppe*. Weinheim, München: Juventa Verlag, 1998

[Scerri et al. 2004]
P. Scerri, D. Pynadath, N. Schurr, A. Farinelli, S. Gandhe, M. Tambe: *Team Oriented Programming and Proxy Agents: The Next Generation*. In: M. Dastani, J. Dix, A. E. Fallah-Seghrouchni: Programming Multi-Agent Systems: First International Workshop, Berlin, Heidelberg, New York: Springer, 2004, S. 131-148

[Schumacher 2001]
M. Schumacher: *Objective Coordination in Multi-Agent System Engineering*. Berlin, Heidelberg, New York: Springer, 2001

[Singh 1992]
M. P. Singh: *A Critical Examination of the Cohen-Levesque Theory Intentions*. In: B. Neumann: 10th European Conference on Artifcial Intelligence (Proceedings), Chinchester, New York, Brisbane, Toronto, Singapore: John Wiley & Sons, 1992, S. 364-368

[Smith u. Davis 1988]
R. G. Smith, R. Davis: *Frameworks for Cooperation in Distributed Problem Solving*. In: A. H. Bond, L. Gasser: Distributed Artificial Intelligence, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988, S. 61-70

[Steegmans et al. 2004]
E. Steegmans, K. Schelfthout, T. Holvoet, Y. Berbers, P. Valckenaers, B. S. Germain: *A Basic Taxonomy for Role Composition*. In: C. Lucena, A. Garcia, A. Romanovsky, J. Castro, P.S.C. Alencar: Software Engineering for Multi-Agent Systems II, Berlin, Heidelberg, New York: Springer, 2004, S. 92-110

[Tambe 1997a] M. Tambe: *Towards flexible teamwork*. In: Journal of Artificial Intelligence Research 7, 1997, S. 83-124

[Tambe 1997b]
M. Tambe: *Agent Architectures for Flexible, Practical Teamwork*. In: AAAI-97 / IAAI-97: Proceedings, Menlo Park, California: AAAI Press, 1997, S. 22-28

[Vanzin und Barber 2006]
M. M. Vanzin, K. S. Barber: *Decentralized Partner Finding in Multi-Agent Systems*. In: P. Scerri, R. Vincent, R. Mailer: Coordination of Large-Scale Multiagent Systems, Berlin, Heidelberg, New York: Springer, 2006, S. 75-98

[Weiß u. Jakob 2005]
G. Weiß, R. Jakob: *Agentenorientierte Softwareentwicklung*. Berlin Heidelberg New York: Springer, 2005

[Wiendieck 1994]
G. Wiendieck: *Arbeits- und Organisationspsychologie*. Berlin: Quintessenz, 1994

[Wooldridge 2002]
M. J. Wooldridge: *An Introduction to MultiAgent Systems*. Chichester, England: John Wiley & Sons Ltd, 2002

[Wooldridge u. Jennings 1995]
M. Wooldridge, N. R. Jennings: *Intelligent Agents: theory and practice*. In: Knowledge Engineering Review 10 (2), 1995, S. 115-152

[Wooldridge u. Jennings 1997]
M. Wooldridge, N. R. Jennings: *Formalizing the Cooperative Problem Solving Process*. In: M. N. Huhns, M. P. Singh: Readings in Agents, San Francisco: Morgan Kaufmann Publishers Inc., 1997, S. 430-440

[Yen et al. 2001]
J. Yen, J. Yin, T. Ioerger, M. Miller, R. Volz: *CAST: Collaborative Agents for Simulating Teamwork*. In: B. Nebel: Seventeenth International Joint Conference on Artifcial Intelligence (IJCAI-01), San Francisco: Morgan Kaufmann Publishers Inc., 2001, S. 1135-1142

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Ich bin mit einer Einstellung in den Bestand der Bibliothek des Departments Informatik einverstanden.

Hamburg, den 17.12.2007

_____

Christian Poulter