14 Multiagentensysteme

Franziska Klügl

Multiagentensysteme bilden das grundlegende Konzept, intelligente Entitäten (Software, Hardware, Menschen) zu einem kohärenten Gesamtsystem zu vernetzen. Zur individuellen Intelligenz kommt die soziale. Die Entitäten interagieren, kommunizieren, teilen Information, kooperieren, koordinieren ihre Aktionen. Aus dem Blickwinkel der Künstlichen Intelligenz liegen die Ursprünge der Multiagentensysteme in der Verteilten Künstlichen Intelligenz, die sich ab etwa 1980 mit der Idee konstitutionalisiert hat, Koordination und Interaktion von intelligenten Problemlösern auf einer abstrakteren Ebene jenseits von technischen, maschinennahen Problemen der Parallelität zu behandeln [90]. Während in den ersten 20 Jahren die Unterscheidung in „Verteilte Problemlöser" und „Multiagentensystem" noch wichtig war (siehe [34] für eine Diskussion), bezeichnet man mittlerweile – nach dem Hype der 90iger Jahre – jede Form von System mit mehreren, interagierenden „Agenten" als Multiagentensystem. Heute erhebt auch nicht mehr nur die Künstliche Intelligenz den Anspruch, Beiträge zur Multiagentensystem-Forschung zu liefern, sondern auch Verteilte Systeme als technische Grundlage, Wirtschaftsinformatik mit ihrer Beziehung zu den Wirtschafts- und Sozialwissenschaften als Ideengeber für Metaphern, Interaktionsprotokolle oder Analyse oder das Software Engineering.

So identifizierten [92] eine Revolution im Software Engineering indem sie vier Charakteristika von modernen Systemen beschrieben: Situiertheit, Offenheit, lokale Kontrolle und lokale Interaktion, also genau die Aspekte, die Multiagentensysteme charakterisieren.

Beispiele für Multiagentensysteme findet man in allen Anwendungsgebieten, bei denen diese Charakteristika wichtig sind:

- Systeme, bei denen Software- oder Hardware-Entitäten in einer Umwelt existieren und mit dieser und anderen Entitäten interagieren;
- bei denen die einzelnen Entitäten ihre eigenen Ziele verfolgen und neue Entitäten von außen dazukommen können;
- eine Verteiltheit von Kontrollen und Daten inhärent gegeben ist oder es sinnvoll ist eine solche Verteilung einzuführen, weil die Komplexität nicht mehr durch eine zentrale Steuerung behandelt werden kann;
- bei denen parallele, asynchrone Prozesse aktiv sind.

Mit solchen Charakteristika werden eine Vielzahl moderner Anwendungsgebiete beschrieben, von Ambient Intelligence (Umgebungsintelligenz) zu Verkehrsmanagement, von modernen Fabriken zu Werkzeugen für Soziale Netzwerke.

Es gibt mittlerweile eine Reihe von sehr guten Kompendien und Monographien, die sich für eine weiterführende Einführung eignen. [89][1] versammelte einführende Kapitel zu allen wich-

[1] Eine zweite Edition erscheint 2013.

tigen Teilgebieten. [90] ist ein hervorragendes, breites Lehrbuch. [80] führt sehr fundiert in den aktuellen Stand der Verteilten Entscheidungsfindung ein. Eine konzentrierte, formalere Einführung bietet [87]. Darüber hinaus gibt es sehr viele Bücher und Artikel, die sich mit speziellen Teilgebieten der Multiagentensystem-Forschung befassen.

Dieses Kapitel kann nur einen oberflächlichen Einblick zu Multiagentensystemen liefern. Dabei wollen wir wie folgt vorgehen: Zunächst soll der Begriff des „Agenten“ erklärt werden, welche Eigenschaften damit verbunden sind und mit welchen Architekturen für einen Agenten diese Eigenschaften erreicht werden können. Danach werden Agenten zu Multiagentensystemen zusammengefügt. Im Abschnitt 14.2 wird näher betrachtet, wie Agenten interagieren können und wie diese Interaktion strukturiert werden kann. Dabei gehen wir zunächst von Agenten aus, die als kooperierende Problemlöser konzipiert sind. Im darauf folgenden Abschnitt werden verschiedene Interaktionsprotokolle dargestellt, mit denen egoistische Agenten gemeinsam Entscheidungen treffen können, um so auch bei nicht kooperativen Agenten zu einer sinnvollen Problemlösung zu kommen. In Abschnitt 14.4 werden zunächst Methoden vorgestellt, mit denen Multiagentensysteme entwickelt werden können, danach werfen wir einen kurzen Blick auf Werkzeuge und Anwendungen. Das Kapitel endet mit einer kurzen Diskussion von spannenden und aktuellen Themen, die nicht näher behandelt werden können.

14.1 Vom Agenten zum Multiagentensystem

14.1.1 Begriff und Charakteristika

Ausgangspunkt für jede Behandlung von Multiagentensystemen ist der Begriff „Agent“. In den Anfangsjahren herrschte noch terminologische Verwirrung. Jede Publikation begann mit einer entsprechenden Definition des dabei verwendeten Agentenbegriffs. Der Höhepunkt war [41], die verschiedene Definitionen miteinander verglichen haben. Auch in der Künstlichen Intelligenz selbst wurde mit [74] eine Agentensicht etabliert. Sie geben in ihrem bekannten Lehrbuch zur Künstlichen Intelligenz – das vom Agentenkonzept ausgehend organisiert ist – eine Definition des Begriffs „Agent“ als „An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.“ (p. 34). Diese sehr einfache Definition wird erweitert zu rationaler Agent etc. Eine andere wichtige alternative Sichtweise des Agentenbegriffs stammt aus der Principal-Agent Theorie der Wirtschafts- und Sozialwissenschaften [35]. Dabei wird „Agent“ als jemand gesehen, der im Auftrag seines Prinzipals handelt. Ein Mensch delegiert Aufgaben an „seinen“ Agenten. Diese Interpretation spielt heute noch bei Interface Agenten oder ähnlichem eine Rolle. Mittlerweile hat sich die Begriffsbildung praktisch konsolidiert: Die Definition, die beide Sichtweisen in sich vereinigt und sich durchgesetzt hat, ist folgende:

> „An agent is a computer system that is situated in some environment and that is capable of independent (autonomous) action in this environment on behalf of its user or owner (figuring out what needs to be done to satisfy design objectives, rather than constantly being told)“ [90, p. 15]

Die wichtigsten Elemente dieser Definition sind

- **Situiertheit** in einer Umwelt beinhaltet, dass der Agent in einer Umwelt existiert und von dieser klar abgrenzbar ist. Das „In-einer-Umwelt-Sein“ hat als Konsequenz, dass ein Agent an einer bestimmten Position lokalisiert ist, von der aus er (lokal zugängliche) Informationen erhält und auch nur innerhalb eines bestimmten Aktionsradius Elemente seiner Umwelt manipulieren kann.
- **Unabhängigkeit/Autonomie** definiert M. Wooldridge klar als eigenständige Zielerfüllung. Das ist mehr als die (energetische) Unabhängigkeit eines einfachen Roboters, aber deutlich einfacher als die vollständige Autonomie, wie sie bei [74] beschrieben wird. Dort muss ein autonomer Agent sein Verhalten vollständig selbst bestimmen können, z.B. durch Lernen. [52] geben in ihrem einführenden Kapitel eine kurze Diskussion von verschiedenen Formen der Autonomie: von absoluter Autonomie, die man auch als Autismus bezeichnen könnte, zu Ausführungsautonomie, die nur fordert, dass der Agent seine Aktionen ohne Hilfe durchführen kann.

Diese grundlegende Definition wird erweitert zum „intelligenten Agenten“, der in der Lage ist flexibel zu handeln. Flexibilität bedeutet dabei, dass der Agent sein Verhalten an eine dynamische Umwelt anpassen und dabei immer seine Ziele im Auge behalten kann. Die reine Anpassungsfähigkeit wird auch als Reaktivität bezeichnet, das eigenständige Zielerfüllen als Proaktivität. Interaktionsfähigkeit mit anderen Agenten ist eine weitere zentrale Komponente eines intelligenten Agenten. Das bedeutet nicht unbedingt eine Beschränkung auf nachrichtenbasierten Austausch, Interaktion kann auch über die Umwelt geschehen: ein Agent verändert die Umwelt absichtlich oder unabsichtlich, andere Agenten nehmen diese Änderung wahr und adaptieren daraufhin ihr Verhalten.

Autonomie, Reaktivität, Zielgerichtetheit und Interaktionsfähigkeiten sind die zentralen Eigenschaften, die einen intelligenten Agenten von einfacheren Einheiten unterscheiden. Ein grundsätzliches Problem ist dabei das Zusammenspiel zwischen Reaktivität und Zielgerichtetheit: Ein Agent, der sein Verhalten permanent und sein aktuell verfolgtes Ziel opportunistisch ändert, wird es womöglich nicht schaffen, dieses Ziel zu erreichen. Ein Agent, der stur auf sein Ziel hinarbeitet und seine Pläne und Aktivitäten nicht den dynamischen Umweltbedingungen anpasst, wird ebenso scheitern. Im nächsten Abschnitt sollen die wichtigsten Agentenarchitekturen vorgestellt werden, die insbesondere eine Lösung für das Dilemma zwischen Reaktivität und Zielgerichtetheit anbieten.

14.1.2 Wichtige Agentenarchitekturen

Das Grundprinzip aller Agentenarchitekturen ist gleich und wird in Abbildung 14.1 dargestellt. Der Agent bekommt Eingaben aus seiner Umwelt – über die Wahrnehmung oder Nachrichten – und benutzt diese Eingaben und den Inhalt seines internen Speichers, um die nächste Aktion zu bestimmen. Diese Aktion bestimmt dann, wie der Agent auf seine Umwelt einwirken will. Es liegt dann an der Umwelt, welche Effekte diese Aktion tatsächlich hat. Dies wird in [38] schön formalisiert.

Die Frage ist, wie der Agent im konkreten Fall seine nächste Aktion bestimmt. Eine Agentenarchitektur ist ein Muster, das beschreibt, wie interne Komponenten eines Agenten zusammenspielen, um aus Sensoreingaben im Endeffekt Verhalten zu generieren (für eine ausführlichere

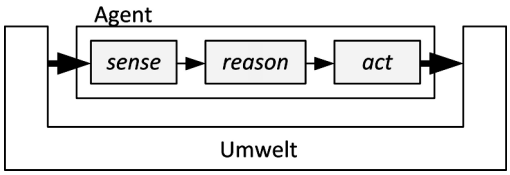

Abbildung 14.1: *Der Wahrnehmen-Reasoning-Aktion-Zyklus eines Agenten.*

Definition siehe [59]). Im Laufe der Jahre wurden sehr viele verschiedene Architekturen für Agenten vorgeschlagen. Die allgemein bekanntesten sind die Agentenarchitekturen – oder besser Architekturkategorien –, die in [74] vorgeschlagen wurden:

- Einfache Reflexagenten verknüpfen die Wahrnehmung des aktuellen Umweltzustandes direkt mit Aktionen, die in diesem Zustand ausgeführt werden sollen.
- Ein modellbasierter Reflexagent besitzt zudem einen Speicher, in dem Informationen über den Zustand der Umwelt gespeichert werden können. So muss der Agent nicht mehr in jedem Aktualisierungszyklus die vollständige Information in der Umwelt finden, um die geeignete Aktion auszuwählen.
- Ein zielbasierter Agent erweitert die Architektur um explizites Wissen über Ziele, für deren Erreichen Pläne erzeugt werden können. Situationen und Aktionen sind nicht mehr direkt verknüpft, sondern flexibel in einem oder mehreren Plänen kombiniert. Dies erlaubt komplexere Probleme zu lösen.
- Eine vierte Ebene an Komplexität stellen die nützlichkeitsbasierten Agenten dar. Ein Agent besitzt mehrere Ziel gleichzeitig, die er bewertet und verfolgt. Es gibt interessante Kombinationen von Architekturen: Ein Agent kann die Nützlichkeit eines jeden Zustands lernen. Auf der Basis dieses Wissens kann er im deterministischen Fall direkt bestimmen, welches die Aktion ist, die aus dem aktuellen Zustand zu dem nächsten mit der höchsten Nützlichkeit führt und dieses Wissen in Regeln organisieren. So wird aus einem nützlichkeitsbasierten Agenten ein einfacher Reflexagent mit oder ohne internen Zustand abhängig davon, wie viel Zugriff auf Zustandsinformation die Sensoren des Agenten bieten.

Diese allgemeinen Architekturkategorien kann man auch bei Multiagentensystemen finden. Reflexagenten wurden vor allem mit dem Ziel, eine schnelle Reaktion zu produzieren, verwendet; ein interner Zustand ist jedoch in einer nur lokal zugänglichen Umwelt unerlässlich, z.B. um Sensoreingaben zu puffern und zum Gesamtbild des aktuellen Umweltzustands zusammen zu fügen. Seit den 90igern wurden einige Architekturen vorgeschlagen. Regeln werden zum Beispiel aus Plänen kompiliert, indem alle Eventualitäten in einer baumartigen Struktur integriert wurden, z.B. in [76]. Andere Architekturen organisieren Aktivitäten, Aufgaben oder nur Aktionen in Graphstrukturen (z.B. [17,31,59,62]). Diese sollen dann in der aktuellen Situation in Konkurrenz miteinander die nächst passende Aktion auszuwählen. Mittlerweile haben sich vor allem zwei Architekturformen herauskristallisiert: BDI-Architekturen und geschichtete Architekturen.

BDI (Belief Desire Intention) Architektur

BDI Architekturen gehören zu den wichtigsten Agentenarchitekturen. Die Idee stammt aus der Alltagspsychologie. [90, Seite 65ff] nennt es auch „praktisches Reasoning“. Die Ideen stammen

ursprünglich von M. E. Bratman, der sich als Philosoph mit dem menschlichen Denken beschäftigt hat: Um über Aktionen zu entscheiden, muss man sich zunächst bewusst machen, was man erreichen will und erst danach, wie man das Gewünschte erreichen kann. Das bedeutet, der Gesamtprozess ist bei Agenten in zwei Teile unterteilt: Zielfindung und Plangenerierung: Ein Agent generiert eine Menge von Zielen („Desire“ oder „Option“), die er verfolgen könnte, dann filtert er diese Ziele und legt sich auf das Verfolgen einiger weniger Ziele fest. Diese werden zu Intentionen. Ein Studierender beispielsweise kann für die Zeit nach dem Studium verschiedene Ziele haben: a) viel Geld verdienen, b) promovieren, oder c) auf Bali eine Bar aufmachen. Die Gesamtmenge an Zielen muss nicht konsistent sein. Jedoch, wenn sich der Agent auf eine Teilmenge festlegt, dann muss diese Teilmenge erreichbar sein, bzw. der Agent muss glauben, dass alle Intentionen in dieser Menge für ihn erreichbar sind. Das dritte Element der BDI-Architektur ist das interne Weltmodell des Agenten, die „Beliefs“[2]. Damit ist das beschrieben, von dem der Agent denkt, dass es wahr ist: seine Annahmen, die Information in seinem internen Weltmodell, das er zu vorherigen Zeitpunkten aus seinen Wahrnehmungen oder Information von anderen Agenten aufgebaut hat. Die Welt kann sich aber geändert haben, Information kann fehlerhaft sein. Demzufolge müssen seine Beliefs nicht mit der tatsächlichen Situation übereinstimmen.

Auf der Basis dieser Überlegungen, wurde nicht nur eine BDI-Logik entwickelt [71], sondern die wohl einflussreichste Agentenarchitektur: Das „Procedural Reasoning System“ (PRS, [54]). Abbildung 14.2 zeigt den schematischen Aufbau von PRS.

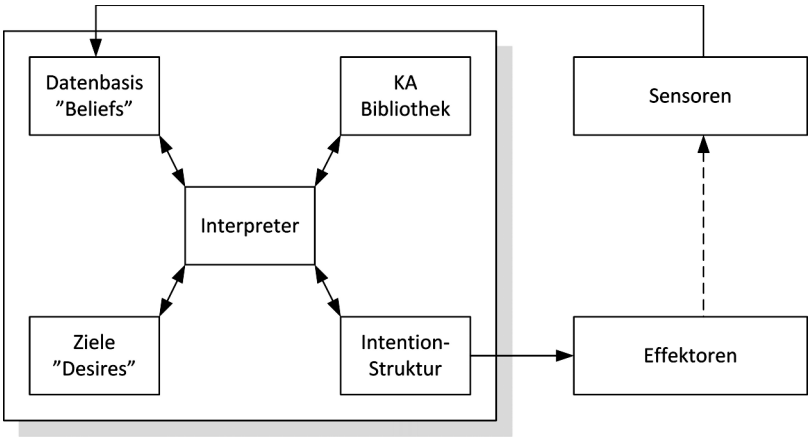

Abbildung 14.2: *Schematischer Aufbau der PRS Architektur (nach [54]).*

Intentionen werden dabei in Form sogenannter „Knowledge Areas (KA)“ repräsentiert. Das sind Planschablonen, mit denen in einer Graphstruktur Verhalten beschrieben wird. KAs können hierarchisch aufgebaut sein. Ausgewählte Intentionen werden auf einem Stack organisiert und abgearbeitet. Änderungen dieses Stacks werden durch neue Information oder neue Ziele getriggert. Der Stack kann auch über Meta-KAs manipuliert werden. PRS hat diverse Nachfolger-Architekturen, die zur Zeit bekannteste ist AgentSpeak(L) [70], auch wegen der Entwicklungsplattform JASON [15].

[2] „Beliefs“ sinnvoll in das Deutsche zu übersetzen, ist nicht ganz einfach. Die direkte Übersetzung wäre „Glaube“ oder „Überzeugungen“, was eine nicht ganz angemessene religiöse Konnotation hat.

Schichtenarchitekturen

Während bei den BDI-Architekturen das Balance-Halten zwischen Reaktivität und Zielgerichtetheit eine Frage des Reasoning-Mechanismus' und der Repräsentation des Verhaltens in den Planschablonen ist, wird in geschichteten Architekturen reaktives und ziel-orientiertes Verhalten explizit getrennt. Abbildung 14.3 zeigt die beiden grundsätzlichen Alternativen: horizontale und vertikale Schichtung. Als Vorbilder für diese etwas vereinfachten Darstellungen dienen die TouringMachines [39] für die horizontale Schichtung und InterRAP [61] für die vertikale Schichtung. Es gibt eine weitere Variante einer vertikalen Schichtung, die sogenannte Ein-Pass vertikale Schichtung. Im Gegensatz zu der hier dargestellten Zwei-Pass Schichtung, werden die Daten immer zur nächsten Schicht weitergereicht. In der Zwei-Pass Architektur kann die reaktive Schicht alleine bereits das Verhalten bestimmen.

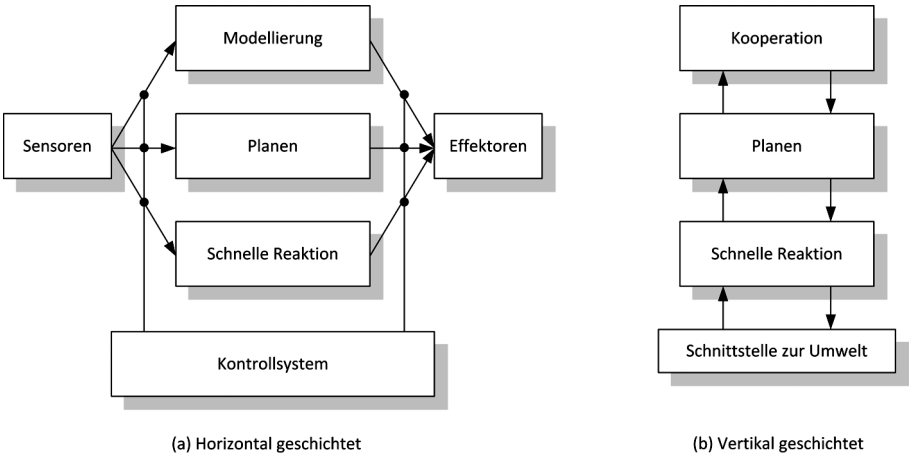

Abbildung 14.3: *Schematischer Aufbau einer horizontal (a) und vertikal geschichteter (b) Agentenarchitektur.*

Die verschiedenen Schichten haben klar getrennte Verantwortlichkeiten. Die unterste Schicht ist für die schnelle Reaktion in Notfällen verantwortlich, die Planer-Schicht für zielgerichtetes Verfahren. Eine dritte Schicht beschäftigt sich mit anderen Agenten. Der Unterschied liegt in der Zusammenschaltung der verschiedenen Schichten:

Bei der horizontal geschichteten Architektur werden alle Schichten gleichzeitig mit Sensorinformation versorgt und produzieren Vorschläge für die nächste Aktion. Welche von den Aktionen dann letztendlich zur Ausführung kommt, bestimmt eine Zensorkomponente. Diese mächtige, zentrale Komponente kann auch Sensorinformation für einzelne Schichten manipulieren, um z.B. zu verhindern, dass der Agent bei der Annäherung an ein interessantes, näher zu untersuchendes Objekt, kein Verhalten zur Kollisionsvermeidung aktiviert. Der kritische Punkt dabei ist klar diese Zensorkomponente, die sehr viel über den Aufbau und das Wissen der einzelnen Schichten kennen muss, um sinnvoll einzugreifen. Das Beispiel für eine horizontal geschichtete Architektur findet sich bei [39]. Auch die bekannte Subsumptionsarchitektur [17] kann als eine geschichtete Architektur gelten, allerdings ist die Funktionalität der Zensorkomponente implizit in die Verknüpfungen der Schichten eingebaut.

Prominenter sind vertikal geschichtete Architekturen. Ihr Prototyp war InterRAP [61]. Die unterste Schicht sorgt für eine schnelle Reaktion in Notfallsituationen. Erzeugt diese Schicht keine Aktion, delegiert sie die Aktionsgenerierung an die nächst-höhere Schicht, die aus dem aktuell verfolgten Plan die nächste Aktion liefert, bzw. für das aktuelle Ziel einen Plan generiert. Auf der obersten Schicht werden für die Arbeit in einem Team gemeinsame Ziele und die aktuelle Kooperationssituation verwaltet. Daraus werden die lokalen Ziele generiert, falls die Planungsebene kein aktuelles Ziel verfügbar hat. In InterRAP besitzt jede Schicht ihre eigene Wissensrepräsentation, z.B. Situation-Aktion-Regeln für schnelle Reaktionen, oder Skelettpläne für die Planungskomponente. Geschichtete Architekturen sind mittlerweile allgegenwärtig, z.B. auch in Simulationsanwendungen, in denen so taktisches und strategisches Verhalten getrennt wird. Ein Beispiel in der Verkehrssimulation findet man in [3].

14.1.3 Multiagentensystem

Es gibt im Bereich der agenten-basierten Systeme viele, die aus einem einzelnen Agenten bestehen. Dieser Agent ist oft ein persönlicher Assistent, mit dem ein Nutzer interagiert und an ihn Aufgaben delegiert, z.B. als Mail-Dämon. Der Übergang zu Multiagentensystemen ist dabei fließend: Ein Agent, der für seinen Benutzer bei einem Online-Auktionshaus in Konkurrenz zu vielen anderen Agenten Dinge ersteigert, macht dies als persönlicher Assistent in einem Multiagentensystem. Auch ein Mail-Agent kann sich mit anderen Mail-Agenten über aktuelle Spam-Signaturen austauschen. Ein Multiagentensystem ist ein System aus interagierenden Agenten. Diese Definition ist allerdings etwas zu kurz gedacht, da sie verschiedene Aspekte impliziert, aber nicht klar darstellt.

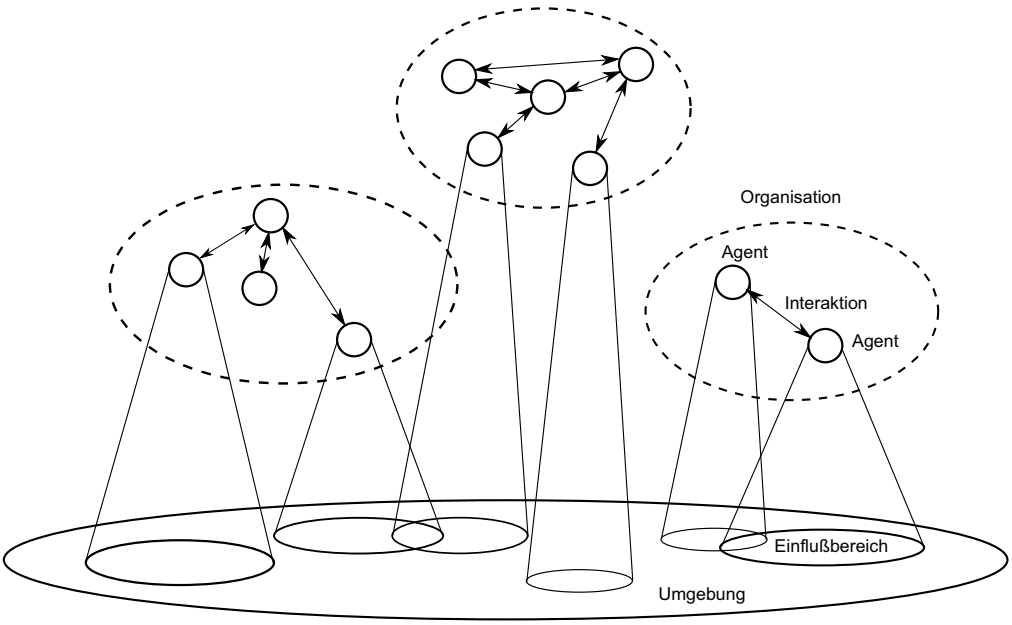

Abbildung 14.4: *Multiagentensystem und Kontext (nach [90]).*

Abbildung 14.4 zeigt eine abstrakte Darstellung eines Multiagentensystems in seinem Kontext: Jeder Agent kann in der allen Agenten gemeinsamen Umwelt nur ein beschränktes Areal wahrnehmen und hat ebenso nur einen beschränkten Einflussbereich. Wahrnehmung und Aktion sind also lokal. Die Agenten stehen miteinander in Beziehung. Sie interagieren, d.h. sie sind in einer dynamischen Relation zueinander, die durch eine Menge von auf einander bezogenen Aktionen entsteht. Die Beziehungen der Agenten untereinander sind üblicherweise weder zufällig noch interagiert jeder Agent mit jedem anderen, sondern wer mit wem in Beziehung steht wird entweder durch die jeweiligen Aufgaben oder Positionen bestimmt und wird vom Entwickler des Systems festgelegt. Das Arrangement der Beziehungen zwischen Agenten, die zusammen ein System bilden, nennt man auch Organisation. Die meisten Autoren verlangen zudem als definierendes Element einer „Organisation“, dass die Organisation als Einheit ein Ziel hat, also das mehr oder weniger vordefinierte Muster der Agenteninteraktionen einem gemeinsamen Zweck dient [19]. Dies soll im Folgenden näher erläutert werden.

14.2 Interaktion, Kommunikation, Organisation

Interaktion zwischen Agenten bedeutet, dass die Aktionen eines Agenten das zukünftige Verhalten der anderen Agenten beeinflussen [36]. Es gibt direkte Interaktionen zwischen Agenten – z.B. durch direkte, nachrichtenbasierte Kommunikation – und indirekte Interaktion, bei der ein Agent die gemeinsame Umwelt verändert. Die Grundidee ist, dass die Manipulation oder Nachricht des einen Agenten das Verhalten der anderen Agenten beeinflusst.

Interaktionsfähigkeit ist eine der definierenden Eigenschaften des Agent-Seins. Zunächst soll nun nachrichtenbasierte Kommunikation darstellt werden. Nachrichten müssen in einen sinnvollen Zusammenhang gebracht werden. Das Senden einer Nachricht generiert Erwartungen, welche Nachrichten der Sender als Antwort empfangen könnte. Sinnvolle Konversationen werden mit Protokollen spezifiziert. Die nächste Frage ist dann, wer wann an welchen Protokollen teilnimmt und warum. Dies führt zu Organisation und Organisationsstrukturen. Zuletzt muss noch die Frage beantwortet werden, wie die Agenten letztendlich gut zusammenarbeiten können.

14.2.1 Kommunikation und Koordinationsinfrastruktur

Direkter Nachrichtenaustausch zwischen zwei oder mehreren Agenten ist die häufigste Form der Kommunikation. Weitere Formen sind die Multicast-Nachricht – es gibt mehrere Empfänger, deren Anzahl ist beschränkt, die Empfänger sind bekannt – oder die Broadcast-Nachricht, die wie im Radio an „alle“ geht, die zuhören. Das bedeutet, der Sender weiß nicht, wer und wie viele Agenten die Nachricht empfangen. Eine Alternative zum Nachrichtenaustausch ist die Verwendung eines Blackboards (siehe Seite 536).

Nachrichtenbasierte Interaktion

Kommunikation ist eine besondere Form der Interaktion, die sowohl das Empfangen von Nachrichten als Wahrnehmung und das Senden von Nachrichten als eine Aktion beinhaltet.

Die Basis für Kommunikation in Multiagentensystemen bildet die *Sprechakttheorie*. Die Basisidee dabei ist, dass Kommunikation eine Form von Aktion darstellt. Begründet wurde die

Sprechakttheorie von den Philosophen John Austin [2] und John Searle [77]. Ein Sprechakt entspricht einer Äußerung, die die Welt verändert. Ein oft verwendetes Beispiel ist die Aussage des Standesbeamten „Hiermit erkläre ich Sie zu Mann und Frau“ am Ende einer Trauung als einen Sprechakt, der Fakten schafft. Laut Sprechakttheorie besitzt jeder Sprechakt (etwas vereinfacht) drei Elemente: Die lokutive Komponente, die die Aussprache betrifft, die illokutive Komponente, die die Absicht des Sprechers darstellt und die perlokutive Komponente, die den Effekt auf den Empfänger beschreibt. Für die Entwicklung von Agentenkommunikationssprachen war insbesondere die Repräsentation der illokutiven Komponente wichtig. Sie wird explizit repräsentiert in Form von `Performativ(Inhalt):Request(„Hol' Kaffee“)` oder `Inform(„Kaffee ist aus“)`. Das Performativ stellt explizit die Absicht des Senders dar.

Eine Agentenkommunikationssprache beschreibt die atomaren Bestandteile der Agentenkommunikation: die Nachrichten. Der sinnvolle Zusammenhang zwischen den einzelnen Nachrichten wird über Protokolle und ihre Spezifikation erreicht. Alle Nachrichtensprachen für Agentenkommunikation basieren auf der Idee, dass alle Information, die zum Verstehen des Inhalts einer Nachricht notwendig sind, mit der Nachricht verschickt werden soll. Eine solche Nachricht sieht typischerweise wie folgt aus:

```
(REQUEST
     :sender   Professor
     :receiver Student
     :ontology Universität-Ontologie
     :language Deutsch
     :content  "Kommen Sie pünktlich in die Vorlesung!"
)
```

`REQUEST` bestimmt die Intention des Senders der Nachricht, das Performativ. Im Beispiel ist es eine Anfrage oder Forderung (abhängig von der „Macht“ des Senders). Der eigentliche Inhalt steht im Feld `:content`. Um diesen Inhalt verstehen zu können, benötigt man zwei weitere Attribute: `:language`, das die Sprache angibt, in der der Inhalt geschrieben ist, und `:ontology`, mit der die Bedeutung der verwendeten Wörter spezifiziert wird. `:sender` und `:receiver` sind Informationen zum Sender und Empfänger der Nachricht. Es gibt weitere Felder, mit denen z.B. der Zusammenhang zwischen Nachrichten hergestellt werden kann, die im Beispiel nicht dargestellt sind.

KQML („Knowledge Query and Manipulation Language“) – entstanden aus der Knowledge Sharing Initiative [66] – war die erste Agentenkommunikationssprache [40], die eine weite Anwendung gefunden hat. Ihr Hauptproblem war, dass es weder eine Einigung auf einen minimale Menge von Nachrichtenprimitive (Performative) gab, noch die Performative formal klar definiert waren. Je nach Anwendungsfall wurden neue Primitive hinzugefügt, so auch auch Performative für Infrastrukturaufgaben. Welchen Effekt ein Performativ haben sollte, konnte jedoch von Agentensystem zu Agentensystem unterschiedlich sein. Das Ziel der Unterstützung von Interoperabilität wurde dadurch nicht erreicht.

Aus diesem Grund wurde ACL („Agent Communication Language“) eingeführt und von FIPA[3] standardisiert (siehe dazu `www.fipa.org`). Einen ausführlichen Vergleich zwischen KQML

[3] FIPA steht etwas irreführend für „Foundation of Intelligent Physical Agents“ und zeichnet sich, 1996 gegründet, verantwortlich für Standardisierung bei Multiagentensystemen aus. Ziel ist, tatsächlich interoperable Multiagentensysteme zu schaffen.

und FIPA-ACL findet man in [57]. Die Anzahl der verfügbaren Nachrichtentypen ist beschränkt, es beinhaltet keine Infrastruktur-Performative, dafür aber Performative, mit denen Verpflichtungen („Commitments“) behandelt werden können. Somit können Verhandlungsprotokolle klarer realisiert werden. Ein weiterer Vorteil ist die formal definierte Semantik, die Interpretationsspielräume bei der Verwendung der Nachrichten verringert. Die Syntax von KQML wurde absichtlich wiederverwendet, um weiterhin die umfangreich vorhandene KQML-basierte Middleware benutzen zu können.

Indirekte Interaktion, Blackboard Systeme

Nachrichtenbasierte Interaktion ist meist direkt, d.h. von einem Agenten zum anderen. Viele Protokolle arbeiten mit „Multicast“ oder „Broadcast“ Nachrichten, um zum Beispiel mehreren Agenten gleichzeitig die Aufforderung zum Einreichen von Geboten zukommen zu lassen. Bei ersterem („Multicast“) ist die Anzahl der Empfänger begrenzt und die Empfänger sind dem Sender bekannt. Beim zweiten weiß der Sender nicht, wer und ob überhaupt jemand seine Nachricht empfängt. Eine andere Form der Interaktion geschieht indirekt über eine absichtliche oder unabsichtliche Manipulation der gemeinsamen Umwelt. Ein Agent ändert einen Aspekt der Umwelt, z.B. legt er eine Spur. Ein anderer Agent nimmt diese geänderte Umwelt wahr und passt sein Verhalten an, folgt der Spur. Diese Form der indirekten Interaktion wird auch Stigmergie genannt. Sie hat in so fern Ähnlichkeit mit Broadcast Nachrichten, als die tatsächlichen Empfänger unbekannt sind. Interaktionen mittels Stigmergie sind dazu noch zeitlich entkoppelt. Der Sender weiß weder, wer die Nachricht erhält, noch, wann die geänderte Umwelt wahrgenommen wird. Für ein robustes System muss die in der Umwelt abgelegte Nachricht wieder gelöscht werden. Ein prominentes Beispiel für stigmersive Interaktion ist die Massenrekrutierung bei Ameisen durch Pheromonspuren, die im Bereich der Ant Colony Optimization [30] in Multiagentensystemen zur Lösung von Routingproblemen nachgebildet werden.

Bei vielen emergenten Phänomenen (siehe [48]) liefert eine stigmersive Interaktion einen wichtigen Beitrag zur Entstehung. Kurz charakterisiert, geschieht Emergenz, wenn das Ganze mehr ist als die Summe seiner Teile. Durch Interaktionen zwischen Einheiten auf einer unteren, disaggregierten Ebene des Gesamtsystems werden nicht-lineare Rückkopplungen erzeugt, die auf einer höheren Ebene etwas „Neues“ erzeugen. Das Neue ist dadurch charakterisiert, dass es nicht mit den Mitteln beschreibbar ist, mit denen man das Verhalten der Entitäten auf der unteren Ebene beschreibt. Klassische Beispiele sind Neuronen im Gehirn, die ein Bewusstsein entstehen lassen, oder Stau, der sich als Ganzes in die entgegensetzte Richtung zur Bewegungsrichtung seiner Bestandteile bewegt. Emergente Phänomene sind schwer zu analysieren, da sie erst bei der Simulation oder Ausführung der Verhaltensweisen auf der unteren Ebene entstehen. Multiagentensysteme eignen sich in besonderer Weise, Emergenz zu untersuchen. Wichtig ist andererseits, bei der Entwicklung von Multiagentensystemen unerwünschte emergente Phänomene auszuschließen.

Eine etablierte Form der stigmersiven Interaktion findet man bei Blackboardsystemen [21]. Diese wurden als Hilfsmittel beim gemeinsamen Problemlösen durch heterogene Agenten erfunden und basieren auf der Tafel-Metapher: Eine Gruppe von „Experten“ steht zusammen vor einer Tafel, ein Agent erweitert oder adaptiert einen Teil der gemeinsamen Lösung, die anderen nehmen diese Änderung direkt auf der Tafel wahr und adaptieren ihre Schlussfolgerungen und ändern ihrerseits. Die Tafel entspricht dabei dem gemeinsamen Speicherbereich der Agenten. Im Gegensatz zur Protokollen zur Aufgabenteilung unterstützen Blackboardsysteme das gemeinsame Bearbeiten von Ergebnissen, das sogenannte Result-Sharing. In den letzten Jahren wurde die

Implementierung von Blackboards auf der Basis von Tuplespaces vorgeschlagen [73]. Damit wurde Koordinationsinfrastruktur aus dem Bereich der Verteilten Systeme auf Multiagentensysteme angepasst.

14.2.2 Interaktionsprotokolle

Um eine sinnvolle Interaktion zu erreichen, reicht die Definition einer Nachrichtensprache nicht aus. Ein Agent und seine Interaktionspartner müssen einem gemeinsamen Muster von Nachrichten folgen, um zu einer sinnvollen Konversation zu kommen. Mit dem Schicken einer Nachricht erzeugt ein Agent Erwartungen, welche Antworten auf seine Nachricht möglich sind. Solche sinnvollen Nachrichtenfolgen werden in Protokollen spezifiziert. Nicht die Nachrichtentypen, die ein Agent verstehen kann machen seine Schnittstelle aus, sondern die Protokolle, an denen sich der Agent beteiligen kann. Auch Protokolle wurden von FIPA standardisiert. Zur Beschreibung von Interaktionsprotokollen wurden verschiedene Sprachen vorgeschlagen. [36, Kapitel 6] gibt einen schönen Überblick. Beispiele für Sprachen zur Modellierung von Protokollen basieren auf endlichen Automaten, wie z.B. COOL [4] oder auf Petri-Netzen, wie [23]. Meist wird jedoch AgentUML [63], eine Erweiterung der UML Sequenzdiagramme, verwendet. Auch FIPA benutzt diese Sprache zur Spezifizierung von standardisierten Protokollen. Ein leicht vereinfachtes Beispiel findet man in Abbildung 14.5. Die wichtigsten Erweiterungen betreffen zum einen die Behandlung von flexiblen Reaktionen auf eine Nachricht (wie im Bild dargestellt), zum anderen mögliche Verzweigungen der Lebenslinien. Zusätzliche Informationen, wie Performative statt Methodennamen oder Rollen statt konkreten Objekten, betonen das höhere Abstraktionsniveau. Mit der Veröffentlichung von UML 2 wurde die weitere Entwicklung von AgentUML eingestellt [5].

Es gibt verschiedenste Protokolle zu unterschiedlichen Zwecken, von der einfachen `Request-Agree` Interaktion, bis zu komplexen, iterativen Verhandlungen. Ein wichtiger Schritt beim Entwickeln eines Multiagentensystems ist die Definition der verwendeten Protokolle (siehe dazu auch Abschnitt 14.4.1). Das Contract Net Protocol [82] ist das bekannteste Protokoll. Es dient zur Verteilung von Aufgaben auf Agenten und bildet einen Ausschreibungsmechanismus nach. Ein „Manager“ schickt eine Aufforderung zur Einreichung von Vorschlägen an eine Menge von möglichen Bearbeitern. Diese berechnen, wie schnell, mit welcher Qualität sie die Aufgabe erfüllen können, bzw. wollen. Auf dieser Basis schicken die möglichen Auftragnehmer Gebote an den Manager. Dieser evaluiert die Gebote und wählt das beste aus. Das Contract Net Protocol ist flexibel, robust und funktioniert, auch wenn mögliche Auftragnehmer ausfallen. Es kann hierarchisch angewendet werden. Natürlich gibt es auch Schwachpunkte zusätzlich zur zentralen Funktion des Managers: das Protokoll geht von einer vorgegebenen Unterteilung der Gesamtaufgabe aus. Wenn diese Unterteilung nicht zu den während der Verhandlung verfügbaren Agenten passt, dann ist das Protokoll hilflos. Dies gilt ebenso, wenn aus verschiedenen Gründen die besten Agenten nicht auf die Ausschreibung reagieren. Abbildung 14.5 zeigt eine Darstellung des Contract Net Protocols in einer etwas vereinfachten Version von AgentUML.

14.2.3 Organisation

Ein zentraler Aspekt – insbesondere wenn das Multiagentensystem als solches gezielt für einen bestimmten Zweck entwickelt wird – ist, wer mit wem zu welchem Zweck interagiert. Um die Entwicklung solcher kooperativer Multiagentensysteme zu verbessern, kann man eine explizite

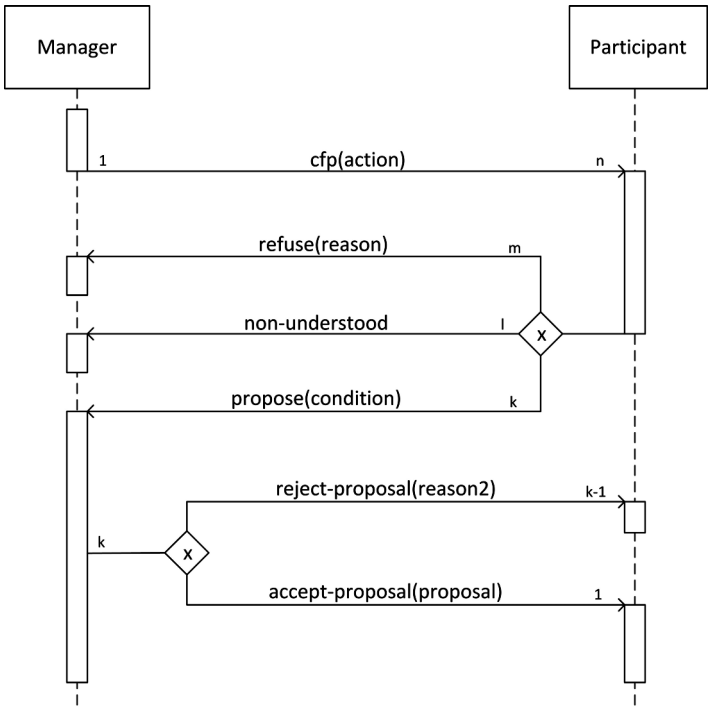

Abbildung 14.5: *Spezifikation des Contract Net Protocols.*

Sicht auf die Organisation von Multiagentensystemen verwenden. Ausgehend von der Idee, dass eine Organisation eine gemeinsames Ziel hat, kann ein Entwickler versuchen, die Menge aller Agenten in der Organisation zu strukturieren, Gruppen zu bilden, diesen Gruppen Teilziele zuzuordnen usw. Ein solches explizites Vorgehen ist typisch für eine Reihe von Methodologien im Bereich des Agentenorientierten Software Engineerings. [60] und [49] geben einen Überblick über die Organisationskonzepte, die auf verschiedenen Ebenen eines Multiagentensystems benutzt werden können. Die wichtigsten sind dabei folgende:

- Die *Rolle* als eine abstrakte Darstellung der Aktivitäten bzw. Services, die ein Agent im Rahmen einer Organisation bereitstellen muss. Die Rolle fokussiert also darauf, „was“ zu tun ist, ein Agent dagegen kümmert sich dann um das „wie“, wenn er die Rolle akzeptiert hat. Eine Rolle kann von mehreren Agenten gleichzeitig gespielt werden. Oft werden mit Rollen Verantwortlichkeiten und Pflichten assoziiert, ebenso wie Rechte und Ziele. [18] geben einen aktuellen Überblick über den Rollenbegriff in Multiagentensystemen, sowie über verschiedene Mechanismen, wie Rollen einem Agenten zugeordnet werden können.
- Abhängigkeiten und Protokolle verbinden Rollen in einer klar definierten und systematischen Art und Weise. Abhängigkeiten können dabei verschiedenste Formen annehmen – z.B. eine Informationsbeziehung, bei der ein Agent zu einem gewissen Zeitpunkt Informationen von einem anderen benötigt oder eine Autoritätsbeziehung, bei der ein Agent Anweisungen vom anderen akzeptiert. Protokolle realisieren die Abhängigkeiten oder lösen gegebenenfalls Konflikte auf.

- Rollen werden in *Gruppen* zusammengestellt. Gruppen sorgen für die Strukturierung der Interaktion, da meist ein Agent nur mit anderen Agenten aus der gleichen Gruppe interagieren darf. Interaktionen zwischen Gruppen können über Agenten geschehen, die Rollen in mehreren Gruppen spielen. Einer Gruppe kann dabei auch ein spezielles gemeinsames Ziel zugeordnet werden.

Eine Organisationsstruktur bestehend aus der Spezifikation von Gruppenstrukturen und Rollen, kann so als Muster oder Architektur für ein Multiagentensystem gesehen werden. Dies ist vergleichbar zu Agentenarchitekturen, die die Prozesse in einem Agenten steuern. Es wurden viele Spezifikationssprachen für Organisationen vorgeschlagen, die eine Entwicklung von organisationsbasierten Multiagentensystemen unterstützen sollen. Die bekanntesten sind das AGR-Modell [37], das eine recht pure Sicht auf Gruppen und Rollen bietet und sich auf einfache Konzepte beschränkt, und MOISE+ [51], das eine ausgefeilte Sprache bietet, unter anderem mit Rollenvererbung, verschiedenen Beziehungstypen oder hierarchischen Gruppenstrukturen. In einer zusätzlichen Sicht wird auch eine Zieldekomposition und Zuteilung zu Rollen unterstützt. Eine weitere Spezifikationssprache ist OperA [29]. Man findet auch verschiedene Ansätze zur Modellierung von Organisationen in Kombination mit Methoden zum Agentenorientierten Software Engineering. [85] zeigen, wie man eine dynamischere normative Sicht mit Agenten und Organisationen in einem flexibleren Modellierungsrahmen vereinigt. Informationen, wie mit Multiagentensysteme mit Organisationskonzepten programmiert werden können, findet man z.B. in [50].

14.2.4 Koordinierte Aktivitäten

Als letzter Schritt bei der Behandlung von Kommunikation und Koordination sollen nun Ansätze beschrieben werden, wie Agenten, nachdem sie wissen, mit wem sie zusammenarbeiten sollen, ihre Aktivitäten koordinieren können, um diese Zusammenarbeit zu realisieren (für eine Übersicht siehe [32]). Im Allgemeinen geht man davon aus, dass jeder Agent sein Teilziel kennt. Die Verteilung kann inhärent sein, z.B. auf der Basis der Rolle in einer Organisation, aufgrund von räumlicher Verteilung oder sie kann mittels z.B. eines Contract Net Protocols erfolgt sein. Auf der Basis dieser Aufgabenverteilung erzeugt jeder Agent einen Plan seiner zukünftigen Aktivitäten. Wie dies gemacht wird, ist abhängig von der verwendeten Agentenarchitektur. Der zentrale Aspekt ist die Koordination der Aktivitäten, um Abhängigkeiten zwischen den Plänen aufzulösen. Die Koordination kann vor dem Planen, während des Planens oder nach dem Planen geschehen. Bei der Koordination vor dem Planen versucht man, durch Organisationsstrukturen oder durch zusätzliche Constraints die Handlungsmöglichkeiten der Agenten so weit einzuschränken, dass Interferenzen gar nicht auftauchen, auch wenn jeder Agent unabhängig vom anderen plant. Solche Constraints werden auch als Soziale Gesetze („Social Laws“) oder Normen bezeichnet. Ein klassisches Beispiel ist die Regel, dass Fahrzeuge am rechten Rand der Straße fahren sollen. Dieses Rechtsfahrgebot schränkt die Freiheit der Verkehrsteilnehmer ein; sie dürfen nicht mehr überall fahren. Es sorgt aber, sofern sich jeder dran hält, für eine konfliktfreie Wegeplanung und koordiniertes Fahren. Ein wichtiger Aspekt bei Normen insbesondere bei offenen Systemen (siehe nächster Abschnitt) ist die Durchsetzung dieser normativen Regeln. Einen generellen Rahmen und Formalisierung von „Coordination by Design“ – also der Koordination vor dem eigentlichen Planen findet man in [83].

Die beiden anderen Möglichkeiten, d.h. die Koordination nach und während des Planens, erfordern, dass Agenten in der Lage sind, so flexible Pläne zu entwickeln, dass sie für die Koor-

dination angepasst werden können (im Folgenden nach [32]). Bei der Koordination nach dem Planen erzeugt jeder Agent unabhängig von den anderen einen partiell geordneten Plan. Danach wird dieser Plan entweder an eine zentrale Instanz geschickt, die die Pläne der einzelnen Agenten aufeinander abstimmt, oder die Agenten erledigen diese Abstimmung durch massive Kommunikation, in dem sie sich z.B. schrittweise darauf einigen, wer wann welche Aktion ausführen darf. Die dritte Form des Multiagentenplanens ist, wenn das Planen und die Koordination miteinander verzahnt sind, wie z.B. im Partial Global Planing (PGP) [33] (oder seiner verallgemeinerten Version, dem Generalized Partial Global Planing (GPGP) [27]). Die Grundidee dabei ist, dass die Agenten gleichzeitig abstrahierte Pläne koordinieren, während sie diese abstrakten Pläne auf lokaler Ebene ausführen.

14.3 Von der Kooperation zum Wettbewerb

Bisher haben wir Systeme betrachtet, bei denen ein oder eine Gruppe von Entwicklern ein Gesamtsystem aus interagierenden Agenten entwickelt. Dabei kann angenommen werden, dass die Agenten, die gemeinsam ein bestimmtes Ziel erreichen sollen, dies wenn möglich auch tun wollen. Die Agenten kooperieren, weil sie so gemacht wurden. Eine andere Form der Multiagentensysteme sind die offenen System [47]. Dabei kann jeder Agent von einem anderen Entwickler entworfen worden sein und damit ein Ziel verfolgen, das nicht kompatibel mit den Zielen der anderen Agenten ist. Der Agent kann im Wettbewerb mit den anderen Agenten stehen. Ein Beispiel sind Auktionsplattformen im Internet: Benutzer (Agenten) können jederzeit hinzukommen, um Dinge zu verkaufen oder auf Dinge zu bieten. Bieten zwei Agenten auf das gleiche Gut, stehen sie im Wettbewerb um dieses Gut. Keiner weiß vom anderen, wie viel derjenige letztendlich bereit ist zu bieten, aber beide wollen das Gut so günstig wie möglich ersteigern. Solche *offenen* Multiagentensysteme haben andere Charakteristika als Multiagentensysteme, die mit einem Systemziel als Gesamtheit entworfen wurden. Man kann keine Annahmen über Kooperativität treffen, sondern muss davon ausgehen, dass die Agenten egoistisch und rational sind. Natürlich können obige Routinen, Protokolle oder Organisationsstrukturen auch in offenen Systemen verwendet werden, aber grundsätzlich müssen die egoistischen Agenten einen Vorteil im kooperativen Verhalten erkennen. Auch bei offenen Systemen verfolgen die Entwickler des Gesamtsystems bzw. der Interaktionsinfrastruktur ein Ziel. Das kann eine „gerechte“ Verteilung von Ressourcen oder eine bestimmte Funktionalität sein, die nur zusammen erreicht werden kann. Der Entwurf und die Analyse von Protokollen oder Mechanismen, die die Interaktion der Agenten so regeln, dass dieses Ziel erreicht werden soll, geschieht im sogenannten „Mechanism Design“ (siehe [56, 72, 75, 80]). Dabei versucht man, einen Mechanismus so zu entwerfen, dass er garantiert zum Ziel, d.h. einer Einigung kommt, und dass er von keinem Agenten so manipuliert werden kann, dass das Ergebnis nicht mehr für alle das beste ist. Außerdem soll es für den einzelnen Agenten besser sein, teilzunehmen als außen vor zu bleiben, d.h. individuell rational sein. Weitere Anforderungen sind unter anderem Verteiltheit (es gibt keinen Agenten, der eine zentrale Stelle einnimmt und so zum Flaschenhals wird), Effizienz und Einfachheit. Die Analyse solcher Mechanismen geschieht auf der Grundlage der Spieltheorie. Eine ausreichend tiefe Einführung in die Spieltheorie ist im Rahmen dieses Kapitels nicht möglich. [80] geben eine gut verständliche, aber sehr präzise Darstellung der Grundlagen der Spieltheorie soweit sie für Multiagentensysteme relevant ist. Im Folgenden soll daher nur kurz die grundlegende Idee der rationalen Agenten erläutert werden.

14.3.1 Idee des rationalen Agenten

Rationalität als Eigenschaft eines Agenten bedeutet zunächst (siehe oben), dass der Agent nichts macht, was seinen Zielen entgegen wirkt. Bei intelligenten Softwareagenten bedeutet dies, dass wenn ein Agent die Wahl hat, 10,0 Euro mit einer Lüge und 9,99 Euro für eine wahre Aussage zu verdienen (und sein Ziel ist möglichst viel Geld einzusammeln), dann wird dieser Agent lügen. Die Wahl der richtigen Aktion ist meist schwieriger, da das Ergebnis nicht nur von der Aktion eines Agenten abhängt, sondern auch von den Aktionen der anderen Agenten, die gleichzeitig ihre Aktion wählen und ausführen. Die für ihn beste Aktion bestimmt ein Agent, indem er alle Ergebnisse betrachtet, die bei allen jeweils möglichen Aktionskombinationen aller Agenten erzeugt werden. Diese möglichen Ergebnisse werden für zwei Agenten üblicherweise in einer Payoff-Matrix repräsentiert. Tabelle 14.1 ist ein Beispiel für eine Payoff-Matrix im Gefangenendilemma – dem bekanntesten Szenario der Spieltheorie.

Tabelle 14.1: *Beispiel für eine Payoff-Matrix. Sie stellt vollständig die Resultate für beide Agenten für jede Aktionskombination dar. Die Werte entsprechen einem Gefangenendilemma.*

		AgentB	
		C	D
AgentA	C	2 2	1 4
	D	4 1	3 3

Die Tabelle ist so zu lesen: Entscheidet sich der Agent A (der „Zeilenspieler“) für die Aktion „C“ (Kooperation) und der Agent B (der „Spaltenspieler“) für die Aktion „D“ (Verraten), erhält Agent A einen Punkt, während der Agent B 4 Punkte bekommt. Diese Payoff-Matrix ist ein Beispiel für das wohl bekannteste spieltheoretische Szenario: dem Gefangenendilemma. Es gibt zwei Aktionen – eine „Verraten“ (engl. defect, Aktion D) und „kooperieren“ (engl. cooperate, C). Das besondere an diesem Szenario ist der Wert des Vertrauens: Wenn beide kooperieren, ist das individuelle Ergebnis besser, als wenn beide nicht kooperieren (verraten). Die Versuchung zu verraten ist hoch, weil damit der höchste Gewinn erzielt werden kann. Ähnliche Situationen findet man häufig im realen Leben: Alle investieren in ein gemeinsames Projekt und kooperieren. Die Versuchung existiert, die Projektergebnisse auszunutzen ohne vorher in den gemeinsamen Topf investiert zu haben. Ein Agent der „C“ spielt, zahlt, ein Agent der „D“ spielt, nutzt das gemeinsame Gut ohne zu zahlen.

Die Spieltheorie zeigt, wie man solche Szenarien analysieren kann. Wichtige Mittel sind dabei: das Konzept der dominanten Strategie/Aktion, das Nash-Gleichgewicht und die Idee von Pareto-Optimalität. Alle drei – soweit vorhanden – sind Anhaltspunkte für die Stabilität des Spiels: Die dominante Strategie oder Aktion ist diejenige, die das beste Ergebnis für den Agenten liefert egal, welche Strategie der andere Agent auswählt. Im obigen Szenario ist dies die Aktion „D“. Wenn der Agent „C“ wählen würde, wäre das schlechtest mögliche Ergebnis für den Agenten gleich 1, wenn der „D“ wählt ist es 2. Das Nash-Gleichgewicht dagegen ist die Situation, in der die Strategie des einen Agenten die beste Antwort auf die des anderen Agenten ist und vice versa. Im Szenario ist das Nash-Gleichgewicht die Situation, in der beide Agenten „D“ wählen. Die beste Reaktion auf ein „D“ ist ein „D“ (ebenso ist „D“ die beste Reaktion auf ein „C“). Der pareto-effiziente Zustand dagegen, ist wenn beide „C“ wählen: Durch ein Abweichen von „C“ kann ein Agent nur dann etwas gewinnen, wenn ein anderer Agent verliert. Solche

Analysen ermöglichen es, die Entscheidungen von Agenten vorher zu sagen. Dies geschieht aber unter der Annahme, dass die Agenten vollständig rational sind und vollständiges Wissen über alle möglichen Ergebnisse als Reaktion der Aktionen aller Agenten kennen. Das Ergebnis widerspricht jedoch manchmal einer menschlichen Intuition von „Fairness“. [26] zeigen, dass „Fairness“ formal dargestellt werden kann und bei bestimmten Scenarien sehr nützlich sein kann.

Spieltheorie findet einen intensiven Einsatz in der theoretischen Ökonomie und anderen Sozialwissenschaften. Im Bereich der Multiagentensysteme bildet sie, wie oben erwähnt, die Grundlage für den „Mechanism Design“. Dieser Forschungsbereich beschäftigt sich mit der Entwicklung von Interaktionsmechanismen, also mit den Regeln des Spiels, die bestimmte Eigenschaften besitzen. Die hier dargestellte, einfache Form von spieltheoretischen Szenarien reicht dazu nicht aus. In [80] findet man eine Einführung in komplexere Formen, die tatsächlich für die Interaktionsanalyse verwendet werden.

Im Folgenden sollen die wichtigsten Interaktionsprotokolle für verschiedene, wichtige Szenarien vorgestellt werden. Es ist interessant zu sehen, wie viele dieser Mechanismen durch ein Vorbild aus menschlichen Gesellschaften inspiriert sind. In den 90igern wurde deswegen eine neue Forschungsrichtung „Sozionik“ gegründet, die Multiagentensystem- und Soziologische Forschung zusammenbringen sollte. Trotz großer Hoffnungen (und Fördermittel) hatten die Ergebnisse bisher noch kaum Einfluss auf den Stand der Forschung. Der Grund ist wohl auch, dass das hohe Abstraktionsniveau, auf dem Agenteninteraktionen behandelt werden, nur schlecht zu der Komplexität der von der Soziologie behandelten Realwelt passt.

14.3.2 Voting

Das Abstimmen – englisch „Voting“ – bezeichnet eine Gruppe von Mechanismen, bei denen die Agenten über die Auswahl einer Lösung aus einer Menge von möglichen Optionen abstimmen. Die gewählte Option ist dann für alle Agenten bindend. Die Agenten fällen also eine Gruppenentscheidung. Wie in menschlichen Gesellschaften kann über gemeinsame Aktivitäten oder einen „Anführer“ abgestimmt werden. Jeder Agent muss eine im einfachsten Fall vollständige Präferenzrelation besitzen, die für jedes Paar an Optionen angibt, welche Option der Agent im Vergleich zur anderen bevorzugt. Bei manchen Mechanismen muss jeder Agent nur öffentlich machen, welches seine bevorzugte Option ist, bei anderen Mechanismen muss der Agent seine gesamte Präferenzrelation offen legen. Es gibt eine Reihe von Abstimmungsmechanismen, jede mit spezifischen Vor- und Nachteilen. Eine tiefergehende Behandlung findet man in [75]:

- Pluralitätsprotokoll: Jeder Agent teilt als Eingabe dem Mechanismus mit, welche die von ihm bevorzugte Option ist. Der Mechanismus selektiert die Option mit den meisten Stimmen. Dieser Mechanismus hat eine sehr klare Entsprechung bei Wahlen in menschlichen Gesellschaften. Wie im richtigen Leben kann auch bei Agenten die Wahl in einem Unentschieden resultieren, wenn die Anzahl der Stimmen für mehrere, meist-gewählte Optionen gleich sind; dann wird eine zusätzliche Regel benötigt, um das Unentschieden auflösen zu können. Varianten des Pluralitätsprotokolls sind das kumulative Pluralitätsprotokoll, bei dem jeder Agent eine festgelegte Anzahl von Stimmen verteilen darf oder das Pluralitätsprotokoll mit Elimination, bei dem die Agenten wiederholt wählen: Nach jedem Wahlgang wird die Option mit den wenigsten Stimmen aus dem Pool der möglichen Optionen entfernt. Dies wird wiederholt, bis nur noch eine Option übrig ist.

- Bordaprotokoll: Jeder Agent teilt dem Mechanismus als Eingabe seine vollständige Präferenzrelation mit. Für jeden Agenten ordnet der Mechanismus jeder Option eine Zahl zu, die der Position im persönlichen Ranking des Agenten entspricht. Diese Zahlen werden für alle Agenten aufsummiert. Die Option mit der höchsten Summe ist das Ergebnis des Mechanismus.
- Binärprotokoll: Die Agenten stimmen paarweise über die Optionen ab. Immer zwei Optionen werden miteinander verglichen. Die Option mit den meisten Stimmen wird mit der nächsten Option verglichen. Dies wird so lange wiederholt, bis nur noch eine Option übrig ist.

Folgendes Beispiel illustriert die verschiedenen Protokolle: Gegeben sind 3 Optionen $\{a, b, c\}$. Agent A besitzt folgende Präferenzrelation: $a > b > c$, Agent B $c > a > b$, ebenso wie Agent C, und Agent D $b > a > c$. Bei Verwendung eines Pluralitätsprotokoll teilt jeder Agent seine erste Wahl mit und die Option c gewinnt mit zwei Stimmen. Beim Borda-Protokoll wird für Agent A, der Option a eine 3 zugeordnet, 2 zu b und 1 zu c. Dies wird für alle Agenten entsprechend gemacht. Diese Zahlen werden summiert. Das Ergebnis ist, dass a den Borda-Score 9, b 7 und c 8 bekommt und somit a gewählt wird. Das Binärprotokoll vergleicht die Optionen paarweise: Nimmt man z.B. die Reihenfolge: a-b-c kann folgendes passieren: In der ersten Runde stimmen z.B. die Agenten über a oder b ab: a ist der klare Gewinner mit 3:1 Stimmen. In der nächsten Runde a–c, gibt es ein Unentschieden, das durch zusätzliche Regeln aufgelöst werden muss. Man kann für jeden dieser Mechanismen ein Szenario konstruieren, bei dem Schwächen klar werden oder ein sogenanntes Voting Paradox auftritt. So kann sich beim Borda-Protokoll das Gesamtergebnis ändern, wenn irrelevante Optionen entfernt werden. Irrelevante Optionen sind solche, die kein Agent als bevorzugte Wahl hat. Das Binärprotokoll ist abhängig davon, in welcher Reihenfolge die Optionen miteinander verglichen werden. Auch hier kann man ein Szenario konstruieren, bei dem jede Reihenfolge zu einem anderen Gesamtergebnis führt. [80] schildert verschiedene Szenarien mit erstaunlichen Voting Paradoxien.

Alle diese Mechanismen können von Agenten manipuliert werden. In [80] findet sich ein schönes Beispiel, wie ein Pluralitätsprotokoll manipuliert werden kann: Ein Agent kennt die Präferenzrelationen aller anderen Agenten und weiß, dass seine bevorzugte Option keine Mehrheit finden wird. Die Manipulation besteht nun daraus, dass er nicht ehrlich seine bevorzugte Option angibt, sondern eine andere, und so den eigentlichen Gewinner der Abstimmung verhindert. Die Agenten haben einen Anreiz über mögliche Präferenzrelationen der anderen Agenten zu spekulieren, um die Abstimmung zu ihren Gunsten zu manipulieren. Derartiges Spekulieren und Manipulieren will man durch einen guten Mechanismus verhindern: Es gibt Abstimmungsverfahren – den Vickrey-Clarke-Groves Mechanismus [80] –, bei denen die dominante Strategie ist, ehrlich zu sein, d.h. immer die bevorzugte Option als solche zu benennen: Dabei gibt ein Agent nicht nur die Option an, sondern eine numerische Präferenz für jede Option. Der Mechanismus ermittelt dann nicht nur die Option mit der höchsten Bewertungssumme als den Gewinner der Abstimmung, sondern auch eine Art Steuer, die jeder Agent zahlen muss. Die individuelle Höhe der Steuer richtet sich danach, wie viel Einfluss der Agent auf das Ergebnis genommen hat. Die einzig sinnvolle Strategie für einen Agenten ist dabei, ehrlich zu sein. Würde er einen höheren Wert angeben, als die Option für ihn wirklich wert ist, würde er mehr Steuern zahlen als eigentlich notwendig. Würde er zu wenig angeben, um Steuern zu sparen, würde er riskieren, dass etwas anderes gewählt würde.

14.3.3 Auktionen

Ebenso wie Abstimmungsmechanismen orientieren sich auch Auktionen zunächst an Vorbildern aus der menschlichen Gesellschaft. Das Ziel ist bei Auktionen nicht, als Gruppe eine für alle bindende Option aus zu wählen, sondern, ein Gut dem Agenten zu geben, für den dieses am wichtigsten ist, bzw. der bereit ist am meisten dafür zu bezahlen. Im Endeffekt ist auch das Herzstück des oben beschriebenen Contract Net Protokolls eine Auktion um einen Auftrag. Die Grundaktionen sind dabei das Abgeben von Geboten, das Auswählen eines Gebots und das Festlegen/Zahlen eines Preises. Wie beim Abstimmen gibt es auch bei den Auktionen eine Vielzahl von Möglichkeiten. Diese unterscheiden sich auf einer technischen Ebene dadurch, ob die Gebote verdeckt oder offen abgegeben werden, ob alle Gebote gleichzeitig abgegeben werden oder iterativ, oder welcher Preis gezahlt werden muss. Für die Analyse einer Auktion ist noch dazu wichtig, wie das Auktionsgut bewertet wird, ob nur die persönliche Wertschätzung der Agenten relevant ist, oder ob die Auktion benutzt wird, um den Wert zu bestimmen. Ein Beispiel für ersteres findet man, z.B. bei der Auktion eines Werkzeugs oder einer Aufgabe; Beispiele für letzteres sind z.B. Kunstauktionen. Bekannte Formen von Auktionen sind folgende:

- Die Englische Auktion ist eine klassische Auktion, bei der ein Auktionator den Preis so lange erhöht, bis kein Bieter-Agent mehr ein höheres Gebot einreicht. Die dominante Strategie ist hierbei, das letzte Gebot um eine kleine Summe zu erhöhen. Das Protokoll ist nicht robust gegen eine versteckte Koalition der Bieter. Der Auktionator kann den Mechanismus betrügen, indem er Preistreiber unter den Bietern versteckt.
- Einfache verdeckte Auktionen („sealed bid, first bid“) sind nicht iterativ. Jeder Bieter reicht ein verdecktes Gebot ein. Der Auktionator wählt das höchste Gebot aus, der Bieter zahlt den Preis seines Gebots. Der Bieter hat einen Anreiz, über die Gebote der anderen Agenten zu spekulieren, um den Preis zu drücken, den er zahlen wird. Dies ist umso interessanter, je höher sein Gebot als das der anderen sein würde. Wegen der verdeckten Gebote sind Koalitionen weniger attraktiv, da sie einfach gebrochen werden können, ohne dass der eigentlich Höchstbietende etwas dagegen machen kann.
- Bei einer Holländischen Auktion fängt der Auktionator mit einem hohen Gebot an und verringert den Preis schrittweise, bis der erste Agent den aktuellen Preis akzeptiert. In der Analyse ist diese Auktion äquivalent zur einfachen verdeckten Auktion.

Die wichtigste Form der Auktion in Multiagentensystemen ist allerdings keine der obigen drei, sondern die Vickrey-Auktion („sealed bid, second price“). Dabei geben alle Agenten ein verdecktes Gebot ab. Der Agent mit dem höchsten Gebot erhält das Gut, muss aber nur den Preis des zweithöchsten Gebots zahlen. Dadurch, dass die Höhe des zu zahlenden Preises vom gewinnenden Gut entkoppelt ist, ist die dominante Strategie für einen Agenten, seine wahre Wertschätzung des Gutes weiterzugeben. Bei der Vickrey-Auktion hat ein Agent keinerlei Motivation beim Gebot zu sparen: Ist sein Gebot niedriger als das, was er eigentlich zahlen würde, läuft er Gefahr, dass ein anderer Agent mehr bietet und das Gut erhält. Würde der Agent mehr bieten, als er eigentlich zahlen würde, könnte es sein, dass sein Gebot so knapp unterboten wird, dass der Agent mehr zahlt, als er eigentlich wollte.

Vickrey Auktionen findet man eigentlich in realen Gesellschaften nicht. Dies liegt vor allem daran, dass die Bieter dem Auktionator vertrauen müssen. Der gewinnende Bieter kennt die Höhe des zweithöchsten Gebots nicht. Der Auktionator könnte irgendeinen Preis knapp unter dem

höchstbietenden nennen. Interessant ist allerdings das Auktionsprotokoll von Ebay: Betrachtet man den Mechanismus im Gesamten mit automatischen Bietagenten, dann kann man ihn als Vickrey Auktion interpretieren, obwohl an sich eine klassische Englische Auktion verwendet wird: Der Bieter reicht den Wert, den er zu zahlen bereit ist, an den Bietagenten weiter und zahlt beim Gewinn der Auktion einen Preis, der minimal höher ist als der zweithöchste Preis. Der Bietagent nimmt an der eigentlichen Englischen Auktion teil, die stoppt, wenn kein anderer das letzte Gebot überbietet.

Wirklich kompliziert werden Auktionen, wenn der Wert der versteigerten Güter davon abhängt, was der Agent zuvor bereits ersteigert hat. In solchen Situationen macht es keinen Sinn, die Güter separat voneinander zu betrachten. Idealerweise schnürt der Auktionator ein Paket. Wenn das aus irgendwelchen Gründen nicht geht, dann sind die Agenten gezwungen, über ihre Gebote zu lügen, um zu einer effizienten Verteilung der Güter zu kommen. Ein wichtiger Ansatzpunkt ist dabei die Vorausschau: Z.B. werden zwei Güter G1 und G2 in dieser Reihenfolge versteigert. Macht es nun für einen Agent Sinn, dass er beide Güter hat, dann kann er seine Gebote durch Vorausschau modifizieren, um sicher zu stellen, dass er wirklich beide ersteigert: Bei der Berechnung des Gebots für G1 bezieht der Agent mit ein, welchen Vorteil er bei der Auktion um G2 dadurch haben wird, dass er dann bereits G1 hat. Dieser Vorteil von G2 wird auf das Gebot für G1 zurückgerechnet. Der Agent macht eine falsche Angabe bei G1, das Gesamtergebnis wird aber besser.

14.3.4 Verhandlungen

Ein weiteres Interaktionsprotokoll ist das „Bargaining“, das Handeln. Die Aufgabe ist hierbei, eine Ressource zu teilen. Beim strategischen Bargaining machen die Agenten abwechselnd oder gleichzeitig Vorschläge, die vom jeweils anderen Agenten akzeptiert oder abgelehnt werden. Kommt keine Einigung zustande, dann tritt die sogenannte „Fallback“-Lösung ein. Eine interessante Anwendung des Handelns ist die Verteilung von Aufgaben. [72] formalisieren dies als aufgaben-orientierte Domäne für Verhandlungen. Dabei besitzt jeder Agent eine initiale Zuordnung von Aufgaben, die er erfüllen muss. Über Verhandlungen mit anderen Agenten wird versucht, die Verteilung von Aufgaben zu verbessern, indem die einzelnen Agenten versuchen, Aufgaben abzugeben oder zu tauschen um die individuellen Kosten zu verringern. [72] schlagen ebenfalls ein Verhandlungsprotokoll vor, das monotone Zugeständisprotokoll („monotonic concession protocol“): Die Menge möglicher Abmachungen („Deals“) ist dabei die Menge von Aufgabenverteilungen zwischen den Agenten, die für die Agenten jeweils individuell rational und pareto-effizient sind. Individuell rational bedeutet hier, dass die Kosten für die im Deal zugeordneten Aufgaben geringer als die ursprünglich zugeordneten Aufgaben sind. Pareto-effizent bedeutet, dass keine weitere Abgabe oder Tausch möglich ist, bei der ein Agent besser und ein anderer nicht schlechter gestellt wird. Das monotone Zugeständnisprotokoll geht nun wie folgt vor: Die Agenten beginnen damit, den für sie jeweils besten Deal aus der Menge der zulässigen vorzuschlagen. In der nächsten Runde ändert derjenige Agent seinen Vorschlag, der beim Rückfall auf die ursprüngliche Verteilung mehr verlieren würde. Im nächsten Vorschlag macht dieser Agent ein Zugeständnis für den anderen Agenten. Dies wird wiederholt, bis ein Agent den Vorschlag des anderen besser oder zumindest gleich gut wie den eigenen findet. Falls kein Zugeständnis möglich ist, dann wird die ursprüngliche Verteilung von Aufgaben realisiert. Eine kurze, aber gut verständliche Erläuterung findet man in [90].

14.3.5 Bildung von Koalitionen

Die Bildung von Koalitionen ist ein Problembereich, in dem sich egoistische Agenten zu einer Gruppe zusammen schließen, um ein Problem gemeinsam zu lösen. Eine Liste von Beispielen findet man in [80], z.B. eine Menge von Gemeinden, die gemeinsam einen Flughafen bauen könnten, der größer und effizienter ist, als wenn jede Gemeinde den eigenen Flughafen bauen würde. Wer beteiligt sich nun an dem gemeinsamen Flughafen, wer baut alleine bzw. wer baut mit wem? Eine vollständige Einteilung der Agenten in Gruppen nennt man auch Koalitionsstruktur. Die „Grand Coalition" – die große Koalition – ist die Koalitionsstruktur, bei der alle Agenten in einer Koalition zusammenarbeiten.

Im Prinzip gibt es drei Phasen bei der Lösung des Koalitionsproblems (siehe auch [75]): 1) das Generieren der Gruppenstruktur 2) das tatsächliche Bilden eines Teams aus den Mitgliedern der Koalition, mit Verteilung der Aufgaben und deren Lösung und 3) das Verteilen des Gewinns, den man in der Gruppe gemacht hat. Das Problem ist dabei, dass der Gewinn, den ein Agent durch die Problemlösung in einer Gruppe machen wird, notwendig ist, um zu entscheiden, ob der Agent an der Koalition teilnimmt oder nicht. Es kann also sehr aufwändig sein, eine mögliche Koalitionsstruktur zu evaluieren. Für die Verteilung des Gewinns gibt es verschiedene Ansatzpunkte und Bedingungen. Insgesamt muss der Gewinn so verteilt werden, dass keine Subgruppe eine Motivation hat, aus der Koalition auszubrechen. Der Shapley-Wert[4] bietet eine „faire" Verteilung des Koalitionsgewinns: Jeder Agent erhält die Summe, die seinem Grenznutzen für die Gemeinschaft entspricht. Je mehr der Agent beiträgt, umso mehr erhält er. Wichtige Bedingungen sind, dass Agenten, die austauschbar sind und gleiches beitragen, auch gleich viel aus dem Koalitionsgewinn erhalten oder dass sogenannte „Dummy" Agenten, deren Beitrag den Gesamtnutzen der Koalition nicht steigert, genau so viel erhalten, wie sie erhalten würden, wenn sie alleine arbeiten würden. Eine dritte Bedingung betrifft die Additivität der Bewertungsfunktion. Der Shapley-Wert für einen Agenten berechnet sich dann aus dem Gewinn der Koalition mit und ohne den Agenten und weitere Faktoren, um unterschiedliche Reihenfolgen des Hinzunehmens der Agenten in die Koalition abzudecken.

14.4 Entwicklung und Praxis

14.4.1 Agentenorientiertes Software Engineering

Eine zentrale Frage ist die, wie ein Multiagentensystem entwickelt werden kann. Dabei müssen im Endeffekt zwei Probleme gelöst werden: a) das Systemdesign und b) das Agentendesign. Bei ersterem muss das Gesamtsystem aus interagierenden Agenten so entwickelt werden, dass es bestimmte Anforderungen erfüllt. Das bedeutet, dass vom Multiagentensystem als Ganzes eine bestimmte Funktionalität in einer bestimmten Qualität bereitgestellt werden muss. Dabei müssen die Agenten entsprechend „organisiert" werden. Das zweite Problem betrifft die untere Ebene, die der Agenten: Wie muss ein einzelner Agent entwickelt werden, welche Funktionalität und Eigenschaften muss ein Agent durch sein Verhalten bereitstellen, um das Ziel des Gesamtsystems zu erreichen. Methoden und Sprachen für eine systematische Lösung dieser Probleme zu entwickeln, ist das Ziel des Agentenorientierten Software Engineering [10]. [8] und [46] sind

[4] Benannt nach Lloyd Shapley, der 2012 für diese Arbeiten den Nobel-Preis für Wirtschaftswissenschaften bekommen hat

Sammlungen mit Beschreibungen unterschiedlicher Methoden, die für die systematische Entwicklung von Multiagentensystemen vorgeschlagen wurden. Beide Kompendien enthalten auch Vergleiche zwischen den Methoden. Dies zeigt die Vielzahl vorhandener Ansätze. Bekannte, mehr oder weniger willkürlich ausgewählte Methodologien sind dabei im Folgenden ausgeführt. Weitere sind Prometheus [64] oder O-MaSE [28]. 2012 wurde bereits der 13te Internationale Workshop zum Agentenorientierten Software Engineering durchgeführt.

- GAIA [91] kann als die erste Methodologie zur Entwicklung von Multiagentensystemen betrachtet werden. Ausgehend vom Systemziel wird eine Organisation mit konzeptioneller Beschreibung von Rollen und Interaktionsprotokollen erstellt. Diese werden in weiteren Phasen konkretisiert. Agenten werden hinzugefügt, um diese Rollen ausfüllen zu können. Der Anwendungsbereich ist klar definiert als Systeme mit wenigen heterogenen und kooperierenden Agenten.
- TROPOS [16] legt einen besonderen Fokus auf das frühe Anforderungsmanagement als eine von fünf Phasen der Methodologie. Dort werden Akteure und ihre Abhängigkeiten untersucht und daraus funktionale und nicht-funktionale Anforderungen abgeleitet. Weitere Phasen sind ein verfeinertes Anforderungsmanagement, eine Designphase für die Gesamtsystemarchitektur und eine detaillierte Designphase für die Agenten. Eine fünfte und letzte Phase betrifft die Implementierung. Die wichtigsten Konzepte der dabei verwendeten Modellierungssprache sind unter anderem ein „Akteur“, der eine Entität mit strategischen Zielen darstellt, die Ziele selbst und die dabei auftretenden Abhängigkeiten.
- MAS-CommonKADS [53] hat seine Ursprünge im Wissensmanagement. Die Methodologie besteht aus fünf Phasen, von einer Konzeptionalisierungsphase als erstem Schritt zur Identifikation der funktionalen Anforderungen des Systems, über Analyse und Design zur Implementierung, Testen und zum tatsächlichen Einsatz. Verschiedene Aspekte des zu entwickelnden Multiagentensystems werden in einer Vielzahl von Modellen behandelt: Agentenmodell für die Eigenschaften und Aufbau der beteiligten Agenten; Taskmodell, das die Ziele und Methoden der Agenten strukturiert; das Expertisemodell für das notwendige Wissen; das Organisationsmodell für die Agenten-Gesellschaft; das Koordinationsmodell für die Interaktionen der Agenten untereinander; das Kommunikationsmodell für die Schnittstelle zum menschlichen Benutzer und zuletzt das Designmodell zur Konkretisierung des Netzwerkes, der Agenten und der verwendeten Plattform.
- MESSAGE [44] bieten einen Entwicklungsprozess auf der Basis des *Rational Unified Process*(RUP). Das Hauptaugenmerk liegt dabei auf der Erweiterung von UML durch Konzepte, die für ein Agentensystem relevant sind, wie Agent, Organisation, Rolle und Ziel. Es gibt zwei Typen von Aktivitäten: Aufgaben und Interaktionen/Interaktionsprotokolle; für jedes dieser Konzepte werden graphische Notationen in UML integriert. In einer Analysephase werden Ziele identifiziert, zerlegt und Rollen zugeordnet, Interaktionen werden beschrieben, die für das Erfüllen der Ziele notwendig sind. Abstrakte Modelle werden schrittweise verfeinert und in der Designphase zu Software-Elementen weiter verfeinert. INGENIAS [67] kann als Erweiterung von MESSAGE/UML gesehen werden. Es wird eine Umwelt-Sicht hinzugefügt und die Agentenmodelle stärker auf eine BDI-Architektur orientiert.
- Der PASSI [22] Prozess besteht aus fünf Elementen, die iterativ verfeinert werden: Systemanforderungsmodell, Agentengesellschaftsmodell, Agentenspezifikation, Implementierung und Verwendung. Bei jedem Schritt kommen entsprechende Beschreibungen in Standard UML zum Einsatz. Die Produktion von Programmcode wird unterstützt durch wiederverwendbare

Muster und Codegenerierung auf der Basis der standardisierten FIPA Multiagentensystem-Architektur.

- Adelfe [11] nimmt eine Sonderstellung im AOSE Bereich ein. Zum einen ist diese Methode deutlicher durch einer Bottom-up-Vorgehensweise geprägt als die anderen Ansätze, zum anderen zielt sie auf adaptive, lernfähige Agenten in offenen Multiagentensystemen ab. Basisprinzip ist das Konzept eines „Adaptiven Multiagentensystems“, das ein „Living Design“ ermöglicht: Agenten versuchen permanent kooperativ zu sein, was durch nicht-kooperative Situationen verhindert wird. Ein Beispiel ist eine Situation, in der ein Agent Informationen schickt, die der Empfänger nicht verstehen kann. Durch Adaptionen auf verschiedenen Ebenen – vom Parameter-Tuning zur Reorganisation –, lernen die Agenten solche Nicht-kooperativen Situationen zu vermeiden.

Aktuelle Entwicklungen gehen dahin, diese mehr oder weniger verschiedenen Ansätze miteinander zu kombinieren. Dazu werden die unterschiedlichen Methodologien in sogenannte Methodenfragmente zerlegt. Ein Fragment entspricht dabei einem Schritt in einem übergeordneten Prozess, mit dem z.B. eine bestimmte Sicht auf das Gesamtsystem spezifiziert wird oder mit dem Anforderungen auf die Agentenebene abgebildet werden. Diese Fragmente sollen mittels geeigneter Metamodelle verknüpft werden [9]. Sogar eine Selbstorganisation von Fragmenten ist angedacht [13]. Auf diese Weise können vollständige Methoden aus den Bausteinen zusammengesetzt werden, die für eine spezielle Anwendung am besten geeignet sind. Weitere, neuere Entwicklungen betreffen das modellbasierte Entwickeln von Multiagentensystemen [45]. Weitere Metamodelle sind FAML [12] oder [24]. Diese Metamodelle sollen als explizite Verallgemeinerung der verwendeten Abstraktionen die Grundlage für Schnittstellen zwischen Methodenfragmenten oder Softwaremodellen bilden.

14.4.2 Werkzeuge und Wettbewerbe

Seit 2002 gibt es eine eigene Serie von jährlichen Workshops, die sich ausschließlich mit Werkzeugen zur Programmierung von Agenten und Multiagentensystemen beschäftigt. Dies zeigt, dass es mittlerweile eine große Vielfalt von Werkzeugen und speziellen Programmiersprachen gibt. Es gibt viele Projekte, die die bekannten objekt-orientierte Programmiersprachen, wie z.B. `JAVA` oder `C++` benutzen. Besonders interessant für Multiagentensystem-Projekte ist die funktionale Programmiersprache `Erlang` für verteilte Prozesse (`www.erlang.org`). `Agent0` [79] gilt als erste Programmiersprache speziell für Agenten. Es gab eine Lisp-basierte Implementierung, von der heute keine funktionierende Version mehr gefunden werden kann. Das besondere an `Agent0` ist, dass es eine logikbasierte Programmiersprache war, deren Programme auf sogenannten Commitment-Regeln beruhen. Das sind Regeln, mit denen der Agent sich selbst oder anderen das Ausführen einer bestimmten Aktion zu einem bestimmten Zeitpunkt verspricht. Eine ausgefeiltere Programmiersprache für BDI-Agenten ist JASON [15], ein Interpreter für AgentSpeak(L) Programme. Einen Überblick über Programmiersprachen für Multiagentensysteme findet man unter anderem bei [14].

Es wurden mittlerweile auch viele Spezialwerkzeuge entwickelt. Einige der oben genannten Methodologien wurden mit Werkzeugen zur Spezifikation, Code Generierung oder für Simulation und Testen ausgestattet. Darüber hinaus gibt es viele Plattformen, die dem FIPA-Architekturmodell folgen und auch standardisierte Services für Agentenkommunikation bereitstellen. Die wohl bekannteste ist JADE (`http://jade.tilab.com/`, [7]). Das einfache

Agentenmodell wurde von L. Braubach und A. Pokahr [68] mit JADEX zu einer BDI Architektur erweitert.

Multiagentensimulation ist eine besondere Form der Anwendung, die mehr und mehr Beachtung findet. Laut der AgentLink III Roadmap [58] ist Simulation sogar das am besten etablierte Anwendungsgebiet von Multiagentensystemen. Ausgehend von Swarm (`www.swarm.org`) – das in den frühen 90igern für Simulationen von komplexen Systemen im Artificial Life Umfeld entwickelt wurde – haben sich eine Reihe von Simulationsplattformen etabliert. Die heute wichtigsten sind Repast (`repast.sourceforge.net/`) und NetLogo (`ccl.northwestern.edu/netlogo/`). Beim Einsatz im industriellen Umfeld haben Multiagentensysteme nach [69] den großen Vorteil, dass Agenten aus simulierten Umgebungen ohne großen Aufwand in die Realwelt überführt werden können. Dazu ist allerdings eine Simulationsplattform notwendig, die die Umwelt in ausreichendem Detaillierungsgrad repräsentieren kann, wie z.B. ihre AgentFly Plattform [81].

Information über Werkzeuge findet man insbesondere im Zusammenhang mit Wettbewerben, die dazu dienen die Leistungsfähigkeit von Werkzeugen, bzw. den damit erzeugten Artefakten zu vergleichen. Was 1997 mit dem RoboCup (`www.robocup.org`) und kleinen und großen fußballspielenden Robotern begann, hat in der Forschungslandschaft der Multiagentensysteme weite Kreise gezogen. Der RoboCup ist mittlerweile eine Sammlung von Wettbewerben, die nicht nur die Forschung im Bereich der komplexen, kooperierenden Roboter über Jahre angetrieben hat, sondern auch im Bereich des Mensch-Maschine Interaktion und Kooperation. Weitere Wettbewerbe folgten: Die „Trading Agent Competition“ (`http://www.sics.se/tac/`) für Agenten wurde 1999 initiiert und seit 2002 regelmäßig durchgeführt. Zu Beginn war es nur ein komplexes Auktionsszenario mit multiplen, von einander abhängigen Auktionen, um ein attraktives Reisepaket für einen Menge von Kunden zu schnüren. 2003 ist ein Supply Chain Scenario hinzugekommen. Mittlerweile gibt es weitere Szenarien mit Agenten, die möglichst gut handeln sollen. Der „Multiagent Programming Contest“ (`multiagentcontest.org/`) konzentriert sich seit 2005 auf interessante Herausforderungen zur Agentensteuerung und Koordination in abstrakten Benchmark-Szenarien. Ein wichtiger Aspekt ist dabei, dass nicht derjenige Teilnehmer mit der ausgefeiltesten Hardware gewinnt, sondern derjenige mit den besten Konzepten.

14.4.3 (Zu) kurzer Blick auf die Anwendungen

Schon vor zehn Jahren war es fast ein Ding der Unmöglichkeit, einen umfassenden Überblick über erfolgreiche Anwendungen von Multiagentensystemen zu geben. Einen recht kondensierten Überblick aus dieser Zeit findet man in [55].

Pěchouček und V. Mařík geben einen Überblick zur industriellen Anwendung von Multiagentensystemen und vier Fallstudien im Bereich der verteilten Kontrolle und Diagnose in technischen Systemen, der Produktionsplanung, der Luftverkehrskontrolle und Materialflusskontrolle. Passende Anwendungsgebiete für Multiagentensysteme sind nach [69]:

- Szenarien, in denen Daten und Wissen für die Problemlösung nicht an einem zentralen Ort vorhanden sind. Gründe dafür können in der geographischen Verteiltheit oder in Beschränkungen im Datenaustausch liegen, weil die Agenten in einer Konkurrenzsituation stehen.

- Szenarien, die besondere Robustheit und schnelle Reaktionen in einer verteilten Umgebung benötigen, z.B. in zeitkritischer Fertigung. In solchen Szenarien kann lokale Kontrolle oder lokales Umplanen sehr viel zur Robustheit des Gesamtsystems beitragen.
- Simulation und Modellierungszenarien, bei denen eine einfache Übertragbarkeit in die Realwelt gewünscht wird.
- Offene Systeme, die eine Integration und Interoperabilität zwischen Systemen erfordern, die nicht a priori bekannt sind, sondern erst während der Laufzeit hinzukommen.
- Komplexe Systeme generell, bei denen eine Aufteilung in Teilprobleme Sinn macht, die dann verteilt auf verschiedenen Agenten, mit Verhandlungen und gegebenenfalls parallel bearbeitet werden.
- Autonomie-orientierte Szenarien, bei denen ein Nutzer einen substantiellen Teil der Entscheidung an einen Agenten delegiert.

Diese Eigenschaften treffen auf viele System in unterschiedlichsten Domänen zu. Sie reichen vom Taxi-Management [43] zu neuen Systemkonzepten zur lokalen Energieversorgung (z.B. [88]), von innovativen Ampelsteuerungen (siehe [6]) zum elektronischem Handeln (z.B. [25]), vom Supply Chain Management ([20]) zur Produktionssteuerung ([65] als Pionier), von virtuellen Gegnern oder Helfern in Computerspielen [84] zu Sicherheit in Flughäfen [78], von selbstorganisierenden Sensornetzen (für eine Übersicht siehe [86]) zur Diagnose in Telekommunikationsnetzen (siehe [1], z.B. [42]). Dies ist nur eine winzige, zufällige Auswahl von erfolgreichen Projekten, die Multiagentsystemtechnologie anwenden. Für jedes dieser Gebiete ließe sich ein Übersichtsartikel finden/schreiben, der den Umfang dieses Kapitels übertrifft.

AgentLink (`www.agentlink.org`), das Europäische Network of Excellence of Agent-based Computing, schaffte es über fast zehn Jahre, weitgehend alle Europäischen Forschungsinstitute, die sich mit Multiagentensystemen beschäftigen, zu vereinigen. Am Ende der Laufzeit, 2006, wurde eine Roadmap[5] [58] publiziert, die Erreichtes dokumentierte und neue Trends identifizierte. Ein Schwerpunkt wurde dabei auch auf den industriellen Einsatz von agentenbasierter Technologie gelegt. Die Agentlink-Roadmap dokumentiert sehr einprägsam den Stand verschiedener agentenbasierter Technologien und Anwendungsgebiete. AgentLink sammelte zudem prominente Fallstudien für den industriellen Einsatz von Agententechnologie.

14.5 Aktuelle Trends

In diesem Kapitel sollte ein Überblick über Multiagentensysteme im Kontext der Künstlichen Intelligenz gegeben werden. Hierbei haben wir den Fokus hauptsächlich auf Interaktionen gelegt und weniger auf die Ideen, die möglicherweise zentraler für die Künstliche Intelligenz sind. Wir haben auf Themen aus dem Bereich der formalen Behandlung von Multiagentensysteme, z.B. Multiagentenlogiken, verzichtet. Mit dem Fortschritt bei der Entwicklung und Anwendung von Multiagentensystemen, wird die Frage nach der formalen Verifikation immer interessanter. Die Beweisbarkeit, dass ein Multiagentensystem wirklich die vorgegebenen Eigenschaften erfüllt, ist und bleibt eine wichtige Voraussetzungen, derart komplexe Systeme in der Realität auch in sicherheitskritischen Anwendungen einsetzbar zu machen. Ebenso haben wir den interessanten Bereich des Multiagentenlernens ignoriert, in dem Agenten voneinander, miteinander oder

[5] Die Roadmap ist immer noch von `www.agentlink.org` downloadbar.

in Konkurrenz zueinander lernen. Multi-Robot Systeme oder Multiagentensysteme aus Virtuelle Agenten haben in den letzten Jahren ihren Platz in der Mainstream Multiagentensystem-Forschung eingenommen, auch weil die technologischen Hürden durch kommerzielle Robotikplattformen, wie z.B. die Nao-Robots (`http://www.aldebaran-robotics.com`), ebenso wie effizientere und mächtigere Game Engines, wie z.B. die MASSIVE Plattform (`http://www.massivesoftware.com/`), niedriger geworden sind.

Eine interessante Forschungsrichtung, die wir ebenfalls kaum betrachtet haben, hat sich unter dem Schlagwort „Agreement Technology" formiert. Dabei sollen Agenten entwickelt werden, die auf menschlichem Niveau miteinander verhandeln, zu einer Einigung kommen und entsprechende Abmachungen treffen können. Forscher aus den Bereichen Normative Multiagentensysteme, Multiagentenorganisationen, Argumentation und Verhandlung und Technologien, die Vertrauen und Reputation zwischen und von Agenten unterstützen, sind an einer gleichnamigen COST-Aktion beteiligt (`http://www.agreement-technologies.org/`). Ein Buch, das die Ergebnisse dieser COST Aktion zusammenfasst, erscheint Ende 2012.

Literaturverzeichnis

[1] Albayrak, S., editor (1998). *Intelligent Agents For Telcommunications Application*. IOS Press.

[2] Austin, J. (1962). *How to do things with words*. Cambridge.

[3] Balmer, M., Nagel, K., und Raney, B. (2004). Large scale multi-agent simulations for transportation applications. *J. of Intelligent Transport Systems*, 8:205–223.

[4] Barbuceanu, M. und Fox, M. S. (1995). COOL: A language for describing coordination in multi agent systems. In *Proc. of the 1st. Int. Conf. on Multiagent Systems*, pages 17–24. AAAI.

[5] Bauer, B. und Odell, J. (2005). UML 2.0 and agents: how to build agent-based systems with the new UML standard. *Eng. Appl. Artif. Intell.*, 18(2):141–157.

[6] Bazzan, A. L. C. (2009). Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multi-Agent Systems*, 18(3):342–375.

[7] Bellifemine, F. L., Caire, G., und Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. John Wiley.

[8] Bergenti, F., Gleizes, M.-P., und Zambonelli, F., editors (2004). *Methodologies and Software Engineering For Agent Systems – The Agent-Oriented Software Engineering Handbook*. Kluwer Academic Publishing, New York.

[9] Bernon, C., Cossentino, M., Gleizes, M.-P., Turci, P., und Zambonelli, F. (2005a). A study of some multi-agent meta-models. In Odell, J., Giorgini, P., und Müller, J. P., editors, *Agent-Oriented Software Engineering V, 5th International Workshop, AOSE 2004, New York, NY, USA, July 19, 2004, Revised Selected Papers*.

[10] Bernon, C., Cossentino, M., und Pavon, J. (2005b). Agent oriented software engineering. *Knowledge Engineering Review*, 20:99–116.

[11] Bernon, C., Gleizes, M.-P., Peyruqueou, S., und Picard, G. (2003). Adelfe: A methodology for adaptive multi-agent systems engineering. In Petta, P., Tolksdorf, R., und Zambonelli, F., editors, *Engineering Societies in the Agents World III*, volume 2577 of *Lecture Notes in Computer Science*, pages 70–81. Springer Berlin / Heidelberg.

[12] Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomez-Sanz, J. J., Pavón, J., und Gonzalez-Perez, C. (2009). FAML: A generic metamodel for MAS development. *IEEE Transactions on Software Engineering*, 35(6):841–863.

[13] Bonjean, N., Gleizes, M.-P., Maurel, C., und Migeon, F. (2012). Forward self-combined method fragments. In *Proc. of the 13th International Workshop on Agent-Oriented Software Engineering, Valencia, 2012*.

[14] Bordini, R., Braubach, L., Dastani, M., Seghrouchni, A. E. F., Gomez-Sanz, J. J., Leite, J., G. O'Hare, G., Pokahr, A., und Ricci, A. (2006). A survey of programming languages and platforms for multi-agent systems. *Informatica*, 30:33–44.

[15] Bordini, R., Hübner, J., und Wooldridge, M. (2005). *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley.

[16] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., und Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Systems and Multi-Agent Systems*, 8:203–236.

[17] Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23.

[18] Camphell, A. und Wu, A. (2011). Multiagent role allocation: issues, approaches and multiple perspectives. *Autonomous Agents and Multiagent Systems*, 22:317–355.

[19] Carley, K. M. und Gasser, L. (1999). Computational organization theory. In Weiss, G., editor, *Multiagent systems*, pages 299–330. MIT Press, Cambridge, MA, USA.

[20] Chaib-Draa, B. und Müller, J. P., editors (2006). *Multiagent based Supply Chain Management*. Springer.

[21] Corkill, D. D. (1991). Blackboard systems. *AI Expert*, 6:40–47.

[22] Cossentino, M. (2005). From requirements to code with the PASSI methodology. In Henderson-Sellers, B. und Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 79–106. Idea Group.

[23] Cost, R. S., Chen, Y., Finin, T. W., Labrou, Y., und Peng, Y. (2000). Using colored petri nets for conversation modeling. In *Issues in Agent Communication*, pages 178–192, London, UK, UK. Springer-Verlag.

[24] da Silva, V. T., Choren, R., und de Lucena, C. J. P. (2008). Mas-ml: a multi-agent system modelling language. *Int. J. Agent-Oriented Softw. Eng.*, 2(4):382–421.

[25] Dasgupta, P., Narasimhan, N., Moser, L. E., und Melliar-Smith, P. M. (1999). MAgNET: mobile agents for networked electronic trading. *Knowledge and Data Engineering, IEEE Transactions on*, 11(4):509 –525.

[26] de Jong, S. und Tuyls, K. (2011). Human-inspired computational fairness. *Autonomous Agents and Multiagent Systems*, 22:102–126.

[27] Decker, K. S. und Lesser, V. R. (1992). Generalizing the partial global planning algorithm. *Int. Journal of Intelligent and Cooperative Information Systems*, 1:319–346.

[28] DeLoach, S. A. und Garcia-Ojeda, J. C. (2010). O-MaSE: a customizable approach to designing and building complex, adaptive multiagent systems. *Int. Journal of Agent-Oriented Software Engineering*, 4:244–280.

[29] Dignum, V. (2004). *A Model for Organizational Interaction: Based on Agents, Founded in Logic*. PhD thesis, Utrecht University.

[30] Dorigo, M. und Stützle, T. (2004). *Ant Colony Optimization*. MIT Press.

[31] Drogoul, A. und Dubreuil, J. (1992). Eco-problem solving: Results of the n-puzzle. In Demanzeau, Y. und Werner, E., editors, *Decentralized AI III*. North Holland.

[32] Durfee, E. H. (1999). Distributed problem solving and planning. In Weiss, G., editor, *Multiagent systems*, pages 121–164. MIT Press, Cambridge, MA, USA.

[33] Durfee, E. H. und Lesser, V. R. (1991). Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:1187–1183.

[34] Durfee, E. H. und Rosenschein, J. S. (1994). Distributed problem solving and multi-agent systems: Comparisons and examples. In *AAAI Technical Report WS-94-02*, pages 52–62.

[35] Eisenhardt, K. M. (1989). Agency theory: An assenssment and review. *The Academy of Management Review*, 14:57–74.

[36] Ferber, J. (1999). *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*. Addison Wesley Longman.

[37] Ferber, J., Gutknecht, O., und Michel, F. (2004). From agent to organizations: an organizational view of multi-agent systems. In Giorgini, P., Müller, J., und Odell, J., editors, *Agent-Oriented Software, AOSE IV, Melbourne 2003*, volume LNCS 2935, pages 214–230. Springer.

[38] Ferber, J. und Müller, J.-P. (1996). Influences and reactions: a model of situated multiagent systems. In *Proc. of the ICMAS'96, International Conference on Multi-Agent Systems*. AAAI Press.

[39] Ferguson, I. A. (1992). Towards an architecture for adaptive, rational, mobile agents. In Werner, E. und Demazeau, Y., editors, *Decentralized A.I. 3 – 3rd European Workshop on Modellings Autonomous Agents in a Multi-Agent World, MAAMAW'91*, pages 249–263. North Holland.

[40] Finin, T., Fritzson, R., McKay, D., und McEntire, R. (1994). KQML as an agent communication language. In *Proc. of the 3rd Int. Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463. ACM Press.

[41] Franklin, S. und Graesser, A. (1997). It is an agent, or just a program? a taxonomy for autonomous agents. In Jennings, N. und Wooldridge, M., editors, *Intelligent Agents*, volume III. Springer.

[42] Garcia-Gomez, S., Gonzalez-Ordas, J., Garcia-Algarra, F. J., Toribio-Sardon, R., Sedano-Frade, A., und Buisan-Garcia, F. (2009). KOWLAN: A multi agent system for bayesian diagnosis in telecommunication networks. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09*, pages 195–198.

[43] Glaschenko, A., Ivaschenko, A., Rzevski, G., und Skobelev, P. (2009). Multi-agent real time scheduling system for taxi companies. In Decker, K., Sichman, J., Sierra, C., und Castelfranchi, C., editors, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, May 2009*, pages 29–36.

[44] Gómez-Sanz, J. J. und Pavón, J. (2002). Agent oriented software engineering with message. In Giorgini, P., Lespérance, Y., Wagner, G., und Yu, E. S. K., editors, *AOIS '02, Agent-Oriented Information Systems, Proceedings of the Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002 at CAiSE*02)*.

[45] Hahn, C., Madrigal-Mora, C., und Fischer, K. (2009). A platform independent meta model for multi-agent systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 18(2):239–266.

[46] Henderson-Sellers, B. und Giorgini, P. (2005). *Agent-Oriented Methodologies*. IDEA Group, Hershey.

[47] Hewitt, C. und Inman, J. (1991). DAI betwixt and between: from 'intelligent agents' to open systems science. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(6):1409–1419.

[48] Holland, J. H. (1999). *Emergence: From Chaos to Order*. Helix Books.

[49] Horling, B. und Lesser, V. (2004). A survey of multi-agent organizational paradigms. *Knowledge Engineering Review*, 19(4):281–316.

[50] Hübner, J. F., Boissier, O., Kitio, R., und Ricci, A. (2010). Instrumenting multi-agent organizations with organisational artifacts and agents: "giving the organisational power back to the agents". *Autonomous Agents and Multi-Agent Systems*, pages 369–400.

[51] Hübner, J. F., Sichman, J. S., und Boissier, O. (2002). A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence*, SBIA '02, pages 118–128, London, UK, UK. Springer-Verlag.

[52] Huhns, M. N. und Singh, M. P., editors (1999). *Readings in Agents*. Morgan Kaufmann.

[53] Iglesias, C. A. und mercedes Garijo (2005). The agent-oriented methodology MAS-CommonKADS. In Henderson-Sellers, B. und Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 46–78. Idea Group Publishers.

[54] Ingrand, F. F., Georgeff, M. P., und Rao, A. S. (1992). An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6):34–44.

[55] Klügl, F. (2004). Applications of software-agents. *Künstliche Intelligenz*, 18:5–10.

[56] Kraus, S. (2001). Automated negotiation and decision making in multiagent environments. In *Multiagent Systems and Applications, Selected Tutorial Papers*, pages 150–172. Springer-Verlag.

[57] Labrou, Y. (2001). Standardizing agent communication. In Luck, M., Marik, V., Stepankova, O., und Trappl, R., editors, *Multiagent Systems and Applications, Selected Tutorial Papers*, LNAI 2086, pages 74–91, Berlin, Heidelberg. Springer.

[58] Luck, M., McBurney, P., Shehory, O., und Willmott, S. (2005). *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink.

[59] Maes, P. (1992). Behavior-based artificial intelligence. In Meyer, J.-A. und Wilson, S. W., editors, *From Animals to Animats 2: 2nd Conference on the Simulation of Adaptive Behavior*, pages 2–10. MIT Press.

[60] Mao, X. und Yu, E. (2005). Organizational and social concepts in agent oriented software engineering. In *Proceedings of the 5th international conference on Agent-Oriented Software Engineering*, AOSE'04, pages 1–15, Berlin, Heidelberg. Springer-Verlag.

[61] Müller, J. P. (1996). *The Design of Intelligent Agents – A Layered Approach*, volume 1177 of *LNAI*. Springer.

[62] Nilsson, N. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1:139–158.

[63] Odell, J., Parunak, H. V. D., und Bauer, B. (2000). Extending UML for agents. In *Proc. of the Agent-Oriented Information Systems, Workshop at the 17th National Conference on Artificial Intelligence*, pages 3–17.

[64] Padgham, L. und Winikoff, M. (2005). Prometheus: A practial agent-oriented methodology. In Henderson-Sellers, B. und Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 107–135. Idea Group Inc.

[65] Parunak, H. V. D. (1987). Manufacturing experience with the contract net. In Huhns, M. N., editor, *Distributed Artificial Intelligence*, pages 285–310.

[66] Patil, R. S., Fikes, R. E., Patel-Schneider, P. F., MacKay, D., Finin, T., Gruper, T., und Neches, R. (1992). The DARPA knowledge sharing effort: Progress report. In *Proceedings of the 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 777–787. Reprinted in M. Huhns and M. Singh. Readings in Agents, 1999.

[67] Pavón, J., Gómez-Sanz, J. J., und Fuentes, R. (2005). The INGENIAS methodology and tools. In Henderson-Sellers, B. und Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 236–276. Idea Group Publishing.

[68] Pokahr, A. und Braubach, L. (2009). From a research to an industrial-strength agent platform: Jadex v2. In Hansen, H. R., Karagiannis, D., und Fill, H.-G., editors, *Business Services: Konzepte, Technologien, Anwendungen – 9. Internationale Tagung Wirtschaftsinformatik (WI 2009)*, pages 769–778. Österreichische Computer Gesellschaft.

[69] Pěchouček, M. und Mařík, V. (2008). Industrial deployment of multi-agent technologies: review and selected case studies. *Autonomous Agents and Multi-Agent Systems*, 17(3):397–431.

[70] Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away: agents breaking away*, MAAMAW '96, pages 42–55. Springer-Verlag New York, Inc.

[71] Rao, A. S. und Georgeff, M. P. (1991). Modeling rational agents within a bdi-architecture. In Allen, J. F., Fikes, R., und Sandewall, E., editors, *Proc. of the 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22–25, 1991*, pages 473–484. Morgan Kaufmann.

[72] Rosenschein, J. S. und Zlotkin, G. (1994). *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT Press.

[73] Rossi, D., Cabri, G., und Denti, E. (2000). Tuple-based technologies for coordination. In Omicini, A., Zambonelli, F., Klusch, M., und Tolksdorf, R., editors, *Coordination of Internet Agents: Models, Technologies and Applications*. Springer, Berlin.

[74] Russell, S. und Norvig, P. (1995). *Artificial Intelligence – A Modern Approach*. Prentice Hall, 3rd edition.

[75] Sandholm, T. (1999). Distributed rational decision making. In Weiss, G., editor, *Multiagent systems*, pages 201–258. MIT Press, Cambridge, MA, USA.

[76] Schoppers, M. (1987). Universal plans for reactive robots in unpredictable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-87, 1987*.

[77] Searle, J. (1962). *Speech Acts*. Cambridge.

[78] Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., und Meyer, G. (2012). PROTECT: a deployed game theoretic system to protect the ports of the united states. In van der Hoek, W., Padgham, L., Conitzer, V., und Winikoff, M., editors, *Int. Conf. on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 2012*, pages 13–20. IFAAMAS.

[79] Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, 60:51–92.

[80] Shoham, Y. und Leyton-Brown, K. (2009). *Multiagent Systems – Algorithmic, Game-Theoretic and Logical Foundations*. Cambridge University Press.

[81] Sislak, D., Volf, P., Kopriva, S., und Pechoucek, M. (2012). *AgentFly: Scalable, High-Fidelity Framework for Simulation, Planning and Collision Avoidance of Multiple UAVs*. John Wiley&Sons.

[82] Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput.*, 29(12):1104–1113.

[83] ter Mors, A., Yadati, C., Witteveen, C., und Zhang, Y. (2010). Coordination by design and the price of autonomy. *Autonomous Agents and Multiagent Systems*, 20:308–341.

[84] Torres, D. (2008). On virtual environments and agents in next-generation computer games. *The Knowledge Engineering Review*, 23:389–397.

[85] Vazquez-Salceda, J., Dignum, V., und Dignum, F. (2005). Organizing multiagent systems. *Autonomous Agents and Multiagent Systems*, 11:307–360.

[86] Vinyals, M., Rodriguez-Aguilar, J. A., und Cerquides, J. (2005). A survey on sensor networks from a multiagent perspective. *The Computer Journal*, 54:455–470.

[87] Vlassis, N. (2007). *Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence*. Synthesis Lectures in Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

[88] Vytelingum, P., Ramchurn, S. D., Voice, T. D., Rogers, A., und Jennings, N. R. (2010). Trading agents for the smart electricity grid. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, Toronto, Canada*, pages 897–904.

[89] Weiss, G., editor (1999). *Multiagent Systems – A Modern Appoarch to Distributed Artificial Intelligence*. MIT Press, Cambridge, USA.

[90] Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. John Wiley, Chichester, 2nd edition.

[91] Zambonelli, F., Jennings, N. R., und Wooldridge, M. (2005). Multiagent systems as computational organisations: the Gaia methodology. In Henderson-Sellers, B. und Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 136–171. Idea Group Publishers.

[92] Zambonelli, F. und Parunak, H. V. D. (2003). Signs of a revolution in computer science and software engineering. In *Proceedings of the 3rd international conference on Engineering societies in the agents world III*, pages 13–28, Berlin, Heidelberg. Springer-Verlag.