Dezentrale Steuerung verteilter Anwendungen mit rationalen Agenten

Alexander Pokahr, Lars Braubach, Winfried Lamersdorf

Univ. Hamburg, FB Informatik, Verteilte Systeme und Informationssysteme
Vogt-Kölln-Str. 30, 22527 Hamburg
{pokahr, braubach, lamersd}@informatik.uni-hamburg.de

Abstract Herkömmliche Methoden für die Steuerung und Koordination verteilter Anwendungen mit weitgehend autonomen Diensten und Prozessen in heterogenen, sich dynamisch ändernden Umgebungen beruhen oft auf *zentralen* Steuerungskomponenten und *statischen* Zustands- und Prozessbeschreibungen. Sie berücksichtigen damit sowohl die Autonomie der Teilprozesse als auch die Dynamik des Anwendungskontextes noch zu wenig. Die *Agententechnologie* wird als ein viel versprechender Ansatz gesehen, Probleme der Konstruktion komplexer verteilter Softwaresysteme in heterogenen Systemumgebungen mit *dezentralen* Steuerungsstrukturen besser beherrschbar zu machen. Durch die naturanaloge Aufteilung des Gesamtproblems in autonome, miteinander interagierende Einheiten lassen sich derartige verteilte Anwendungen adäquat modellieren, simulieren und auch implementieren. Eine *systemtechnische Unterstützung* solcher Anwendungen hat zum Ziel, die aus autonomen Teilaufgaben bestehenden Teilfunktionen und -prozesse rechnergestützt ausführbar und kontrollierbar zu machen. Grundlage dafür sind Middleware-Standards für die Agentenkommunikation sowie entsprechende standardkonforme Plattformen. Zur Realisierung dezentraler Steuerungsprozesse in derartigen Umgebungen ist darüberhinaus die Unterstützung der *Entscheidungsprozesse* (Reasoning) der einzelnen als Agenten modellierten Komponenten besonders wichtig, da diese mangels globaler Sicht oft allein auf Basis lokaler Informationen handeln müssen. Vor diesem Hintergrund stellt das Papier eine im Rahmen eines laufenden DFG-Schwerpunktprogramms entwickelte, agentenbasierte Systemplattform vor, die sowohl die dezentrale Steuerung verteilter Anwendungen als auch die Dynamik der Anwendungsdomäne unter Verwendung *rationaler Agenten* systemtechnisch unterstützt und so die Vorteile erweiterter Middleware-Funktionen mit denen eines agentenorientierten Reasoning-Ansatzes verbindet.

1 Einleitung

Agententechnologie wird heutzutage als ein vielversprechender Ansatz gesehen, Probleme der Konstruktion komplexer Softwaresysteme besser beherrschbar und systemtechnisch umsetzbar zu machen. So kann Agententechnologie z.B. die Modellierung komplexer verteilter Problemstellungen inkl. ihrer Umsetzung in verteilte Softwarelösungen erleichtern, da dabei abstrakte naturanaloge Konzepte als Grundlage einer anwendungsadäquaten Problemlösung verwendet werden. Aufgrund des höheren Abstraktionsgrades bei der agentenorientierten Modellierung solcher Anwendungen

können so auch Problemstellungen adressiert werden, die mit herkömmlichen Techniken - wie z.B. der objektorientierten Programmierung - nur schwer handhabbar sind. Als Beleg für diese These können nach Jennings [9] eine Reihe von Argumenten angeführt werden:

1. Agentenorientierte Dekompositionen sind ein effektiver Weg den Problemraum eines komplexen Systems zu partitionieren.
2. Die Schlüsselabstraktionen der agentenorientierten Denkweise sind ein natürliches Mittel, komplexe Systeme zu modellieren.
3. Die agentenorientierte Philosophie zum Modellieren und Verwalten organisationeller Beziehungen ist dazu geeignet, mit den in komplexen Systemen existierenden Abhängigkeiten und Interaktionen besser umzugehen.

Technisch gesehen sind Agenten Softwarebausteine, die über Eigenschaften verfügen, die sie von anderen Komponenten wie z.B. Objekten unterscheiden [11]. Die wichtigsten Eigenschaften sind dabei: Die *Autonomie*, d.h. die Fähigkeit selbstständig und ohne Benutzerinteraktion über einzelne Aktivitäten zu entscheiden, die *Reaktivität*, d.h. die Fähigkeit, die Umgebung wahrzunehmen, und adaptiv auf Ereignisse zu reagieren, sowie die Fähigkeit, ausgehend von abstrakt vorgegebenen Zielen zu jedem Zeitpunkt die jeweils am besten geeignete Handlungsalternative auszuwählen (*zielgerichtetes Verhalten*) und dabei, wenn nötig, auch selbst die Initiative ergreifen zu können (*Proaktivität*). Interaktion zwischen Agenten erfolgt über Nachrichtenaustausch, wobei die Bedeutung des Nachrichteninhalts mittels Ontologien und Sprechakten unabhängig vom konkreten Sender und Empfänger festgelegt wird.

Im Folgenden werden nun Besonderheiten der *agentenorientierten Entwicklung verteilter Softwareanwendungen* näher betrachtet. Dabei werden insbesondere zwei Aspekte der Agententechnologie untersucht: Zunächst werden in Abschnitt 2 Middleware-Standards und konforme Plattformen für Agenten als bereits vorhandene systemtechnische Basis für die Realisierung verteilter Agentenapplikationen vorgestellt. Darauf folgend wird in Abschnitt 3 ein kurzer Überblick über verschiedene Reasoning-Mechanismen für Agenten gegeben. In Abschnitt 4 wird mit der selbst entwickelten Plattformerweiterung Jadex vorgeschlagen, eine direkte Verbindung zwischen Agenten-Middleware und Reasoning herzustellen. Anhand eines Fallbeispieles aus dem Bereich Krankenhauslogistik wird dann in Abschnitt 5 die praktische Anwendbarkeit des Ansatzes aufgezeigt. Kapitel 6 enthält eine Einordnung des Ansatzes in den Kontext verwandter Arbeiten bevor das Papier mit einer Zusammenfassung schließt.

2 Middleware für Agenten

Die Agententechnologie erweitert die Modellierungskonzepte verteilter Systeme und erfordert aufgrund des Paradigmenwechsels hin zu autonomen Softwarebausteinen in offenen verteilten Umgebungen sowohl neue Standards (um die Interoperabilität sicher zu stellen) als auch neue Middleware-Produkte (als systemtechnische Unterstützung um diese Standards umsetzen). Agenten können dabei als Softwarebausteine gesehen werden, die sich auf der Anwendungsebene befinden und mittels Middleware Zugriff auf standardisierte systemtechnische Dienste und Leistungen erhalten.

Für die Interoperabilität von Agenten verschiedener heterogener Plattformen ist derzeit vor allem die Foundation for Intelligent Physical Agents (FIPA) [14] wichtig - eine internationale Non-profit-Organisation, die sich zur Aufgabe gemacht hat, Standards für heterogene interagierende Agenten und Multiagentensysteme (MAS) bereitzustellen. Seit 1997 wurden dazu eine Reihe von Spezifikationen veröffentlicht, die in unregelmäßigen Abständen ersetzt oder ergänzt werden. Dabei konzentrieren sich die Arbeiten auf einerseits Anwendungs- und andererseits Middleware-bezogene Aspekte: Die anwendungsbezogenen Spezifikationen bieten systematisch untersuchte Beispieldomänen mit Dienst- und Ontologiebeschreibungen; die Middleware-bezogenen Spezifikationen behandeln detailliert alle notwendigen Bausteine zur Ausgestaltung einer abstrakten Plattformarchitektur. Dazu gehören sowohl die Mechanismen zur Agentenverwaltung auf der Plattform, als auch Infrastrukturelemente wie Verzeichnisdienste und der Nachrichtentransport. Daneben existieren umfangreiche Spezifikationen auf semantischer und syntaktischer Ebene, welche die Kommunikation zwischen Agenten auf eine einheitliche Basis stellen. Die FIPA-Standards wurden bereits in einer Reihe von Agentenplattformen umgesetzt, welche auch prinzipiell interoperabel sind. Zusätzlich zu den oben aufgeführten Spezifikationsthemen greifen viele dieser Plattformen weitere Middleware-Aspekte auf und bieten spezialisierte Lösungen für z.B. Sicherheits- und Persistenzproblematiken. FIPA-kompatible Middleware-Plattformen ermöglichen es daher, interoperable, offene und zukunftsfähige Systeme zu erstellen.

Dennoch decken weder FIPA-Standards noch die Middleware-Plattformen alle für die Kostruktion verteilter Anwendungen wichtigen Aspekte ab: So erfordern komplexe verteilte Anwendungen häufig nicht nur eine Dekomposition des Gesamtproblems in kleinere Teilprobleme, sondern zusätzlich auch eine adäquate Adressierung der oftmals problemimmanenten Dynamik der Anwendungsdomäne. Nur unter Berücksichtigung dieser Dynamik sowohl auf Modellierungs- als auch auf Realisierungsebene können flexible, adaptive Systeme erstellt werden, die den Ansprüchen komplexer Aufgabenstellungen gerecht werden. Dabei erfordert die für viele verteilte Anwendungen typische Dynamik insbesondere Mechanismen zur Entscheidungsfindung auf Basis der jeweiligen Situation der einzelnen Agenten. Neben der von der Middleware erbrachten Abstraktion von generischen Problemen der Verteilung wird damit also zusätzlich auch eine Abstraktion von Problemen der einzelnen Entscheidungsprozesse notwendig. Für die Beschreibung dieser Entscheidungsprozesse existieren verschiedene Modelle für so genannte *rationale Agenten*, auf die im nächsten Abschnitt näher eingegangen wird.

3 Rationale Agenten

Als rationale Agenten werden Agenten angesehen, die ihre Aktionen auf Grundlage der verfügbaren Informationen fortlaufend neu abwägen und auf Basis kognitiver Fähigkeiten selbständig Entscheidungen treffen können. Dabei beruhen die kognitiven Architekturen auf Theorien zur Beschreibung von Individualverhalten, die ihre Wurzeln in unterschiedlichen Disziplinen wie z.B. der Psychologie, Philosophie oder Biologie haben. Die wichtigsten Konzepte sind in diesem Bereich das Modell Belief-Desire-Intention (BDI) [2,15], die Theorie des Agent Oriented Programming (AOP) [16] und die Unified Theories of Cognition (UTC, SOAR) [10]. Im Folgenden wird speziell

auf das BDI-Modell näher eingegangen. Es ist allgemein anerkannt und aufgrund der konzeptuellen Klarheit auch im Bezug auf praktisch eingesetzte Softwaresysteme interessant [8].

Bratman [2] führt im BDI-Modell die Attitüden *Beliefs* (Wissen), *Desires* (Wünsche) und *Intentions* (Absichten) als elementare Bausteine zielgerichteten Handelns ein. Dabei repräsentieren *Beliefs* das subjektiv wahrgenommene Wissen eines Agenten über sich und seine Umwelt und spiegeln somit seine lokal verfügbaren Informationen wider. Diese Informationen sind jedoch kein direktes Abbild der aufgenommenen Reize, sondern zeichnen sich vielmehr durch eine domänenabhängige Abstraktion der Elemente aus, die wichtige Eigenschaften betont und weniger wichtige ausblendet. Dieser Filter- und Interpretationsmechanismus ermöglicht dem Agenten eine persönliche Weltsicht.

Desires repräsentieren die Wünsche eines Agenten, welche die Richtung seines Handelns maßgeblich steuern. Dabei müssen Desires keine konsistente Menge untereinander verträglicher Zielvorgaben sein, sondern können konfligieren und damit nicht gleichzeitig erreichbar sein. Eine konsistente Menge konkreter Ziele wird oft mit dem Begriff *Goals* bezeichnet. Bei einem zielorientierten Vorgehen werden Ziele explizit durch die Angabe von z.B. einem zu erreichenden Zustand formuliert. Dadurch kann nach der Ausführung von Aktionen überprüft werden, ob ein Ziel bereits erreicht wurde, oder Handlungsalternativen gesucht werden müssen. Mit Hilfe von Zielen wird ein Agent in die Lage versetzt proaktiv im Sinne seiner Vorstellungen vorzugehen, anstatt nur auf Ereignisse reagieren zu können.

Ziele versucht ein Agent durch vorgegebene Handlungsfolgen zu erreichen, die in Plänen formuliert werden. Entschließt sich ein Agent ein bestimmtes Ziel mit Hilfe eines Planes zu verfolgen, legt er sich auf die Art und Weise der Zielerfüllung fest. Dieses Festlegen bezüglich der aktuell eingeschlagenen Handlungsstränge wird *Intention* genannt und bildet die dritte Säule des BDI-Modells. In der informatischen Sicht sind die Pläne von vorrangigem Interesse, da sie das Fundament des möglichen Verhaltens bilden. Die Flexibilität im Handeln entsteht bei BDI-Agenten durch die Kombination zweier Facetten. Einerseits ist dies die dynamische Auswahl von Plänen in Bezug auf ein zu erreichendes Ziel. Durch den Prozess des Meta-Level Reasoning kann unter Einbezug auch von Domänenwissen und der aktuellen Informationslage der jeweils am besten geeignetste Plan ausgewählt werden. Scheitert dieser Plan, kann der Auswahlprozess auf Basis der neuen Informationen wiederholt werden, um einen weiteren Plan zu probieren. Andererseits kann die Ausgestaltung der Pläne je nach Bedarf abstrakter oder konkreter ausfallen. Einfache Aktionen können zur Planspezifikation ebenso verwendet werden wie Unterziele, über deren Verfolgungsstrategie erst zur Laufzeit durch das Meta-Level Reasoning entschieden wird.

Grundlage für die meisten Implementierungen von BDI-Systemen ist dabei der abstrakte Interpreter nach Rao und Georgeff (siehe Listing 1). In diesem werden zu Beginn jedes Schleifendurchlaufs die zu einem aufgetretenen Ereignis prinzipiell passenden Pläne bestimmt; danach wird aus diesen verfügbaren Optionen eine Untermenge selektiert (Meta-Level Reasoning), diese der Intentionenstruktur hinzugefügt und schließlich ausgeführt. Alle bis zu diesem Zeitpunkt aufgetretenen externen Ereignisse werden im nächsten Schritt in die Ereignisliste aufgenommen. Abschließend werden die Ziel- und Intentionenstrukturen des Agenten aktualisiert indem erfüllte oder unmögliche gewor-

```
BDI-interpreter
Initialize-state();
repeat
   options := option-generator(event-queue);
   selected-options := deliberate(options);
   update-intentions(selected-options);
   execute();
   get-new-external-events();
   drop-successful-attitudes();
   drop-impossible-attitudes();
end repeat
```

Listing 1. Abstrakter BDI Interpreter aus [15]

dene Attitüden entfernt werden. In den umgesetzten Systemen existieren zum Teil erhebliche Unterschiede - vor allem hinsichtlich der softwaretechnischen Ausgestaltung der zu Grunde liegenden Konzepte (Beliefs, Goals, Plans). Im Folgenden wird eine konkrete Umsetzung dieser Konzepte im Rahmen einer Erweiterung der standardkonformen Agentenplattform JADE näher beschrieben.

4 Jadex: Eine standardardkonforme BDI-Plattform

Die vorangegangenen Abschnitte motivieren den Bedarf nach einer Agentenplattform, die neben der Interoperabilität auf einer abstrakten Agentenebene durch standardisierte Kommunikation (z.B. nach FIPA) auch die Entwicklung rationaler Agenten durch eine systemtechnische Unterstützung der BDI-Konzepte ermöglicht. Gerade im Bereich der verteilten, aus autonomen Teildiensten bestehenden Anwendungen in dynamischen, d.h. sich laufend ändernden, Umgebungen ist die ausreichende Unterstützung *beider* Aspekte Voraussetzung für eine problemadäqute Anwendungsentwicklung. Im Projekt Jadex [13,3] wird daher der Ansatz verfolgt, bestehende agentenorientierte Middleware-Plattformen um die ihnen bisher fehlenden Konzepte rationaler Agenten zu erweitern. Damit wird ein generisches Framework bereitgestellt, mit dessen Hilfe verteilte Softwaresysteme in unterschiedlichen Anwendungsdomänen unter Verwendung des Agentenparadigmas realitätsnah modelliert und effizient umgesetzt werden können.

Bei der Konzeption wurde deshalb darauf geachtet, den mit dem des BDI-Modell erreichten Abstraktionsgrad auch auf Implementierungsebene möglichst direkt zu unterstützen, um so Anwendungen in möglichst natürlicher d.h. der menschlichen Denkweise ähnlicher Art und Weise modellieren zu können: Anstatt starrer, komplexer Abläufe mit fest vorgegebenen Vorgängen sollen hier nur relativ einfache, in sich geschlossene Funktionalitäten realisiert werden, die die allgemeinen Handlungsmöglichkeiten eines Agenten darstellen. Die Anwendbarkeit dieser Pläne für spezielle Situationen kann dabei über Vorbedingungen näher spezifiziert werden. Derartige Pläne erreichen damit nicht nur einen hohen Grad an Wiederverwendbarkeit, sondern können zudem aufgrund ihrer geringen technischen Komplexität von Domänenexperten verstanden werden. Abstrakte Ziele bilden die Motivation für die Ausführung von Plänen. Sie beschreiben einerseits die im System auftretenden Ereignisse, auf die

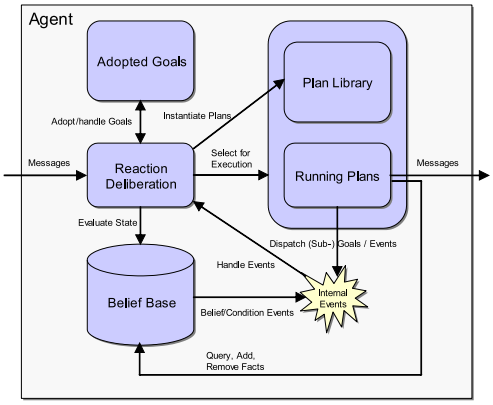

Abb. 1. a: Jadex Architektur

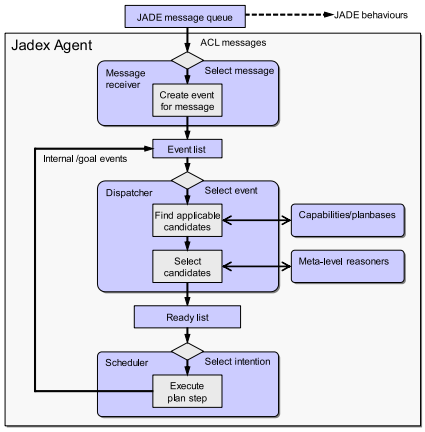

Abb. 1. b: Jadex Verarbeitungszyklus

von Agenten reagiert werden muss. Andererseits dienen sie auch der Festlegung allgemeiner Rahmenbedingungen, die vom System automatisch überwacht und im Fehlerfall durch geeignete Maßnahmen wiederhergestellt werden. Das notwendige Wissen zur Ausführung der Pläne und Auswertung von Zielen und Bedingungen wird dabei in der Wissensbasis eines Agenten gehalten und repräsentiert. Unterschiedliche Akteure bzw. Rollen mit spezifischen Zielen, Handlungsmöglichkeiten und Wissen können so auf natürliche Weise durch verschiedene Agenten repräsentiert werden.

4.1 Architektur des Systems Jadex

Abbildung 1.a gibt einen Überblick über die generelle Architektur des auf den oben genannten Grundsätzen realisierten Jadex-Systems: Vom außen betrachtet ist ein Agent eine Black-Box, die lediglich durch Nachrichten mit der Außenwelt kommuniziert und interagiert. Herzstück jedes Agenten ist der Mechanismus zur Reaktion und Deliberation. Ein Agent reagiert auf externe Ereignisse (z.B. Nachrichten anderer Agenten) sowie interne Ereignisse (wie neue Ziele) indem er geeignete Pläne (*Plans*) aus einer entsprechenden Bibliothek auswählt und zur Ausführung bringt. Die Auswahl und Ausführung der Pläne geschieht dabei auf Basis der aktuellen Überzeugungen (*Beliefs*) des Agenten, so dass vor der Ausführung eines Plans überprüfbar ist, ob ein Plan in der aktuellen Situation überhaupt angewendet werden kann, und nach der Ausführung, ob das gewünschte Ergebnis eingetreten ist. Ein Agent ist damit sowohl zu *reaktivem* Verhalten als auch zu *proaktivem* Verhalten, d.h. der Ausführung von Plänen zum Erreichen der eigenen Ziele (*Goals*), fähig.

Der grundlegende Reaktions- und Deliberationsmechanismus wird in Jadex von der sog. *Reasoning Engine* bereitgestellt und ist damit für alle Agenten identisch. Das spezifische Verhalten eines Agenten jedoch wird ausschließlich durch seine konkreten Beliefs, Goals und Plans bestimmt. Im Folgenden wird deshalb näher betrachtet, wie diese drei fundamentalen Konzepte eines BDI-Agenten im Jadex-System realisiert wurden.

Überzeugungen (Beliefs) Im Sinne einer einfachen Verwendbarkeit und Erlernbarkeit verzichtet Jadex auf die (in anderen BDI-Realisierungen übliche) Repräsentation von Beliefs in einer formalen logikbasierten Sprache. Um die Verbindung der objektorientierten Programmierung mit der agentenorientierten Programmierung für den Entwickler so einfach wie möglich zu gestalten, können beliebige Java-Objekte als Beliefs eines Agenten verwendet werden. Damit ist es möglich, von Werkzeugen wie Datenbankschichten und Ontologieentwurfswerkzeugen generierte Klassen direkt im Agenten weiterzuverwenden.

Die Reasoning Engine verwendet die Beliefs als Grundlage zur kontextsensitiven Auswertung von Vor- und Nachbedingungen von Plänen und Zielen. Dazu überwacht die Engine die Beliefs in Bezug auf relevante Änderungen und sorgt automatisch dafür, dass sich diese in den laufenden Zielen und Plänen widerspiegeln. Der Vorteil dieses Modells liegt in der Entkopplung der Manipulation von Daten (Beliefs) an einer Stelle (z.B. in einem Plan) und der dadurch ausgelösten Anpassung des Agenten (durch Erzeugen oder Löschen von Goals und Plans). Daher muss in einem Plan nicht berücksichtigt werden, welche Auswirkungen etwaige Änderungen auf mögliche weitere Ziele bzw. Pläne des Agenten haben. Pläne wie auch Ziele können als in sich geschlossene, wieder verwendbare Komponenten unabhängig voneinander entwickelt werden.

Ziele (Goals) Ziele erlauben die Definition anzustrebender Zustände, ohne dabei bereits konkrete Aktionen festzulegen. Erst zur Bearbeitungszeit werden für Ziele geeignete Pläne ausgewählt und ausgeführt, wobei für jedes Ziel verschiedene alternative Pläne gleichzeitig oder nacheinander ausgeführt werden können bis das Ziel letztendlich erreicht ist. Da ein Agent aber nicht immer alle seine Ziele gleichzeitig verfolgen kann, werden aktive und inaktive Zustände der Zielbearbeitung unterschieden.

In Jadex werden zudem vier verschiedene Arten von Goals unterstützt: Perform-, Achieve-, Query- und Maintain-Goals. Diese Typen unterscheiden sich in der Art und Weise, wann Pläne für diese Ziele ausgeführt werden, und wie entschieden wird, ob ein Ziel erreicht ist. Einzelheiten dazu finden sich in [5]. Diese vier Goal-Typen erlauben es für jede im System vorliegende Aufgabe, die jeweils optimale Repräsentation zu benutzen. So können z.B. für einmalige Aktivitäten Perform-Goals definiert werden, die - basierend auf dem aktuellen Kontext - den richtigen Plan ausführen (z.B. um eine Benachrichtigung per Email oder SMS zu versenden). Achieve- und Query-Goals legen Ergebnisse fest, die benötigt werden bevor weitere Aufgaben fortgesetzt werden können (wie z.B. das Aushandeln eines Termins oder das Finden der Adresse eines Kunden). Mit Maintain-Goals können Bedingungen überwacht werden (z.B. das Einhalten von Fristen durch automatisch vom System ergriffene Maßnahmen - wie etwa die Benachrichtigung des zuständigen Bearbeiters).

Pläne (Plans) Pläne repräsentieren Handlungsalternativen eines Agenten, um Ziele zu erreichen und auf Ereignisse zu reagieren. Dabei besteht ein Plan immer aus zwei Teilen: dem Plankopf (Head) und dem Plankörper (Body). Der Plankörper besteht aus einer Folge von Aktionen und Aktivitäten, und stellt damit im Prinzip ein prozedurales Programm im herkömmlichen Sinne dar. Dagegen ist der Plankopf eine deklarative Beschreibung der Situationen, in denen der Plan eingesetzt werden kann.

Die Auswahl von Plänen erfolgt in einem mehrstufigen Prozess (vgl. Abschnitt 4.2). Dazu wird im Plankopf zunächst angegeben, zur Bearbeitung welcher Ziele oder Ereignisse ein Plan grundsätzlich geeignet ist. Zusätzlich kann die Anwendbarkeit über Vorbedingungen weiter auf einen bestimmten Anwendungskontext eingeschränkt werden. So kann ein Agent z.B. für die Abwicklung einer Bestellung unterschiedliche Pläne auswählen, je nachdem ob es sich um eine externe oder interne Bestellung handelt. Scheitert ein ausgewählter Plan, so werden für das Ziel so lange weitere Pläne ausgeführt bis das Ziel erreicht ist oder keine weiteren Pläne vorhanden sind.

Die eigentliche Anwendungsfunktionalität wird in den einzelnen Plankörpern realisiert. Dies bestehen aus einer Folge von Anweisungen und werden als Java-Klassen implementiert. Dabei kann der Programmierer auf die gesamte Mächtigkeit der Java-Programmiersprache zurückgreifen. Dies vereinfacht die Integration mit bestehenden Anwendungen und Systemen, wo entsprechende Programmbibliotheken oftmals bereits vorliegen.

Agentendefinition Wie gezeigt, besteht ein Jadex Agent aus lose gekoppelten Elementen wie Beliefs, Goals und Plans. Die Zusammenstellung eines Agenten aus seinen initialen Elementen wird in Jadex mittels eines XML-basierten Deskriptors formuliert. Das damit erstellte, so genannte Agent Definition File (ADF) enthält Verweise auf die Java-Klassen, welche z.B. die Belief-Datentypen oder Plankörper implementieren. Alle weiteren Informationen über den Aufbau des Agenten werden deklarativ im ADF angegeben. Dazu gehören die Beschreibungen der Goal-Typen - u.a. mit Parametern und zu erreichenden Zielzuständen - sowie die Planköpfe, die festlegen welche Pläne in welchen Situationen ausgeführt werden können.

4.2 Reasoning Prozess

Die Beliefs, Goals und Plans eines Agenten beschreiben zunächst seine statischen Eigenschaften. Das Verhalten eines Agenten wird dabei maßgeblich durch den Reasoning-Prozess bestimmt, der auf diesen Elementen operiert. Primär dient das Reasoning dazu, auf Ereignisse wie neu entstandene Goals oder eingetroffene Nachrichten durch die Ausführung geeigneter Pläne angemessen zu reagieren.

Abbildung 1.b zeigt den Reasoning Prozess in Jadex. Das Reasoning wird von drei unabhängig voneinander operierenden Komponenten (Message receiver, Dispatcher, Scheduler) durchgeführt. Dennoch ist die Verwandtschaft zum abstrakten Interpreter aus Sektion 3 unverkennbar. Analog zu *get-new-external-events()* verarbeitet der *Message receiver* eingehende Nachrichten und erzeugt entsprechende Ereignisse, die in die Ereignisliste des Agenten eingefügt werden. Der Dispatcher realisiert die Funktionen *option-generator()* und *deliberate(options)* indem er diese und weitere Ereignisse konsumiert, passende Pläne sucht und aus der Menge der passenden Pläne diejenigen auswählt, die zunächst zur Ausführung gelangen sollen. Diese Pläne werden in die Ausführungsliste (*Ready list*) eingetragen und schließlich vom Scheduler im Rahmen der Operation *execute()* schrittweise ausgeführt. Die Ausführung von Plänen kann dabei aufgrund von geänderten Beliefs sowie bearbeiteten oder neu erzeugten Goals zu neuen internen Ereignissen führen, die zur Bearbeitung durch den Dispatcher in der Ereignisliste abgelegt werden.

4.3 Zur Realisierung des Jadex-Systems

Grundlage der Realisierung des Jadex-Systems ist die unter der Führung der Telecom Italia Laboratories (TILab) entwickelte FIPA-konforme Agentenplattform JADE [1]. Aufgrund der breiten Benutzerbasis sowohl im akademischen als auch im industriellen Bereich und der freien Verfügbarkeit unter Open Source-Lizenz bietet die Plattform eine ideale Infrastruktur zur Realisierung verteilter Systeme mit Agenten. Die Plattform legt sich auf keine spezialisierte Technik zur Umsetzung der einzelnen Agenten fest, und bietet somit auch eine gute Ausgangsbasis für Erweiterungen.

Zudem bietet JADE bereits Werkzeuge u.a. zum Abhören der Kommunikation zwischen Agenten an. Im Rahmen des Projekts Jadex wurden weitere Werkzeuge entwickelt, die es z.B. erlauben, den Zustand eines Agenten zur Laufzeit in einer verteilten Umgebung zu überwachen, und auf seine aktuellen Beliefs, Goals und Plans z.B. zu Testzwecken Einfluss zu nehmen. Einzelheiten zur Realisierung des Jadex-Systems findet sich in [13,3]. Das so entstandene System ist unter einer Open Source Lizenz frei verfügbar und wird inzwischen sowohl lokal als auch von anderen Instituten international in verschiedenen Forschungs- und Lehreprojekten eingesetzt.

5 Anwendungsbeispiel

Zur Demonstration der Leistungsfähigkeit der bisher vorgeschlagenen und implementierten Konzepte rationaler Agenten soll abschließend ein Anwendungsbeispiel vorgestellt werden. Dieses stammt aus dem Projekt *MedPAge*, dass im Rahmen des von der Deutschen Forschungsgemeinschaft (DFG) initiierten Schwerpunktprogramms (SPP) *Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien* [7] durchgeführt wird. Ziel des SPPs mit Beteiligten aus den Gebieten der Wirtschaftsinformatik und der Informatik ist es, Fortschritte auf dem Gebiet der Agententechnologie dadurch zu fördern, dass agentenbasierte Anwendungen in realistischen betriebswirtschaftlichen Szenarien umgesetzt und evaluiert werden. Vor diesem Hintergrund werden im SPP die beiden Anwendungsdomänen Gesundheits- und Fertigungswesen vor allem unter logistischen Gesichtspunkten näher betrachtet.

Dabei setzt sich das in den Anwendungsbereich Krankenhauslogistik einzuordnende Teilprojekt MedPAge (Medical Path Agents) [12] mit der Terminplanung in verteilten, autonomen Krankenhaussystemen auseinander: Angenommen wird dabei, dass Krankenhäuser in viele einzelne teilautonome Funktionsbereiche unterteilt sind, die von den Patienten in Abhängigkeit ihrer jeweiligen Erkrankung durchlaufen werden. Hierbei muss u.a. die Nutzung der verschiedenen Geräten und Ressourcen durch die Patienten nach vorgegebenen Optimierungskriterien vorgeplant werden. Dazu ist es wichtig, jederzeit auch auf unerwartete Ereignisse wie Notfälle und Komplikationen reagieren zu können, um mittels dezentraler Koordination jeweils schnell und flexibel die notwendigen Umplanungen vornehmen zu können. Einen wichtigen Vergleichsmaßstab bei diesen Untersuchungen bilden dabei auch die derzeit in Krankenhäusern verwendeten - meist zentralen - Verfahren zur Terminvergabe.

Vor allem aufgrund der Komplexität derartiger Patientensteuerungsprobleme im Krankenhaus und deren inhärenter Dynamik sollten *dezentralen Steuerungsmechanis-*

men für die verschiedenen (Teil-) Prozesse verwendet werden. Dazu sollte ein Multiagentensystem eingesetzt werden, in welchem alle Koordinationsobjekte wie z.B. Patienten und Krankenhausressourcen als autonome Softwareagenten modelliert werden. Da eine für derartige Zwecke geeignete Agentenplattformen jedoch nicht zur Verfügung steht, wurden bisherige Konzepte erweitert und so das beschriebene System *Jadex* entwickelt und zur Realisierung der genannten verteilten Steuerungsmechanismen eingesetzt. Dabei wird durch die Repräsentation der einzelnen Koordinationsobjekte mit ihren jeweils eigenen Zielen der existierenden verteilten Struktur von Krankenhäusern mit relativ autonomen Teildiensten adäquat Rechnung getragen. Sowohl die Patienten- als auch die Ressourcenagenten treffen lokale Entscheidungen zur Zielerreichung autonom unter Berücksichtigung sowohl ihrer eigenen Ziele als auch der aktuellen Situation (ihren Beliefs) und können damit u.a. sehr flexibel auf Änderungen in der Umwelt (wie z.B. unerwartete Ereignisse) reagieren. So kann z.B. ein Ressourcenagent auf natürliche Art und Weise akute Notfälle oder unvorhergesehene Gerätedefekte berücksichtigen und dementsprechend dynamisch geeignete Umplanungsmaßnahmen einleiten, um das Ziel der jeweiligen Behandlung in jedem Fall bestmöglich zu erreichen. Insbesondere im Zusammenhang mit unerwarteten Ereignissen erweist sich hier der Jadex zugrunde liegende BDI-Mechanismus als sehr vorteilhaft, da z.B. keine starren Verarbeitungsabläufe gespeichert werden. So kann z.B. ein Ressourcenagent ggfs. den Plan zur Durchführung einer Untersuchung wiederholen, wenn zunächst keine relevanten Befunde erzielt wurden. Die Jadex zugrunde liegende FIPA-konforme Agenten-Middleware JADE ermöglicht zudem systemübergreifende Interoperabilität und vereinfacht die Umsetzung der Verhandlungsapekte durch standardisierte Interaktionsprotokolle wie z.B. das FIPA Contract-Net. Weitere Details zur Modellierung und Realisierung des MedPAge-Systems findet man z.B. in [4].

6 Verwandte Arbeiten

Abb. 2 gibt eine allgemeine Übersicht über verschiedene existierende Agentenplattformen, und ordnet diese Systeme mittels der Dimensionen *Einsatzfeld* (Industrie oder Forschung) und *Ausrichtung* (Middleware oder BDI Ansatz) ein. [1] Aus dieser Klassifikation ist ersichtlich, dass eine Verbindung zwischen Middleware-orientierten Systemen und BDI-Systemen derzeit kaum besteht. Insbesondere für eine industrielle Nutzung der Agententechnologie in offenen verteilten Umgebungen ist es aber von entscheidender Bedeutung, dass gleichermaßen Middleware-Aspekte wie z.B. Interoperabilität und Sicherheit sowie rationale Entscheidungsprozesse in adäquater Weise unterstützt werden. Eine Zusammenführung der entsprechenden beiden Forschungsrichtungen erscheint vor diesem Hintergrund besonders interessant.

Zum Schließen dieser Kluft zwischen Middleware und Reasoning gibt es derzeit grundsätzlich zwei unterschiedliche Herangehensweisen: Zum einen setzen Plattformen wie Agentis und Whitesteins TAP1 auf den etablierten Standard *Java J2EE* und integrieren so Agententechnologie in Applikationsserverumgebungen. [2] Die zweite

[1] Weiterführende Informationen zu den abgebildeten Systemen sind über http://vsis-www.informatik.uni-hamburg.de/projects/jadex/links.php zu erreichen.

[2] http://www.agentissoftware.com/ bzw. http://www.whitestein.com/pages/index.html

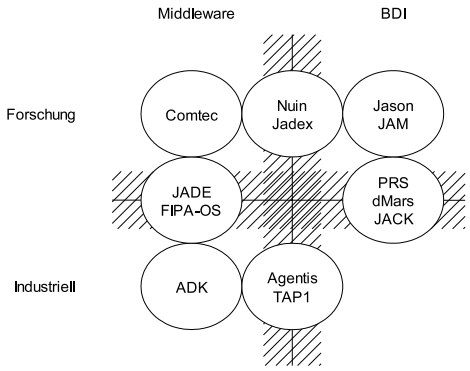

Abb. 2. Klassifikation von Agentensystemen nach Einsatz und Ausrichtung

Möglichkeit besteht darin, auf bestehende Middleware-Plattformen aufzusetzen und diese um BDI-spezifische Charakteristika zu erweitern. Dieser Ansatz wird von Nuin [6] und dem hier beschrieben Jadex verfolgt. Im Unterschied zu Jadex unterstützt Nuin's Architektur insbesondere Anwendungen aus dem Bereich des Semantic Web.

Beide Integrationsarten bieten unterschiedliche Vor- und Nachteile, so dass nicht pauschal eine Herangehensweise als insgesamt überlegen angesehen werden kann. Vielmehr hängt die Eignung der einen oder anderen Lösung insbesondere von den individuellen Charakeristika des jeweiligen Problemfeldes ab: Generell bietet die Herangehensweise auf der Basis von J2EE Vorteile deshalb, weil sie die Möglichkeit eröffnet, auf etablierte Standards und Werkzeuge zurückgreifen zu können. Problematisch ist dabei ist jedoch, dass diese Standards auf Softwarekomponenten und nicht auf Agenten abzielen und somit starke Anpassungen erfordern. Genau entgegengesetzt verhält es sich bei der Nutzung existierender Middleware-Agentenplattformen, die zwar agentenbasierte Services (z.T. auf Basis von FIPA-Standards) bereitstellen, aber noch nicht den Reifegrad der komponentenbasierten Umgebungen erreicht haben. Für Jadex wurde auch deshalb der letztgenannte Ansatz gewählt, da er näher am aktuellen Stand der Forschung in Bezug auf Agenten-Middleware liegt und es zudem erlaubt, direkt an Fortschritten im Rahmen der aktuellen Standardisierungsbemühungen im Rahmen der der FIPA und der darauf aufbauenden Systemplattformen zu partizipieren.

7 Zusammenfassung

Dieses Papier stellt einen Ansatz zur dezentralen Steuerung und Koordination verteilter Anwendungen vor, der deren steigende Komplexität besser d.h. vor allem problemadäquater zu beherrschen erlaubt indem er neben der Verteilung insbesondere die bisher nur schwer handhabbare Dynamik mit Hilfe der Agententechnologie adressiert. Wichtige Grundlagen dafür bieten *agentenorientierte Middleware-Plattformen*, die Basisdienste z.B. für die Verwaltung und Kommunikation von Agenten zur Verfügung stellen und so Teile der notwendigen Infrastruktur zur Dekomposition komplexer Anwendungen in autonome interagierende Teilkomponenten realisieren. Diese werden im Projekt Jadex um Aspekte der *rationalen Entscheidungsfindung* ergänzt.

Die für die Steuerung autonomer Anwendungsprozesse in verteilten Umgebungen typische Dynamik der Anwendungsdomäne wirkt sich direkt auf die Komplexität des abzubildenden Softwaresystems aus: Sowohl Art und Umfang der Interaktion als auch die notwendigen Entscheidungsprozesse innerhalb der beteiligten Agenten werden dadurch maßgeblich beeinflusst. Mit Hilfe der *Jadex Reasoning Engine*, die als Erweiterung der weit verbreiteten Agentenplattform JADE entwickelt wurde, können Entscheidungsprozesse von Agenten als dezentrale Entscheidungsträger bei einer derartigen Koordination systematisch und auf hohem Abstraktionsniveau formuliert und direkt ausgeführt werden. Konzeptionelle Basis dafür bildet das BDI-Modell zur Beschreibung von rationalem menschlichen Individualverhalten. Damit realisiert das Jadex-System konzeptionell eine Symbiose aus Middleware und Rationalität.

Anhand des Fallbeispiels zur funktionsübergreifenden Terminplanung im Krankenhaus aus dem Projekt MedPAge wurde abschließend kurz exemplarisch verdeutlicht, welche Vorteile der Einsatz zielorientierter Agenten für eine ausgewählte Anwendungsdomäne mit sich bringt und wie die in Jadex vorhandenen BDI-Konzepte angewendet werden können.

References

1. F. Bellifemine, G. Rimassa, and A. Poggi. JADE – A FIPA-compliant agent framework. In *4th Int. Conf. Practical Applications of Agents and Multi-Agent Systems (PAAM-99)*, 1999.
2. M. Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, 1987.
3. L. Braubach, A. Pokahr, and W. Lamersdorf. Jadex: A Short Overview. In *Net.ObjectDays 2004: AgentExpo*, 2004.
4. L. Braubach, A. Pokahr, and W. Lamersdorf. MedPAge: Rationale Agenten zur Patientensteuerung. *Künstliche Intelligenz*, (2):33–36, 2004.
5. L. Braubach, A. Pokahr, D. Moldt, and W. Lamersdorf. Goal Representation for BDI Agent Systems. In *Proc. 2nd Workshop on Programming Multiagent Systems (ProMAS04)*, 2004.
6. I. Dickinson and M. Wooldridge. Towards practical reasoning agents for the semantic web. Technical Report HPL-2003-99, Hewlett Packard Laboratories, May 15 2003.
7. S. Kirn et al. DFG-Schwerpunktprogramm 1083: Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien. In *Informatik 2003*. Köllen Druck+Verlag, 2003.
8. M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge. The Belief-Desire-Intention Model of Agency. In *Agent Theories, Architectures, and Languages (ATAL) '98*. Springer, 1999.
9. N. R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, April 2001.
10. A. Newell. *Unified Theories of Cognition*. Harvard University Press, 1990.
11. J. Odell. Objects and Agents Compared. *Journal of Object Technology*, 1(1), 2002.
12. T. O. Paulussen, N. R. Jennings, K. S. Decker, and A. Heinzl. Distributed Patient Scheduling in Hospitals. In *Proc. 18th Int. Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003.
13. A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: Implementing a BDI-Infrastructure for JADE Agents. *EXP – in search of innovation*, 3(3):76–85, 2003.
14. S. Poslad and P. Charlton. Standardizing Agent Interoperability: The FIPA Approach. In *Multi-Agent Systems and Applications*. Springer, 2001.
15. A. Rao and M. Georgeff. BDI Agents: from theory to practice. In *Proc. of 1st International Conference on Multi-Agent Systems (ICMAS'95)*. The MIT Press, 1995.
16. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.