Autonome Multi-Agenten-Systeme in der Industrie

Hochschule Offenburg
University of Applied Sciences

Informatik Master

Fakultät Elektrotechnik & Informationstechnik

Wintersemester 2014/2015

Referent	:	Prof. Dr. Jan Münchenberg
Koreferent	:	Prof. Dr. Joachim Orb
Vorgelegt am	:	30.04.2015
Vorgelegt von	:	Steffen Ritter Sautierstraße 59, 79104 Freiburg sritter@stud.hs-offenburg.de

Abstract

Diese Arbeit beschäftigt sich mit den Grundlagen zu Multi-Agenten-Systemen in der Industrie. Der Begriff “Industrie 4.0“ wird eingeführt und es wird eingehend auf die Potentiale und Herausforderungen diesbezüglich eingegangen. Außerdem wird ein Überblick über aktuelle Entwicklungen und Ansätze zur Entwicklung von sogenannten autonomen Agenten gegeben. Diese werden auch im Hinblick auf die Themen Holonic Manufacturing und Multi-Agenten-System besprochen. Im praktischen Teil der Arbeit wird ein System bestehend aus vier BDI-Agenten entwickelt, um einen beispielhaften Geschäftsprozess zu bearbeiten. Die Entwicklung basiert dabei auf Java und dem Jadex Agenten-Framework. Es wird gezeigt, dass sich damit autonome BDI-Agenten umsetzen lassen, die über Rechnergrenzen hinweg koordiniert werden können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

BPMN Business Process Model and Notation

FIPA Foundation for Intelligent Physical Agents

JADE Java Agent Development Framework

MOM Message Oriented Middleware

MAS Multi-Agenten-System

CPS Cyper-Physical Systems

HMS Holonic Manufacturing Systems

PRS Procedural Reasoning System

BDI Belief-Desire-Intention

1 Einleitung

Deutschland ist für viele Dinge bekannt. Für Pünktlichkeit, Fleiß, Höflichkeit, Ordnung und Korrektheit und nicht zuletzt für seine vielen Produkte *made in Germany*. Doch die hiesige Industrie ist durch viele Wettbewerber bedroht. Zur Zeit findet ein Umbruch statt, mit dem Ziel die deutsche Industrie weiterhin weltweit an der Spitze zu halten und Konsumenten rund um den Globus zu versorgen.

1.1 Motivation

Vom 13. bis zum 17. April 2015 fand in Hannover wie jedes Jahr die bedeutendste Industriemesse[1] der Welt statt: die HANNOVER MESSE.
Kaum ein Tag verging ohne Pressemeldungen[2] wie diese:

- ZVEI: Hannover Messe setzt Ausrufezeichen bei Industrie 4.0 und Smart Grids
- Vernetzte Industrie beflügelt HANNOVER MESSE
- VDMA: Maschinenbau geht auf Kurs 4.0
- Wirtschaftsforum diskutiert Chancen der digitalen Vernetzung
- Industrie 4.0 – In Hannover erleben, begreifen und investieren

Deutschland ist ein international wichtiger Industriestandort mit vielen produzierenden Unternehmen. Die Bandbreite erstreckt sich über die Automobilindustrie, Maschinenbau, Pharmazie, Chemie, Lebensmittel, Mess- und Regeltechnik und viele Bereiche mehr. Im Jahr 2014 wurden Waren im Wert von über einer Billion Euro exportiert [Spi15]. Doch trotz langjähriger Erfahrung in der Entwicklung und Produktion haben viele Unternehmen mit einem Problem zu kämpfen: die Anforderungen an die Unternehmen steigen laufend. Es wird eine höhere Flexibilität, Schnelligkeit und Individualität sowie geringere Kosten erwartet. Gleichzeitig darf die Qualität jedoch nicht darunter leiden. Während die Entwicklungsabteilungen möglichst in Deutschland verbleiben, wird die Produktion mehr und mehr ausgelagert. Seit Mitte der 90er Jahre verschiebt sich die Produktion von Gütern immer mehr in Länder mit günstigeren Produktionsmöglichkeiten. Zwar ist dieser Trend seit 2003 rückläufig, doch trotzdem

[1] http://www.hannovermesse.de/de/info/ueber-uns/
[2] http://www.hannovermesse.de/de/info/fuer-journalisten/pressemitteilungen/

werden nach [ZKM13] Waren im Wert von 389 Milliarden Euro an Bruttoproduktionswert im Ausland produziert. Dies entspricht etwa 21% des gesamten deutschen verarbeitenden Gewerbes. Es ist daher wichtig, neue, innovative Ansätze für die Produktion zu entwickeln und einzuführen, um Arbeitsplätze und technisches Know-How in Deutschland halten zu können.

Ein großer Teil der Produktion ist heute bereits vollständig oder teilweise automatisiert. Dieser bisher hauptsächlich auf Basis von Maschinen basierende Ansatz reicht zunehmend nicht mehr aus. Es wird daher eine Revolution der Industrie, die sogenannte *Industrie 4.0*, gefordert. Ein vielversprechender Ansatz auf diesem Weg ist die Integration der Informationstechnologie. Die heutige Automatisierungstechnik beschränkt sich häufig auf einfache Aufgaben wie zum Beispiel die Nutzung von Fließbändern oder Verpackungsanlagen. Bei komplexeren Aufgaben werden Industrieroboter eingesetzt, die selbständig bestimmte Arbeitsschritte ausführen können. Doch bisher arbeiten all diese Systeme in der Regel ohne eigene Intelligenz. Ein strikter Ablaufplan muss für jeden Prozess erstellt werden. Obwohl inzwischen Sensoren unzählige Parameter während der Produktion messen, sind die Systeme nicht in der Lage, selbst darauf zu reagieren. Die Vision für die Zukunft ist eine intelligente Verknüpfung aller beteiligten Akteure, die sich selbst auf verändernde Bedingungen einstellen und zielorientierte Entscheidungen treffen können. Die Entwicklung sogenannter Agenten Systeme steht hierbei seit einiger Zeit im wachsenden Fokus. Ein Agent repräsentiert einen beliebigen Akteur z.B. in Form eines Computerprogramms. Ein Agent kann eine Produktionsmaschine steuern, einen Transport organisieren oder einen Bestellprozess abwickeln. Kommen mehrere gleichartige oder aber auch verschiedene Agenten zum Einsatz spricht, man von einem Multi-Agenten-System (MAS). Diese Systeme sollen die genannten Anforderungen erfüllen. Durch eine konsequente Vernetzung werden die Produktionsteilnehmer somit in die Lage versetzt, autonom und zielgerichtet zu arbeiten. Die Einführung solcher Systeme und die dafür nötigen Grundlagen sind ein wichtiges Teilstück zur nächsten industriellen Revolution.

1.2 Ziele der Arbeit

Diese Arbeit soll einen Überblick über die aktuellen Herausforderungen und Entwicklungen in der Industrie, speziell den autonomen Agentensystemen, geben. Dabei werden zum einen die anstehenden Anforderungen und Lösungsmöglichkeiten für die Industrie erläutert. Zum anderen wird ein Blick auf unterschiedliche Herangehensweisen an das Thema der Agenten geworfen. Hier soll vor allem eine Grundlage für das Verständnis von Agenten und der zugrundeliegenden Theorie gelegt werden. Neben der Einführung verschiedener Konzepte zur Umsetzung von Agenten, wird auch ein spezifisches Frame-

work auf Java Basis vorgestellt Anschließend werden die Anwendungsfelder aufgezeigt und einige Umsetzungen für reale Projekte vorgestellt. Auf Basis eines einfachen Beispielprozesses wird anschließend ein einfaches Agentensystem auf Basis von Java konzipiert und implementiert.

1.3 Aufbau der Thesis

Nach der Einleitung in Kapitel 1, bearbeitet Kapitel 2 die Grundlagen, die zum Verständnis dieser Arbeit notwendig sind. Es wird auf das Modewort "Industrie 4.0" eingegangen, sowie die Thematik der Agentensysteme erläutert. Kapitel 3 nimmt Bezug auf den aktuellen Stand der Technik. Es wird das im praktischen Teil der Arbeit eingesetzte Framework Jadex vorgestellt, sowie zwei Beispiele von Agentensystemen in der Praxis bzw. Forschung. Kapitel 4 beschäftigt sich mit der Konzeption und den Anforderungen des zu entwickelnden Agentensystems, Kapitel 5 erläutert Teile der Implementierung des Systems. In Kapitel 6 wird das Ergebnis des Systems kurz vorgestellt und schließlich in Kapitel 7 ein Resümee gezogen und ein Ausblick in die Zukunft gewagt.

2 Grundlagen

Dieses Kapitel soll einen Überblick über einige grundlegenden Themen geben. Es soll ein Verständnis für einzelne Abschnitte schaffen und einige Begrifflichkeiten näher erläutern und definieren. Neben der Einführung in den neumodischen Begriff Industrie 4.0 wird auch ein Blick auf darauf basierende Konzepte wie Cyper-Physical Systems und Holonic Manufacturing geworfen. Außerdem wird der Begriff des Agenten eingeführt und dessen Ideen und Vorteile vorgestellt. Neben einem Blick auf unterschiedliche Architekturen wird das für diese Arbeit wichtige Konzept der BDI-Agenten besprochen. Abschließend folgt eine kurze Definition der Multi-Agenten-Systeme.

2.1 Industrie 4.0

Deutsche Produkte werden auf der ganzen Welt hoch geschätzt und nachgefragt. Dies liegt nicht zuletzt an der leistungsstarken Industrie, sondern auch an stetigen Innovationen und Weiterentwicklungen. Dabei ist das produzierende Gewerbe für nahezu die Hälfte der deutschen Exporte verantwortlich [Bun13].

Damit unter anderem die deutsche Wirtschaft weiterhin mit in der internationalen Spitze steht, wurde 2006 die Hightech-Strategie für Deutschland erarbeitet [Bun06]. Diese wurde 2010 von der deutschen Bundesregierung durch die Hightech-Strategie 2020 [Bun10] erweitert, deren Hauptaufgabe die Formulierung von Zielen für verschiedene Innovationsfelder in der Zukunft ist.

Eine der erarbeiteten Zukunftsvisionen ist die sogenannte Industrie 4.0. Dabei ist "Industrie 4.0" ein deutscher Begriff, der 2011 durch die Promotorengruppe KOMMUNIKATION der Forschungsunion Wirtschaft - Wissenschaft geprägt wurde. Er wird hauptsächlich als Marketingbegriff im Zusammenhang mit der im November 2011 verabschiedeten Hightech-Strategie 2020 verwendet [Pro13].

Der Aktionsplan der Bundesregierung von 2012 konkretisiert die Ziele der Hightech-Strategie 2020 in 10 großen Themenfeldern. Neben Mobilität, Energieversorgung, alternativen Rohstoffe und weiteren Projekten, ist die Industrie 4.0 ebenfalls erstmals enthalten. Der Aktionsplan fordert die Evolution zur nächsten industriellen Stufe und fördert diese mit dafür vorgesehenen 200 Mio. € [Bun12].

Sieht man sich Abbildung 1 an, wird die immer größere Dynamik in der Entwicklung sichtbar. Erst Ende des 18. Jahrhunderts begann die Produktion mit Hilfe von Wasser- und Dampfkraft. Vorher war es unmöglich große, bewegliche Maschinen zur Unterstützung zu bauen. Bereits ein Jahrhundert später wurde durch die Elektrifizierung und der daraus folgenden veränderten Arbeitsteilung die Massenproduktion möglich. Von da an ging alles sehr schnell. Wenige Jahrzehnte später verhalf die Elektronik und Informationstechnik der Industrie zu einer erneuten Revolution. Heute, lediglich 40 Jahre später, steht die nächste, die 4. Stufe an.

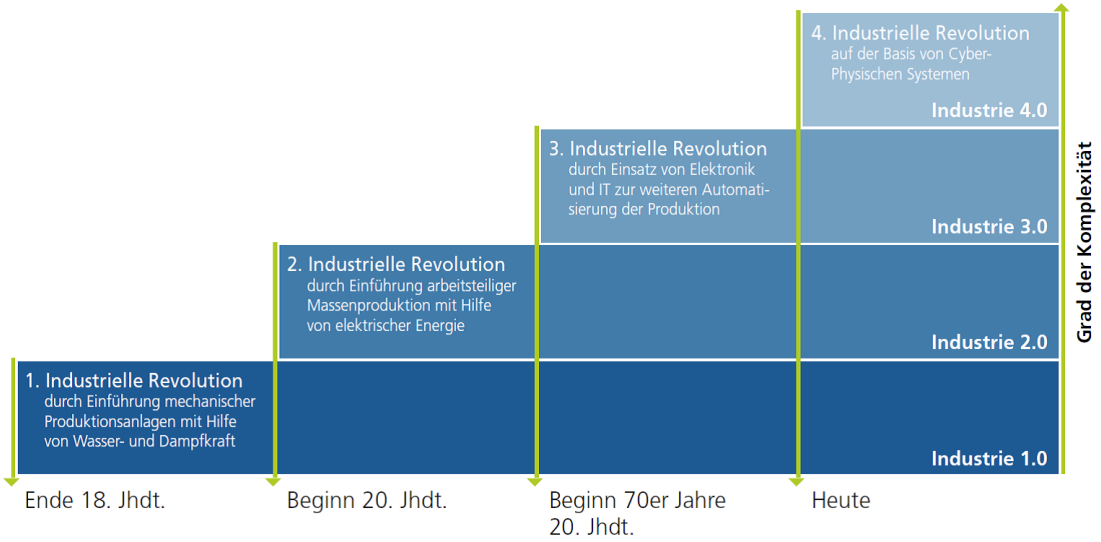

Abbildung 1: Stufen der indust. Revolution [Spa13]

Um weiter auf diese nächste Stufe eingehen zu können bedarf es zunächst einmal einer Definition. Eine mögliche Definition von Industrie 4.0 liefert [Spa13] wie folgt:

Definition 1 *Unter »Industrie 4.0« wird die beginnende vierte industrielle Revolution nach Mechanisierung, Industrialisierung und Automatisierung verstanden. Zentrales Element sind vernetzte Cyber-Physische Systeme (CPS).*

Dabei wird deutlich, dass aus dieser Begriffsdefinition wenig neue Erkenntnis gewonnen werden kann. Der letzte Satz allerdings weißt auf den großen Unterschied zur bisherigen Industrie hin. Die sogenannten Cyper-Physical Systems (CPS) sind ein zentraler Bestandteil, auf die später noch genauer eingegangen wird.

Es gibt allerdings noch weitere Konzepte, die maßgeblich für die neue Industrie stehen wie z.B. Selbstorganisation durch dezentrale Produktionssysteme, neuartige Systeme, um Produkte weiter zu individualisieren, an die Bedürfnisse des Menschen angepasste Fertigungssysteme sowie Entwicklungen, um die Nachhaltigkeit und Ressourceneffizienz weiter zu steigern. [Las+14].

Ein weiterer Ansatz sind sogenannte *Smart Factories*. [Pro13] erwartet, dass Unternehmen ihre Maschinen, Lagersysteme und andere Betriebsmittel mit Hilfe von CPS weltweit miteinander vernetzen werden. Die so entstehenden intelligenten Fabriken bilden das Rückgrat der neuen Industrie. In der Vision sind hier die Produkte und auch deren Teilprodukte jederzeit eindeutig identifizierbar. Ein Produkt kennt nicht nur seinen aktuellen Standort (bzw. Standpunkt in der Produktionskette), sondern führt auch einen Verlauf aller bisher ausgeführten Aktionen. Hierbei findet sowohl eine Verknüpfung der vertikal angeordneten betriebswirtschaftlichen Prozesse, als auch der horizontal angeordneten Wertschöpfungsprozesse statt. Somit kann sich ein Produkt intelligent und selbstständig von Auftragseingang bis zur Auslieferung durch die Wertschöpfungskette bewegen.

Die Einführung solcher Prozesse ist jedoch mit Schwierigkeiten verbunden. Zum einen steigt die Komplexität der Prozesse durch die Kombination von Informationstechnologie und Maschinenbau. Je nach Auswahl der Technologie und des Umsetzungsgrads bei der Einführung entstehen mannigfaltige Kombinationsmöglichkeiten, die jeweils aneinander angepasst werden müssen. Zum anderen wird die Umstellung häufig auf bereits bestehende Fabriken mit eingespielten Arbeitsabläufen, Prozessen und Maschinen angewandt. Wichtig wäre hier jedoch eine Planung im voraus, also bevor eine Fabrik in Betrieb geht und feste Strukturen besitzt. [BS14]

[Las+14] sieht in der Industrie 4.0 zwei Entwicklungsrichtungen. Zum einen sind dies der Druck von außen auf Unternehmen immer schneller und dabei wirtschaftlicher zu produzieren. So werden immer kürzere Entwicklungszeiten und die Individualisierung von Produkten erwartet. Der letzte Punkt führt im Extremfall zur sogenannten Losgröße 1. Das heißt, es ist wirtschaftlich unerheblich, ob eine Produktvariante in Serien von mehreren tausend Stück oder nur einmal, individuell gefertigt wird. Der Kunde ist König und kann in Zukunft sein Auto direkt ab Werk mit individueller Lackierung bestellen, ohne dass die Produktion geändert werden muss. Daraus resultierend müssen Unternehmen flexibler und ressourceneffizienter werden. Zudem nimmt der Technologiedruck weiter zu. Es ist notwendig, innovative Techniken zur Steigerung der Automatisierung, Miniaturisierung, Digitalisierung und Vernetzung zu entwickeln und einzusetzen, um den oben genannten Anforderungen zu begegnen.

Das ganze Potential zeigt sich in einer flächendeckenden Durchdringung dieser Vision. Aufträge besorgen sich selbständig benötigte Materialen, buchen Bearbeitungsmaschinen oder organisieren ihre eigene Auslieferung zum Kunden. [Spa13]

Eine notwendige Voraussetzung ist die konsequente Vernetzung der dezentral aufgebauten Systeme. Der Fortschritt im Bereich der energie- und kostengünstigen Funkverbindungen ermöglichen den schnellen Aufbau von verteilten Infrastrukturen. [Spa13]

Nach [Pro12] ist es für den Produktionsstandort Deutschland enorm wichtig, "die vom Internet getriebene 4. Industrielle Revolution mit zu gestalten und autonome, selbststeuernde, wissensbasierte und sensorgestützte Produktionssysteme zu entwickeln, zu vermarkten und zu betreiben."

Zusammenfassend lässt sich sagen, dass Industrie 4.0 mehr wie ein platter Werbeslogan ist. Hinter dem Begriff verstecken sich viele wichtige Konzepte und Ideen, um die Produktion weltweit zu verbessern. Zum einen für die Unternehmen, die schneller reagieren und günstiger produzieren können. Aber auch für den Kunden, der in Zukunft individuelle Produktvarianten ohne großen Aufpreis oder Zeitverzug bestellen können wird.

2.1.1 Cyper-Physical Systems

Ein Schlüsselelement dieser Revolution ist eine umfassende Vernetzung der an der Produktion beteiligten Akteure. Zentraler Punkt dabei sind sogenannte CPS. Diese Systeme umfassen Sensorik, Aktorik, funktionelle Abläufe sowie Schnittstellen zwischen den Systemen selbst und dem Benutzer des Systems. [Min14]

Definition 2 *»Cyber-Physische Systeme« (CPS, engl. Cyber-Physical Systems) sind mit einer eigenen dezentralen Steuerung (engl. embedded systems) versehene intelligente Objekte, welche in einem Internet der Daten und Dienste miteinander vernetzt sind und sich selbstständig steuern. [Spa13]*

Dabei kann ein praktisch alles einCPS sein. Über einen intelligenten Toaster mit Webinterface, den Kühlschrank, der selbständig beim Supermarkt bestellt oder eben die Produktionsmaschine, die ein Bauteil erkennt, entsprechend verarbeitet und weiterleitet. Auch eine ganze Fabrik kann ein CPS sein, wenn sie in der Lage ist, selbstständig zu arbeiten und mit anderen Fabriken über standardisierte Schnittstellen zu kommunizieren.

2.1.2 Holonic Manufacturing Systems

Während die später besprochenen Multi-Agenten-Systeme aus dem Forschungsfeld der künstlichen Intelligenz stammen, sind Holonic Manufacturing Systems (HMS) aus philosophischen Überlegungen heraus entstanden. HMS sind ein Konzept, das auf den Begriff des "Holons" zurückgeht. Holon ist ein von Arthur Koestler geprägter Begriff,

den er 1967 in seinem Buch "The Ghost in the Machine" verwendete. Ein Holon beschreibt dabei eine stabile Struktur, die selbst wieder aus Holonen besteht. Als ein Beispiel führt Koestler den menschlichen Körper an. Dieser besteht aus Organen, welche wiederum aus Zellen bestehen, die ebenfalls weiter unterteilt werden können. Die einzelnen Bestandteile können nicht ohne die über- bzw. untergeordneten Teile verstanden werden. [Koe89]

Mehrere Holonen bilden eine sogenannte "Holarchie". Diese Überlegungen haben auch Auswirkungen auf moderne Produktionssysteme. Holarchien bieten Vorteile wie Stabilität, Adaptivität, Flexibilität und Effizienz. Holonen können wie Agenten autonom sein und miteinander kooperieren. In einem HMS sind Holonen meist aus zwei Teilen zusammengesetzt. Aus einem virtuellen, informationsverarbeitenden Teil und aus einem physischen Teil. Der physische Teil transportiert und verarbeitet Objekte, während der andere Teil für kooperative Zwecke zwischen den Holonen zuständig ist. Das Besondere ist nun, dass diese Produktionseinheiten aus anderen Produktionseinheiten aufgebaut werden können. [GSV99]

Prinzipiell sind HMS und Multi-Agenten-Systeme recht ähnlich. Die Konzepte sind aus unterschiedlichen Forschungsrichtungen entstanden und haben sowohl Gemeinsamkeiten als auch Unterschiede. Einige Forschungsarbeiten versuchen aus beiden Richtungen Vorteile zu ziehen und beide Ansätze miteinander zu verbinden (vgl. [Ada+11]).

2.2 Agenten

Schon seit längerer Zeit beschäftigt sich die Forschung mit sogenannten *Agenten*. Obwohl der Begriff von vielen verwendet wird, besteht doch häufig keine Einigkeit über die genaue Begriffsdefinition und Abgrenzung. Ein Definitionsversuch von Michael Wooldridge lautet folgendermaßen:

"An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives."[Woo02a]

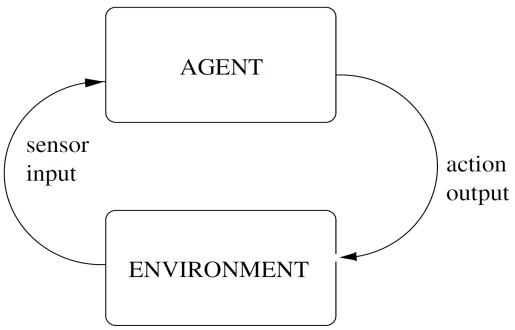

Abbildung 2: Abstrakte Vorstellung eines Agenten und seiner Umgebung [Woo02a]

Ein Agent ist also ein System, das autonome Handlungen in seiner Umgebung ausführt, um vorbestimmte Ziele zu erreichen. Außerdem zeigt Abbildung 2, dass ein

Agent seine Umgebung wahrnehmen und durch seine Aktionen auch beeinflussen kann. Um einen Agenten jedoch vollständig zu beschreiben und von anderen Computersystemen zu unterscheiden, fehlen noch einige wichtige, zusätzliche Eigenschaften.

[MN95] unterscheidet zwei Auffassungen, die die Eigenschaften der oben genannten Definition miteinschließt und erweitert. Nach der ersten, schwachen Vorstellung ist ein Agent ein Computersystem mit folgenden Eigenschaften:

- Autonomie
- Soziale Fähigkeiten
- Reaktivität
- Proaktivität

Ein Agent arbeitet demnach autonom, das heißt, er besitzt zu einem gewissen Grad die Kontrolle über seine Handlungen und wird nicht unmittelbar von außen gesteuert. Zudem kommuniziert er und arbeitet mit anderen Agenten zusammen. Er kann seine Umgebung wahrnehmen und reagiert entsprechend auf deren Änderung. Eine der wichtigsten Eigenschaften ist die Proaktivität. Im Gegensatz zur Reaktivität wird nicht nur auf Änderungen reagiert, sondern es werden zielgerichtete Aktionen ausgeführt, um die Umgebung zu beeinflussen und einen gewünschten Zustand zu erreichen. Die drei letztgenannten Eigenschaften sind ein entscheidender Unterschied zu vielen anderen autonomen Systemen und bilden die Voraussetzung für intelligentes Verhalten [Woo02a].

Die zweite Auffassung geht stärker auf die Forschungen im Bereich der künstlichen Intelligenz ein. Zusätzlich zu den oben genannten Eigenschaften besitzt ein Agent menschenähnliche Eigenschaften wie z.B. knowledge, belief, intention und obligation[1].

Eine der momentan populärsten Agententheorien ist das Belief-Desire-Intention (BDI) Modell basierend auf den Überlegungen von Michael Bratman [Bra87][AM91] und das Procedural Reasoning System (PRS) von Michael P. Georgeff und Amy L. Lansky [GL86][GL87].

2.2.1 Agenten Architekturen

Im Laufe der Zeit wurden unterschiedliche Ansätze zur Entwicklung von Agenten entwickelt. [MN95] gliedert diese in drei wesentliche Bereiche.

[1] de: Wissen, Eindrücke, Absichten, Pflichten. Da in der Literatur im Allgemeinen die englischen Begriffe verwendet werden, wird in dieser Arbeit dies beibehalten.

Deliberative Architekturen zeichnen sich durch eine detaillierte, logische Repräsentation der Welt aus. Innerhalb dieses Modells können nun Entscheidungen mit Hilfe von logischen Operationen getroffen und überprüft werden. Obwohl dieser Ansatz verlockend klingt, gibt es dabei jedoch ein Problem. Es ist sehr schwierig, die reale Welt, ihre Objekte und Prozesse in ein formales Modell zu transformieren. Ebenso ist die Repräsentation von Entscheidungen und Schlussfolgerungen nicht einfach lösbar. [AM95]

Reaktive Architekturen sind ein alternativer Lösungsvorschlag, um die Probleme von Deliberativen Architekturen zu vermeiden. Zum Beispiel beschrieb Rodney A. Brooks bereits 1986 seine *Subsumption Architecture* [Bro86]. Er geht dabei davon aus, dass Intelligenz auch ohne ein formales Weltmodell auskommt. Intelligentes Verhalten resultiert ihm zufolge aus der Interaktion von Agenten mit ihrer Umwelt, also dem Zusammenspiel von komplexen Systemen [AM95]. Sein Vorschlag basiert dabei auf unterschiedlichen Schichten, die aufeinander aufbauen [Bro91].

Hybride Architekturen sind eine Mischform aus beiden Ansätzen. Dabei wird häufig ein reaktiver Ansatz benutzt, um schnell auf Änderungen der Umgebung reagieren zu können. Längerfristige Ziele und komplexere Probleme können dann mit Hilfe einer deliberativen Herangehensweise bearbeitet werden. Solch ein hybrider Ansatz ist zum Beispiel das sogenannte PRS von Georgeff und Lansky [GL87].

Eine aktuellere Übersicht über die vielfältigen Theorien und Architekturen im Zusammenhang mit Agenten zeigt Abbildung 3. Dabei wird nicht zwischen den drei oben genannten Architekturansätzen, sondern nach den vier Disziplinen Soziologie, Biologie, Psychologie und Philosophie unterschieden. Nach [BPL08] sind somit *Bratman's BDI-Agenten* eher aus philosophischen Überlegungen heraus entstanden, während *Brooks's Subsumption Architecture* aus dem Bereich der Biologie kommt. Dies soll nur einen Überblick darstellen. Im praktischen Teil dieser Arbeit wird mit BDI-Agenten gearbeitet.

2.2.2 BDI

1991 entwickelten Anand Rao und Michael Georgeff auf Basis der Überlegungen von Michael Bratman ein formales BDI Modell. BDI steht hierbei für die Begriffe Belief, Desire und Intention, also frei übersetzt: Glaube, Verlangen und Absicht. Häufig werden der Begriff Desire und Goal gleich behandelt, ebenso wie Intention und Plan. Im Gegensatz zu vielen anderen Ansätzen werden Intentions hier auf eine Ebene mit Belief und Desire gestellt. BDI nutzt ein sogenanntes *possible worlds*-Konzept. Es gibt eine Vergangenheit, aber viele mögliche Versionen der Zukunft. Formal wird dies durch

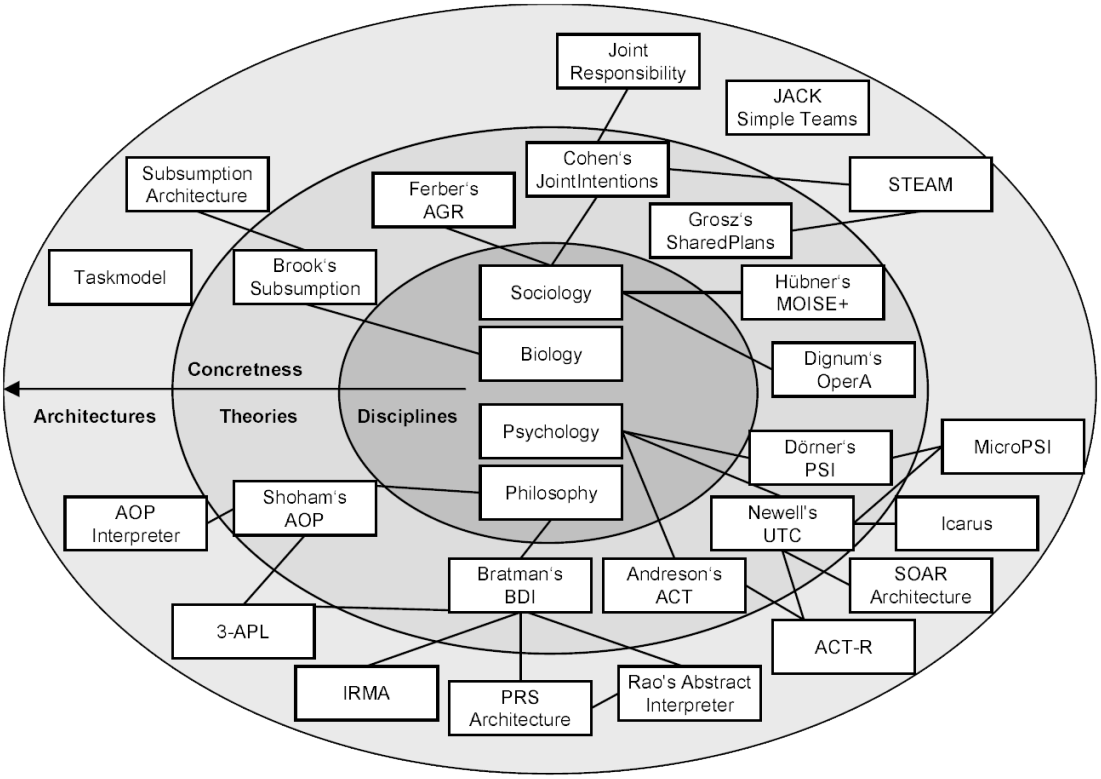

Abbildung 3: Überblick über versch. Agenten Architekturen (vgl. [BPL08])

einen verzweigten "Zeitbaum" ausgedrückt. Jeder Knoten repräsentiert eine mögliche Situation. Mit Hilfe auftretender Ereignisse können die Situationen dann transformiert werden. Das Procedural Reasoning System ist eine Implementierung der abstrakten BDI-Architektur von Georgeff und Lansky. [Mül99]

2.2.3 Multi-Agenten-Systeme

> "*The idea of a multiagent system is very simple. An agent is a computer system that is capable of independent action on behalf of its user or owner. [...] A multiagent system is one that consists of a number of agents, which interact with one another, typically by exchanging messages through some computer network infrastructure.*"[Woo02b]

Die bisherige Forschung hat sich vor allem mit den internen Abläufen eines Agenten auseinander gesetzt. Wie können Ziele definiert, Wissen repräsentiert oder darauf basierende logische Entscheidungen getroffen werden. Wirklich sinnvolle Anwendungsmöglichkeiten entstehen jedoch erst durch die Kooperation von mehreren Agenten. Damit aber mehrere Agenten miteinander kooperieren und zu einem Multi-Agenten-System zusammengesetzt werden können, bedarf es in der Regel einer gemeinsamen Kommunikationsplattform. Die 1996 gegründete Foundation for Intelligent Physical Agents (FIPA)[2] befasst sich mit der Standardisierung von Agentensystemen und im

[2] http://fipa.org/

Besonderen auch mit der Kommunikation zwischen den Agenten. Mit der bekannteste Standard ist die FIPA Agent Communication Language (FIPA-ACL) [Pos07]. Ein Agentenframework, das die FIPA-ACL nutzt, ist zum Beispiel das Java Agent Development Framework (JADE)[3]. Auch das später in einem eigenen Punkt behandelte Framework Jadex ist zu diesem Standard konform und ermöglicht somit eine Agenten übergreifende Kommunikation. Ein MAS ist im Grunde nichts anderes als ein verteiltes, nebenläufiges System, bestehend aus Agenten. Diese Agenten können auf unterschiedlicher Hardware zeitgleich, das heißt nebenläufig arbeiten. Je nach Art des Systems ist auch eine einfache Skalierung möglich. Im Idealfall sollten neue Agenten dem System einfach beitreten und mitarbeiten können. Die Schwierigkeit besteht darin, dass die Agenten einerseits möglichst autonom bleiben und ihre eigenen Ziele verfolgen sollen, sich dabei andererseits aber nicht gegenseitig behindern.[Fre09]

Ein zentraler Punkt bei der Entwicklung von MAS ist also nicht nur die Kommunikation, sondern vor allem die Koordination. Im Idealfall erhält man aber ein System, das Probleme lösen kann, die für einen einzelnen Akteur nicht möglich wären. Besonders dieser Punkt macht solche Agentensysteme so interessant für Produktionsprozesse. In einem Produktionsablauf ist nur in Ausnahmefällen ein einzelner Akteur ausreichend, um ein Ziel zu erreichen. Bisher muss enorm viel Aufwand betrieben werden, um die vielen unterschiedlichen Maschinen bei der Produktion zu koordinieren. MAS versprechen durch ihre Fähigkeit sich selbst zu koordinieren nicht nur eine Zeitersparnis bei der Behandlung von Abläufen. Sondern auch Ergebnisse, die ein Mensch angesichts der immer dynamischeren Prozesse kaum erreichen kann, da unvorhergesehene Ereignisse im Vorhinein nicht in ihrer Gänze berücksichtigt werden können.

[3] http://jade.tilab.com/

3 Stand der Technik

Es gibt unterschiedliche Frameworks um Agenten zu entwickeln wie z.B. JACK oder JADE. In diesem Kapitel wird jedoch ausschließlich auf das Framework Jadex eingegangen, da dieses im praktischen Teil der Arbeit eingesetzt wird. Jadex ist erprobt, gut dokumentiert und aktuell. Außerdem bietet es einen relativ einfach Einstieg in die Entwicklung von BDI-Agenten, weswegen es schließlich ausgewählt wurde. Der zweite Teil des Kapitels widmet sich den möglichen Anwendungsfeldern für Agentensysteme sowie einigen Einsatzbeispielen aus der Praxis und Forschung.

3.1 Jadex

Jadex[1] ist ein Projekt der Universität Hamburg[2] und wird maßgeblich von Lars Braubach und Alexander Pokahr entwickelt.

Das Framework entstand um das Jahr 2005, mit dem Ziel eine Plattform für BDI-Agenten zu schaffen. Zu Beginn basierte es noch auf dem Framework JADE für die Kommunikation zwischen den Agenten. Inzwischen ist das Projekt jedoch vollständig eigenständig und bietet eine eigene Kommunikationsplattform. Dabei ist es jedoch weiterhin möglich, über standardisierte FIPA-Messages (siehe Unterabschnitt 2.2.3) mit anderen konformen Plattformen Nachrichten auszutauschen (wie z.B. JADE).

Bisherige Entwicklungen konzentrierten sich meist entweder auf die sehr spezielle Entwicklung von Agenten oder aber deren standardisierte Kommunikation [ALW05]. Jadex vereint beide Disziplinen. Das Framework selbst bietet die Möglichkeit, Agenten in Java zu implementieren. Außerdem wird ein Container zur Verfügung gestellt, in dem die Agenten ausgeführt werden und mit dem der Kontakt zu entfernten Plattformen und deren Agenten hergestellt werden kann. Zu Beginn des Projekts wurden die Agenten mit Hilfe von XML-Dokumenten beschrieben und danach in Java eingebunden. Mittlerweile lassen sich aber über eine Vielzahl von Annotationen Agenten direkt im Java-Code erzeugen.

[1] http://www.activecomponents.org/
[2] https://vsis-www.informatik.uni-hamburg.de/vsis/research/lookproject/27

Eine Übersicht über die grundlegenden Bestandteile von Jadex zeigt Abbildung 4. Die einzelnen Teile werden im Folgenden genauer betrachtet.

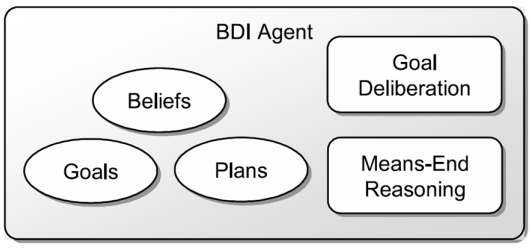

Abbildung 4: Jadex Übersicht

Abbildung 5 zeigt das Zusammenspiel der Komponenten. Der rechte Teil bildet den eigentlichen Agent, oder wie hier dargestellt eine *Capability*. Eine Capability ist im Grunde nichts anderes wie eine gekapselte Funktionalität. Ein Agent kann dann eine für seine Funktion benötigte Capability einbinden und verwenden. Somit entsteht eine gute Modularisierung.

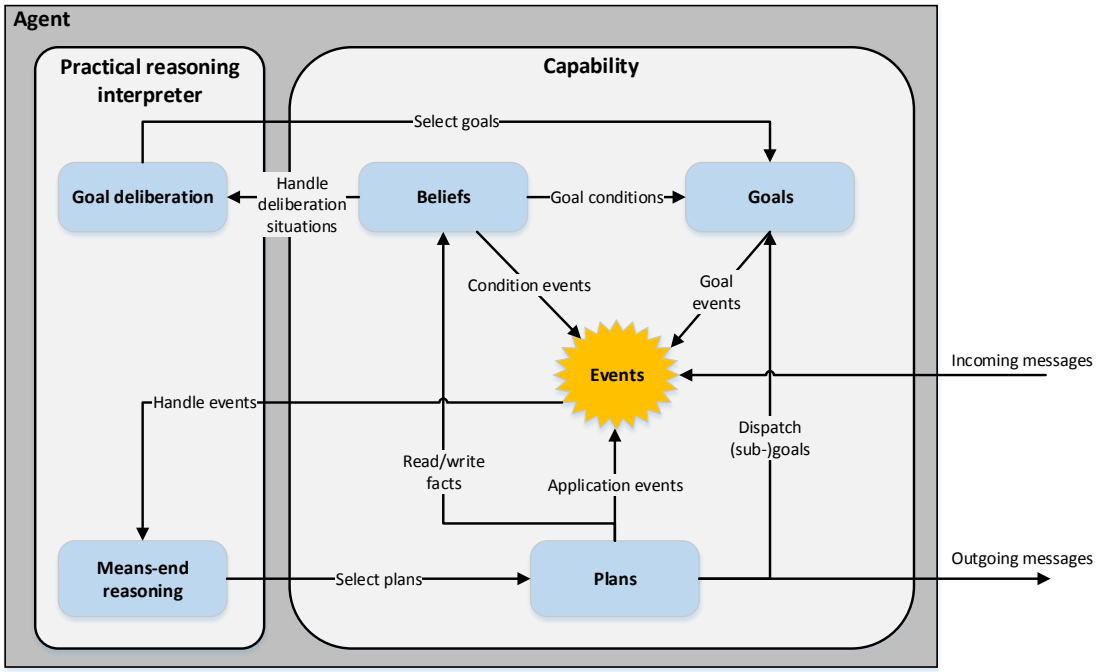

Abbildung 5: Jadex Architektur

Der linke Teil mit *Goal deliberation* und *Means-end reasoning* bildet das eigentliche PRS-System. Wenn sich zum Beispiel ein Belief ändert, wird die *Goal deliberation* aktiv und wählt entsprechende Goals aus. Das *Means-end reasoning* ist für die Auswahl geeigneter Plans zuständig.

Außerdem gibt es noch sogenannte Events. Events sind alle möglichen Ereignisse, die innerhalb des Systems ausgelöst werden. Dies können z.B. eingehende Service-Aufrufe von außen sein oder die Veränderung des Aktivitätszustands eines Goals. In den meisten Fällen wird bei einem Event das *Means-end reasoning* aktiv. Dies wählt eine geeignete Handlungsroutine in Form eines Plan aus, die das ausgelöste Event

bearbeitet, bzw. sich davor für diese Art von Event registriert hat.

Ein Agent (bzw. eine Capability) besteht im Wesentlichen aus vier Teilen.

Beliefs drücken die Eigenschaften der Umgebung aus. Ein Belief kann praktisch jede Programmvariable sein. Es können sogar Listen oder andere Collections als Belief definiert werden und somit vielfältige Anwendungsmöglichkeiten bedienen. Sobald sich ein Belief ändert, wird dies dem *Practical reasoning interpreter* mitgeteilt. Dieser prüft Auswirkungen auf definierte Ziele. Außerdem wird ein Event ausgelöst. *Means-end reasoning* sorgt dann dafür, dass Plans, die diesen Belief als Trigger definiert haben, aktiviert werden.

Goals sind die Ziele, die ein Agent hat. Sie werden als einfache Java-Klassen implementiert. Dabei können sie wie gewohnt intern in der Klasse des Agenten selbst oder aber auch extern definiert werden. Ist ein Ziel für mehrere Agenten von Bedeutung, vermeidet die Modularisierung somit die unnötige Wiederholung von Code. Jeder Agent kann dann dieses Goal nutzen. In Jadex gibt es unterschiedliche Goal-Typen.

- PerformGoal
- AchieveGoal
- QueryGoal
- MaintainGoal

Dabei entscheidet die Verwendung der genutzten Annotationen über den Typ. Zum Beispiel kann ein PerformGoal als gänzlich leere Klasse definiert werden. Dabei werden ausgelöste Pläne einmalig ausgeführt. Besitzt das Ziel eine *@GoalTargetCondition*, so wird es automatisch zu einem AchieveGoal. Dieser Goal-Typ bleibt solange aktiv, bis der Zielwert (Target) erreicht wird. Ein QueryGoal wird häufig zur Informationsbeschaffung eingesetzt. Es besitzt Parameter für Input und Output. Erst wenn der Output von einem Plan bedient wurde, ist das Ziel erreicht. Ein MaintainGoal ist ähnlich einem AchieveGoal. Allerdings besitzt es zusätzlich zur TargetCodition noch eine MaintainCondition. Als Beispiel zum besseren Verständnis bietet sich der Ladeprozess eines Akkus an. Der Zielwert könnte “100% Ladezustand“ sein, während die MaintainCondition 5% definiert. Das Ziel befindet sich solange im Ruhemodus, bis 5% Akkukapazität unterschritten werden. Dies könnte z.B. nötig sein, um das Gerät sicher herunterzufahren oder im Falle eines Roboters, zu einer Ladestation zu fahren. Die TargetCondition stellt dann sicher, dass der Akku vollständig geladen wird und nicht bereits bei 6% der Vorgang abgebrochen wird. Erst danach wird das Ziel wieder inaktiv.

Plans sind die eigentlichen Handlungsroutinen. Ebenso wie Goals können Plans als Klasse (sowohl intern als auch extern) definiert werden. Zusätzlich lassen sich Plans aber auch als einfache Methode umsetzen. Dies ist vor allem bei sehr einfachen Plans sinnvoll, vermeidet unnötigen Code und erhöht die Lesbarkeit. Ein Plan kann Beliefs ändern, indem die zugrundeliegende Variable verändert wird. Außerdem können Subgoals aktiviert werden, um aufeinander aufbauende Ziele und Aktionen zu realisieren. Jeder Plan kann unterschiedliche Trigger besitzen. Dies können z.B. Veränderungen eines Beliefs sein oder die Aktivierung eines bestimmten Goals.

3.2 Anwendungsfelder

Die Möglichkeiten der Verwendung von Agentensystemen ist beinahe unüberschaubar groß. Im Prinzip können überall dort Agenten eingesetzt werden, wo intelligentes und kooperatives Verhalten bei der Nutzung von computergesteuerten Systemen benötigt wird.

Mit der Hilfe von Agenten können komplexe und dynamische Anwendung in vielen Bereichen umgesetzt werden. Dazu gehören unter anderem (vgl. [Mül99], [Lei09]):

- Produktionsplanung
- Produktionssteuerung
- Verkehrssteuerung
- Prozessmanagement
- Telekommunikation
- Gesundheitsbranche
- Internethandel
- ...

Aber auch viele weitere Gebiete wie die Logistik, das Stromnetz, der Flugverkehr oder die Raumfahrt profitieren von den vielversprechenden Möglichkeiten, die autonome Agentensysteme bieten.

3.3 Praxisbeispiele und Forschungsprojekte

Anschließend zu den aufgezeigten Anwendungsfeldern werden noch zwei ausgewählte Beispiele vorgestellt.

Zeitungsverlag

In [BST11] wird der Einsatz eines Multi-Agenten-Systems bei einem großen deutschen Zeitungsverlag besprochen. Das Unternehmen kämpft mit unterschiedlichen Problemen. Zum einen ist die Produktion einer Ausgabe zeitlich schwer abzuschätzen, da häufig noch Beiträge während der Produktion fertiggestellt werden. Außerdem gibt es unterschiedliche Ausgaben je nach Region. Die Entfernung und damit die benötigte Zeit für den Transport der Zeitungen zu ihren Auslieferungsstellen kann relativ gut bestimmt werden. Allerdings ergeben sich auch hier Unwägbarkeiten, je nach Verzögerung und Planung der Routen. Die eigentliche Auslieferung startet beim Sitz der Produktion. Von dort werden die lokalen Verteilstellen angefahren. Jede lokale Lieferstelle benötigt unterschiedliche Ausgaben der Zeitung, teilweise auch mehrere davon.

Das Unternehmen hat dabei mit unterschiedlichen Problemen zu kämpfen. Zum Beispiel fallen Strafkosten an, wenn Transportfahrzeuge auf eine verspätete Produktion warten müssen. Ebenso kann eine verzögerte Auslieferung an die lokalen Verteilstellen zwar durch eine Erhöhung der Transporter ausgeglichen werden. Allerdings fallen hier ebenfalls hohe Kosten an. Deswegen sind die erhofften Ziele durch Einführung eines Multi-Agenten-Systems die Minimierung der Kosten und die Maximierung der Kundenzufriedenheit. Dabei kann ersteres durch verschiedene Unterziele wie z.B. Minimierung der Transportwege, der benötigten Transporter und der anfallenden Strafzahlungen an die Spediteure erreicht werden. Die Maximierung der Kundenzufriedenheit steht zum ersten Ziel meist in Konflikt. Zum einen wird die Zufriedenheit durch die Zuverlässigkeit und Pünktlichkeit, aber auch durch die Aktualität der Ausgabe bestimmt.

Es wurden unter anderem drei Agententypen eingesetzt

- Emergency Agents
- Edition Agents
- Vehicle Agents

Diese drei dynamisch kooperierenden Agenten werden im Zusammenspiel mit anderen, statischen Agenten zur Optimierung von Routenplänen verwendet.

Nach Einführung des Systems und einer 40 tägigen Testphase waren die Ergebnisse durchweg positiv. Die variablen Kosten konnten um 15-17% reduziert, die Kundenzufriedenheit um 3-4% gesteigert und die Anzahl an benötigten Transportfahrzeuge um 16-22% gesenkt werden[3].

[3] Die schwankenden Werte resultieren aus der Verwendung von unterschiedlichen Parametern zur Optimierung

MedPage

Ein weiteres, wenn auch nur auf Forschungsebene betriebenes Projekt ist MedPage ("Medical Path Agents") (vgl. [BPL14], [PBJ13], [Pau+06]). Ziel des Projekts war, die Behandlungspläne von Patienten in Krankenhäusern zu optimieren. In Krankenhäusern oder auch Arztpraxen herrscht selten Normalbetrieb. Zu jeder Zeit können neue Notfälle eintreffen oder Komplikationen bei der Behandlung auftreten, die bestehende Zeitpläne über den Haufen werfen. Trotzdem sollen alle Patienten möglichst kurz auf ihre Behandlung warten müssen. Da Krankenhäuser inzwischen häufig sehr genau kalkulieren müssen um ihre Kosten zu decken, ist es aber auch wichtig, die vorhandenen Ressourcen wie z.B. Röntgengeräte oder andere teure medizinischen Geräte auszulasten.

MedPage basiert auf Java und nutzt Jadex, um BDI-Agenten zu entwickeln. Dabei wurden zwei Agententypen genutzt. Zum einen versuchen *patient agents* die Wartezeit zu minimieren, während *resource agents* die Auslastung der Betriebsmittel des Krankenhauses maximieren sollen. Um diese zwei Ziele, die meist in Konflikt zueinander stehen zu erreichen, sollen die Agenten verhandeln um ausbalancierte Behandlungspläne zu generieren und auf eintretende Ereignisse (Notfälle u.ä.) zu reagieren. Die Schwierigkeit besteht auch darin, dass Patienten nicht nur zeitnah behandelt und zufrieden sein sollen, sondern dass auch Patienten mit schweren Beschwerden priorisiert werden.

Für solche Anwendungsfälle, in denen sich die Umstände jederzeit dynamisch ändern können, bietet sich ein Agentensystem an. Die Akteure wie Patienten oder einzelne Ressourcen lassen sich instinktiv in Agenten übersetzen. Auch wenn das Projekt nur zu Demonstrationszwecken gedient hat, zeigt es doch, dass nicht nur die Industrie, sondern auch die Allgemeinheit spürbar von solchen Systemen profitieren kann.

4 Konzeption

Um die Industrie von der Einsetzbarkeit und den Vorteilen von Agentensystemen zu überzeugen, ist es nötig, möglichst viele Anwendungsbeispiele präsentieren zu können. In einem internen Hochschulprojekt der HS-Offenburg werden Grundlagen für einfache Geschäftsprozesse erarbeitet. Ziel ist es dabei, einfache Simulationen für die Forschung und Lehre zur Verfügung zu stellen.

4.1 Problemstellung

Um unterschiedliche Geschäftsprozesse zu simulieren, benötigt es einer geeigneten Softwareumgebung. Dabei gibt es bereits Software-Plattformen (wie z.B. Activiti[1]), die die einfache Ausführung von BPMN-Prozessen erlauben. Mit Business Process Model and Notation (BPMN) lassen sich Arbeitsabläufe gut modellieren und dokumentieren. Dabei ist der Ablauf jedoch sehr strikt und prozedural gehalten. Die Prozesse arbeiten weder autonom noch proaktiv. Aktionen werden lediglich reaktiv, das heißt bei eintreten eines bestimmten Ereignisse ausgeführt. Wünschenswert wäre jedoch ein System, das im Rahmen vorgegebener Ziele selbständig agieren kann. Auftretende Ereignisse beeinflussen die Ausführung von Aktionen, sind dabei jedoch nicht deren alleiniger Auslöser. Die dargestellten Eigenschaften und Möglichkeiten von Agentensysteme machen diese für viele Anwendungsgebiete und Unternehmen zu einer interessanten Möglichkeit, Prozesse intelligenter und vor allem effizienter zu gestalten.

Im praktischen Teil dieser Arbeit soll solch ein System, bestehend aus unterschiedlichen Agenten, prototypisch entwickelt werden. Die Agenten werden eingesetzt, um einen nachfolgend beschriebenen, beispielhaften Geschäftsprozess umzusetzen. Dabei wird das System als Pilotprojekt betrachtet, auf dessen Grundlage weitere Arbeiten folgen sollen. Dabei wird das zu entwickelnde System kein neues Framework darstellen, mit dessen Hilfe Agenten entwickelt werden können. Vielmehr soll eine Möglichkeit der Umsetzung von Geschäftsprozessen mit Hilfe von Agenten aufgezeigt werden. Folgende Arbeiten können das entwickelte System als Beispiel und Referenz heranziehen, um ähnliche Anforderungen schneller umzusetzen oder Umsetzungsideen für neue Anforderungen zu entwickeln. Auf dieser Basis können dann weitere Prozesse umgesetzt werden. Auch eine Ausweitung von einer rein virtuellen Simulation auf reale

[1] http://activiti.org/

Komponenten mit echten Sensoren und Aktoren ist denkbar.

Einen Überblick über die abstrakte Architektur bestehend aus vier Schichten zeigt Abbildung 6. Auf unterster Ebene stehen ein oder mehrere ERP-Systeme mit Geschäftsaufträgen. Dies sind für den Anfang einfache Produktions- und Transportaufträge, die später genauer spezifiziert werden. Als zweite Schicht folgt ein weiteres, in der Entwicklung befindliches Teilprojekt. Die Message Oriented Middleware (MOM) dient als Kommunikationsschnittstelle zwischen den Agenten und den ERP-Systemen. Außerdem soll sie, falls nötig, eine Möglichkeit der Kommunikation zwischen den Agenten selbst ermöglichen. Die vorliegende Arbeit beschäftigt sich mit der dritten Schicht. Ein Geschäftsprozess soll mit Hilfe eines zielorientierten Agentensystems realisiert werden. Die oberste Schicht verdeutlicht, dass es sich in der Zukunft hierbei nicht zwingend um eine simulierte Umgebung handeln muss.

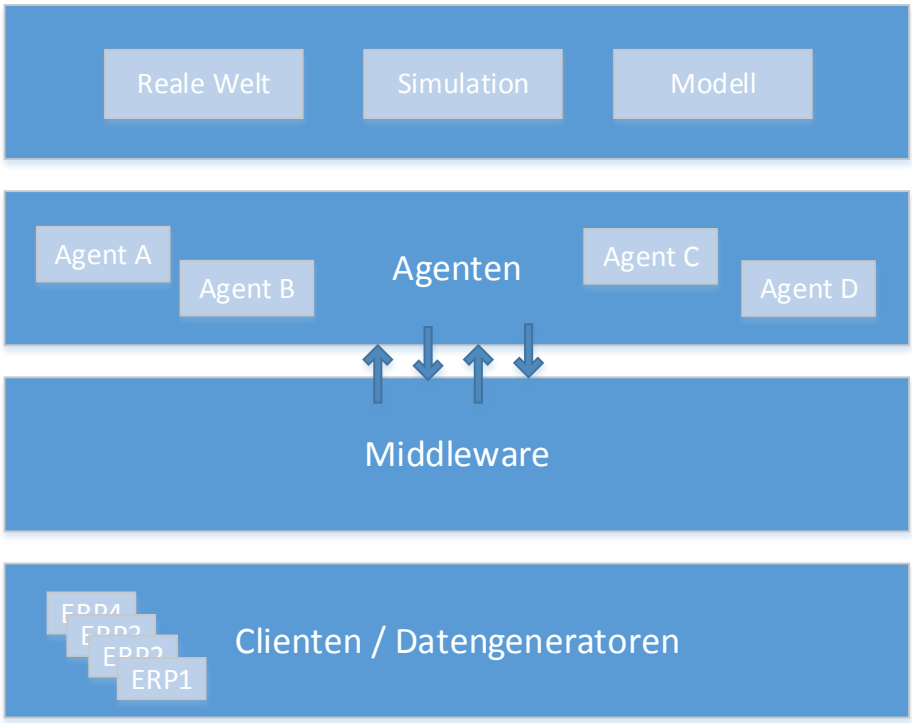

Abbildung 6: Architektur

4.2 Prozessbeschreibung

Ein einfacher Geschäftsprozess (siehe Abbildung 7) dient als Vorlage für das zu entwickelnde Agentensystem. Dieser bildet einen einfachen Auftrag, z.B. zur Produktion eines bestimmten Produktes, ab. Er besteht aus zwei Akteuren.

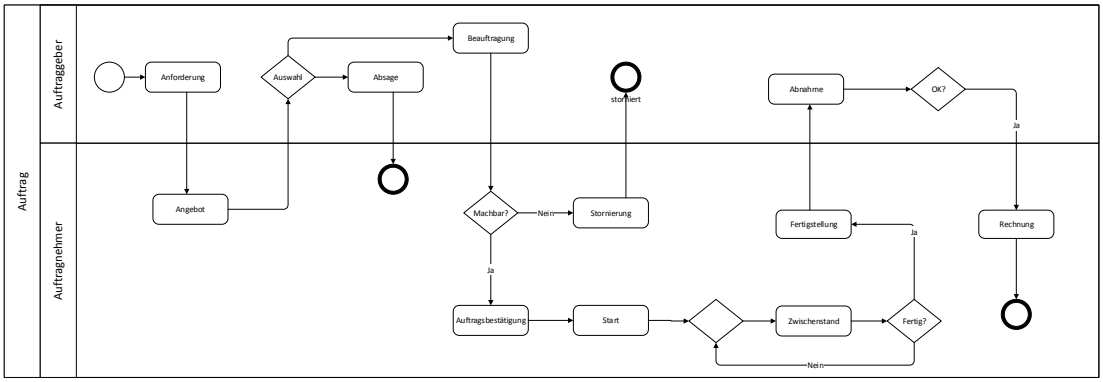

Abbildung 7: Prozessablauf

Der Auftraggeber ist prinzipiell nicht Teil des zu entwickelnden Systems, sondern ein menschlicher Akteur. Er erstellt einen Auftrag und übergibt ihn an ein ERP-System. Auf diesen Auftrag werden anschließend Angebote von potentiellen Auftragnehmern abgegeben. Der Auftraggeber wählt ein Angebot aus und bestätigt dieses. Die anderen Angeboten werden abgelehnt. Wenn die geforderte Bestellung produziert und abgegeben wurde, stellt er abschließend noch eine Rechnung. Da der Input in Form von Aufträgen sowie der Auftraggeber selbst als Kommunikationsgegenstelle benötigt wird, wird der Auftraggeber ebenfalls durch einen sehr einfachen Agenten repräsentiert, der diese Aufgaben übernimmt.

Der Auftragnehmer ist das eigentliche Agentensystem. Er repräsentiert ein Unternehmen zur Produktion beliebiger Produkte. Auf Basis seiner Umgebungseindrücke (Beliefs) und seiner vorgegebenen Ziele (Desires) wählt er geeignete Handlungsroutinen (Intentions) aus. Er ruft neue Aufträge ab, gibt Angebote ab, prüft die Machbarkeit, führt den Auftrag aus und informiert den Auftraggeber über die Fertigstellung.

Als Kommunikationsgegenstelle dient dem Auftragnehmer später ein MOM-Interface. Dieses Interface kapselt mehrere, darunterliegende ERP-Systeme und übernimmt die Kommunikation zwischen dem Agenten- und den ERP-Systemen. Da sich dieses Interface noch in der Entwicklung befindet, wird auf dieses vorläufig verzichtet und auf eine direkte Kommunikation des Auftraggeber-Agenten und des Auftragnehmer-Agenten gesetzt.

4.3 Ansatz

Die Entwicklung von BDI-Agenten ist kein völlig neues Programmierparadigma. Es bedeutet zum Beispiel nicht auf Objektorientierung zu verzichten. Ganz im Gegenteil - viele Prinzipien und Entwurfsmuster gelten für beide Welten. Vielmehr ist es ein auf Objektorientierte Programmierung aufbauendes Konzept. Allerdings erfordert die

Entwicklung von Agenten eine andere Herangehensweise als an übliche Software. Die BDI-Theorie versucht menschliche Verhaltensweisen bzw. die Entscheidungsfindung von Menschen nachzuahmen. Infolgedessen kann man sich einen Agent selbst während der Entwicklung ein wenig wie eine Person vorstellen. Durch "Fragen an diese Person" kann man sich wichtigen Eigenschaften des Agenten annähern. Zum Beispiel, *"Welche Ziele hast du? Was für Aktionen kannst du ausführen? Welche musst du ausführen? Was für Informationen benötigst du, um eine Entscheidung über eine Aktion treffen zu können?"*

Innerhalb des Auftragnehmers werden mehrere unterschiedliche Agententypen implementiert.

- BusinessAgent
- ProductionAgent
- TransportAgent
- UserAgent

Der BusinessAgent spielt die Rolle des Auftragnehmers, des Unternehmens als Ganzes. Dieser Agent folgt dem oben gezeigten Prozess. Das Unternehmen verfolgt meist Ziele, um den Gewinn zu maximieren oder nur profitable Aufträge anzunehmen und Ähnliches. Die beiden Agenten ProductionAgent und TransportAgent sind die eigentlichen Arbeiter bzw. Maschinen. Sie produzieren oder transportieren Produkte je nach Auftrag. Für diese beiden Typen lassen sich ebenfalls instinktiv mögliche Ziele definieren. Eine Produktionsmaschine darf z.B. nicht heiß laufen oder nur nachts arbeiten. Ein Transporter dagegen sollte nicht extra wegen eines kleinen Päckchens fahren, sondern beispielsweise nur, wenn der Laderaum gefüllt oder die Gewichtsgrenze beinahe erreicht wurde. Der UserAgent ist der bereits angesprochene Agentenersatz für den eigentlich nicht mehr zum System gehörenden Auftraggeber. Im Prinzip generiert er lediglich einige Aufträge und akzeptiert diese zufällig oder nach Eingangsreihenfolge.

Ziel dieser Arbeit ist es, ein System zu entwerfen, das nicht nur den vorgestellten Geschäftsprozess verarbeitet, sondern auch möglichst alle Kriterien (vgl. Abschnitt 2.2) eines Agenten erfüllt. Im Folgenden wird dargestellt, wie diese Eigenschaften umgesetzt werden sollen.

Autonomie Die Agenten arbeiten selbstständig an der Erfüllung ihrer vorgegebenen Ziele. Sie werden lediglich indirekt von außen beeinflusst, z.B. durch die Annahme eines Auftragsangebots durch den Auftraggeber. Es findet jedoch zu keinem Zeitpunkt eine direkte Steuerung in Form eines Direktzugriffs statt. Die Entscheidung wann und

welche Aktion ausgeführt wird, basiert ausschließlich auf Grundlage der Umgebung und der vorgegebenen Ziele.

Soziale Fähigkeiten Die Agenten müssen Informationen austauschen und miteinander kommunizieren. Dies kann zum einen indirekt passieren, z.B. durch das Abgeben eines Angebots. Das Angebot verändert den Status des Auftrags im ERP-System und führt somit zu einer indirekten Kommunikation. Eine andere Möglichkeit ist die direkte Kommunikation über Service-Schnittstellen. Dies wird zum Beispiel beim Zusammenspiel zwischen dem BusinessAgent und dem MachineAgent der Fall sein, da der MachineAgent keine Kenntnis des ERP-Systems hat. Er bekommt seine Produktionsaufträge direkt vom BusinessAgent mitgeteilt.

Reaktivität Eigenschaften wie z.B. die Anzahl an momentanen Aufträgen oder die Produktionsgeschwindigkeit werden als *Beliefs* implementiert. Bei einer Änderung dieser Eigenschaften ist das Agentenframework dann zuständig, einen neuen *Goal deliberation*-Prozess anzustoßen, um auf die veränderte Umgebung zu reagieren.

Proaktivität Zu Beginn werden einige Ziele für jeden Agententyp definiert. Diese Ziele können einmalig auszuführende Operation sein oder auch wiederkehrend, um einen bestimmten Zustand zu erreichen. Das Agentenframework sorgt dafür, dass für ein Ziel geeignete Aktionen ausgewählt und ausgeführt werden. Es hängt ganz von der Definition der Ziele ab, ob nur auf Veränderungen der äußeren Umgebung reagiert wird oder ob es Ziele gibt die proaktive Handlungen auslösen.

5 Implementierung

Das in Kapitel 4 vorgestellte System wird auf Basis von Java implementiert. Dies ermöglicht eine breite Kompatibilität über verschiedene Hardware-Plattformen und Software-Frameworks hinweg. Für die Entwicklung der BDI-Agenten wird das in Abschnitt 3.1 vorgestellte Framework *Jadex* verwendet. In diesem Kapitel wird auf die Umsetzung des Systems näher eingegangen.

5.1 Auftragstypen

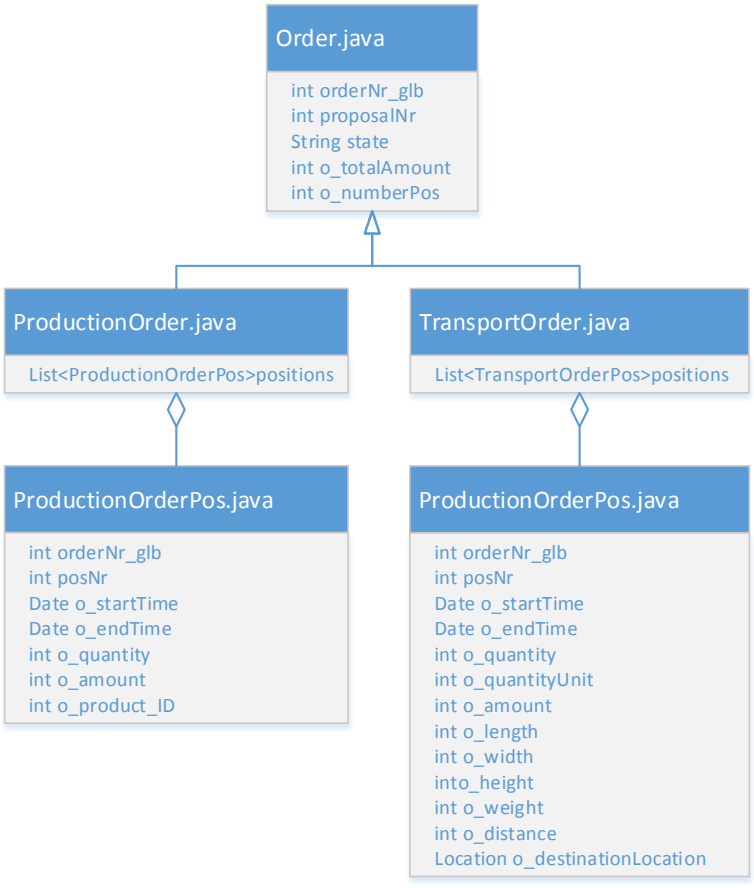

Abbildung 8: Ordertypen

Es werden insgesamt zwei Auftragstypen implementiert. Einmal für die Produktion von Produkten beliebiger Art und einmal für den Transport von Gütern. Abbildung 8 zeigt die beiden Typen als Java Klassen. Dabei sind einige Eigenschaften noch nicht vollständig umgesetzt. Diese können dann später noch erweitert werden. Jeder Auftrag,

egal ob für Produktion oder Transport, kann beliebig viele Positionen besitzen. Eine vollständige Auflistung aller möglichen Eigenschaften findet sich in den Tabellen 1, 2 und 3.

Innerhalb von Java wird die Klasse *Order.java* mit den für beide Typen gemeinsamen Eigenschaften implementiert. Die Klassen *ProductionOrder.java* und *TransportOrder.java* werden davon abgeleitet. Beide spezialisierten Order-Klassen besitzen eine Liste mit beliebig vielen Auftragspositionen (siehe Abbildung 8).

5.1.1 Produktionsauftrag

Eine Beschreibung der Eigenschaften eines Produktionsauftrags und einer Produktionsposition zeigen die Tabellen Tabelle 1 und Tabelle 2. Ein Produktionsauftrag besitzt zur eindeutigen Identifikation eine globale Auftragsnummer. Die Vorschlagsnummer ist wichtig, um den Agenten zu identifizieren, der das angenommene Angebot abgegeben hat. Wichtig ist zudem der Status, der bestimmt, ob ein Auftrag angenommen, abgelehnt oder fertiggestellt wurde (es gibt noch weitere Stati). Die restlichen Eigenschaften erklären sich aus den Beschreibungen der Tabellen von selbst.

Tabelle 1: production_order

Technischer Name	**Name**	**Beschreibung**
orderNr_glb	Globale Auftragsnummer	Global vergebene Auftragsnummer
proposalNr	Vorschlagsnummer	Jedes Angebot für eine Anfrage erhält eine aufsteigende Nummer (ab 1)
state	Status	Gesamtstatus des Auftrags (entspricht dem Status der hinterherhängendsten Position)
customer_ID	Auftraggeber	Globale ID
o_startTime	Startdatum und -uhrzeit	Frühstmöglicher Beginn der Auftragsbearbeitung (frühstes StartDate der Positionen)
o_endTime	Endedatum und -uhrzeit	Spätestes Ende der Auftragsbearbeitung (spätestes endDate der Positionen)
o_totalAmount	Gesamtbetrag	Preis
o_currency	Währung	ISO-Liste
o_numberPos	Anzahl der Position	Anzahl der Einzelpositionen
t_timestamp	Zeitstempel	Zeitstempel des Datensatzes

Tabelle 2: production_orderPos

Technischer Name	**Name**	**Beschreibung**
orderNr_glb	Globale Auftragsnummer	Global vergebene Auftragsnummer
proposalNr	Vorschlagsnummer	Jedes Angebot für eine Anfrage erhält eine aufsteigende Nummer (ab 1)
posNr	Positionsnummer	
state	Status	Gesamtstatus des Auftrags (entspricht dem Status der hinterherhängendsten Position)
o_startTime	Startdatum und -uhrzeit	Frühstmöglicher Beginn der Auftragsbearbeitung
o_endTime	Endedatum und -uhrzeit	Spätestes Ende der Auftragsbearbeitung
o_quantity	Menge	Zu produzierende Menge
o_unit	Mengeneinheit	ISO-Liste
o_product_ID	Produkt	Globale ID
o_amount	Betrag	Preis
o_currency	Währung	ISO-Liste
t_timestamp	Zeitstempel	Zeitstempel des Datensatzes

5.1.2 Transportauftrag

Der Transportauftrag besitzt nahezu die selben Eigenschaften wie auch der Produktionsauftrag. Daher wird auf die Tabelle dieses Typen verzichtet. Die Position eines Transportauftrags enthält jedoch einige kleine Veränderungen. So gibt es Eigenschaften, die die Art der Fracht näher beschreiben wie z.B. die Einheit, ob das Transportgut flüssig oder fest ist, sowie die Abmessungen und das Gewicht. Auch die Entfernung zum Zielort ist wichtig.

Tabelle 3: transport_orderPos

Technischer Name	**Name**	**Beschreibung**
orderNr_glb	Globale Auftragsnummer	Global vergebene Auftragsnummer
proposalNr	Vorschlagsnummer	Jedes Angebot für eine Anfrage erhält eine aufsteigende Nummer (ab 1)
posNr	Positionsnummer	
state	Status	Gesamtstatus des Auftrags (entspricht dem Status der hinterherhängendsten Position)
o_startTime	Startdatum und -uhrzeit	Frühstmöglicher Beginn der Auftragsbearbeitung
o_endTime	Endedatum und -uhrzeit	Spätestes Ende der Auftragsbearbeitung
o_quantity	Menge	Zu produzierende Menge
o_unit	Mengeneinheit	ISO-Liste
o_transportMode	Transportart	Schüttgut, flüssig, gasförmig, gekühlt, ...
o_length	Länge [m]	Länge einer Mengeneinheit (je nach TransportMode)
o_width	Breite [m]	Breite einer Mengeneinheit (je nach TransportMode)
o_height	Höhe [m]	Höhe einer Mengeneinheit (je nach TransportMode)
o_weight	Gewicht	Gewicht einer Mengeneinheit
o_weightUnit	Gewichtseinheit	ISO-Liste
o_distance	Entfernung in [km]	Entfernung in Kilometern
o_amount	Betrag	Preis
o_currency	Währung	ISO-Liste
t_timestamp	Zeitstempel	Zeitstempel des Datensatzes

5.2 ProductionAgent

Der ProductionAgent stellt eine gedachte Produktionseinheit dar. Im Anwendungsfall könnte das eine Maschine sein, die z.B. Löcher stanzt, Metalle zurechtbiegt oder ein Bauteil einsetzt oder Ähnliches. Hierbei kann natürlich unterschieden werden zwischen Maschinen, die einfach eine Aktion durchführen und Maschinen, die aus Teilprodukten oder Bauteilen ein neues Teilprodukt erstellen. Im ersten Fall könnte man sich eine Säge in einem Holzsägewerk vorstellen. Eine Maschine die Getränkeflaschen verschließt, benötigt dagegen eine Ressource (Kronkorken), um einen Auftrag zu bearbeiten.

Bei dem Beispiel des Holzsägewerks lassen sich recht einfach potentielle Ziele identifizieren. Das Sägeblatt läuft und teilt Baumstämme, sobald sie über ein Fließband ankommen. Man könnte sich vorstellen, dass ein Sägeblatt bei hoher Temperatur, d.h. hoher Benutzung schneller verschleißt. Ein Ziel dieser Maschine könnte also sein, die Temperatur unter einem bestimmten Wert zu halten, um die Lebenszeit des Sägeblatts zu optimieren. Ein weiteres Ziel wäre gleichzeitig, nicht mehr wie zwei Baumstämme auf dem Fließband warten zu lassen, da sonst die restliche Produktion still steht.

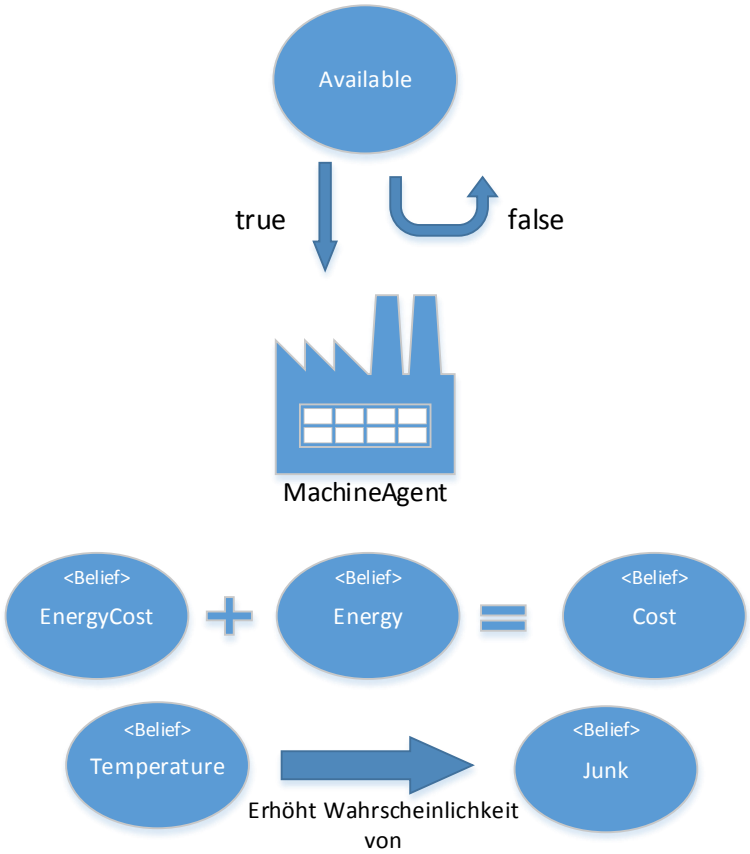

Abbildung 9: ProductionAgent

Der innere Aufbau eines Produktionsagenten wird in Abbildung 9 dargestellt. Jede Produktionseinheit kann immer nur ein Bauteil auf einmal produzieren. Ist die Maschi-

ne belegt, wird die Anfrage zurückgewiesen. Die Maschine kalkuliert ihre Kosten auf Basis des aktuellen Strompreises und ihres bekannten Stromverbrauchs. Sobald sich die Stromkosten (z.B. in der Nacht weniger als am Tag) oder der Stromverbrauch (Leerlauf vs. Beschäftigung) ändern, sollen die Kosten neu berechnet werden. In Listing 5.1 wird ein einfacher Plan gezeigt, der bei der Änderung einer der zwei genannten Beliefs aktiviert wird. Dabei handelt es sich um eine rein reaktive Maßnahme. Eine zielorientierte Aktion wird im Fall der Temperatur genutzt. In einem realen Fall, würden Sensoren die Temperatur messen. Da in diesem Beispiel die Maschine nur "virtuell" existiert, wird die Temperatur bei jeder Produktion programmatisch erhöht. Dabei führt eine erhöhte Temperatur zu einer erhöhten Ausschussrate (Junk). Um dies zu vermeiden wird das in Listing 5.2 dargestellte Ziel eingeführt. Solange die Temperatur unter 90°C liegt, ist das Ziel deaktiviert. Sobald die Schwelle erreicht wird, wird das Ziel aktiviert und nach geeigneten Handlungsroutinen gesucht. Das Ziel bleibt dabei solange aktiv, bis die Temperatur unter 80°C fällt. Ein auf dieses Ziel reagierender Plan könnte dann die Produktionsgeschwindigkeit reduzieren oder zeitweise pausieren, bis die Temperatur gefallen ist.

Listing 5.1: calcCosts

```
1  @Belief
2  int energy;
3  @Belief
4  int energyCost;
5  @Belief
6  int cost;
7
8  @Plan(trigger=@Trigger(factchangeds={"energy", "energyCost"}))
9  protected void calcCosts() {
10   cost = energyCost * energy;
11 }
```

Listing 5.2: MaintainLowTemp

```
1  @Goal
2  public class MaintainLowTemp {
3    /**
4     *  When the temperature is over 90 degree
5     *  the Agent will activate this goal.
6     */
7    @GoalMaintainCondition(beliefs="temp")
8    public boolean checkMaintain() {
9      return temp<90;
10   }
11
12   /**
```

```
13      *  The target condition determines when
14      *  the goal goes back to idle.
15      *  (wait until temperature is below 80 degree)
16      */
17    @GoalTargetCondition(beliefs="temp")
18    public boolean checkTarget() {
19      return temp<=80;
20    }
21  }
```

Die Pläne, die von den Goals bzw. Beliefs getriggert werden, setzen dann die eigentliche Funktionalität um.

5.3 UserAgent

Der UserAgent (vgl. Abbildung 10) spielt die Rolle des Auftraggebers und liefert Input in Form von Aufträgen. Außerdem dient er als Kommunkationspartner, um den Prozess in der dargestellten Form ablaufen zu lassen. Er implementiert ein Service Interface, um mit den anderen Agenten über Services zu kommunizieren. Das Interface *IImportOrders* ermöglicht Aufträge aus unterschiedlichen Quellen zu importieren. Zum Beispiel können Aufträge aus einem Excel Dokument geladen oder einfach generiert werden. Die generierten Aufträge sind alle identisch bis auf eine fortlaufende *orderNr_glb*.

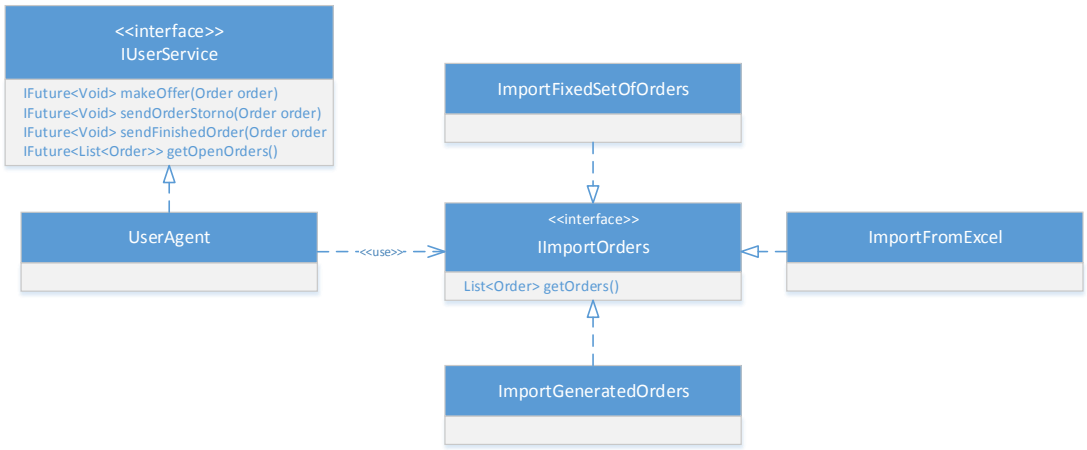

Abbildung 10: UserAgent

5.4 BusinessAgent

Der BusinessAgent (vgl. Abbildung 11) ist der eigentliche Auftragnehmer, also das Unternehmen. Er besitzt zwei Ziele. Zum einen sollen Aufträge von der Gegenseite abgerufen werden, zum anderen soll auf die erhaltenen Aufträge geboten werden. Die zugehörigen Pläne können entweder als Methode direkt in der Klasse definiert oder als eigene Klasse implementiert werden. Letzterer Fall hat den Vorteil, dass die Agenten-Klasse übersichtlicher bleibt und der gleiche Plan auch von anderen Agenten genutzt werden kann.

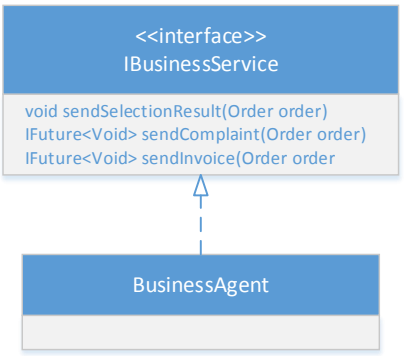

Abbildung 11: BusinessAgent

Listing 5.3: BusinessAgent

```
1  @Agent
2  @Plans({
3    @Plan(trigger=@Trigger(goals=BusinessAgentBDI.
         QueryGoal_RetrieveOpenOrders.class),
4      body=@Body(plans.GetOrdersPlan.class)),
5    @Plan(trigger=@Trigger(goals=BusinessAgentBDI.
         PerformGoal_MakeOffer.class),
6      body=@Body(plans.BidForProductionOrder.class)),
7    @Plan(trigger=@Trigger(goals=BusinessAgentBDI.
         PerformGoal_MakeOffer.class),
8      body=@Body(plans.BidForTransportOrder.class))
9  })
10 public class BusinessAgentBDI implements IBusinessService {
11 ...
12 }
```

Die Aufträge haben einen Start- und ein Endzeitpunkt. In diesem Zeitfenster müssen sie bearbeitet werden. Bevor der BusinessAgent die Produktion an einen Machine-Agent abgibt, wird bis zum Startzeitpunkt gewartet. Bisher entspricht die Wartezeit der Realzeit bis zum Eintreffen des definierten Startzeitpunkts. Für die Zukunft ist eine Funktion zur Zeitraffung geplant, um die Simulationszeit zu beschleunigen.

Der Ablauf zwischen Produktions- und Transportaufträgen unterscheidet sich voneinander. Bei Produktionsaufträgen wird geprüft, ob sich der Auftrag für das Unternehmen überhaupt lohnt. Hierzu wird der angenommene Ertrag des Auftrags errechnet. Ist der Auftrag profitabel, wird ein Angebot abgegeben. Bei Transportaufträge entfällt dies zunächst.

5.5 TransportAgent

Der TransportAgent kann beliebige Aufträge transportieren. Er repräsentiert einen LKW oder ein anderes denkbares Transportmittel. Er besitzt drei Beliefs (vgl. Abbildung 12). Sobald er einen neuen Auftrag erhält, berechnet er seinen Ladezustand neu. Sein definiertes Ziel hält ihn davon ab, mit praktisch leerer Ladefläche loszufahren. Erst wenn entweder seine Gewichtsgrenze oder sein Laderaum zu 90% erreicht bzw. gefüllt ist, wird er aktiv. Er ermittelt seine Zielkoordinaten für jede Auftragsposition. Jeder Fahrtweg aktiviert ein neues Subgoal mit dem Parameter *o_destinationLocation* der Auftragsposition. Das Ziel ist erfüllt, sobald alles Subgoals ebenfalls erfüllt wurden (vgl. Listing 5.4).

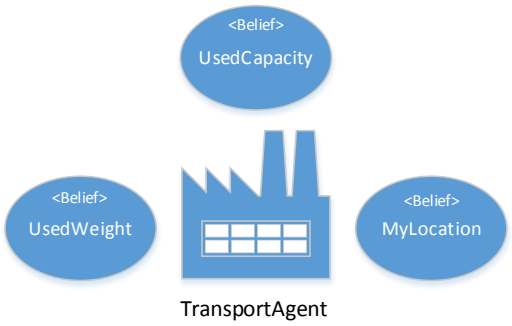

Abbildung 12: TransportAgent

Listing 5.4: TransportPlan

```
1  @Plan(trigger=@Trigger(goals=PerformGoal_Transport.class))
2  public class TransportPlan {
3
4    @PlanAPI
5    IPlan plan;
6
7    @PlanBody
8    public void transport() {
9      for(TransportOrder order : transportOrders) {
10       List<TransportOrderPos> orderPos = order.positions;
11
12       for(TransportOrderPos pos : orderPos) {
13         plan.dispatchSubgoal(new AchieveGoal_MoveTo(pos.
               o_destinationLocation)).get();
14         System.out.println("Order position delivered " + pos);
15       }
16     }
17   }
18 }
```

6 Evaluation

Nach der besprochenen Implementierung des Systems sollte es nun möglich sein, den Geschäftsprozess mit Hilfe der Agenten zu durchlaufen. Dabei kann nicht nur von einem, sondern auch von mehreren Agenten gleichzeitig gearbeitet werden. Außerdem sollte es auch keine Rolle spielen, auf welchem System sich die Agenten befinden. Über die Serviceaufrufe können sich die Agenten auch über Rechnergrenzen hinweg finden und kommunizieren.

6.1 Einzelne Agenten

Zunächst wird der einfachst mögliche Aufbau gewählt. Sobald das Projekt kompiliert und gestartet wird, startet als erstes die Jadex-Plattform. Diese ermöglicht das Ausführen der Agenten und ermöglicht das Finden und Kommunizieren mit Agenten auf anderen Plattformen (z.B. im lokalen LAN oder Internet). Es wird auch automatisch ein Passwort generiert, das den unberechtigten Zugriff von außen verhindert.

Listing 6.1: Output

```
1 Using stored platform password: 2fb00eb9-92b
2 Ice-PC_d5c platform startup time: 2876 ms.
3 Started platform: Ice-PC_d5c
4 [UserAgent@Ice-PC_d5c, Machine-1@Ice-PC_d5c, TransportAgent@Ice-
    PC_d5c, BusinessAgent@Ice-PC_d5c]
```

In der nächsten Ausgabe ist der Prozessablauf mit zwei einfachen Produktionsaufträgen abgebildet. Der BusinessAgent hat zwei neue Aufträge abgerufen. Anschließend wurden für beide Aufträge Angebote abgegeben und vom UserAgent positiv bestätigt. Der BusinessAgent gibt die Aufträge anschließend zur Produktion an einen Machine-Agent weiter. Nach jeweils 25% wird eine kurze Meldung ausgegeben.

Listing 6.2: Output

```
1 BusinessAgent: received new open orders (2)
2 -----------------
3 Machine-1:  Produce:  POrderPos[1|1] - [Thu Jan 01 00:00:00 CET
    2015 - Wed Dec 30 00:00:00 CET 2015] - 1 for 1000$
4 Machine-1: step(1/4) - OrderNo.: 1
5 Machine-1: step(2/4) - OrderNo.: 1
6 Machine-1: step(3/4) - OrderNo.: 1
```

```
7 Machine-1: step(4/4) - OrderNo.: 1
8 BusinessAgent: Machine returned: Fertigstellung
9 UserAgent: Received finished order: POrder[1]  - [1000$, 1Pos] -
     Fertigstellung
10 Machine-1:  Produce:  POrderPos[2|1] - [Thu Jan 01 00:00:00 CET
    2015 - Wed Dec 30 00:00:00 CET 2015] - 1 for 1000$
11 Machine-1: step(1/4) - OrderNo.: 2
12 Machine-1: step(2/4) - OrderNo.: 2
13 Machine-1: step(3/4) - OrderNo.: 2
14 Machine-1: step(4/4) - OrderNo.: 2
15 BusinessAgent: Machine returned: Fertigstellung
16 UserAgent: Received finished order: POrder[2]  - [1000$, 1Pos] -
     Fertigstellung
```

6.2 Multiple Agenten

Das vorige Beispiel bestand aus jeweils einem Agenten pro Agententyp. Im nächsten Fall werden mehrere MachineAgents gestartet. Die Verteilung der Aufträge auf die Agenten ist dann zufällig. Machine-1 hat den einen Auftrag bearbeitet, Machine-3 den anderen Auftrag. Auf die gleiche Art verhält es sich auch mit den TransportAgents.

Listing 6.3: Output

```
1 Machine-3:  Produce:  POrderPos[2|1] - [Thu Jan 01 00:00:00 CET
    2015 - Wed Dec 30 00:00:00 CET 2015] - 1 for 1000$
2 Machine-3: step(1/4) - OrderNo.: 2
3 Machine-3: step(2/4) - OrderNo.: 2
4 Machine-3: step(3/4) - OrderNo.: 2
5 Machine-3: step(4/4) - OrderNo.: 2
6 Machine-1:  Produce:  POrderPos[1|1] - [Thu Jan 01 00:00:00 CET
    2015 - Wed Dec 30 00:00:00 CET 2015] - 1 for 1000$
7 Machine-1: step(1/4) - OrderNo.: 1
8 Machine-1: step(2/4) - OrderNo.: 1
9 Machine-1: step(3/4) - OrderNo.: 1
10 Machine-1: step(4/4) - OrderNo.: 1
11 BusinessAgent: Machine returned: Fertigstellung
12 BusinessAgent: Machine returned: Fertigstellung
```

6.3 Verteilte Agenten

Die Abbildungen 13 und 14 zeigen, dass sich die Agenten auch durchaus auf unterschiedlichen Systemen befinden und trotzdem miteinander kommunizieren können. Die linke Grafik zeigt die Ausgabe auf einer lokalen Plattform, die rechte Grafik die Ausgabe von einer Plattform auf einem anderen Computer im lokalen LAN. Lokal wurden der UserAgent, BusinessAgent und TransportAgent gestartet. Auf der entfernten Plattform wurde lediglich der MachineAgent aktiviert. Die Agenten finden sich über die Discovery Mechanismen der Jadex-Plattformen und kommunizieren wie gewohnt über Serviceaufrufe miteinander. Somit kann theoretisch jede Maschine und jeder Transporter auf einem eigenständigen System laufen, solange eine Netzwerkverbindung besteht. Es könnten natürlich auch alle Agenten auf einem System laufen und jeder Agent übernimmt die Steuerung eines einzelnen externen Akteurs wie z.B. einem Lego Roboter.

```
Using stored platform password: 2fb00eb9-92b
Ice-PC_573 platform startup time: 4276 ms.
Started platform: Ice-PC_573
UserAgent
TransportAgent...
[UserAgent@Ice-PC_573, TransportAgent@Ice-PC_573, BusinessAgent@Ice-PC_573]
in getOrders
BusinessAgent: received new open orders (2)
bidForProductionOrder
bidForProductionOrder
BusinessAgent: Incoming order with state: Beauftragung
BusinessAgent: Order can be produced. Change state to: Auftragsbestätigung
POrder[1]  - [1000$, 1Pos] - Auftragsbestätigung
Used time slots: 2
-----------------
BusinessAgent: Incoming order with state: Beauftragung
BusinessAgent: Order can be produced. Change state to: Auftragsbestätigung
POrder[2]  - [1000$, 1Pos] - Auftragsbestätigung
Used time slots: 4
-----------------
BusinessAgent: Machine returned: Fertigstellung
BusinessAgent: Machine returned: Fertigstellung
UserAgent: Received finished order: POrder[1]  - [1000$, 1Pos] - Fertigstellung
UserAgent: Received finished order: POrder[2]  - [1000$, 1Pos] - Fertigstellung
```

Abbildung 13: Lokal

```
Using stored platform password: c1843aa9-8c4
TV-Client_97c platform startup time: 1599 ms.
Started platform: TV-Client_97c
MachineAgent...
[Machine-1@TV-Client_97c]
Machine-1:  Produce:  POrderPos[1|1] - [Thu Jan 01 00:00:00 CET 2015 - Wed Dec
Machine-1: step(1/4) - OrderNo.: 1
Machine-1: step(2/4) - OrderNo.: 1
Machine-1: step(3/4) - OrderNo.: 1
Machine-1: step(4/4) - OrderNo.: 1
Machine-1:  Produce:  POrderPos[2|1] - [Thu Jan 01 00:00:00 CET 2015 - Wed Dec
Machine-1: step(1/4) - OrderNo.: 2
Machine-1: step(2/4) - OrderNo.: 2
Machine-1: step(3/4) - OrderNo.: 2
Machine-1: step(4/4) - OrderNo.: 2
```

Abbildung 14: Remote

7 Fazit

7.1 Zusammenfassung

In dieser Arbeit wurde das Thema von Multi-Agenten-Systemen in Zusammenhang mit den Herausforderungen für die kommende Revolution zur Industrie 4.0 besprochen. Ziele wie höhere Flexibilität, geringere Kosten und die Möglichkeit zur Individualisierung von Produkten sind wichtige Bausteine, um in der Zukunft konkurrenzfähig zu sein.

Das in dieser Arbeit implementierte System hat gezeigt, dass es inzwischen relativ einfach möglich ist, autonome Multi-Agenten-Systeme zu entwickeln. Durch die Nutzung eines Frameworks wie Jadex, ist die Implementierung für Java-Entwickler verhältnismäßig einfach zu erlernen. Trotzdem gibt es auch hier zahlreiche Tücken. Die Umstellung in der Herangehensweise an die Entwicklung ist gerade zu Anfang doch enorm. Obwohl gerade der BDI-Ansatz an die menschliche Art des Denkens und Entscheidens angelehnt ist, so ist es nichtsdestotrotz schwierig, sich in die Konzepte von Zielen und Plänen einzudenken. Außerdem kann es schnell unübersichtlich werden. Häufig ist nicht ersichtlich, welcher Plan getriggert oder welches Ziel gerade aktiviert wird, da viel über Annotationen geregelt wird.

7.2 Bewertung

Obwohl schon seit über 30 Jahren geforscht wird, wie intelligente Agentensysteme entwickelt werden können, ist die Verbreitung leider noch relativ gering. Die Förderung der Regierung in Form von Forschungsgeldern ist durchaus ein Treiber der Technik. Jedoch muss sie auch von der Industrie akzeptiert und eingeführt werden. Hierfür ist es neben der Erforschung der Grundlagen wichtig, auch interessante Demonstrationsprojekte zu schaffen, die den Mehrwert für Unternehmen aufzeigen. Häufige Hemmnisse sind der oftmals hohe Kostenaufwand um solche Systeme für die einzelnen Bedürfnisse zu entwickeln und schließlich einzuführen. Vorhandene Prozessabläufe und Maschinen können nicht ohne große Anstrengung einfach ersetzt werden. Deswegen ist es umso wichtiger, den Entscheidungsträgern den Nutzen und die Vorteile für ihr Unternehmen der nächsten industriellen Revolution bewusst zu machen. Je größer die Akzeptanz und das Verständnis für die neuen Techniken sind, umso eher werden sie bei Entscheidungen, wie zum Beispiel beim Bau neuer Produktionswerke, berücksichtigt.

7.3 Ausblick

In der Zukunft werden durch die zunehmende Vernetzung von Komponenten viele neuartige Einsatzzwecke entstehen. Da die Umstellung von IPv4 auf IPv6 langsam an Fahrt gewinnt, werden nicht nur in der Industrie, sondern auch im Heimbereich immer mehr Geräte miteinander vernetzt werden. Daher wird es in Zukunft einen großen Bedarf an Systemen geben, die den Gerätepark intelligent koordinieren. Beispielsweise auch die Entwicklung im Automobilsektor mit immer mehr Assistenzsystemen fordert neuartige Konzepte, die sich auf unterschiedliche Anforderungen dynamisch einstellen können. Es ist daher durchaus denkbar, dass Multi-Agenten-Systeme sehr schnell Einzug halten. Allerdings weniger in der Industrie als im privaten Umfeld. Bis neue Produktionsanlagen entworfen und Fabriken gebaut werden, sind hohe Kosten und Entwicklungszeiten zu bewältigen. Im Konsumentenmarkt dagegen ist der Weg frei für sich schnell entwickelnde, neuartige Technologien. Wohin diese letzten Endes führen, wird sich vermutlich in wenigen Jahren zeigen.

Literaturverzeichnis

[Ada+11] Emmanuel Adam u. a. „Role-based manufacturing control in a holonic multi-agent system“. In: *International Journal of Production Research* 49 (5 2011), S. 1455–1468. ISSN: 0020-7543. DOI: 10.1080/00207543.2010.522086 (siehe S. 8).

[ALW05] Alexander Pokahr, Lars Braubach und Winfried Lamersdorf. „Jadex: A BDI Reasoning Engine“. In: *Multi-Agent Programming*. Hrsg. von R. Bordini, M. Dastani, J. Dix und A. El Fallah Seghrouchni. Springer Science+Business Media Inc., USA, 2005, S. 149–174 (siehe S. 13).

[AM91] Anand S. Rao und Michael P. Georgeff. *Modeling Rational Agents within a BDI-Architecture*. 1991 (siehe S. 9).

[AM95] Anand S. Rao und Michael P. Georgeff. „BDI Agents: From Theory to Practice“. In: *Proceedings of the First International Conference on Multi-agent Systems*. 1995, S. 312–319 (siehe S. 10).

[BPL08] Lars Braubach, Alexander Pokahr und Winfried Lamersdorf. „A Universal Criteria Catalog for Evaluation of Heterogeneous Agent Development Artifacts“. In: *From Agent Theory to Agent Implementation (AT2AI-6)* (2008), S. 19–28 (siehe S. 10, 11).

[BPL14] Lars Braubach, Alexander Pokahr und Winfried Lamersdorf. „Negotiation-Based Patient Scheduling in Hospitals“. In: *Advanced Intelligent Computational Technologies and Decision Support Systems*. Hrsg. von Barna Iantovics und Roumen Kountchev. Bd. 486. Studies in Computational Intelligence. Cham: Springer International Publishing, 2014, S. 107–121. ISBN: 978-3-319-00466-2. DOI: 10.1007/978-3-319-00467-9_10 (siehe S. 18).

[Bra87] Michael Bratman. *Intention, Plans, and Practical Reason*. Center for the Study of Language and Information, 1987. ISBN: 9781575861920 (siehe S. 9).

[Bro86] R. A. Brooks. „A robust layered control system for a mobile robot“. In: *Robotics and Automation, IEEE Journal of* 2 (1 1986), S. 14–23. ISSN: 0882-4967. DOI: 10.1109/JRA.1986.1087032 (siehe S. 10).

[Bro91] R. A. Brooks. „Intelligence Without Representation“. In: *Artificial Intelligence* 47 (1991), S. 139–159 (siehe S. 10).

[BS14] Andreas Bildstein und Joachim Seidelmann. „Industrie 4.0-Readiness: Migration zur Industrie 4.0-Fertigung“. In: *Industrie 4.0 in Produktion, Automatisierung und Logistik*. Hrsg. von Thomas Bauernhansl, Michael ten Hompel und Birgit Vogel-Heuser. Wiesbaden: Springer Fachmedien Wiesbaden, 2014, S. 581–597. ISBN: 978-3-658-04681-1. DOI: 10.1007/978-3-658-04682-8_30 (siehe S. 6).

[BST11] Dominik Böhnlein, Katharina Schweiger und Axel Tuma. „Multi-agent-based transport planning in the newspaper industry“. In: *Innsbruck 2008* 131 (1 2011), S. 146–157. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2010.04.006. URL: http://www.sciencedirect.com/science/article/pii/S092552731000126X (siehe S. 17).

[Bun06] Bundesministerium für Bildung und Forschung. *Die Hightech-Strategie für Deutschland*. Hrsg. von Bundesministerium für Bildung und Forschung. 2006. URL: https://www.bmbf.de/pubRD/bmbf_hts_lang.pdf (besucht am 04. 04. 2015) (siehe S. 4).

[Bun10] Bundesministerium für Bildung und Forschung. *Ideen. Innovation. Wachstum. Hightech-Strategie 2020 für Deutschland*. Hrsg. von Bundesministerium für Bildung und Forschung. 2010. URL: http://www.bmbf.de/pub/hts_2020.pdf (besucht am 05. 04. 2015) (siehe S. 4).

[Bun12] Bundesministerium für Bildung und Forschung. *Bericht der Bundesregierung. Zukunftsprojekte der Hightech-Strategie (HTS-Aktionsplan)*. Hrsg. von Bundesministerium für Bildung und Forschung. 2012. URL: http://www.bmbf.de/pub/HTS-Aktionsplan.pdf (besucht am 07. 04. 2015) (siehe S. 4).

[Bun13] Bundesministerium für Bildung und Forschung. *Zukunftsbild "Industrie 4.0"*. Hightech-Strategie. Hrsg. von Bundesministerium für Bildung und Forschung. 2013. URL: http://www.bmbf.de/pub/Zukunftsbild_Industrie_40.pdf (besucht am 08. 04. 2015) (siehe S. 4).

[Fre09] Vitalij Freese. „Portierbarkeit eines Multiagentensystems am Beispiel von Jadex und Whitestein LS/TS“. Hamburg: Hochschule für Angewandte Wissenschaften, 2009. 118 S. URL: http://opus.haw-hamburg.de/volltexte/2009/854 (siehe S. 12).

[GL86] Michael P. Georgeff und Amy L. Lansky. „Procedural knowledge“. In: *Proceedings of the IEEE* 74 (10 1986), S. 1383–1398. DOI: 10.1109/PROC.1986.13639 (siehe S. 9).

[GL87] Michael P. Georgeff und Amy L. Lansky. „Reactive Reasoning and Planning“. In: *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 2*. AAAI'87. AAAI Press, 1987, S. 677–682. ISBN: 0-934613-42-7. URL: http://dl.acm.org/citation.cfm?id=1863766.1863818 (siehe S. 9, 10).

[GSV99] Christian Gerber, Jörg Siekmann und Gero Vierke. „Holonic multi-agent systems“. In: (1999) (siehe S. 8).

[Koe89] Arthur Koestler. *The ghost in the machine*. London: Arkana, 1989. 384 S. ISBN: 9780140191929 (siehe S. 8).

[Las+14] Heiner Lasi u. a. „Industrie 4.0“. In: *Wirtschaftsinformatik* 56 (4 2014), S. 261–264. ISSN: 0937-6429. DOI: 10.1007/s11576-014-0424-4 (siehe S. 5, 6).

[Lei09] Paulo Leitão. „Agent-based distributed manufacturing control: A state-of-the-art survey“. In: *Distributed Control of Production Systems* 22 (7 2009), S. 979–991. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2008.09.005. URL: http://www.sciencedirect.com/science/article/pii/S0952197608001437 (siehe S. 16).

[Min14] Ministerium für Finanzen und Wirtschaft. *Strukturstudie "Industrie 4.0 für Baden-Württemberg"*. Hrsg. von Ministerium für Finanzen und Wirtschaft. 2014. URL: https://mfw.baden-wuerttemberg.de/de/service/publikation/did/strukturstudie-industrie-40-fuer-baden-wuerttemberg/ (siehe S. 7).

[MN95] Michael Wooldridge und Nicholas R. Jennings. „Intelligent agents: Theory and practice“. In: *The Knowledge Engineering Review* 10 (2 1995), S. 115–152 (siehe S. 9).

[Mül99] Jörg P. Müller. „Architectures and applications of intelligent agents: A survey“. In: *The Knowledge Engineering Review* 13 (04 1999), S. 353–380 (siehe S. 11, 16).

[Pau+06] Torsten O. Paulussen u. a. „Agent-Based Patient Scheduling in Hospitals“. English. In: *Multiagent Engineering*. Hrsg. von Stefan Kirn u. a. International Handbooks on Information Systems. Springer Berlin Heidelberg, 2006, S. 255–275. ISBN: 978-3-540-31406-6. DOI: 10.1007/3-540-32062-8_14. URL: http://dx.doi.org/10.1007/3-540-32062-8_14 (siehe S. 18).

[PBJ13] Alexander Pokahr, Lars Braubach und Kai Jander. „The Jadex Project: Programming Model“. In: *Multiagent Systems and Applications*. Hrsg. von Maria Ganzha und Lakhmi C. Jain. Bd. 45. Intelligent Systems Reference Library. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, S. 21–53. ISBN: 978-3-642-33322-4. DOI: 10.1007/978-3-642-33323-1_2 (siehe S. 18).

[Pos07] Stefan Poslad. „Specifying Protocols for Multi-agent Systems Interaction“. In: *ACM Trans. Auton. Adapt. Syst.* 2 (4 2007). ISSN: 1556-4665. DOI: 10.1145/1293731.1293735. URL: http://doi.acm.org/10.1145/1293731.1293735 (siehe S. 12).

[Pro12] Prof. Dr. Henning Kagermann, Prof. Dr. Wolfgang Wahlster, Dr. Johannes Helbig. *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0. Abschlussbericht des Arbeitskreises Industrie 4.0*. Vorabversion. Hrsg. von Promotorengruppe Kommunikation der Forschungsunion Wirtschaft – Wissenschaft. 2012. URL: http://www.forschungsunion.de/pdf/industrie_4_0_umsetzungsempfehlungen.pdf (besucht am 07. 04. 2015) (siehe S. 7).

[Pro13] Prof. Dr. Henning Kagermann, Prof. Dr. Wolfgang Wahlster, Dr. Johannes Helbig. *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0. Abschlussbericht des Arbeitskreises Industrie 4.0*. Hrsg. von Promotorengruppe Kommunikation der Forschungsunion Wirtschaft – Wissenschaft. 2013. URL: http://www.forschungsunion.de/pdf/industrie_4_0_abschlussbericht.pdf (besucht am 07. 04. 2015) (siehe S. 4, 6).

[Spa13] Dieter Spath, Hrsg. *Produktionsarbeit der Zukunft - Industrie 4.0*. Unter Mitarb. von Oliver Ganschar u. a. Stuttgart: Fraunhofer-Verl, 2013. 150 S. ISBN: 3839605709 (siehe S. 5–7).

[Spi15] Spiegel Online. *Exportrekord: Deutschland knackt wieder die Billionenmarke*. 2015. URL: http://www.spiegel.de/wirtschaft/soziales/exporte-neuer-rekord-im-jahr-2014-a-1017416.html (besucht am 19. 04. 2015) (siehe S. 1).

[Woo02a] Michael Wooldridge. „Intelligent Agents: The Key Concepts“. In: *Multi-Agent Systems and Applications II*. Hrsg. von G. Goos u. a. Bd. 2322. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, S. 3–43. ISBN: 978-3-540-43377-4. DOI: 10.1007/3-540-45982-0_1 (siehe S. 8, 9).

[Woo02b] Michael J. Wooldridge. *An introduction to multiagent systems*. New York: J. Wiley, 2002. 348 S. ISBN: 9780470849392 (siehe S. 11).

[ZKM13] Christoph Zanker, Steffen Kinkel und Spomenka Maloca. *Globale Produktion von einer starken Heimatbasis aus. Verlagerungsaktivitäten deutscher Unternehmen auf dem Tiefstand*. Karlsruhe, 2013. URL: http://www.isi.fraunhofer.de/isi-wAssets/docs/i/de/pi-mitteilungen/PI63.pdf (besucht am 02. 04. 2015) (siehe S. 2).

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Stellen der Arbeit die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

Freiburg, den 30.04.2015 Steffen Ritter