Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Anwendungen 1

Gregor Balthasar

Dezentrale Koordination von Jadex–BDI–Agenten

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1. Einführung

Da *Multiagentensysteme* (MAS), ein noch junger Bereich der Informatik, für spezifische Problemstellungen einer immer größer werdenden Bedeutung unterliegen, ist die *Agentenorientierte Softwareentwicklung* (AOSE) ein bedeutendes Forschungsfeld im *Multimedia Systems Laboratory* (MMLab) der HAW Hamburg. Der Forschungsschwerpunkt liegt dabei auf der Entwicklung von Methodiken und Modellen für dezentral koordinierte, selbstorganisierende MAS. Für Lehr- und Forschungszwecke wird dabei die Jadex[1] Multiagentenplattform zur Entwicklung von MAS eingesetzt. Jadex wird entwickelt vom Labor für *Verteilte Systeme und Informationssysteme* (VSIS)[2] am Department Informatik der Universität Hamburg. Ebenfalls in Kooperation mit VSIS läuft das DFG-finanzierte Gemeinschaftsprojekt *Selbstorganisation durch Dezentrale Koordination in Verteilten Systemen* (SodekoVS) [Sudeikat u. a. (2009)]. Dieses Projekt beschäftigt sich mit der Nutzung von naturinspirierten Selbstorganisationsprinzipien zur Erstellung adaptiver verteilter Systeme. Für diese Arbeit relevante Teile des Projektes werden in einem späteren Abschnitt näher erläutert.

Da ein umfassender Überblick über Multiagentensysteme den Rahmen dieser Arbeit sprengen würde, muss an dieser Stelle noch eine Eingrenzung erfolgen. Durch langjährige Erfahrungen im Bereich der Multiagentensysteme haben sich einige Probleme herauskristallisiert, die der Entwicklung von dezentral koordinierten MAS erschwerend im Weg stehen.

Infolgedessen soll diese Arbeit ein Verständnis für diese Problematik schaffen und in einer Vision einen möglichen Lösungsansatz kurz diskutieren.

1.1. Inhalt dieser Arbeit

Um dies zu erreichen, werden im folgenden Abschnitt 2 zuerst grundlegende Begrifflichkeiten aus dem Forschungsfeld AOSE geklärt und ggf. erläutert, um ein grundsätzliches Verständnis zu ermöglichen, sowie eine kurze Einführung in die Jadex Multiagentenplattform gegeben. Zuletzt werden in diesem Abschnitt noch die Kommunikationsmedien von SodekoVS vorgestellt, auf deren möglichen Einsatz dann in einem späteren Abschnitt noch zurückgekommen wird. In Abschnitt 3 werden dann Probleme bei der Implementation von Koordinationsaufgaben mit Jadex anhand der Erfahrungen im Multiagent-Contest aufgezeigt, für die dann in Abschnitt 4 erste mögliche Lösungsansätze formuliert werden. Abschließend wird in Abschnitt 5 eine Zusammenfassung dieser Arbeit geliefert.

[1] zu finden unter: http://jadex.informatik.uni-hamburg.de/
[2] http://vsis-www.informatik.uni-hamburg.de/

2. Grundlagen

2.1. Multiagentensysteme

Als Multiagentensysteme bezeichnet man solche Systeme, die aus mehreren homogenen oder heterogenen, autonom agierenden und kommunizierenden, kleineren Subsystemen, bekannt als *Agenten*, bestehen [Wooldridge (2001)]. MAS fallen in den Bereich der Verteilten Künstlichen Intelligenz und beschäftigen sich damit, komplexe Probleme durch Koordination und Kooperation von Agenten zu lösen. Dabei spielt die Abstimmung des spezifischen Wissens, sowie der Ziele und Pläne einzelner Agenten eine große Rolle.

Ein tieferer Einblick in die Thematik kann durch Lektüre des oben zitierten Buches von Wooldridge gewonnen werden; weiterhin befinden sich im Literaturverzeichnis auch noch zusätzliche, nicht zitierte aber dennoch empfehlenswerte Werke [Ferber (2001); Weiss (2000)].

Es bestehen verschiedenste Konzepte zur Realisierung von Agenten, deren *weiche* Anforderungen laut [Wooldridge und Jennings (1995)] folgende sind:

- Autonomität: Agenten operieren ohne direktes Eingreifen durch Menschen.
- Soziale Fähigkeit: Agenten kommunizieren mit anderen Agenten über eine *Agent Communication Language* (ACL).
- Reaktivität: Agenten nehmen ihre Umgebung wahr und reagieren auf Veränderungen in dieser.
- Pro-Aktivität: Agenten reagieren nicht nur, sondern können ein zielgerichtetes Verhalten entwickeln und somit *die Initiative ergreifen*.

2.2. BDI-Agenten

Ein Ansatz, der sich immer stärker werdender Beliebtheit erfreut, ist die sogenannte BDI-Architektur (*belief, desire, intention*). Der grundlegende Gedanke dieses Modells ist philosophischer Natur; man geht dabei davon aus, dass Menschen ebenfalls nur ein theoretisches Modell der Welt haben und anhand dessen ihr *Reasoning*[3] betreiben. BDI-Agenten könnten somit eine Abstraktion eines menschlichen Akteurs in einem System repräsentieren.

Beliefs repräsentieren dabei das „Wissen“ des Agenten über die ihn umgebende Welt. *Desires* oder auch *Goals* stehen für die Motivationen eines Agenten und sind meist hierarchisch angeordnet. *Intentions* spiegeln den deliberativen Status eines Agenten wieder, sie zeigen also an, wofür sich ein Agent entschieden hat. Schließlich gibt es noch *Plans*, diese bestehen aus

[3] Reasoning = Entscheidungsfindung

Abfolgen von Aktionen und ermöglichen dem Agenten bestimmte *Intentions* zu erfüllen. Auf diese Elemente wird in Abschnitt 2.5 nochmals näher eingegangen.

2.3. Koordination in MAS

Koordination ist ein zentrales Problem kooperativer Arbeit, welches immer dann auftritt, wenn sich Aktionen mehrerer Agenten in irgendeiner Form überschneiden oder Einfluss aufeinander nehmen können. Dabei kann man dem Problem Koordination sowohl mit zentralisierten, als auch mit dezentralisierten Ansätzen entgegengehen. Im Folgenden werden nur die dezentralisierten Ansätze betrachtet werden, da diese tatsächlich eher dem Grundgedanken eines Multiagentensystems entsprechen (viele Agenten arbeiten gemeinsam an der Lösung eines Problems) als zentralisierte Ansätze; weiterhin sind zentralisierte Ansätze immer mit der Problematik des *Single point of failure* (SPOF) behaftet.

[Wooldridge (2001)] nennt folgende Ansätze zur Lösung des Problems der dezentralen Koordination:

- Koordination durch partielles globales Planen.
- Koordination durch verbundene *Intentions*.
- Koordination durch wechselseitige Modellierung.
- Koordination durch Normen und soziale Gesetze.
- Koordination durch Kommunikation (Wissensaustausch, Abstimmung, etc.).

Eine genaue Beschreibung dieser Ansätze findet sich ebenfalls in [Wooldridge (2001)].

2.4. MAS: State of the Art

Multiagentensysteme werden heutzutage hauptsächlich unter Nutzung von Multiagentenplattformen entwickelt. Solche Plattformen sind eine Form von Middleware und bieten grundlegende Dienste zur Realisierung eines Multiagentensystems. Die *Foundation for Intelligent Physical Agents* (FIPA)[4] hat dabei einen Standard für die generelle Architektur einer Multiagentenplattform formuliert [FIPA b], Abbildung 1 auf Seite 7 zeigt ein zugehöriges Architekturmodell. Die Hauptanforderungen dieser Spezifikationen an eine Multiagentenplattform bestehen darin, dass sie Möglichkeiten zur Erstellung, Lokalisierung und Zerstörung von Agenten bietet. Weiterhin ist spezifiziert, dass sie Agenten die Möglichkeit zur Kommunikation untereinander bereitstellt.

[4] zu finden unter: http://www.fipa.org

Es existieren verschiedene Multiagentenplattformen, die die FIPA Standards nutzen, z.B. JACK[5], JADE[6] oder Jadex[1]. Nähere Informationen zu diesen und noch weiteren Plattformen lassen sich auch auf der Homepage der FIPA finden. Auffällig ist, dass ein Großteil dieser Plattformen auf Java™ aufsetzt.

Weiterhin bestehen verschiedene Methodiken zur Entwicklung von Multiagentensystemen, prominente Vertreter sind z.B. Tropos [Troposproject], Prometheus [RMIT] oder AUML [FIPA a]. Diese stellen verschiedene Toolsets und Leitfäden zur Entwicklung von MAS bereit. Den „einen“, festen Standard gibt es allerdings nicht.

[5] zu finden unter: http://www.agent-software.com
[6] zu finden unter: http://jade.tilab.com

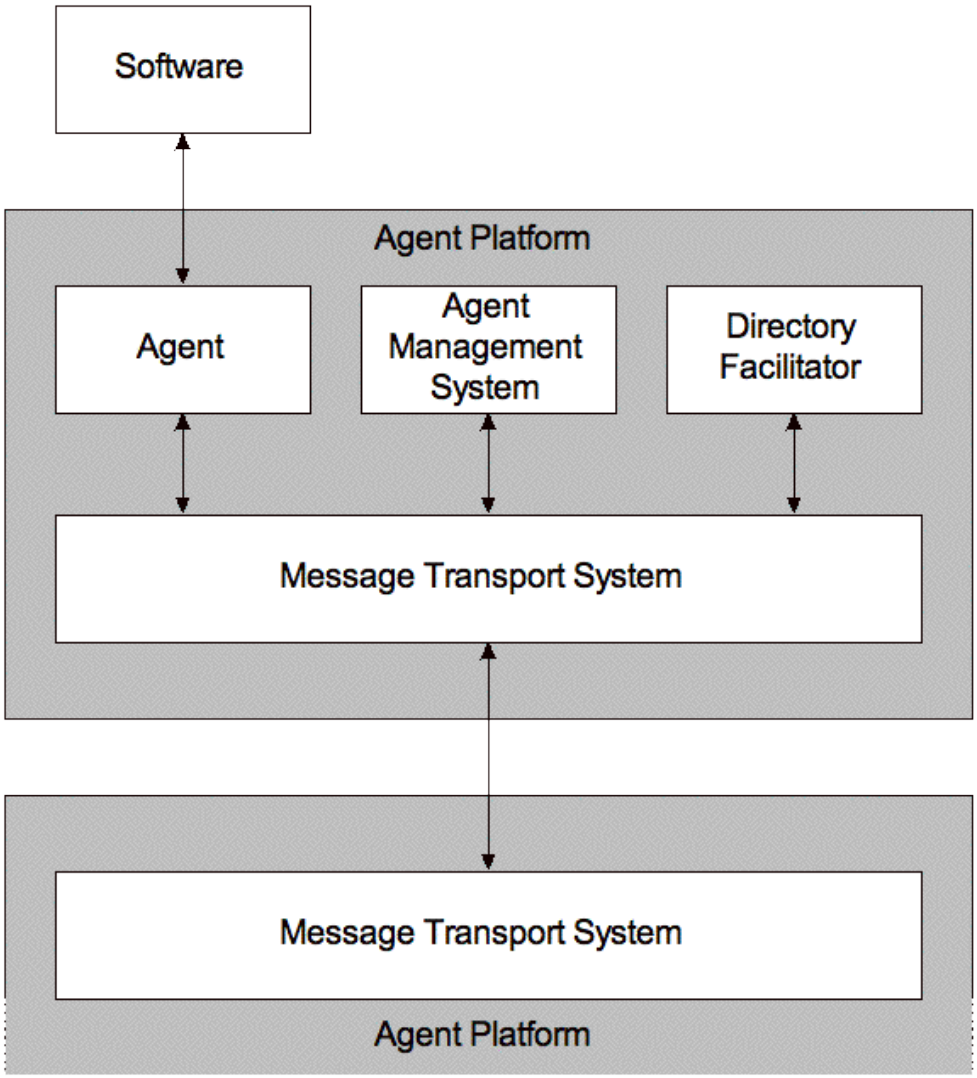

Abbildung 1: Agent Management Reference Model [FIPA b]

2.5. Jadex

Die *Jadex* Multiagentenplattform [Pokahr u. a. (2005a,b) stellt ein Programmier-Framework und eine Ausführungs-Middleware zur Entwicklung von Goal-orientierten Agenten, die der BDI-Architektur folgen, bereit. Die grundlegenden Programmierkonzepte sind dabei *Beliefs*, *Goals* und *Plans*. Beliefs strukturieren das Wissen des Agenten, sowohl über seinen eigenen Status als auch über den Status der von ihm wahrgenommenen Umgebung. Beliebige Java™Objekte können in der sogenannten *Beliefbase* eines Agenten gespeichert werden. Goals sind Beschreibungen von Zuständen, die aktiv werden können, um den Agenten zum Erreichen eines solchen Zustands zu veranlassen. Dabei können Konditionen und Invarianten an Goals annotiert werden, um zu kontrollieren, wann ein Agent veranlasst wird, ein Goal zu verfolgen. Plans realisieren die prozeduralen Mittel eines Agenten und sind in der Regel mit Goals oder Events verknüpft. Sie beinhalten die prozedurale Logik der Aktivitäten eines Agenten, wobei das agenteninterne Reasoning die Ausführung von Plänen zum Erreichen von Goals steuert. Zusätzlich wurde das *Capability* Konzept [Busetta u. a. (2000)] erweitert [Braubach u. a. (2005)], um die Modularisierung von Jadex-Agenten zu ermöglichen, bzw. zu vereinfachen. Capabilities kapseln Sets aus Beliefs, Goals und Plans, wobei die resultierenden Module dann wiederverwendbare funktionale Blöcke bereitstellen.

Jadex-Agenten verwenden *Reactive Planning* (Reaktives Planen) [Wooldridge (2001)]. Pläne werden manuell zur Design-Zeit entwickelt und der Jadex-Reasoner vollzieht zur Laufzeit sowohl die *Goal-Deliberation*, als auch das *Means-Ends Reasoning* [Wooldridge (2001)]. Die Goal-Deliberation betrifft die Selektion von Zielen, die verfolgt werden sollen, und das Means-Ends Reasoning kontrolliert, welche Plans ausgewählt werden, um die momentan aktiven Goals zu erreichen.

Ein Jadex-Agent besteht aus zwei Hauptkomponenten. Da ist zum einen die Struktur des Agenten, die in einem XML-Dokument definiert wird, dem sogenannten *Agent Definition File* (ADF).[7] Diese Datei beschreibt die initiale Struktur von Agententypen (die zuvor erwähnten Konzepte wie *Beliefs*, *Goals*, etc. werden an dieser Stelle deklariert). Die Plans, also das prozedurale Wissen eines Agenten, zum anderen werden in normalen Java™Klassen implementiert.[8] Die konsistente Adaption von weithin akzeptierten Programmierungstechnologien impliziert hier die (Wieder-)Verwendbarkeit von Java™-basierten Programmierumgebungen und Frameworks. Weiterhin bietet Jadex verschiedene Werkzeuge zum Plattform-Management, zur Observation von Agenten und deren Zuständen sowie zur Entwicklung von Agenten. Weitere Details lassen sich in [Pokahr u. a. (2005a,b)] sowie auf der Homepage des Projektes[1] finden.

[7] vgl. Abbildung 2, die die Grundzüge eines ADFs zeigt
[8] vgl. Abbildung 3, die beispielhaft die Nutzung der Jadex *Plan* Klasse zeigt

```
<agent xmlns="http://jadex.sourceforge.net/jadex"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex-0.96.xsd"
       name="myName"
       package="jadex.samplepackage.agent">

 "..."

 <beliefs>
  "..."
 </beliefs>

 <goals>
  "..."
 </goals>

 <plans>
  "..."
 </plans>



 <events>
  "..."
 </events>

 "..."
</agent>
```

Abbildung 2: Ein beispielhaftes Jadex *Agent Definiton File* (ADF).

```
package jadex.samplepackage.agent;

import jadex.samplepackage.agent.*;
import jadex.runtime.*;
import jadex.util.SUtil;


/**
 *  SamplePlan Class which extends the Jadex provided Plan Class
 */
public class SamplePlan extends Plan
{
        /**
         *  Create a new plan.
         */
        public SamplePlan(){}

        /**
         *  The plan body. Programcode of the plan is placed here.
         */
        public void body()
        {
                // accessing the Beliefbase
                getBeliefbase().getBelief("beliefname").getFact());

                // dispatching Subgoals out of a plan
                IGoal samplegoal = createGoal("goalname");
                samplegoal.getParameter("parametername").setValue("somevalue");
                dispatchSubgoalAndWait(samplegoal);
        }
}
```

Abbildung 3: Ein beispielhafter Jadex Plan.

2.6. Koordinationsmedien in SodekoVS

SodekoVS stellt eine Referenz-Architektur sowie eine Beispielimplementation in Jadex zur Koordination von selbstorganisierenden Systemen bereit [Sudeikat u. a. (2009)]. Hauptaugenmerk liegt hierbei auf den sogenannten Koordinationsmedien. Deren Funktion orientiert sich unter anderem an natürlichen Koordinationsmechanismen, z.B. der Funktionsweise von Ameisenpheromonen. Somit steht eine fein granulierte Auswahl der Adressaten für Nachrichten bzw. Sprechakte zur Verfügung, die sich an einmalig festzulegenden Parametern (wie z.B. der Halbwertszeit und der Entfernung im Falle der Ameisenpheromone) orientiert. Die Koordinationsmedien greifen beim Transport der Nachrichten auf die plattformspezifischen Kommunikationsmittel zurück. Zur Verfügung stehen die Koordinationsfunktionen von SodekoVS in Form eines Interfaces, gegen das bei der Entwicklung eines Agenten programmiert werden muss.

3. Implementation dezentraler Koordinationsaufgaben in Jadex am Beispiel Multiagent-Contest

Der Multiagent-Contest ist ein von einem Team rund um Prof. Dr. Jürgen Dix[9] jährlich ausgerichteter Wettbewerb. Ausgeschriebenes Ziel des Wettbewerbs ist es, die Forschung im Bereich der Multiagentensystementwicklung und -programmierung durch Identifizierung von Schlüsselproblemen und der Sammlung passender Benchmarks anzuregen.

Der Wettbewerb wird seit 2005 jährlich durchgeführt, wobei er seit 2007 im Rahmen des *Programming Multiagent Systems Languages and tools* (ProMAS) Workshops[10] stattfindet, die zwei vorherigen Wettbewerbe fanden im Rahmen des *Computational Logic in Multi-Agent Systems* (CLIMA) Workshops[11] statt. Der genannte ProMAS Workshop ist wiederum Teil der *International Conference on Autonomous Agents and Multiagent Systems* (AAMAS)[12].

Das MMLab der HAW Hamburg hat in den Jahren 2008 und 2009 mit jeweils speziell für diese Wettbewerbe entwickelten MAS am Multiagent-Contest teilgenommen und konnte dabei in beiden Jahren den zweiten Platz erreichen.[13] Im Folgenden werden die dabei gewonnenen Erfahrungen bezüglich der Implementation dezentraler Koordinationsaufgaben in Jadex kurz diskutiert.

[9] Professor an der Universität Clausthal, Lehrstuhl für „Computational Intelligence“
[10] zu finden unter: http://www.cs.uu.nl/ProMAS/
[11] zu finden unter: http://research.nii.ac.jp/climaVII/
[12] zu finden unter: http://www.cse.yorku.ca/AAMAS2010/
[13] nähere Informationen zum Multiagent-Contest 2008 finden sich in [Behrens u. a. (2008)]

3.1. Hürden bei der Implementation

Aufgabe im Contest war es, virtuelle Kuhherden mit einer Gruppe von sechs bzw. zehn Cowboy-Agenten[14] in ein Gatter zu treiben, wobei es galt Hindernissen auszuweichen, Zäune offenzuhalten und mit einer Gruppe von gegnerischen Cowboy-Agenten zu konkurrieren. Um die dafür benötigte Koordination erzielen zu können, war es nötig, dass die Agenten Wissen über die Umgebung miteinander austauschten, sowie weitere Kommunikation zur Abstimmung ihrer Aktivitäten[15] betrieben. Um Sprechakte mit Jadex zu implementieren, müssen für jeden Sprechakt sowohl umfangreiche Messageevent-Konstrukte im ADF erstellt werden, sowie Plans, die das Senden bzw. das Empfangen einer Nachricht handhaben. Dabei ist jeder Sprechakt auf ein mitzusendendes Java™-Objekt einer definierten Klasse beschränkt, wodurch kaum Adaptivität gegeben ist. Es ist also nicht möglich, Goals direkt kooperativ zu bestreiten, große Umwege über Artefakte in der Beliefbase eines Agenten und dadurch ausgelöste Konditionen von Goals müssen für jede Koordinationsaktivität einzeln und händisch implementiert werden. Auch späte kleine Änderungen werden dadurch sehr aufwendig, da diese ebenfalls händisch in allen genannten Komponenten durchgeführt werden müssen.[16]

Insgesamt gesehen war der Implementationsaufwand, der aus der geringen Modularität und Adaptivität dieser Konstrukte und der fehlenden Koordinationsunterstützung in Jadex entstanden ist, deutlich höher, als der Aufwand für die verschiedenen funktionalen Aktivitäten der Agenten im MAS.

3.2. Problem: Kommunikationsoverhead

Ein weiteres signifikantes Problem, das während der Entwicklung der Prototypen für den Contest immer wieder auftrat, war entstehender Kommunikationsoverhead durch zu viele Sprechakte mit zu großen Datenmengen, der den Nachrichtentransport bis zur nicht mehr gegebenen Funktionalität des MAS verlangsamen konnte. Hierzu muss erklärt werden, dass bei den FIPA-konformen Sprechakten in Jadex mitgesendete Java™-Objekte serialisiert und als XML transportiert werden. Dies ist aus Sichtweise der Interoperabilität natürlich positiv zu bewerten, schränkt die Performance allerdings ein. Entgegenwirken ließ sich dem nur durch die drastische Reduktion der Menge der transportierten Daten, was wiederum Einschränkungen bei der Koordination zur Folge hatte.

[14] sechs Agenten im Multiagent-Contest 2008, zehn Agenten 2009

[15] z.B. welcher Agent welche Herde treibt

[16] IDEs wie z.B. Eclipse besitzen auch keine Verknüpfung zwischen XML und Java-Klassen zur Entwicklerunterstützung, was das Refactoring nochmals bedeutend erschwert

3.3. Fazit

In diesem Abschnitt wurde gezeigt, dass die Implementation von Koordinationsaufgaben in Jadex ein sehr aufwendiger Vorgang ist und dass zu viel Kommunikation der Plattform Probleme bereiten kann. Da zentrale Koordination durch Kommunikation allerdings in nahezu jedem MAS wiederzufinden ist bzw. benötigt wird, ist es wichtig hierfür Lösungen zu finden, die die Modularität und Adaptivität in diesem Bereich erhöhen. Im folgenden Abschnitt 4 wird nun ein erster möglicher Lösungsansatz für beide Problematiken vorgestellt.

4. Vision: Koordination „out-of-the-box“

Die grundsätzliche Idee besteht darin, beide beschriebenen Problematiken mit einer Lösung abzudecken. Wie das zu bewerkstelligen sein könnte, wird im Folgenden erläutert.

Das erste Ziel besteht darin, dem Entwickler möglichst viel Arbeit abzunehmen. Deshalb wird davon abgesehen, eine Interface-basierte Lösung zu entwickeln, da auch hier wieder an zu vielen Stellen gegen dieses Interface programmiert werden müsste. Wie bereits weiter oben erklärt wurde, ist das ADF eine der Kernkomponenten eines Jadex-Agenten. Also sollten Möglichkeiten geschaffen werden, die für essentielle Koordinationsaufgaben (wie z.B. die Synchronisation von Beliefs zwischen Agenten oder die kooperative Bewältigung von Goals) nur einfache Modifikationen am ADF benötigen. Denkbar ist es, spezielle Flags bei der Deklaration der Beliefs oder Goals einzuführen, die ggf. durch weitere Parameter modifizierbar sind, sowie dahinterliegende, gekapselte Kommunikations- und Koordinationsstrukturen, die dann automatisiert ausgeführt werden. Folgendes Beispiel zur Erklärung: Ein Goal kann als *cooperative* markiert werden und als Parameter eine Gruppe von Agenten erhalten, mit der die Kooperation stattfinden soll. Weiterhin muss dann das Reasoning von Jadex erweitert werden, so dass es fähig ist, sich für Plans zu entscheiden, die wiederum auch als *cooperative* markiert sind, bzw. bei negativen Rückantworten der möglichen Kooperationspartner, Plans auswählt, die zur autonomen Abarbeitung gedacht sind. Für eine solche Lösung sind tiefere Eingriffe in den Jadex-Kern nötig, was natürlich in einer späteren Risikoabschätzung berücksichtigt werden muss.

Dieser erste Teil der Lösung ist dann leicht kombinierbar mit der zweiten Zielsetzung, der Minimierung des Kommunikationsoverheads. Um dies zu erreichen, kann die gesamte für die Lösung notwendige Kommunikation unter Verwendung der bereits vorgestellten Kommunikationsmedien von SodekoVS realisiert werden. Diese stehen als Jadex-Capability bereit und helfen bei richtiger Nutzung, die Kommunikationslast durch eine automatisierte, gezielte Auswahl der Adressaten möglichst gering zu halten. Gegebenenfalls können auch die weiteren Selbstorganisations-bezogenen Aspekte von SodekoVS in die Gesamtlösung einbezogen werden, wobei auch hier wieder eine Risikoabschätzung erfolgen muss.

Im Anhang findet sich noch der erste Architekturentwurf für einen Prototypen unter Verwendung der SodekoVS Kommunikationsmedien.

5. Zusammenfassung

Die Arbeit hat nach der Einleitung eine kurze Einführung in Multiagentensysteme und noch spezieller in die im MMLab eingesetzte Jadex Multiagentenplattform gegeben, um daraufhin anhand von Erfahrungen mit dem Multiagent-Contest als *Testbed* Defizite der Plattform hinsichtlich der Implementation und Ausführung dezentraler Koordinationsmechanismen aufzuzeigen. Abschließend wurde dann ein erster Lösungsansatz für die Problematiken präsentiert, der sowohl eine Modifikation des Jadex-Kerns, als auch eine darin inbegriffene Einbindung der Koordinationsmedien von SodekoVS beinhaltet.

Literatur

[Behrens u. a. 2008] BEHRENS, Tristan M. ; DASTANI, Mehdi ; DIX, Jürgen ; NOVÁK, Peter: Agent Contest Competition: 4th Edition. In: *In Postproceedings of Sixth International Workshop on Programming Multi-Agent Systems, ProMAS'08, LNAI*, Springer, 2008

[Braubach u. a. 2005] BRAUBACH, Lars ; POKAHR, Alexander ; LAMERSDORF, Winfried: Extending the Capability Concept for Flexible BDI Agent Modularization. In: *Proc. of PROMAS-2005*, 7 2005

[Busetta u. a. 2000] BUSETTA, Paolo ; HOWDEN, Nicholas ; RÖNNQUIST, Ralph ; HODGSON, Andrew: Structuring BDI Agents in Functional Clusters. In: *ATAL '99*, Springer, 2000, S. 277–289. – ISBN 3-540-67200-1

[Ferber 2001] FERBER, Jacques: *Multiagentensysteme*. Addison-Wesley, 2001. – ISBN 3-8273-1679-0

[FIPA a] FIPA A, Modeling-TC: *Agent UML (AUML)*. – URL http://www.auml.org/

[FIPA b] FIPA B, TC-B: *Agent Management Specification*. – URL http://www.fipa.org/specs/fipa00023/SC00023K.pdf

[Pokahr u. a. 2005a] POKAHR, Alexander ; BRAUBACH, Lars ; LAMERSDORF, Winfried: Jadex: A BDI Reasoning Engine. In: *Multi-Agent Programming*, 2005, S. 149–174. – Book chapter

[Pokahr u. a. 2005b] POKAHR, Alexander ; BRAUBACH, Lars ; LAMERSDORF, Winfried: The Jadex BDI Reasoning Engine. In: BORDINI, R. (Hrsg.) ; DASTANI, M. (Hrsg.) ; DIX, J. (Hrsg.) ; SEGHROUCHNI, A. El F. (Hrsg.): *Programming Multi-Agent Systems*, Springer Verlag, 2005

[RMIT] RMIT, University: *The Prometheus Methodology*. – URL http://www.cs.rmit.edu.au/agents/SAC2/methodology.html

[Sudeikat u. a. 2009] SUDEIKAT, Jan ; BRAUBACH, Lars ; POKAHR, Alexander ; RENZ, Wolfgang ; LAMERSDORF, Winfried: Systematically Engineering Self-Organizing Systems: The SodekoVS Approach. In: WAGNER, M. (Hrsg.) ; HOGREFE, D. (Hrsg.) ; GEIHS, K. (Hrsg.) ; DAVID, K. (Hrsg.): *Proceedings des Workshops über Selbstorganisierende, adaptive, kontextsensitive verteilte Systeme (KIVS 2009)*, Electronic Communications of the EASST, 3 2009, S. 12

[Troposproject] TROPOSPROJECT: *Tropos: Requirements-Driven Development for Agent Software*. – URL http://www.troposproject.org/

[Weiss 2000] WEISS, Gerhard: *Multiagent systems : a modern approach to distributed artificial intelligence*. Cambridge, Mass. [u.a.] : MIT Press, 2000

[Wooldridge und Jennings 1995] WOOLDRIDGE, M. ; JENNINGS, N. R.: Intelligent Agents: Theory and Practice. In: *Knowledge Engineering Review 10(2)*, URL http://www.csc.liv.ac.uk/~mjw/pubs/ker95.pdf, 1995

[Wooldridge 2001] WOOLDRIDGE, Michael J.: *Introduction to Multiagent Systems*. New York, NY, USA : John Wiley & Sons, Inc., 2001. – ISBN 047149691X

A. Anhang

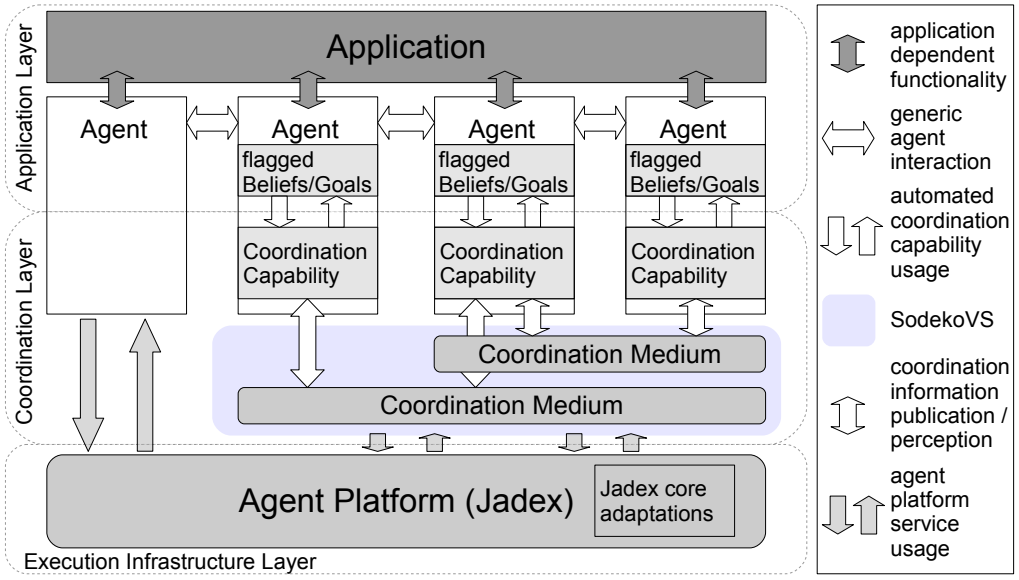

Abbildung 4: Die erweiterte SodekoVS-Architektur (vgl. SodekoVS Referenz-Architektur [Sudeikat u. a. (2009)]).