Vertrauen
als Entscheidungskriterium im Logistikmarkt

Beispiel einer Marktsimulation mit dem Agenten-Framework JADE

Susanne Knoop

Informationssysteme in der Logistik
Sommersemester 2011
Universität Bremen

1. November 2011

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Ziel der Ausarbeitung

Die folgende Abhandlung soll anhand der Beispielimplementierung *MarketSimulation* aufzeigen, wie der Einfluss von Vertrauen auf einen Markt simuliert werden kann. Dabei wird insbesondere untersucht, welchen Einfluss Vertrauen oder Misstrauen auf den Preis und auf die Erfolgsrate von Aufträgen hat. Die Anwendung baut dabei auf dem Agentenframework JADE auf.

1.2. Intelligente Agenten

Als Intelligente Agenten werden Roboter oder Softwareprozesse bezeichnet, denen in gewissem Grad Eigenständigkeit verliehen wurde, um ihre Aufgaben auszuführen.([5], Seite 34, [2], Teil 4, Seite 6) Eigenständigkeit wird von [4], Seite 944, definiert als „in der schwächeren Form die relative Abgeschlossenheit (Kapselung) gegenüber der Umgebung, in der stärkeren Form tatsächlich selbstbetimmtes Handeln (Autonomie).“ Dabei müssen sie ihre Umgebung wahrnehmen (Sensorik) und auch auf sie einwirken (Motorik oder Aktorik). Beides erfordert oftmals, dass der Agent mit anderen Agenten kommunizieren muss.

Die Entwicklung von intelligenten Agenten ist gemäß [5], Seite 5, eine der Hauptaufgaben der Forschung innerhalb der Künstlichen Intelligenz.

1.3. FIPA

Um die Entwicklung von Multiagentenplattformen, auf denen viele Softwareagenten miteinander interagieren, zu fördern, wurde 1996 die *Foundation for Intelligent Physical Agents*, abgekürzt FIPA gegründet. Diese legte unter anderem die *Agent Communication Language* oder ACL fest, eine Art Protokollsprache, mit deren Hilfe Agenten miteinander kommunizieren können. Außerdem definierte FIPA verschiedene Interaktionsprotokolle, d. h. „strukturierte Konversationen“ zwischen Agenten ([2], Teil 4, Seite 48).

1.4. Das FIPA Contract Net

Gemäß [4], Seite 1001, ist das Contract Net Protokol mittlerweile zu einem Standardverfahren in der Informatik geworden, das nicht nur im ökonomischen Bereich Anwendung findet. FIPA definierte hierzu die *FIPA Contract Net Interaction Protocol Specification* ([3]), das von einem Agenten ausgeht, der die Verhandlungen initiiert, indem er um Angebote bittet (*call for proposal*). Andere Agenten können auf diese Anfrage entweder ablehnen oder mit einem Angebot beantworten. Im letzteren Fall kann der initiierende Agent

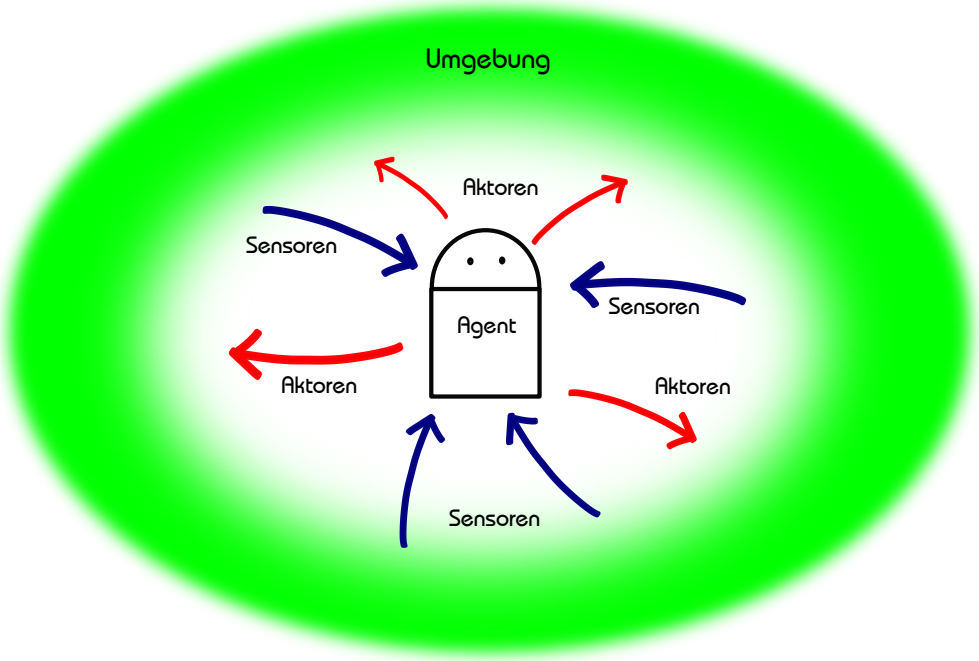

Abbildung 1: Ein Agent interagiert mit seiner Umwelt

wiederum das Angebot annehmen oder ablehnen. Falls er es annimmt, wird er vom Anbieter informiert, ob der Auftrag korrekt ausgeführt wurde oder nicht. Dieses Verfahren wird in Abbildung 2 veranschaulicht.

1.5. Das JADE Agenten-Framework

JADE steht für *Java Agent Development Framework* und ist ein Framework für die Programmiersprache Java, das die FIPA-Spezifikationen erfüllt. Insbesondere enthält es Klassen und Methoden, mit denen FIPA-konforme Agenten erstellt und ACL-Nachrichten ausgetauscht werden können([2], Teil 4, Seite 33 ff, [1]).

JADE-Agentenobjekte werden in einen sogenannten Container geladen, wobei es mehrere Container mit jeweils beliebig vielen Agenten geben kann. Die Agenten können sich außerdem beim DF-Service registrieren, einer Art „Gelbe Seiten Verzeichnis“, sodass andere Agenten ihre Dienste leichter finden können. Mit Hilfe der JADE-GUI kann der Agenten-Lifecycle beobachtet und gesteuert werden und sogar die Kommunikation zwischen den Agenten mit Hilfe eines sogenannten *Snifferagenten* abgehört werden ([1], Seite 32 ff).

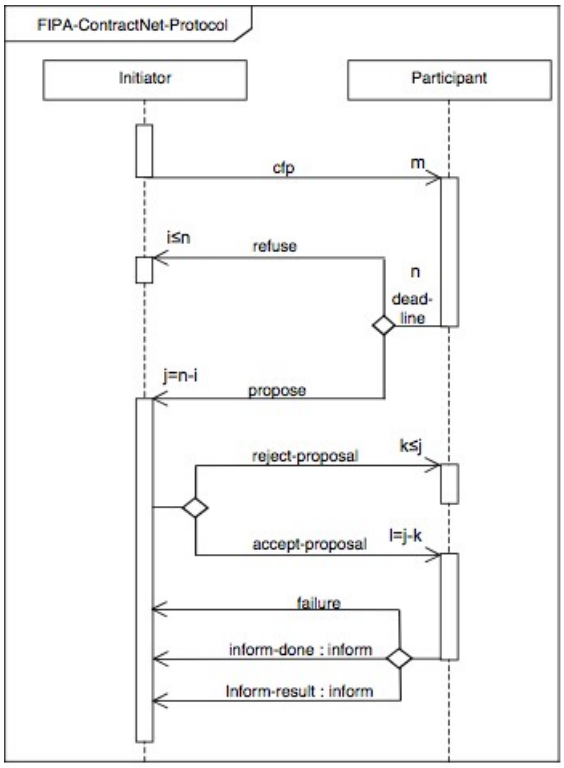

Abbildung 2: Das FIPA Contract Net

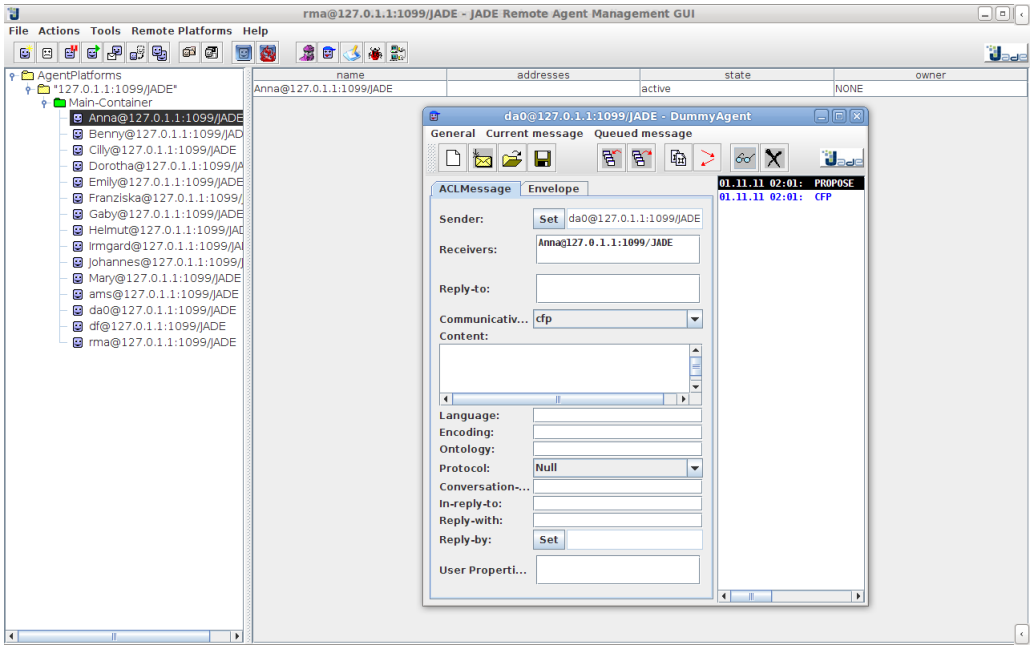

Abbildung 3: Die grafische Benutzeroberfläche von JADE

JADE war für die Anwendung *MarketSimulation* die geeignet Wahl, weil es die FIPA Spezifikationen einhält und dank Open-Source-Lizenz kostenfrei verwendet werden darf. Da in der Simulation außerdem die gesamte Interaktion eines Agenten mit seiner Umgebung über Nachrichtenaustausch mit anderen Agenten stattfindet, war es ein Vorteil von JADE, dass es auf relativ einfache Art und Weise erlaubt, die Kommunikation mittels ACL-Nachrichten zu steuern.

Die wichtigsten Klassen des JADE Framework sind:

Agent ein Softwareagent auf der JADE Plattform

Behaviour implementiert das Verhalten eines Agenten. Einem Agent können mehrere *Behaviour* hinzugefügt werden.

ACLMessage Objekt zur Erzeugung FIPA-konformer ACL-Nachrichten

1.6. Einfluss von Vertrauen auf das Marktgeschehen

Wenn ein Spediteur oder ein anderer logistischer Dienstleister einen Transport an einen Subunternehmer vergibt, dann nur auf Grund des Vertrauens, dass dieser den Transport zuverlässig und pünktlich ausführen wird. Dies gilt allgemein für alle Märkte, doch im Logistikmarkt spielt Vertrauen eine besonders entscheidende Rolle, da sehr viele unvorhersehbare Faktoren den Erfolg einer logistischen Dienstleistung gefährden können. Der Spediteur muss also auch zuversichtlich sein, dass der Subunternehmer auf plötzlich auftretende Probleme richtig reagieren wird und den Auftraggeber jederzeit vom aktuellen Stand der Situation in Kenntnis setzen wird. Falls dies nicht der Fall ist, würde der Spediteur wiederum seinen Auftraggeber enttäuschen, und so weiter, was in der der ganzen Logistikkette zu einem Rückgang des Vertrauens in das jeweilige Folgeglied führen würde.

Wenn Marktteilnehmer anfangen, von Geschäftspartnern, mit denen sie in der Vergangenheit erfolgreich zusammengearbeitet haben, zu bevorzugen, dann werden sie innerhalb eines gewissen Spielraums auch bereit sein, von diesen höhere Preise zu akzeptieren. Dagegen werden sie zögern, Aufträge an neue oder unbekannte Marktteilnehmer zu vergeben, selbst wenn diese günstiger sein mögen. Auf diese Weise verschiebt sich der Gleichgewichtspreis des Markts nach oben. Dafür müsste eine höhere Erfolgsrate zu beobachten sein, da verstärkt Transporte an Marktteilnehmer vergeben werden, die sich in der Vergangenheit als zuverlässig erwiesen haben.

Erfahrene Spediteure wissen, was ein zu spät ausgeführter Transport, ein Schadensfall oder sogar ein Gerichtsverfahren dem Unternehmen kosten kann und sind daher eher bereit, einen höheren Preis im Tausch für höhere Leistung zu zahlen. Es könnte jedoch schwierig sein, diese Entscheidung auch dem eigenen Kunden zu vermitteln. Es ergibt sich außerdem das Problem,

die für den Markt passende „Vertrauensstrategie“ zu finden: Wie „misstrauisch“ sollte ein Käufer sein, um die höchstmögliche Performancesteigerung bei geringstmöglicher Preissteigerung zu erhalten?

Im Logistikmarkt ist allgemein zu beobachten, dass viele Speditionskaufleute sich stark auf ihre Erfahrungen oder die Erfahrungen ihrer Kollegen mit bestimmten Betrieben verlassen und diese auch an neue Mitarbeiter weitergeben. Einmal gemachte negative Erfahrungen werden lange in Erinnerung behalten und der betreffende Anbieter gemieden. Oft verlässt man sich immer wieder auf dieselben Geschäftspartner, die sich in einer längeren Geschäftsbeziehung bewährt haben. Daher könnte man die Marktteilnehmer auf dem Logistikmarkt als eher missstrauisch einstufen.

Natürlich muss auch von der Seite des Verkäufers einer logistischen Dienstleistung aus Vertrauen vorhanden sein, nämlich das Vertrauen, dass der Käufer die Dienstleistung auch bezahlen wird. Hierauf soll aber in dieser Ausarbeitung nicht eingegangen werden.

2. Die Anwendung MarketSimulation

2.1. Ablauf der Simulation

In der Simulation *MarketSimulation* wird ein Transport mehrere Male von einem einzigen einzigen Käuferagenten nachgefragt, wobei mehrere Verkaufsagenten als Anbieter zur Auswahl stehen. Jeder Verkaufsagent besitzt eine andere Erfolgsrate, die aussagt, wieviel Prozent der Transporte problemlos ausgeführt werden und bei wieviel Prozent es dagegen Probleme gibt. Der Käuferagent wählt in jeder Runde den für ihn optimalen Verkäufer aus, wobei sowohl der Preis, als auch das Vertrauen, dass der Käufer in den Verkäufer setzt, in die Berechnung mit einfließen. Wenn es mehrere Verkäufer gibt, in die der Käufer volles Vertrauen setzt, erfolgt die Entscheidung nur über den Preis. Der Marktpreis für den Transport wurde der Einfachheit halber auf 50 Geldeinheiten festgesetzt.

Nachdem ein Verkaufsagent ausgewählt wurde, wird dieser beauftragt, den Transport durchzuführen. Der Verkaufsagent benachrichtigt daraufhin den Käufer, ob er den Auftrag erfolgreich ausgeführt hat oder nicht. Im Erfolgsfall steigt das Vertrauen in den Verkäufer, im Falle eines Misserfolgs jedoch geht das Vertrauen fast komplett verloren. Um zu simulieren, dass auch schlechte Erfahrungen irgendwann wieder in Vergessenheit geraten, steigt das Vertrauen in alle Anbieter (außer dem Verkäufer) um einen kleinen Betrag, so dass auch ein Verkäufer, der einmal einen Mißerfolg hatte, irgendwann wieder eine Chance hat, den Zuschlag zu erhalten,

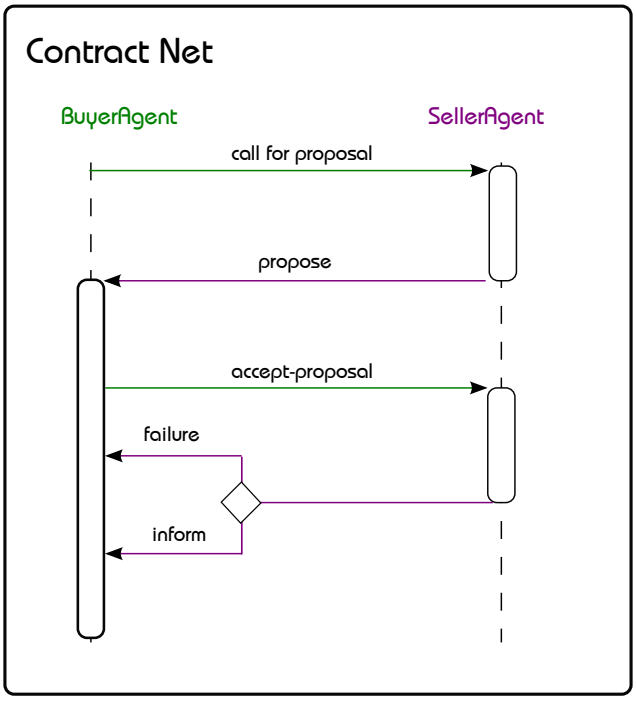

Abbildung 4: Das vereinfachte Contract Net von *MarketSimulation*

Die obige Beschreibung zeigt, dass die Anwendung *MarketSimulation* eine vereinfachte Form des FIPA Contract Net implementiert, dass lediglich die ACL-Messages *cfp, propose, accept-proposal, failure* und *inform* benutzt (siehe Abbildung 4).

Am Ende jeder Runde werden die wichtigsten Daten des Kaufs vom Käuferagenten in eine Tabelle geschrieben. Wenn alle Transportaufträge vergeben und durchgeführt wurden, werden die Daten in der Tabelle ausgewertet und vier Werte errechnet:

- Die zu erwartende Erfolgsrate in Prozent, die aus dem Durchschnitt der Erfolgsraten aller Anbieter errechnet wird.
- Die tatsächliche Erfolgsrate, d. h., wieviel Prozent der Transporte wurden vom Anbieter erfolgreich durchgeführt.
- Der zu erwartende Durchschnittspreis in Geldeinheiten, d. h. der Preis, den der Käufer durchschnittlich bezahlt hätte, wenn er den jeweils günstigsten Anbieter gewählt hätte.
- Der wirklich vom Käufer bezahlte Durchschnittspreis.

Die so ermittelten Werte werden wieder in eine Tabelle geschrieben und gleichzeitig auf dem Bildschirm ausgegeben. Aus einem Vergleich der erwar-

teten Erfolgsrate, bzw. des erwarteten Preises und der tatsächlich beobachteten Werte lassen sich Rückschlüsse auf den Erfolg des Kaufverhaltens des Käuferagenten schließen.

Aus dem Vorangegangenen wird deutlich, dass die Agenten in *MarketSimulation* nicht wirklich als „intelligent“ zu bezeichnen sind, da sie lediglich reaktives Verhalten an den Tag legen ([2] Teil 4, Seite 10). Sensorik und Motorik der Agenten laufen nur über ACL-Messages ab, d. h. sie sind nur über Nachrichtenaustausch in der Lage, die Umgebung wahrzunehmen und auf sie einzuwirken. Sie können aber nicht auf eine veränderte Marktsituation reagieren, indem sie ihre Vertrauensstrategie anpassen oder - im Falle des Verkäufers - die Auswahl des Angebotspreises.

2.2. Implementierung

Bevor die Anwendung *MarketSimulation* gestartet wird, muss sie erst den Wünschen des Benutzers gemäß konfiguriert werden. Der Ordner *MarketSimulation* enthält vier Unterordner, *bin, config, logs* und *src*.

- *bin* enthält die kompilierten Javaklassen der Anwendung.
- *config* enthält die die Konfigurationsdatei *configuration.xml* .
- *logs* enthält die Logdateien und die Tabellen mit den oben erwähnten Daten.
- *src* enthält die Quelldateien der Javaklassen.

In der Konfigurationsdatei *configuration.xml* im Ordner *config* können die folgenden Parameter manuell verändert werden:

- Die Anzahl der Verkaufsagenten (≪*noSellerAgents* ≫)
- Die Anzahl der zu kaufenden Transporte (≪*transports* ≫)
- Die durchschnittliche Erfolgsrate der Transportanbieter in Prozent (≪*meanPerformanceRate* ≫)
- Die Standardabweichung von der durchschnittlichen Erfolgsrate (≪*perfStandardDev* ≫)
- Das Anfangvertrauen, dass der Käufer vor der ersten Verkaufsrunde in alle Verkaufsagenten setzt in Prozent (≪*initialTrust* ≫)
- Der Betrag, um den das Vertrauen in jeden Verkaufsagenten nach jeder Runde steigt (≪*allTrustRaise* ≫)
- Der Betrag, um den das Vertrauen in einen Verkaufsagenten nach einem erfolgreich ausgeführten Transport steigt (≪*sellerTrustRaise* ≫)

Nachdem die gewünschten Parameter eingestellt wurden, kann die JADE Umgebung gestartet werden mittels des Befehls

```
 java -classpath .:/home/user/jade.jar jade.Boot  -agents
"seller1:SellerAgent;seller2:SellerAgent;buyer:BuyerAgent"
```

- wobei */home/user/jade.jar* für den Pfad zu den JADE Klassen steht, *seller1*, *seller2*, usw. für die Namen der Verkaufsagenten stehen und *buyer* für den Namen des Käuferagenten. Es können beliebig viele Verkaufsagenten gestartet werden; es sollte jedoch darauf geachtet werden, dass der gesamte Text nach *-agents* in Anführungszeichen stehen muss, da sonst nur der erste Agent gestartet wird. Die Verkaufsagenten sollten zuerst gestartet werden, da der Käuferagent beim Start ihre Dienste abfragt.

2.2.1. Die Klasse SellerAgent

Wenn ein Objekt vom Typ *SellerAgent* erzeugt wird, wird wie bei allen Agenten im JADE Framework zuerst die *setup*-Methode ausgeführt. Diese richtet eine Logdatei ein, in der alle Aktivitäten des Agenten im folgenden geloggt werden. Danach werden die Parameter des Agenten, nämlich die durchschnittliche Erfolgsrate und die Standardabweichung aus der Konfigurationsdatei ausgelesen. Falls dies nicht möglich ist, z. B. weil die Datei *./config/configuration.xml* nicht existiert, werden als Standardwerte 65 bzw. 5 eingesetzt. Dann wird mit den ermittelten Werten eine normalverteilte Zufallszahl erzeugt, die als die persönliche Erfolgsrate des Agenten gespeichert wird.

Danach wird der neu erzeugt *SellerAgent* mit dem Service Typ *Seller* in den DFService der JADE Plattform eingetragen, damit sein Service später für den *BuyerAgent* sichtbar ist.

Der *SellerAgent* besitzt nur ein *Behaviour* mit Namen *SellerAgentBehaviour*, das als innere Klasse implementiert ist, weil es so auf die privaten Attribute des *SellerAgent* zugreifen kann. Es handelt sich um ein *CyclicBehaviour*, das immer wieder von vorn ausgeführt wird, da immer wieder neue Nachrichten vom *BuyerAgent* eintreffen können.

Der *SellerAgent* reagiert nur auf Nachrichten vom Typ *ACLMessage.CFP* und *ACLMessage.ACCEPT_PROPOSAL*. Im ersten Fall ist ein *Call For Proposal*, also eine Anfrage vom *BuyerAgent* eingegangen und der *SellerAgent* reagiert mit einer Nachricht vom Typ *ACL.PROPOSE*, die als Inhalt den Angebotspreis enthält. Der Angebotspreis wird mit Hilfe einer normalverteilten Zufallszahl berechnet, wobei der Durchschnittspreis bei 50 Geldeinheiten und die Standardabweichung bei 10 liegen:

```
Random r = new Random();
int nextprice = (int) Math.round((r.nextGaussian() * 10) + 50);
reply.setPerformative(ACLMessage.PROPOSE);
```

Zahl der Transporte	10
Anfangsvertrauen	50
allgemeiner Vertrauensanstieg	5
Vertrauensanstieg im Erfolgsfall	20
Zahl der Verkaufsagenten	2

Tabelle 1: Die Standardparameter des *BuyerAgent*

```
reply.setContent(String.valueOf(nextprice));
myAgent.send(reply);
```

Eine Nachricht vom Typ *ACLMessage.ACCEPT_PROPOSAL* dagegen bedeutet, dass der *SellerAgent* den Zuschlag erhalten hat und den Transport ausführen soll. Dies wird simuliert, indem eine Zufallszahl zwischen 0 und 100 erzeugt wird, und geprüft wird, ob sie innerhalb der Erfolgsrate des Agenten liegt. Wenn dies der Fall ist, wird eine ACLMessage vom Typ *ACLMessage.INFORM* zurückgesandt, ansonsten eine Nachricht vom Typ *ACLMessage.FAILURE*. Diese enthalten jeweils keinen weiteren Inhalt, da der *BuyerAgent* die Information bereits aus dem Typ der Nachricht auslesen kann.

In beiden Fällen braucht der Absender der Nachricht nicht geprüft zu werden, da nur der *BuyerAgent* Nachrichten der oben erwähnten Typen versendet. Dies hat den Vorteil, dass leichter getestet werden kann, da auch der auf der JADE Plattform standardmäßig vorhandene *DummyAgent* dem *SellerAgent* Nachrichten schicken kann und von diesem eine Antwort erhält.

2.2.2. Die Klasse BuyerAgent

Beim Start des *BuyerAgent* wird wiederum zuerst die *setup*-Methode ausgeführt, die eine Logdatei einrichtet und die Parameter des Agenten aus der Konfigurationsdatei ausliest. Wenn das Auslesen fehlschlägt, werden die Standardparameter aus Tabelle 1 verwendet.

Der *BuyerAgent* konsultiert daraufhin den DFService der Plattform, um alle mit dem Service Typ *Seller* angemeldeten Agenten zu finden. Die Zahl der gefundenen Verkaufsagenten muss mindestens so groß sein, wie der in der Konfigurationsdatei hinterlegte Parameter, da der *BuyerAgent* sonst immer weiter im DFService sucht und nicht mit dem Kaufen der Transporte beginnt. Der *BuyerAgent* besitzt als Klassenattribut eine *HashMap*, die jedem *SellerAgent* einen ganzzahligen Wert von 0 bis 100 zuordnet, der das aktuelle Vertrauen ausdrückt. Dieses wird mit dem Anfangsvertrauen initialisiert und nach jeder Verkaufsrunde aktualisiert.

Auch das einzige *Behaviour* des *BuyerAgent* wurde als innere Klasse implementiert, um Zugriff auf die Klassenattribute des *BuyerAgent* zu erhalten. Es wird ausgeführt, bis alle Transporte gekauft wurden, was durch die Methode *done()* sichergestellt wird:

```
 @Override
public boolean done() {

if (leftTransports < 1) {
return true;
}
else {
return false;
}
}
```

Jedes Mal, wenn die *action*-Methode des *Behaviour* ausgeführt wird, wird also ein Transport gekauft. Dies beginnt damit, dass eine *Call For Proposal* Nachricht an alle bekannten *SellerAgents* gesandt wird. Für jedes erhaltene Angebot wird ein Objekt vom Typ *Offer* erzeugt, das alle wichtigen Angebotsdaten speichert und in eine *PriorityQueue* eingereiht wird, in der das Angebot mit dem niedrigsten Vertrauenspreis an erster Stelle steht.

Das aktuelle Vertrauen in den jeweiligen Agenten wird dabei wie in Tabelle 2 gezeigt mit dem angebotenen Preis verrechnet. Diese Berechnung hat den Vorteil, dass mehrere Anbieter mit demselben Vertrauen - z. B. einem vollen Vertrauen von 100 % - dann über den Preis konkurrieren und es nicht zu rein zufälligen Kaufsentscheidungen kommt.

Das beste Angebot, das nach Eingang aller Angebote an erster Stelle der *PriorityQueue* steht, erhält den Zuschlag. Der entsprechende *SellerAgent* erhält eine Nachricht vom Typ *ACLMessage.ACCEPT_PROPOSE*. Die Antwort des Agenten gibt Aufschluss über den Erfolg oder Misserfolg des Auftrags. Das Vertrauen wird entsprechend aktualisiert und alle Daten über den Auftrag werden in die Logdatei *Simulation.csv* geschrieben.

Nachdem alle Verkäufe getätigt wurden gibt die Methode *done() true* zurück, sodass die zum *Behaviour* gehörende Methode *onEnd* ausgeführt wird. Diese ruft die Funktion *evaluate* der Klasse *Evaluator* auf, die die Auswertung der Simulation vornimmt.

2.2.3. Die Klasse Evaluator

Die Klasse *Evaluator* bietet die Methode *evaluate()*, die die Logdatei *Simulation.csv* auswertet, und die folgenden Werte errechnet:

Vertrauen in Prozent	Berechnung
0 bis 9	Angebotspreis multipliziert mit 3
10 bis 19	Angebotspreis multipliziert mit 2,5
20 bis 29	Angebotspreis multipliziert mit 2
30 bis 39	Angebotspreis multipliziert mit 1,5
40 bis 49	unveränderter Angebotspreis
50 bis 59	unveränderter Angebotspreis
60 bis 69	Angebotspreis dividiert durch 1,5
70 bis 79	Angebotspreis dividiert durch 2
80 bis 89	Angebotspreis dividiert durch 2,5
90 bis 100	Angebotspreis dividiert durch 3

Tabelle 2: Berechnung des Vertrauenspreis

- die erwartete Erfolgsrate (errechnet aus dem Durchschnitt der Erfolgsraten aller Verkaufsagenten)
- die tatsächliche Erfolgsrate (errechnet aus der Anzahl der erfolgreichen Transporte im Verhältnis zur Anzahl aller Transporte)
- der erwartete Preis, wenn die Kaufentscheidung nur aufgrund des Preises erfolgt wäre
- der tatsächlich vom *BuyerAgent* bezahlte Durchschnittspreis

Die obigen Werte werden dann auf der Konsole ausgegeben und im csv-Format an den Inhalt der Datei *Evaluation.csv* angehängt, die dann mit gängigen Tabellenkalkulationsprogrammen weiterverarbeitet werden kann. Danach wird das Programm beendet.

3. Tests von Marktszenarien

3.1. Die Parameter der Tests

Für jedem Test wurden 10 Verkäufer der Klasse *SellerAgent* und ein Käufer der Klasse *BuyerAgent* gestartet und 300 Verkäufe durchgeführt. Für jedes Szenario wurden 10 solcher Tests durchgeführt. Anschließend wurde die Datei *Evaluation.csv* mit Hilfe von *OpenOffice Calc* ausgewertet.

Es wurden jeweils zwei Marktsituationen und vier Vertrauensstrategien zugrundegelegt, sodass sich insgesamt acht Szenarios ergeben. Die erste Marktsituation beschreibt einen Markt, auf dem kein großer Druck auf die Anbieter herrscht oder ein Markt, der sich noch nicht im Gleichgewicht befindet, so dass die Leistungsunterschiede zwischen den Anbietern groß sind und die durchschnittliche Leistung eher niedrig. In der zweiten Situation dagegen

Vertrauen	Anfangsvertrauen	Vertrauensanstieg allgemein	Vertrauensanstieg bei Erfolg
hoch	70 %	10 %	40 %
mittel	50 %	5 %	30 %
niedrig	10 %	2 %	15 %
sehr niedrig	1 %	1 %	4 %

Tabelle 3: Die Parameter der Simulationstests (Vertrauen)

Anbieterleistung	Markt	durchschnittliche Erfolgsrate	Standardabweichung
niedrig	heterogen	70 %	15
hoch	homogen	90 %	5

Tabelle 4: Die Parameter der Simulationstests (Marktsituation)

herrscht so großer Druck auf die Anbieter, dass fast alle eine hohe Leistung zeigen und die Unterschiede zwischen den Anbietern gering sind. Es ist zu erwarten, dass es im zweiten Fall wesentlich schwieriger wird, die Erfolgsrate durch Einbeziehung des Vertrauens noch zu steigern.

Das Käufervertrauen kann entweder hoch, mittel, niedrig oder sehr niedrig sein, was sich in den Werten der Tabelle 3 widerspiegelt. Es ist zu erwarten, dass ein geringeres Vertrauen sich in höheren Preisen, aber auch höheren Erfolgsraten zeigt, da so eher die zuverlässigen Verkäufer gefunden werden.

3.2. Ergebnisse

Die Ergebnisse in Tabelle 5 zeigen, dass die Erwartungen erfüllt wurden: Je höher das Misstrauen, desto höher die Performancesteigerung, desto höher aber auch der Preis. Möchte man möglichst viel Performancesteigerung zu einem geringen Preis, so ist eine Vertrauensstrategie mit hohem Vertrauen ratsam, da hier der Erfolgsquotient am höchsten ist. Wenn ein Käufer dagegen die Performancesteigerung höher bewertet als den Preis, wäre ein misstrauisches Verhalten optimal, da dann die höchste Performancesteigerung zu erwarten ist.

Bei Szenario vier ist jedoch zu beobachten, dass ein sehr misstrauischer Käufer eine geringere Performancesteigerung verzeichnen konnte als ein misstrauischer. Dies könnte daran liegen, dass selbst erfolgreiche Verkäufer nicht genügend bevorzugt wurden, weil das Vertrauen zu langsam wuchs.

Szenario	Markt	Anbieterperf.	Vertrauen	e[1]	p [2]	q[3]
1	heterogen	gering	hoch	4,02	4,65	0,87
2	heterogen	gering	mittel	6,30	9,46	0,66
3	heterogen	gering	niedrig	20,00	24,20	0,83
4	heterogen	gering	sehr niedrig	14,00	23,19	0,60
5	homogen	hoch	hoch	0,60	1,33	0,45
6	homogen	hoch	mittel	0,30	3,74	0,08
7	homogen	hoch	niedrig	0.8	15,41	0,05
8	homogen	hoch	sehr niedrig	3,30	20,11	0,17

Tabelle 5: Ergebnisse der Simulationstests

Auch die Erwartungen bezüglich der Marktsituation erfüllten sich: Während bei einem heterogenem Markt noch eine hohe Leistungssteigerung bis circa 20 Prozent möglich ist, beträgt die Steigerung bei einem homogenen Markt nur noch maximal drei Prozent.

4. Fazit und Ausblick

Die Anwendung *MarketSimulation* zeigt, wie mit Hilfe des JADE Framwork mit relativ wenig Quellcode ein Transportmarkt unter Einbeziehung des Vertrauens simuliert werden kann. Die Tests zeigen, dass - wie in der Realität zu beobachten - dadurch die Erfolgsrate, aber auch der bezahlte Preis steigen.

Um die Simulation realitätsnäher zu gestalten, könnte man sie wie folgt erweitern:

- Um den Käuferagenten wirklich intelligent zu machen, müsste er in seiner Vertrauensstrategie flexibel auf die Marktsituation reagieren können, um eine gewünschte Performancesteigerung zu erreichen.
- Statt einfach nur den Erfolg oder Misserfolg des Transports zu melden, könnte der Verkaufsagent einen Grad der Kundenzufriedenheit zurücksenden.
- Als zusätzliche Parameter könnten auch der Durchschnittspreis und die Standardabweichung des Preises in die Konfigurationsdatei aufgenommen werden.

[1] Performancesteigerung in Prozent
[2] Preissteigerung in Prozent
[3] „Erfolgsquotient“, $q = \frac{e}{p}$

- Verkäufer mit einer niedrigen Erfolgsrate könnten versuchen, dies durch durchschnittlich niedrigere Preise auszugleichen und umgekehrt.
- Auch die Verkaufsagenten könnten mit der Intelligenz ausgestattet werden, auf das Verhalten des Käufers zu reagieren, um möglichst hohe Preise zu erzielen.

A. Anhang

A.1. Beispiele

A.1.1. Beispiel einer Logdatei eines SellerAgent

```
01.11.2011 02:24:49 SellerAgent setup
INFO: Die Performanz des Agenten Anna liegt bei 62%!
01.11.2011 02:24:49 SellerAgent setup
INFO: Seller Agent Anna hat sich erfolgreich in den Gelben Seiten registriert!
01.11.2011 02:24:49 SellerAgent setup
INFO: Seller Agent Anna initialisiert!
01.11.2011 02:24:49 SellerAgent$SellTransportManager action
INFO: Verkaufsagent Anna hat Call for Proposal erhalten!
01.11.2011 02:24:49 SellerAgent$SellTransportManager action
INFO: Verkaufsagent Anna hat Angebot zum Preis von 41 abgegeben!
01.11.2011 02:24:49 SellerAgent$SellTransportManager action
INFO: Agent Anna hat einen Auftrag erhalten!
01.11.2011 02:24:49 SellerAgent$SellTransportManager action
INFO: Agent Anna hat den Auftrag erfolgreich ausgeführt.
```

A.1.2. Beispiel einer Logdatei Simulation.csv

```
AID;Performanz;Transportnr;günstigsterAgent;günstigster Preis;gewählter
Preis;Vertrauenspreis;Erfolg;VertrauenVorher;VertrauenNachher
Cilly;63;null;null;null;null;null;null;null;null
Benny;84;null;null;null;null;null;null;null;null
Anna;75;null;null;null;null;null;null;null;null
null;null;1;Cilly;52;52;130;ja;10;25
null;null;2;Benny;39;39;98;ja;12;27
null;null;3;Benny;29;29;58;ja;27;42
null;null;4;Benny;52;52;52;nein;42;1
null;null;5;Cilly;44;46;69;nein;31;1
```

A.1.3. Beispiel einer Auswertungsdatei Evaluation.csv

1

```
90,30;87,00;;31,98;37,23;
90,80;93,00;;35,50;40,71;
91,90;95,00;;32,08;37,14;
```

[1] Die Spalten stehen für: erwartete Erfolgsrate, tatsächliche Erfolgsrate, Leerspalte, erwarteter Preis, tatsächlicher Preis

Literatur

[1] Bellifemine, Fabio L. ; Caire, Giovanni ; Greenwood, Dominic: *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. Wiley, 2007

[2] Berndt, Jan O. ; Wagner, Thomas ; Langer, Hagen: *Vorlesungsfolien zu Informationssysteme in der Logistik*. 2011

[3] Communication, FIPA T.: *FIPA Contract Net Interaction Protocol Specification*. `http://www.fipa.org/specs/fipa00029/SC00029H.html`,

[4] Görz, Günther (Hrsg.) ; Rollinger, Claus-Rainer (Hrsg.) ; Schneeberger, Josef (Hrsg.): *Handbuch der Künstlichen Intelligenz*. 4. München : Oldenbourg, 2003

[5] Russell, Stuart J. ; Norvig, Peter: *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003