Multi-Agent Systems in E-Commerce

Marian Zsuzsanna

Abstract

This paper presents the usage of Multi-Agent Systems (MAS) in e-commerce. First some of the first simple search agents like Bargain-Finder, Jango and one of the first Multi-Agent Systems, Kasbah are shortly presented. Then four auction types, which are often used in MAS for the negotiation are described. Next a MAS is presented, whose prototype is implemented in JADE, together with some modifications and improvements done to transform it in a rule-based negotiation system. A MAS that is capable of learning the best strategy, using knowledge discovery is presented in more detail, followed by one that uses fuzzy logic to determine the best strategy.

1 Introduction

The exponential growth of the Internet, and the fact that almost 2.000.000.000 people are using it in the world (which is a 444.8 % growth compared to the values at the end of 2000) [3] lead to the apparition and growth of e-commerce. By the definition of ECA (Electronic Commerce Association) ”electronic commerce covers any form of business or administrative transaction or information exchange that is executed using any information and communications technology” [11]. Nowadays, this term refers almost exclusively to commercial activities over the Internet, meanings like ”ordering an air ticket over the telephone” are no longer in use. In most scenarios, the user uses a web browser and a search engine in order to find the items he or she wants to purchase online. But there is another possibility, the use of the agent technology, which is said to become for e-commerce, what Windows was for the PC - a relatively simple and user-friendly way of utilising the new technology [2].

Different papers characterize agents differently, but all of them agree that they are autonomous software programs, which means that they act independently on the behalf of their user. Besides these they, can also be reactive (which means that they are capable of perceiving their environment and respond to changes in it), and pro-active (which means that besides reacting to the environment’s changes, they are able to exhibit a goal-directed behaviour), to mention only the most frequently enlisted characteristics.

Multi-Agent systems (MAS) are systems that are made of two or more agents, that can interact via a commonly known protocol. Obviously, these agents have to use the same objective language, so that they can communicate, and usually they need to posses some negotiation skill also. In most applications of e-commerce, we can rather talk about a MAS than just a single agent, because usually at least one agent that sells a good and one agent that wants to buy a good is needed.

There are many ways in which agents can be helpful in e-commerce: they can monitor and retrieve useful information and do transactions on the behalf of their users, Secondly, agents can find the best deals for their users with less cost, quicker response, and less user effort. Thirdly, agents can participate in negotiations with other agents, to get a good price of trade for their users [11].

The structure of the paper is the following: Section 2 presents some of the first agents and multi-agent systems. Section 3 starts with a short section about negotiation and presents four auction types (English auction, Dutch auction, First price sealed bid auction and Vickery auction) used ofter in e-commerce models. The next subsections present some MAS prototypes that can be used in e-commerce. The first is a prototype implemented in Jade. The next two subsections are related to this system, and present improvements on it, unfortunately the exact strategy that agents from the MAS should use is not discussed. The next MAS is capable of negotiating even in cases when there are multiple attributes to consider, and uses knowledge discovery to find the best strategy. The last MAS, FINA, uses fuzzy logic to find the best strategy. Finally, in Section 4 the conclusions are drawn.

2 First agents in e-commerce

The first agents that were used in e-commerce were only used to search different online stores, and display the results to the user. One of them was BargainFinder, created in 1995, an agent that performs comparison price shopping among nine predefined on-line CD stores, in such a short time that a human user would only be able to check one site in this time. Unfortunately, Bargainfinder is no longer available on the Internet, but in the first two month after it being released (30 June 1995) it was used over 100.000 times [4].

The problem with BargainFinder was that is only displayed the price of the CD without the other offers that might have been related to it (free shipping for example). Since in this way the price might have been higher than at other places, where there were no offers, some of the stores banned the acces of BargainFinder to their page (but there were shops that sked to be included in the list). Another similar agent, Jango, managed to avoid this problem, by initiating the request from the address of the user, so that stores were unable to tell the difference between requests from the agent or from human users.

One of the first MAS that used agents to sell and buy goods was Kasbah [1]. The system was based on two types of agents: buying and selling agents which can interact in the marketplace. Both the buying and selling agents are created with some initial parameters: the date to sell (buy) the item by, the desired price to sell (buy) the item for, and the lowest (highest) acceptable price. The agents have to figure out themselves what strategy to follow in order to sell (buy) the desired item at the best price. The crude heuristic that is used, is something like this: start with the desired price, and if there are no buyers (sellers) that accept that price, lower (increase) it as time goes by, so that at the end of the period the price should be almost equal to the lowest (highest) acceptable price. The third member of this system was the marketplace itself, where these agents meet and negotiate. In Kasbah, when an agent entered the marketplace, it asked about the product it is buying or selling, and directed it to other agents that could help it (they wanted to buy or sell that given product). The system had a prototype, implemented in CLOS using Harlequin Lisp.

3 MAS in e-commerce

Negotiation and auction

In most papers that describe MAS for e-commerce, the main scenario is very simple: buyer and seller agents are created by the users, which will act on their behalf. These agents are placed in a common environment (usually called masketplace, that has a yellow pages service, which can pair buyer and seller agents, that have the same item), and they will start negotiating, until they reach an agreement. It is easy to realize that the hardest thing to implement is the negotiation strategy itself. Since the agent acts on the behalf of the user that created it, the negotiation strategy should try to maximize the payoff of the user. The only problem is, that in many cases, users don't even know what they exactly want, let alone formulate it in a mathematical or logical way, that an agent should understand.

Probably this is the reason why, many e-commerce application frameworks designed for agents decide to use auctions instead of simple negotiations, because in case of some auctions game theory can provide the ideal strategy to be used.

Some of the main auction types together with their rules, outcomes and optimal strategy are shortly presented in the following list:

- **English auction** - This is an open auction (which means that bidders can see the bids submitted by others), where bidders successively raise bids for the item, until a single bidder remains, which is the winner. The optimal strategy in this auction, is to bid up, until the maximum limit allows.
- **Dutch auction** - Another open auction, where the auctioneer calls out descending bids, and the bidder calls out a bid. The winner is the first bidder who calls the same bid as the auctoneer. The optimal strategy is to shade bid a bit below true willingness to pay.
- **First priced sealed bid** - A sealed auction (participants cannot see the bid of ther people), where bidders submit a single sealed bid, before a deadline. The winner is the highest price. The optimal strategy is to shade bid a bit below true willingness to pay.
- **Vickery auction** - A sealed auction, where bidders submit a single sealed bid before a deadline. The winner is the one who offers the highest price, but will pay the second highest price. The optimal strategy is truthtelling.

MAS implemented in Jade

[5] presents a MAS system, implemented in Jade, which they claim to be usuful in building a large-scale agent system for experiments, because a previous experiment showed that on 8 "antiquated Sun workstations" [5] a system with 1500 agents and 300000 messages was successfully used. They mention as other advantages that Jade is FIPA compliant, runs on a variety of operation systems, and is one of the best modern agent environment.

In this paper negotiation is in fact auction, and they shortly present two auction protocols: the FIPA English Auction Interaction Protocol (where the seller starts with a price below the market value, and as long as there is an agent willing to pay that price, it will increment it, a little. The auction ends, when no buyer accepts the proposed price, in which case the winner is the buyer from the previous round) and the FIPA Dutch Auction Protocol (where the seller starts with a

price above the market value, and decreases it, until a buyer accepts is). The paper presents a MAS, where agents are general enough, so that they are capable of working with different auction strategies. In order to achieve this, an agent is said to be made of several modules:

- **Communication module** - responsable for the communication between the agents. Since the project is implemented in Jade it uses thr ACL communication language.
- **Protocol module** - contains the general rules of the negotiation (auction). When an agent initiates a negotiation, it finds out which negotiation protocol should be used and dinamically loads it from the user's local machine or a server (since these modules are the same for each agent).
- **Strategy module** - contains the proper reasoning module, that leads to the end of the negotiation with a success. In order to find out what reasoning model to use, the agent can consult a table which records the earlier history of transactions with the seller. In case that no history is, or the table is unavailable (since this table is on the user's computer, not in the agent) a default strategy is used.

It is considered, that the strategy module and the protocol modules can be quite large, so that they would impede the agent's mobility. This is why, the agent only contains a skeleton, and the communication module, while the other two modules are downloaded, after the agent arrives to the actual marketplace. Unfortunately, no concrete negotiation strategy is presented, only two very simple ones, used in the test cases.

The proposed system, is made of different types of agents. There is only one *Client Information Center (CIC) agent*, that is responsible for storing, managing and providing information about all participants in the system. All agents, have to register with the CIC agent, which stores their data in the Client Information Database (CICDB). This database is both a client registry, and a yellow pages service which gives information to client agents about the shops they are interested in. A *client agent (CA)* acts int the marketplace on behalf of the user, that attempts to buy something. Each such agent has a bunch of negotiation agents, called *buyer agent*, one for each shop, that sells the good they are looking for. A *shop agent* represents a user that wants to sell something and it has a number of negotiation agents with the "seller role", called *seller agent* for each product sold in the shop.

A general usage scenarion for this environment consists of the following steps.

- A Client agent registers with CIC and receives an ID, which can be a new ID if the agent has never been in the marketplace before, or if it has already been, the old ID is retrieved from the database.
- The Client agent queries the CIC to get a list of Shop agents that sell the product he seeks. For each such shop it creates a Buyer agent to handle the negotiations.
- Buyer agents migrate to the site their corresponding Shop agent exists, and query them about the negotiation protocol used in the given e-store, and dynamically load the corresponding negotiation and strategy protocols.
- The Seller agents periodically check if there are Buyer agents interested in their product, and if there is at least one, an auction is started. At the end of the auction, the Seller agent informs the Shop agent about the winner, which informs the Client agent that the purchase is possible. The Client agent, then decides if he buys the good, and where he buys it, in case that there are more than one possible Shops.

Framework for Automated Negotiation

Paper [9] describes a software framework, that allows automated negotiation. They start with the formal definitions of FIPA for several standard negotiation protocols, but they say that these definitions are not complete (for example the definition of the English auction does not contain that a bid is valid only if it is higher than the highest current bid), so agents will have to contain the rest of the details in the implementation code, which will stop them from being general, and capable of participating in any negotiation. Instead they suggest a method, that includes all these rules into the environment.

As first step they describe the abstract process of a negotiation, in which usually two roles are implied: the negotiation host, and the negotiation participant. In a general setting, communication between the participants is done through a "negotiation locale", which is a kind of blackboard, for which the read and write access is given by the host.

The first action is taken by the participant, which goes to a host and requires admission to the negotiation. If access is granted, the participant is informed about the rules of the negotiation, gathered in a negotiation template that contains the parameters of the auction. This template must be accepted at the admission. The next part is the negotiation itself, during which the participants exchange proposals that represent the agreements that are currently acceptable to them. These proposals are sent to the host, which validates them, before accepting it. A proposal may fail validation in two cases: restriction of the parameter space specified in the template is not respected or it does not follow the rules that govern the negotiation (who can make a proposal, when they can make it, what proposals can be submitted, compared to previous ones and so on). During the negotiation, agreements can be formed based on some agreement formation rules. The negotiation is ended, when some rules are satisfied. For example, in an English auction the termination rule would state that the auction finishes when there are no valid proposals for a periode of time, while the agreement is formed between the seller and the highest bidder, at the price of the highest bid.

Rule-Based Framework for Automated Negotiation

The above presented framework inspired the group of researchers that created the Jade based MAS presented earlier and they decided to add it to their model [7]. Since in the Jade based model, every Shop agent, has a separate seller agent for each product it sells, we can consider each seller agent a separate host, so that the environment will have many instances of the framework described above. For each of these intances we can have different rules for the negotiation. The automated negotiation framwork contains a host agent, but it has a some "sub-agents" that can perform different actions: Gatekeeper (handles the access of new participants), Proposal Validator (validates proposals against the negotiation template, and if they pass, will forward them to the Protocol Enforcer), Protocol Enforcer (checks if the proposal satisfies posting and improvement rules), Negotiation Terminator (checks if the condition to end the negotiation holds), Information Updater (regularly updates information on the blackboard) and Agreement Maker (applies the agreement formation rules to determine which agreements can be made). This host agent and the sub-agents are present here also, but they share the same JESS inference engine which has separate modules for the separate rule categories, instead of each of them having a separate one.

The actual strategies used by agents that particpated in experiments, were very simple: an agent posted a bid immediately after being admitted to the auction, and any other times, when it was informed that somebody else posted a bid higher than his own. The value of the new bid, was the current highest bid, plus an increment, that was private for each agent. If the current bid was above a limit, the agent stopped bidding.

In [8] the rules for this negotiation component for the cases of English and Dutch auction is presented. Unfortunately, there is no detail about the strategy the agent could use.

Probabilistic negotiation agents

Unlike the previous example, real-life negotiations usually include other attributes beside price. A MAS which takes more than one attribute into consideration is presented in [6]. Moreover, agents in this MAS are also capable of learning the right negotiation technique. The basic architecture of the system, is very similar to the previous ones: both the buyer and the seller uses an agent which finds publicly available marketplaces. After finding the marketplace, the agent can be deployed to it, which then can retrieve relevant information about the market from a facilitator agent. Once the right negotiation partners are found, if there already is a previous history with the agent, it is mined to find the preferences of the partner. Knowing these preferences helps the agent, because it can decide easier what offers to propose. If such a history does not exist, the negotiation can start immediately. During the negotiation, the agent's knowledge discovery mechanism can be used to estimate the opponents preferences, based on the recent negotiation history.

In order to understand how knowledge discovery can be used to adapt to the preferences of the partner agent, first, the general structure of the negotiation should be presented, as defined in [6]. The negotiation space is a 5 tuple $Neg =< P, A, D, U, T >$ which is made of the set of agents (P), the set of attributes over which the negotiation takes place (A), the domain for these attributes (D), a utility function defined on the Cartesian product of the domains of the attributes, with values in the [0,1] interval (U) and the set of deadlines for the agents (T). It is supposed that P, A, D are supplied by the marketplace and they are known by everyone.

Using these notations, an offer $o = (d_{a_1}, d_{a_2}, ..., d_{a_n})$ is a vector of attribute values, or attribute value intervals. An example for an offer could be $o_i = (20-30, 1-2, 10-30, 100-500)$ where the first attribute is the price, expressed in dollars, the second attribute is the "warranty periode" expressed in years, the third one is the "shipment time" measured in days, and the fourth is the quantity [6].

To evaluate such an offer, every agent has two valuation functions: one, $U_p^A : A \rightarrow [0,1]$, which shows the weight of a given attribute (for some users, price may be more important than shipment), and a second function $U_p^{D_{a_i}} : D_{a_i} \rightarrow [0,1]$, which shows the importance of the value of an attribute. Using these two functions the utility of an offer can be computed as: $U_p(o) = \Sigma_{a_i \in A, d_{a_i} \in o} U_p^A(a_i) * U_p^{D_{a_i}}(d_{a_i})$.

If the offer of an agent is rejected by its negotiation partner, it will have to make a counter offer. This offer is chosen from the set of all possible, not yet suggested offers, such that the decrease in own utility should be the minimum. The negotiation stops with an agreement if an offer is received that is better than all possible not yet offered counter offers of the agent, or it is an offer, that was already proposed by the current agent.

The probabilistic negotiation agents presented in [6] use the above description of an offer, and they try to achieve two contradictory things: maximize own payoff, and maximize the chance of reaching an agreement. The self payoff can be computed as described above, while the second one can be achived using Bayesian learning, based on previous negotiation history with the current agent, or (if this is not available) the current negotiation dialog. In this model, each possible offer gets a ranking computed in the following way: $Rank(o) = [U_p(o)]^{\alpha} * [P(accept|o)]^{1-\alpha}$, where α is between 0 (agent tries to maximize opponents payoff) and 1 (agent cares only for his payoff) [6]. This parameter can be set by the user, otherwise the 0.5 default value is used. This rank is used to choose the best counter offer in case that the offer received from an agent is not accepted.

The probability of accepting an offer o can be computed using Bayes theorem: $P(c_j|o) = \frac{P(o|c_j)*P(c_j)}{P(o)}$, where $c_j \in \{accept, reject\}$. Using some rules from the probability theory, the probability of accepting an offer o becomes the following:
$P(accept|o) = \frac{P(accept)*\Pi_{i=1}^{|A|} P(d_{a_i}|accept)}{P(accept)*\Pi_{i=1}^{|A|} P(d_{a_i}|accept)+P(reject)*\Pi_{i=1}^{|A|} P(d_{a_i}|reject)}$.

In case that the counter offer does not contain some attributes, the corresponding $P(d_{a_i}|accept)$ and $P(d_{a_i}|reject)$ has the value 1. If we suppose that the preference of an agent does not change during the negotiation, then these probabilities can be estimated based on previous negotiation history with the agent, once before the start of negotiation and consider it unchanged during it. On the other hand, for agents that might change their preference during the negotiation, the value of $P(accept|o)$ is recomputed after each round, based on the most recent negotiation.

In order to compute the prior probabilities P(accept), P(reject), $P(d_{a_i}|accept)$ and $P(d_{a_i}|reject)$ knowledge mining on negotiation history is done. The basic assumption is, that avery offer that an agent receives from its partner is a positive training example, while every offer that the agent proposes, but is rejected, is a negative example. Moreover, recent offers give more information about the partner's preference, than older ones. Using these asspumptions, training examples from the previous sessions, and the current negotiation dialog are weighted. The weight of a session i, w_i^S is computed in the following way: $w_i^S = w_{max} - step * \frac{w_{max}-w_{min}}{|session|-1}$. w_{max} and w_{min} are positive parameters of the method, and represents the maximum and the minimum weight that can be assigned to a negotiation session; step is the number of the session, the most current session having step 0, the previous step 1 and so on.

One negotiation session can contains multiple offers. Each such offer can have different weights. The weight computed with the above formula is assigned to the first counter offer of the partner from the session (based on the idea, that this counter offer, would have been the best from the point of view of the partner). For the rest of the offers, the weight is computed as: $w_{ij}^E = w_i^S - (j-1) * \frac{w_i^S - w_{i+1}^S}{|E|}$, where $|E|$ is the number of entries in this session.

An example of such weights can be seen on Table 1 taken from [6]. The maximum weight is 500 while the minimum one is 200. The sum of all weights is 3650, and the sum of weights for offers where the category is accept (it has Y in the opponent accept column) is 1500, so $P(accept) = \frac{1500}{3650} = 0.41$. The other conditional probabilities can also be computed, for example $P(price = 25-30|accept) = \frac{675}{1500} = 0.45$.
One common problem with the naive Bayesian learning is the presence of zero conditional probabilities, i. e. cases which does not appear in the training set. For such cases, this method uses

Table 1: Example of weights associated with different offers from negotiation sessions

Session	Offers	Price (d_{a1})	Shipment Time (d_{a2})	QTY (d_{a3})	Opponent accept	Weights
4	o1	5 - 10	1 - 2	20 - 30	N	200
	o2	15 - 20	3 - 4	50 - 50	Y	175
	o3	1 - 2	2 - 2	10 - 20	N	150
	o4	25 - 30	5 - 8	60 - 100	Y	125
3	o1	5 - 10	1 - 2	20 - 30	N	300
	o2	15 - 20	3 - 4	50 - 50	Y	275
	o3	1 - 2	2 - 2	10 - 20	N	250
	o4	25 - 30	5 - 8	60 - 100	Y	225
2	o1	5 - 10	1 - 2	20 - 30	N	400
	o2	15 - 20	3 - 4	50 - 50	Y	375
	o3	1 - 2	2 - 2	10 - 20	N	350
	o4	25 - 30	5 - 8	60 - 100	Y	325
1	o1	5 - 10	1 - 2	20 - 30	N	500

a constant, the probability which is computed as $P(d_{a_i}|c_j) = \frac{P(c_j)}{N}$, where N is the sum of weights.

In case of an offer which contains an interval that intersects more than one from the history, two different approaches can be adapted, depending on the optimism characteristic of the agent. If it is optimistic, it can take the highest available accept and the lowest reject probability used, while for pessimistic agents it is vice versa.

Experiments done with this model are described in [6], and they demonstrate that the probabilistic agents perform better than the Pareto optimal agents (agents that use only the technique described at the negotiation part - so no probabilities, and knowledge discovery).

FINA

[10] presents an agent model that uses fuzzy logic in negotiation. This model is designed for multi-issue one-to-one negotiations (no auction). In order to build any negotiation agent three main ideas have to be considered: negotiation protocols (rules of the negotiation), negotiation issues (issues over which agreement must be reached) and the agent reasoning models. The agent reasoning model concentrates on processing incoming offers, generating counter offers sent to the seller (or buyer) and on making the decision to interact with the opponent or not.

The general structure of a reasoning model for such an agent, as described in [10], consist of the following parts: it has as input the incoming offer, which can contains values for more than one attribute, and it has one of the following three outputs: accept, reject (no further communication), counter-offer. Inside the model there are three components: new offer generation engine (which generates new offers), offer evaluation block and the decision making block.
The offer evaluation block receives both the new offer generated by the engine and the incomming offer from the opponent. It does an analysis of them, and computes the degree of satisfaction of them. The result is scaled over the values 0 and 100. Finally, the decision making block makes the final decision, which can be one of the above mentioned three possiblities.

The modeling of the new offer generation engine can be seen as a Distributed Fuzzy Con-

straint Satisfaction Problem [10]. The offer evaluation system can be considered a fuzzy expert system. Since human preferences usually are vague and uncertain, fuzzy logic is a good choice for this module. A simple variant for the decision block can be a function with the following rules:

- if $A_{in} < A_{min}$ REJECT
- if $A_{in} \geq A_{counter}$ ACCEPT
- if $A_{in} < A_{counter}$ COUNTER OFFER

4 Conclusions and discussion

Most of the above described examples are only theoretical, with the exception of the search agents BargainFinder and Jango, which were used online, but now they are not accessible. The MAS implemented in Jade (and extended with JESS) seems to be a nice framework that could scale up to become a large application, but there still is a lot of work to be done, especially because that model has no description about the optimal negotiation strategy (although, that group of researchers has many papers related to the subject, so they might have solved that problem, too). Still, the paper tries to present a general framework that can be used for any auction, which is a positive thing, since some of the early applications, had problems with the lack of standards.

The fuzzy logic based approach seems interesting, because it tries to capture the uncertainty that is characteristic to human users. The most complex, and maybe promising seems the be the model, where the strategy can be learnt during negotiation, although there the whole set of not yet tried possible offers is needed to find the best one to offer, which might be time consuming to compute, and can have many elements, even if attributes intervals are used instead of simple values.

Agent technology is an important improvement for e-commerce and it will definitely improve in the future, because it makes buying things on the internet faster and easier. While negotiation using e-mails could take days, if not weeks, until an agreement is reached, the same thing can be done with agents in a matter of seconds. On the other hand, almost all the above presented examples lack a very important part from the agents: the exact strategy that they should follow during the negotiation. This can partially be explained by the fact, that humans are unable to express their wishes and utility functions clearly, in many cases the actions they take are not based on logic, or computations, but on intuition and feelings, which are unable to teach to an agent. Still, if these utility functions could somehow be expressed using mathematics or logic, due to the computing power of the agents (which is a lot larger than that of humans) they would find better solutions than humans.

Another problem with these agents could be connected to their autonomy. Ideally, because they are autonomous, these agents should take the final decisions in the negotiation (up to the point of making credit card transactions), but I don't know if humans would trust them sufficiently to let them do this. Possible, if the above described problem with the definition of a negotiation strategy would be solved, people's trust grew.

References

[1] Antony Chavez, Pattie Maes: Kasbah: An Agent Marketplace for Buying and Selling Goods, Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology pg. 75-90, 1996

[2] Nir Vulkan: Economic Implications of Agent Technology and E-Commerce, The Economic Jurnal, pg. 67-90, 1999

[3] Internet World Stats: http://www.internetworldstats.com/stats.htm (visited on 6. 12. 2010)

[4] Bruce Krulwich: Information Integration Agents: BargainFinder and NewsFinder, Internet-Based Information Systems: Papers from the 1996 AAAI Workshop, pg. 72-77, 1996

[5] Maria Ganzha, Marcin Paprzycki, Amalia Pirvanescu, Costin Badica, Ajith Abraham: Jade Based Multi-Agent E-Commerce Environment: Initial Implementation, Analele Universitatii din Timisoara, Seria Matematica-Informatica, Vol. XLII, pp. 79-100, 2004

[6] Raymond Y.K. Lau, Yuefeng Li, Dawei Song, Ron Chi-Wai Kwok: Knowledge Discovery for Adeptive Negotiation Agents in E-Marketplaces, Decision Support Systems, Vol.45, Nr. 2, pg. 310-323, 2008

[7] Costin Badica, Adriana Badita, Maria Ganzha, Alin Iordache, Marcin Paprzycki: Rule-Based Framework for Automated Negotiation: Initial Implementation, Proceedings of RuleML, 2005

[8] Costin Badica, Gabriel-George Popa, Mihnea Scafes, Maria Ganzha, Maciej Gawinecki, Pawel Kobzdej, Marcin Paprzycki: Degin Considerations for a Negotiation Component in a Model E-commerce Agent System, Symbolic and Numeric Algorithms for Scientific Computing, 2006

[9] Claudio Bartolini, Chris Preist, Nicholas R. Jennings: A Software Framework for Automated Negotiation, SELMAS 2004, LNCS 3390, 2005, pg.213-235

[10] Xin Wang, Xiaojun Shen, Nicolas D. Georganas: A fuzzy logic based intelligent negotiation agent (FINA) in e-commerce, Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering, 2006

[11] Minghua He, Ho-fung Leung: Agents in E-commerce: State of the Art, Knowledge and Information Systems, Nr. 4, pg. 257-282, 2002