A Prototype for an Agent-based Secure Electronic Marketplace including Reputation Tracking Mechanisms

Boris Padovan, Stefan Sackmann, Torsten Eymann, Ingo Pippow
Albert-Ludwigs-University Freiburg
Institute for Computer Science and Social Studies, Telematics Dept.
Tel. +49-761-403-4963 – Fax +49-761-203-4929
Friedrichstrasse 50 – 79098 Freiburg im Breisgau – Germany
{padovan|sackmann|eymann|pippow}@iig.uni-freiburg.de

Abstract

The future of electronic commerce will be shaped by open, heterogeneous and complex structures, consisting of networked marketplaces. Software agents will interact and negotiate on behalf of their human (or organizational) principals. Principals will be able to implement fraudulent strategies in their agents, which cannot be countered by technical security alone. In the absence of a single correctional institution, agents will have to rely on social mechanisms for assessing reliability and reputation of other, unknown agents.

The multi-agent system AVALANCHE is a prototype for an agent-based secure electronic commerce marketplace environment. The reputation tracking mechanism, which is implemented in AVALANCHE's software agents, evaluates transaction behavior and influences partner selection and negotiation strategy in future transactions, while protecting the privacy of the participants. The successful result is an increasing expulsion of fraudulent agents from the market.

1. Introduction

1.1. Self-Interested Agents in Electronic Marketplaces

Distributed, open, heterogeneous and independent electronic marketplaces will shape the future of electronic commerce over the whole Internet. The increasing number of electronic marketplaces will make it impossible for individual human users to keep track of the ever-changing offer, demand and price situation. This leads to the necessity for humans to find new ways of being simultaneously present at as many markets as possible, in order to conduct profitable transactions. A promising way of fulfilling this challenge is the use of software agents that represent their human principals at the marketplace and are able to conduct a whole business transaction in a satisfying way. Autonomous agents and multi-agent systems represent a new way of analyzing, designing, and implementing complex software systems [2]. Software agents are computer programs which function continuously and autonomously in an environment in which other processes take place and other agents exist [30]. In the context of electronic marketplaces, agents will monitor other trade agents continuously, watching for potential opportunities. They will be able to enter into negotiation with many potential trading partners at once, reaching an acceptable deal and setting up a contract in a matter of milliseconds [13] [25] [31]. Today's software agents (often called shop-bots) are mostly used for information retrieval, while autonomous negotiation and realization of transactions are not supported [12]. The software agents described in this paper can be characterized as performative agents [22], which act autonomously during the negotiation phase (e.g. in the Consumer Buying Behavior (CBB) model [12]).

While current research on agents in electronic markets is focused on idealized computational economies, it should be emphasized that future electronic markets will exhibit self-interested participants, who are completely free to act in their own best interest [27]. Conflicts between the agents will be settled using budget constraints rather than a relative importance of individual goals. In a market world without a centralized institution and a common goal, principals will also be free to enforce their own interest with agents that speculate, act fraudulently, or cheat in negotiation.

Open software-agent-based electronic marketplaces are not a mature technology yet, but they will appear from the technological development of shopbots and pricebots alike, which already receives public attention [1]. Amongst others, we envisage the evolution of the Internet into a free-market information economy (an *information ecosystem*) in which billions of artificial software entities exchange a rich variety of information goods and services with humans and amongst themselves [9][14]. In this environment, agents have to interact with unknown part-

ners. Ad-hoc cooperation is subject to high risks because of the asymmetric distribution of information. To trust an unknown partner is paramount for inter-human and inter-agent business transactions in open, insecure network environments [5]. To reduce the risks of financial loss, a mechanism is needed to obtain information about the earlier cooperative behavior of a certain transaction partner – his reputation.

The remainder of the article is organized as follows: After a brief introduction to the roles of security, reputation and trust in multi-agent systems (chapter 2), a supply chain management scenario is described, where software agents cooperate on electronic marketplaces to maximize their own utility. Chapter 3 shows the multi-agent system (MAS) AVALANCHE as a technical implementation of this scenario. The result shows the coordination of supply chain steps via decentralized market mechanisms. Chapter 4 builds upon this environment with an implementation of a reputation-tracking mechanism, which is able to enforce cooperative behavior between the agents. The article finishes with some results of the prototype's implementation.

2. Reputation and Trust in Multi-Agent Systems

A central goal of open marketplaces will be to show that while market-like coordination can be achieved relatively easily on a solitary marketplace, market-like coordination in open wide-area networks is also possible, regardless of the underlying technology. This extension leads to further problems, e.g. security and trust, and will be realized in three steps. First, systems with a single marketplace will connect themselves within a closed, organized and legally homogenous environment. In the second step, marketplaces will open themselves to the outer world, allowing a priori unknown software agents, e.g. from real world suppliers and customers, to trade within the network. In a last step, the marketplaces are distributed over the whole Internet, just as Web servers are today.

If the electronic marketplace environment is opened to other software agents with a priori unknown owners, security aspects of the system come into consideration. Three major issues concerning security can be identified:

Firstly, the **authentication** of the software agents is not guaranteed, because their real identities might remain unclear. The human owners might use pseudonyms to hide their real identity. These pseudonyms can be changed easily within a short period of time, and the same person can appear under several pseudonyms simultaneously (which in auctions quickly leads to a phantom bidder problem).

The second area concerns the **privacy** of the communication. Apart from the fact that confidential communication is always subject to previous authentication of the communication partners, numerous possibilities to attack still exist in insecure networks like the Internet today.

Thirdly, even with all other solutions applied, software agents are still free to behave in a **non-cooperative** way, e.g., delivering goods of poor quality. In open electronic marketplaces, means of enforcing commitments are mostly non-existent or ineffective, especially if the identity of the non-cooperative contractor cannot be definitively specified. If a participant of an electronic marketplace is known as non-cooperative, it can easily change its pseudonym to obtain a new and better reputation. Even if a static reputation monitoring mechanism is working well, the possibility of changing identities makes it obsolete.

These three topics can be encountered by means of technical and non-technical security measures.

Technical Security. As a prototype, the multi-agent system AVALANCHE already implements a common asymmetric public-key cryptosystem (see [20] and [10] for a closer look at public-key encryption). By digitally signing their messages, software agents can be easily identified. Every agent has one public and one private key. The private key remains in the possession of the principal's host computer. The public key is distributed together with the agent's identity at the marketplaces' directory. The identity, the offers and inquiries, as well as every message sent by the software agent is digitally signed in order to prevent faked messages or changed identities. Software agents then use asymmetric encryption to encrypt the messages, which makes it impossible to monitor the content of the communication between any two software agents.

Non-technical Security. Even if all software-agents can be identified and are able to communicate privately and confidentially, non-cooperative behavior of an entity cannot be prevented. Recent research works have shown that the missing knowledge about the competence and trustworthiness of an unknown transaction partner is the major barrier for new Internet-based business cooperation [5]. Ad-hoc cooperation is subject to high risks because of the asymmetric distribution of information. To trust an unknown partner is paramount for inter-human and inter-agent business transactions in open, insecure network environments.

Definitions of **trust** are very rare. Koller [16] defines trust as the expectation that an interaction partner will show benevolent behavior, although he has the possibility of choosing other, non-benevolent, manners. Deutsch [4] gives a similar, but more detailed definition of trust:

"(a) the individual is confronted with an ambiguous path, a path that can lead to an event perceived to be beneficial (*Va+*) or to an event perceived to be harmful (*Va-*); (b) he perceives that the occurrence of *Va+* or *Va-* is contingent on the behavior of another person; and (c) he perceives the strength of *Va-* to be greater than the

strength of *Va+*. If he chooses to make an ambiguous path with such properties, I shall say he makes a trusting choice; if he chooses not to take the path, he makes a distrustful choice."

Trust is needed to reduce complexity before a transaction [15]; otherwise the transaction cannot be performed. When nothing is known about the transaction partner, it is impossible to compute all potential outcomes because of complexity. The agents have to be content with estimations, which requires trust as replacement for uncertainty. But in these cases, however, transactions can only be performed with high risks. To reduce the risks, a mechanism is designed to obtain information about the former cooperation behavior of a certain transaction partner. Trust is not a timeless feature. In order to conduct a transaction with an unknown partner, it is helpful to obtain information about its previous cooperation behavior, because "an agent, as a trusting entity, has a basic trust 'value', deriving from previous experiences" [18]. This information leads to the transaction partner's **reputation**. Marsh [18] defines reputation as the amount of trust inspired by the particular person in a specific setting or domain of interest. A detailed discussion of reputation can be found at [21].

In multi-agent systems reputation is viewed as the expectation of the cooperative behavior of a single agent by its environment. The reputation value is the rating of the software agent's cooperative behavior by its former transaction partners. "Good" reputation means that the transaction partners expect a cooperative behavior, e.g. that the agent is keeping commitments. A software agent has a "bad" reputation, if it is expected to behave non-cooperatively e.g. by not keeping to commitments. If no information about the agent is available, it has "no" reputation at all. It should be noted that there is a difference between having a bad reputation and no reputation.

3. A Real World Scenario for Agent-Driven Marketplaces

In order to evaluate the future impact and functionality of agent-driven marketplaces today, a prototypical market system called AVALANCHE has been designed, where software agents represent human participants intending to buy, produce and sell different goods or services. A simple and arbitrary real world value chain is modeled in which three different types of software agents produce tables: "lumberjacks", that buy trees to produce and sell boards; "carpenters", that buy the boards, fix it together and sell it as panels; and "cabinet-makers", that buy the panels, build tables out of them and sell them. Typically, a value chain consists of several interconnected organizational units [24], which use raw materials as input and by adding labor and knowledge enhance the value of their manufactured product. This output is then taken as input by the next organizational unit in line until the product reaches the consumer. The AVALANCHE agents buy, sell and produce, truly "miniature automated businesses" [15].

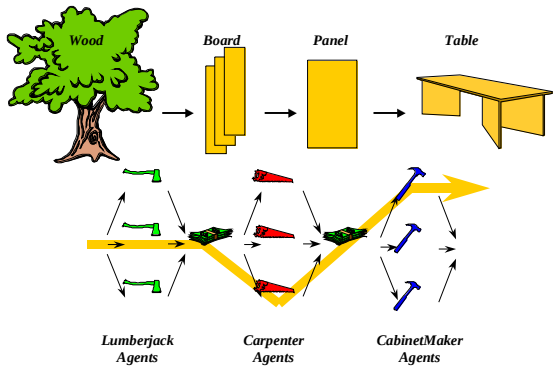

Figure 1. A simple scenario: a value chain from tree to desk.

3.1. Architecture and Classes

AVALANCHE is realized in JAVA. The architecture consists of three basic classes, the marketplaces (*AvLocationAgent*), trader agents (*AvTradeAgent*), and a monitoring and control object (*AvInfoServer*). In the prototype, both marketplaces and trader agents are initialized simultaneously, but due to the openness of the architecture, trader agents can enter marketplaces at any point in time during the course of the simulative experiments. The technical details of the initialization process are documented in [7].

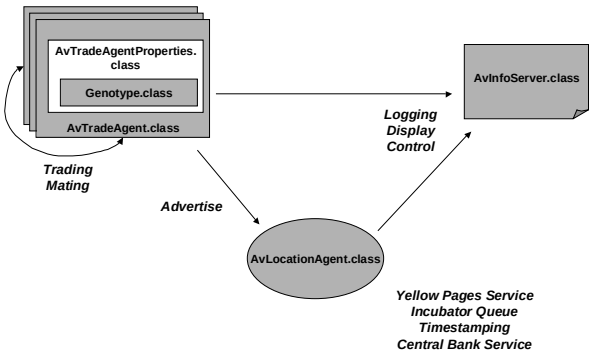

Figure 2. Architecture of AVALANCHE

The marketplaces are Java Virtual Machines, which can be regarded as a sandbox environment for the software agents. Every marketplace holds one *AvLocationAgent* object. The *AvLocationAgent* never actively influences the software agents' behavior, and in order not to interfere with the trading process of the software agents, the functions and services of the location are kept to a minimum. The *AvLocationAgent* thus offers a "white pages" directory service for the software agents, enabling them to find out which other agents are at the same location. For mobile agents, the directory service also provides information about neighboring marketplaces. Marketplaces are addressed and distinguished by a unique IP-port number on a network-connected computer (e.g. 127.0.0.1:7777).

The *AvTradeAgent* class defines the abilities of agents to produce, communicate, negotiate and move between marketplaces. All agents are based on standardized concepts of Java [23], and distinguishable by a unique identity (ID). Each agent runs in parallel as single Java thread, and will not be synchronized explicitly as in common market implementations, which use time-slicing or scheduling through arbitrators or auctioneers, as in Market-Oriented Programming [32].

The *AvInfoServer* class has no effect on the market. At the moment, all relevant transaction data is reported from the trader agents to the *AvInfoServer* and then to a central log-file, which can be visualized afterwards with common analyzing tools.

3.2. Individual Strategies

The software agents' behavior in the *AvTradeAgent* objects is determined by a small set of general rules with probabilistic elements to produce heterogeneity in the population, which is given to them during initialization by the *AvTradeAgentProperties* object. This class simulates the definition of goods, goals and strategies, which human principals would implement in their own software agents.

The software agents communicate unmediated and bilaterally by passing messages to each other. The communication channel can be encrypted and messages can be signed with a digital signature of the software agent, using public-key encryption. The negotiation protocol used has been largely taken from FIPA's Agent Communication Language [11], implementing elements of an iterated contract net [30], and modified to reflect self-interested behavior [6][27] of the single software agent. Comparable automated negotiation efforts in multi-agent systems can be found in the research context of agent-mediated electronic commerce [2][25] and market-oriented programming [32].

The agent's set of actions consists of buying, selling, producing, moving, and termination. Depending on the market situation and its internal state, the software agent decides autonomously what to do. The basic algorithm works as follows: If the software agent has produced output in stock, it tries to sell. If the software agent has all required input factors in stock, it produces output. If the software agent has no input factors in stock, it tries to buy some. If the market situation is not satisfying at all, e.g. if there are no offers or demands within a certain time span, the agent tries to move to another marketplace. If the software agent has spent its entire budget or all marketplaces are shut down, it has to terminate.

The goal of the agents, and the reason to engage in negotiation at all, is to maximize the "equity" capital: the agents try to buy materials for a low price and sell their products at high price to other software agents which in turn use these as input goods. To maximize the spread and thus their utility, the agent follows a certain strategy. This strategy can essentially be conceived as a Markoff model, a non-deterministic finite state machine, in which known decision paths are taken depending on stochastic probes against certain internal parameters. This method is intended to represent real human behavior better than the simple functions of similar agent projects like e.g. KASBAH [3]. The non-determinism leads to heterogeneous negotiation behavior, in certain parameters comparable to that of humans in empirical studies [26]. At present, the following 6 parameters form a *Genotype* object as part of the *AvTradeAgentProperties* object, which encodes the behavior and strategy of the software agents:

- *Acquisitiveness* represents the probability of sticking with the agent's own last offer and not making a concession. A concession means changing an offer price as supplier or a demand price as buyer at the next negotiation step.
- The *In-negotation delta price change (del_change)* parameter calculates the amount of the price concession between two negotiation steps. Both partners calculate a percentage from the price difference of their original offers. If the buyer has *del_change = 0.25* and the supplier *del_change = 0*, the agents will reach agreement after four negotiation steps at the original price of the supplier, if no one drops out.
- *Pre-negotiation delta price change*. To maximize income, the agents will try to raise their initial offer price between negotiations. This parameter is influenced by the past performance and the perceived market information (e.g. prices from unsuccessful negotiations).
- The *Satisfaction* parameter decides with a certain probability if the agent will continue an ongoing negotiation or if it will drop out. The more steps the negotiation takes, or the more excessive the partner's

offers are, the sooner the negotiation will be discontinued.

- All price information is computed into the parameter *Memory*, which thus holds the perceived market price. Negotiations will only be started if an initial offer price is below 200% of this value depending on the agent's *Satisfaction*, or unconditionally if below the *Memory* value.
- *Reputation* affects the cooperative behavior of the software agents (see section 4).

3.3. Negotiation in a monotonic concession protocol

Economic transactions in electronic markets can be divided into three stages: information, agreement, and settlement [29].

In the **information phase** of any transaction, a buyer or seller has to identify his potential trading partners. The specific implementation depends on the environment and the services offered by the marketplace. The marketplaces of AVALANCHE support a directory service, where software agents can register as sellers or buyers, and inquire for other registered agents. Price bids are not posted (as in catalogs), since the price information might partly reveal the agent's strategy to competitors and bidders, and prevents the negotiation of individual prices per bidder. If a seller agent wants to sell its output, it currently advertises its offer with the directory, and waits until a prospective buyer agent demands a negotiation, and vice versa. Simultaneously, the agent will actively search in the directory for potential transaction partners.

After receiving the identity of potential transaction partners present, the buyer agent initiates the **agreement phase** by communicating with (a subset of) the identified supplier software agents, thus realizing a local market survey. Its preference function ranks the suppliers by unit price, for example, and enters into negotiation until its demand is satisfied. The software agents negotiate by using direct, unmediated and bilateral communication, which can be characterized as a monotonic concession protocol [28], a sequence of *propose* and counter-*propose* messages, similar to an one-layer iterated contract-net-protocol [11] (see Figure 3). The negotiation variables cover only the price, as the goods are commodities. Further variables such as quality and terms-of-trade require additional logic, e.g. multi-attribute utility theory [12], which has not been implemented yet. If the negotiation process is successful, both agents will reach an agreement by making price concessions. Whether a compromise is eventually reached, depends on the individual negotiation strategies encoded by the variables in the *Genotype* object.

However, both agents have the opportunity to drop out of the negotiation, if they are not satisfied with the progress. In this case, the information about the price suggestions is retained and both agents are up to start new negotiations with other agents from their shortlist. It is possible that the agents from that list might have satisfied their demand already, so that the agent might have to start with the information phase again.

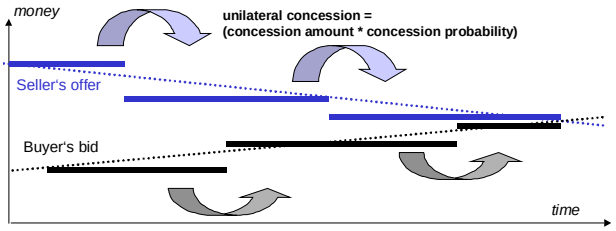

Figure 3. Monotonic Concession Protocol

In the final **settlement phase**, the transaction is carried out and monitored. The transaction partners will exchange goods and money respectively. This phase shows if the confidence in the transaction partner's reputation was justified and modifies the reputation setting accordingly. This information gives some feedback to be used in future information and agreement phases.

After the acquisition of the input goods, the buyer agents will stay idle as long as the corresponding physical production process is running, which is currently simulated as a fixed time span. After that, they will start selling the finished products.

3.4. Experimental Results

In the current implementation of AVALANCHE, typically 250 software agents trade on one electronic marketplace on a 133 MHz Pentium Windows PC. Figure 4 shows how the prices of the traded goods develop over time through the mutual trade of the software agents, when all agents are homogeneous. The horizontal axis measures the time in milliseconds, the vertical axis shows the price level. Every dot in the graph points out time and agreement price of a transaction, e.g. the x-value of a triangle marks the time when some carpenter software agent has sold a table to a cabinet-maker software agent for y-value price.

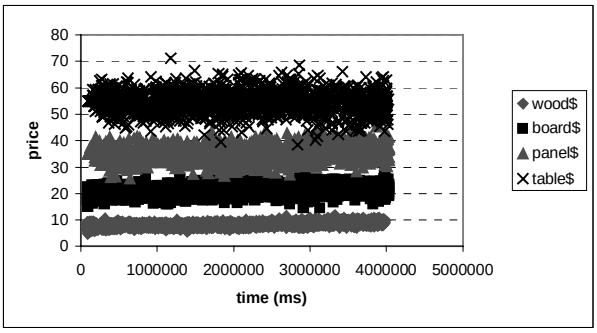

Figure 4. Price development in AVALANCHE

In Figure 5, the absolute settings of the strategy vary for the *acquisitiveness* variable. The carpenter software agents, that buy boards (pink) and sell panels (yellow), have a higher *acquisitiveness* value (0.85) than the other types (0.60). Therefore, they are able to increase their income (the spread between board prices and panel prices) at the expense of both lumberjacks and cabinet-makers. If the agents are able to learn and revise their strategies like in human market experiments [26] (e.g., by using a decentralized evolutionary algorithm [13]), this development can be counterbalanced and a stabilization of price curves can be achieved [8]. A systematic evaluation of the interdependencies between parameter settings and results are the subject of further research.

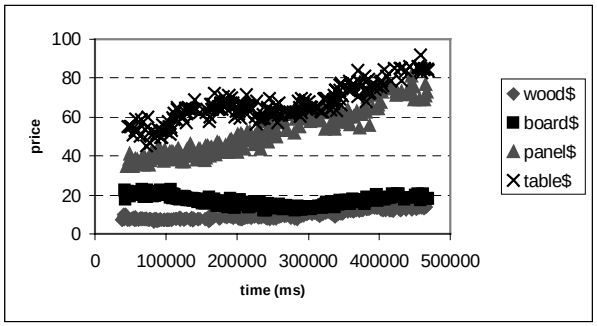

Figure 5. Price development in AVALANCHE with uncooperative carpenter agents

The *reputation* of the agents has been the highest possible to this point, which means that all agents will behave cooperatively and carry out the transactions as committed. An extension of these experiments is a distribution of evil and honest agents, as in the real world. The next section will concentrate on such experiments, and on centralized and decentralized countermeasures against evil agents.

4. An Implementation of a Reputation Tracking Mechanism

The r*eputation* parameter of the software agent's genotype affects the cooperative behavior of the agent. With the probability given by this parameter, the agent will effectively cheat his opponent in the settlement phase. As a buyer, the agent will not pay, and as a seller, it will not deliver the goods. This simple variable is intended to cover many real-world problems, such as delivering poor quality goods or demanding more money afterwards. It should be noted, however, that the reputation is caused by the behavior of the principal, who defines the agent's strategy; if technically possible, it is necessary to rate the principal and his agent(s), but this problem will not be addressed here.

The predisposition of the agent to cooperate in the actual situation cannot be computed by its negotiation partner, but only estimated by the reflection of earlier experiences to give a **reputation coefficient**. Every software agent has the possibility to log its cooperation experiences and to calculate its subjective estimations of the partners' reputation coefficients. The formal notation of the reputation coefficient in the software agents of AVALANCHE is R_Y^X, where R stands for the reputation coefficient, X for the identity of the rating software agent and Y for the identity of the rated software agent. The reputation coefficient obtains $0 \le R_Y^X \le 1$, where $R_Y^X = 1$ stands for good reputation and $R_Y^X = 0$ stands for bad reputation.

A simple example will clarify this approach. Two software agents, A_1 and A_2, want to perform a transaction. A_1 has a certain reputation coefficient for A_2 with the value $R_2^1 = \alpha$. This value is just the estimated probability of the cooperative behavior, as a result of A_1's former experiences with A_2 and is not inevitably corresponding to the real probability. It is possible that a third software agent A_3 might have a different coefficient for A_2's reputation: $R_2^3 \neq R_2^1$.

In general, the reputation coefficient is used to adapt the software agent's negotiation strategy according to its partner's expected cooperative behavior. AVALANCHE uses simple rules to take the cooperative behavior into account, in order to keep the behavior mechanism of the software agents straightforward. Obtaining and using the reputation coefficient can be cut into four stages: obtaining the reputation coefficient, adapting it to the own negotiation strategy, rating the partner's behavior after the transaction and distributing reputation information to others.

4.1. Obtaining Reputation Information

The possibilities of obtaining reputation information depend on the level of awareness of the software agent *Y* with which a transaction is planned:

Case 1: Software agent *Y* is absolutely unknown to the whole environment, called *domain* in this context. In this case, the questioner's software agent *X* has a wide choice of estimating its counterpart's reputation. In the current implementation of AVALANCHE it can either use an average of all already known reputations $R_Y^X = \overline{R}$ (with $\overline{R}$ as the arithmetic average of all reputations) or *X* can use a default value between 0 and 1. Before the first transaction of the whole system has taken place, no reputation information is available at all, so $\overline{R}$ does not exist. Therefore, a default value between 0 and 1 has to be chosen by the owner.

Case 2: *Y* is unknown to *X*, but known to the *domain*, e.g. to a special software agent that runs a rating agency. Information about the reputation can be obtained from this special software agent. In the "real world" conducting businesses with partners, which are unknown to the entrepreneur, but already known to its domain is a very usual case. Better Business Bureau (BBB), Web of Trust, the German Schufa, and lots of other businesses offer ratings about the former cooperation behavior of potential transaction partners.

Case 3*: X* has already made transactions with *Y*. It can use its own information about the cooperation behavior and eventually take further information from the rating agency software agent into consideration.

4.2. Adapting Reputation Information to Negotiation Strategies

At the beginning of a negotiation, a software agent receives offers from potential negotiation partners. All these offers $(i = 1..n)$ are ranked by an assessed offer price p_{Yi}^*, which takes into consideration the originally offered price p_i and the reputation coefficient R_{Yi}^X. In the current implementation of AVALANCHE, the assessed price offer is calculated as the originally offered price plus the expected value of loss:

(1) $$p_i^* = p_i + \left(R_{Yi}^X \cdot 0 + \left(1 - R_{Yi}^X\right) \cdot p_i\right) = p_i \cdot \left(2 - R_{Yi}^X\right)$$

This is explained in the following example (see table 1), where the software agent X has four offers (from software agent 3, 5, 12, and 16) to rank.

Following the assessed ranking, the software agent *X* starts its negotiation with its negotiation partner #16. If the negotiation with #16 fails, *X* continues with # 12 and so on.

Table 1: Assessment of price and reputation

I	Y_i	p_i	R_{Yi}^X	$2 - R_{Yi}^X$	p_i^*	rank
1	12	47	$R_{12}^X = 0.63$	1.37	64.39	2
2	3	52	$R_3^X = 0.65$	1.35	70.20	3
3	16	54	$R_{16}^X = 0.85$	1.15	62.10	1
4	5	56	$R_5^X = 0.44$	1.56	87,36	4

With the further development of the AVALANCHE system, other ranking mechanisms based on the ones recommended by [19] can also be easily implemented.

4.3. Rating Cooperative Behavior

After each settlement of a transaction, the two software agents involved rate each other internally with the factor r_j (with *j* as an index of the transaction). In the current implementation, an unsuccessful transaction, which was caused by the non-cooperative behavior of a software agent, leads to a rating of $r_j = 0$, and a successful transaction leads to a rating of $r_j = 1$.

4.4. Distribution and Updating

The newly obtained rating r_j updates the reputation coefficients of the transaction partners involved. Currently, an exponentially weighted moving average is used, with the weighting factor α. In the current implementation a value for α can be set while configuring the software agent by its owner.

The new reputation coefficient is then calculated as:

(2) $$R_{Y\,j}^X = R_{Y\,j-1}^X \cdot (1-\alpha) + r_j \cdot \alpha$$

If a specialized rating agency is used, both transaction partners may forward the latest rating r_j^V $(V = X, Y)$. The rating agency also uses the exponentially weighted moving average with the factor β. It regards a constant weighting factor γ and the reputation coefficient of the rating software agent R_V. In the current implementation, a global value for γ can be set while initializing the system. The rating agency updates the reputation coefficients of the transaction partners involved with:

(3) $$R_{Y\,j} = R_{Y\,j-1} \cdot (1-\beta) + r_j^X \cdot \beta \quad \text{with} \quad \beta = \gamma \cdot R_{X\,j-1}$$

and

(4) $$R_{X\,j} = R_{X\,j-1} \cdot (1-\beta) + r_j^Y \cdot \beta \quad \text{with} \quad \beta = \gamma \cdot R_{Y\,j-1}$$

In the special case that no information of the rated software agent has been available before the transaction, the rating agency bases its update on the arithmetic average

$\overline{R}$ of all available reputation coefficients previously filed. It is calculated as:

(5) $R_{Y1} = \overline{R} \cdot (1-\beta) + r_j^X \cdot \beta$ with $\beta = \gamma \cdot R_{X\,j-1}$

and/or

(6) $R_{X1} = \overline{R} \cdot (1-\beta) + r_j^Y \cdot \beta$ with $\beta = \gamma \cdot R_{Y\,j-1}$

The results of the usage of the mechanism are shown in the figures below. Each figure shows the development of the "equity" capital a software agent owns, after each transaction., starting with a capital of 200 money units. In Figure 6, a rating agency was used, which centrally stores all reputation information and distributes it to the whole system, while in Figure 7 the software-agents only use their own reputation information. The three curves in the figures show the "equity" capital of two cooperative carpenter agents (box and triangle symbols) with a reputation = 1, and one untrustworthy carpenter agent (plus symbol) with a reputation = 0. In both cases, the agents are competitors, and trade with (and are rated by) lumberjack and cabinetmaker agents, but not with each other.

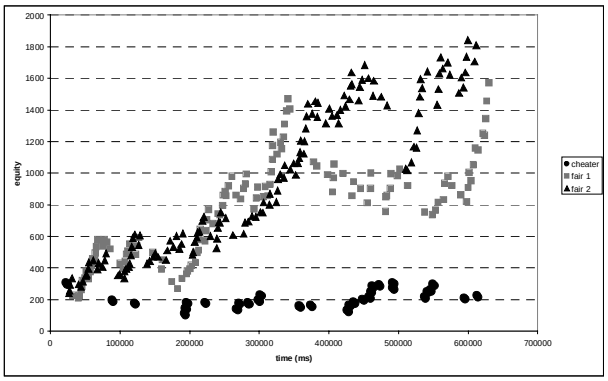

Figure 6. Use of a rating agency

In Figure 6, the uncooperative carpenter agent is not able to generate income. The rating agency soon provides the necessary information to other agents, and the trustful competitors are able to take advantage. This is a simple and expected result of using a central institution.

In Figure 7 a central rating agency is not available. In this case, the untrustworthy agent is able to capitalize in the beginning, at the expense of the fair agents. However, as the market process continues, the cheater agent loses its capital, and the two fair agents are able to recover and to generate income. Apart from the additional time needed to disseminate the reputation information throughout the population, and the added competition pressure because of this, the result is comparable: the fair acting software-agents were also able to increase their income, while the uncooperative agent lost its capital.

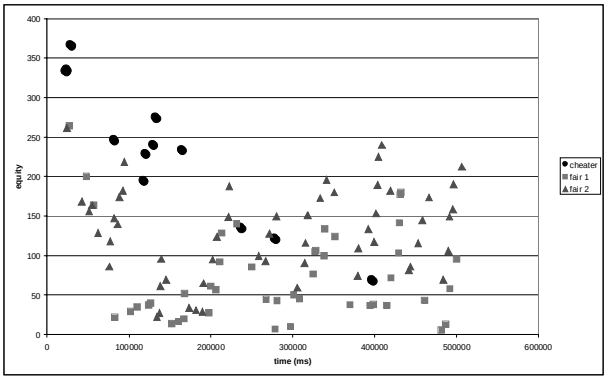

Figure 7. Use of decentralized reputation information

5. Conclusion and Outlook

In this article we have expressed a vision of software agents acting in future electronic marketplaces. These marketplaces will provide open environments, where heterogeneous software agents are free to enter and to leave at any time. In a networked world, no central system designer can enforce whether the agents will use a particular marketplace or auctioneer, and if they cooperate or not. In contrast, human principals all over the network define their own goals and strategies for their agents, and as in the real world, they will be free to behave as good or bad as they like, and act in an autonomous, self-interested and not necessarily cooperative way.

In AVALANCHE, decentralized market coordination is achieved using self-interested, autonomous software agents, and without a central coordination entity, based on market coordination concepts of Neo-Austrian and Evolutionary Economics (for further discussion, see [8]).

Furthermore, the decentralized reputation mechanism of AVALANCHE yields comparable results to a centralized rating agency. When centralized institutions are not available, not trustworthy, or too expensive to use, it is possible to use the distributed knowledge of a market community to expulse fraudulent agents from market coordination. This reputation tracking mechanism evaluates transaction behavior and influences partner selection and negotiation strategy in future transactions, while protecting the privacy of the participants.

With regard to this result, agent-based B2B E-Commerce can be opened to encompass not only static catalog websites and closed auction communities, but also truly dynamic automated negotiation in an open and de-centralized market environment, where a priori unknown agents create new market opportunities in a highly dynamical way.

This development will lead to fast, flexible and very adaptive markets, which have been envisioned (and partly already implemented) first for commodities such as telecommunications bandwidth, electricity, and natural resources (gas and water) [31], where the trade of single units is too expensive for human participants. In likewise applications, the agents in question may coordinate customers and suppliers in supply chain management, organizational units within internal markets, human traders in electronic commerce systems, or distributed materials flow processes in and between companies.

6. References

[1] Bayers, C., "The Bot.Com Future", *WIRED Magazine 8.03*, March 2000, pp. 210-219.

[2] Bradshaw, J.M. (Ed.), *Software Agents*, AAAI Press, Menlo Park, 1997.

[3] Chavez, A., Maes, P., "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*, London, UK, April 1996. http://agents.www.media.mit.edu/groups/agents/ publications/kasbah-paam96.ps.gz

[4] Deutsch, M., *The Resolution of Conflict*, Yale University Press, New Haven and London, 1973.

[5] Eggs, H., and J. Englert, *Electronic Commerce Enquête II 1999/2000 – Vernetzte kleine und mittlere Unternehmen*, Konradin-Verlag, Stuttgart, 1999.

[6] Eriksson, J., N. Finne, and S. Janson (Eds.), *MarketSpace '96 - An Open Agent-Based Market Infrastructure*, UPMAIL Technical Report No. 147, Uppsala, April 7, 1997.

[7] Eymann, T., *AVALANCHE - Einführung und Programmdokumentation*, http://www.iig.uni-freiburg.de/ ~eymann/avalanche/dokumentation.pdf, 1999.

[8] Eymann, T., B. Padovan, and D. Schoder, „The Catallaxy as a new Paradigm for the Design of Information Systems", *Proceedings of The World Computer Congress 2000 of the International Federation for Information Processing (IFIP)*, Beijing, China, August 21-25, 2000.

[9] Flores, L., "Turning the Global Infrastructure into a Universal Information Ecosystem". *AgentLink News 3*, July 1999, pp. 17-18. http://www.agentlink.org/newsletter/3/ newsletter3.pdf

[10] Foner, L., *Political Artifacts and Personal Privacy: The Yenta Multi-Agent Distributed Matchmaking System*, Dissertation, Cambridge, USA, 1999. http://foner.www.media. mit.edu/ people/foner/PhD-Thesis/Dissertation/

[11] Foundation for Intelligent Physical Agents, Agent Communication Technical Committee, "Agent Communication Language", *FIPA '99 Draft Specification*, 1999, http://www.fipa.org.

[12] Guttman, R., A. Moukas, and P. Maes, P.: "Agent Mediated Electronic Commerce: A Survey". *Knowledge Engineering Review*, June 1998. http://ecommerce.media. mit.edu/ papers/ ker98.pdf

[13] Kearney, P.J., R.E. Smith, C. Bonacina, T. Eymann, "Integration of computational models inspired by economics and genetics", *BT Technology Journal*, Vol. 18, No. 4, Kluwer Academic Publishers, October 2000.

[14] Kephart, J.O., J.E. Hanson, D.W. Levine, B.N. Grosof, J. Sairamesh, R.B. Segal, and S.R. White: "Dynamics of an information-filtering economy". In Klusch, M. and G. Weiß (Eds.): *Cooperative Information Agents II*, LNAI No. 1435, Springer, Heidelberg, 1998. http://www.ibm.com/iac/ papers/ cia98/ cia98_public.html

[15] Kephart, J.O., J.E. Hanson, and A.R. Greenwald, "Dynamic Pricing by Software Agents". *Computer Networks*, 2000 (to appear). http://www.research.ibm.com/infoecon/ paps/ html/rudin/rudin.html

[16] Koller, M., *Sozialpsychologie des Vertrauens: Ein Überblick über theoretische Ansätze*. Bielefelder Arbeiten zur Sozialpsychologie, 1990.

[17] Luhmann, N., *Vertrauen. Ein Mechanismus zur Reduktion sozialer Komplexität*, Enke, Stuttgart, 1989.

[18] Marsh, S.: "Trust in Distributed Artificial Intelligence". In Castelfranchi, C. and E. Werner (Eds.), *Artificial Social Societies*, LNAI 830, Springer, Heidelberg, 1994, pp. 94-112.

[19] Moukas, A., G. Zacharia and P. Maes, "Amalthaea and Histos: MultiAgent Systems for WWW Sites and Reputation Recommendations", in Klusch, M. (Ed.), *Intelligent information agents: agent-based information discovery and management on the Internet*. Springer, Heidelberg, 1999.

[20] Müller, G., and K. Rannenberg (Eds.), *Multilateral Security in Communications*, Addison Wesley Longman, München, 1999.

[21] Müller, J., *Diversifikation und Reputation: Transferprozesse und Wettbewerbsbedingungen*, Gabler, Wiesbaden, 1996.

[22] Nissen, M.E., „Supply Chain Process and Agent Design for E-Commerce", Proceedings of the 32st Annual Hawaii International Conference on System Sciences (HICSS'99), Maui, 1999.

[23] ObjectSpace (Eds.), *How Voyager simultaneously supports CORBA, RMI, and DCOM*, White Paper, http://www. objectspace.com/ products/documentation/VgerUnivOrb.pdf, 1999.

[24] Porter, M.E., and V.E. Millar, "How information gives you competitive advantage", *Harvard Business Review*, July-August 1985, pp.149-160.

[25] Preist, C., *Economic Agents for Automated Trading*, HP Technical Reports HPL-98-77. Hewlett Packard Laboratories, Bristol, 1998.

[26] Pruitt, D.G., *Negotiation Behavior*, Academic Press, New York, 1981.

[27] Rasmusson, L., and S. Janson, "Agents, self-interest and electronic markets", *Knowledge Engineering Review*, Vol. 14 (2), 1999, pp. 143-150.

[28] Rosenschein, J.S., and G. Zlotkin, *Rules of Encounter*, MIT Press, Cambridge, 1994

[29] Schmid, B., and M.A. Lindemann, "Elements of a Reference Model for Electronic Markets". *Proceedings of the 31st Annual Hawaii International Conference on System Sciences (HICSS'98)*, Vol. IV, January 1998, pp. 193-201.

[30] Shoham, Y.: "An Overview of Agent-oriented Programming", in Bradshaw, J.M. (Ed.): *Software Agents*. AAAI Press, Menlo Park, 1997, pp.

[31] Sierra, C.: "Agent-mediated Electronic Commerce: Scientific and Technological Roadmap." http://www.iiia.csic.es/AMEC/Roadmap.ps

[32] Wellman, M.P., "Market-Oriented Programming: Some Early Lessons", in Clearwater, S. (Ed.), *Market-Based Control: A Paradigm for Distributed Resource Allocation*, World Scientific, Singapore, 1996.