# The CrocodileAgent: Designing a Robust Trading Agent for Volatile E-market Conditions

Ana Petric, Vedran Podobnik, Gordan Jezic

University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Telecommunications, Unska 3, HR-10000 Zagreb, Croatia
{ana.petric, vedran.podobnik, gordan.jezic}@fer.hr

**Abstract.** The Trading Agent Competition (TAC) simulates a challenging game environment where competing agents engage in complex decision-making activities with purpose of maximizing their profits. One of the TAC competitive scenarios is Supply Chain Management (SCM) where six trading agents compete by buying components, assembling PCs from these components and selling the assembled PCs to customers. In this paper, we briefly describe the TAC SCM environment and present the main features of the CrocodileAgent, our entry in the 2005 TAC SCM Competition. The agent's performances in the 2005 TAC SCM, as well as in a series of controlled experiments, are discussed.

## 1 Introduction

Supply chain management involves several activities such as raw material procurement, producing, selling and shipping manufactured goods. Today's supply chains are still based on static long-term relationships between trading partners while in dynamic supply chains the market is the driving force. Dynamic supply chain management improves the competitiveness of companies since it has a direct impact on their capability of adapting to the changing market demands quickly and efficiently [1]. The annual worldwide supply chain transactions are counted in trillions of dollars what makes this area of research very interesting not just from the scientific, but also from the business point of view, since even the slightest improvement brings a very high profit.

The connection between AI (*Artificial Intelligence*) and economics has received a lot of attention recently [2]. The ideas proposed in this paper are also based on that connection, while the practical implementation of the presented ideas is enabled with the use of intelligent software agent technology and supported by the Internet infrastructure. Although the initial architecture of the Internet was geared towards delivering information visually to humans, currently the Web transforms into an environment filled with the goal-directed applications which intelligibly and adaptively coordinate information exchanges and actions (Web 2.0 and Web 3.0) [3]. At the same time, computers are evolving from single isolated devices to entry points into a worldwide network of information exchange and business transactions [4]. Consequently, the Internet is transforming into an enabler of the digital economy. The digital economy, by proliferation of the use of the Internet, provides a new level and form of connectivity among multiple heterogeneous ideas and actors [5]. Additionally, by utilizing the technology of intelligent software agents, the digital economy automates business transactions.

In the paper we describe the CrocodileAgent, an intelligent agent we developed to participate in 2005 TAC SCM. The paper is organized as follows. Section 2 presents the TAC SCM game. Section 3 describes the CrocodileAgent's architecture and functionalities. Section 4 comments the agent's ranking in the 2005 TAC SCM, elaborates the results of an experiment we conducted in our laboratory and includes a detailed analysis of the results. Section 5 proposes directions for future work and concludes the paper.

## 2 The TAC SCM Game

The Trading Agent Competition (TAC) is an international forum that promotes high-quality research on the trading agent problem. One of its game scenarios is Supply Chain Management (SCM). In the TAC SCM game [6, 7] scenario, each of the six agents included in the game has its own PC manufacturing company. During the 220 TAC days, agents compete in two different markets. In B2B market, agents compete in buying raw materials necessary to produce personal computers. Participants in this market are agents and eight suppliers which produce four types of components (CPUs, motherboards, memory, hard drives) with different performances. In its factory, an agent can manufacture 16 different types of PCs. In the B2C market, the agents try to sell all the PCs they produced to customers and, at the same time, earn as much money as possible. The purpose of the TAC SCM (*Trading Agent Competition Supply Chain Management*) game is to explore how to maximize the profit in the stochastic environment of volatile market conditions. Thus, it is important to develop an agent capable of reacting quickly to changes taking place during the game. Furthermore, it is critical to implement predictive mechanisms which enable an agent's proactive behaviour. The idea is to build a robust, highly-adaptable and easily-configurable mechanism for efficiently dealing with all SCM facets, from material procurement and inventory management to goods production and shipment [8]. Additionally, TAC SCM tournaments provide an opportunity to analyze effects common in real-world business transacting, such as the bullwhip effect, and its relationship with company profits [9]. Furthermore, the tournament can help in developing methods for identifying the current economic regime and forecasting market changes [10].

The architecture of the TAC SCM system is shown in Figure 1. The TAC SCM game server simulates suppliers (PC component manufacturers), customers (PC buyers) and the bank. The game server also controls agents' factories and warehouses. In order to participate in the game, an agent has to connect to the game server. Each TAC SCM agent has a bank account and receives a daily report regarding its current bank balance. At the beginning of the game, the agent has no money and must hence loan money from the bank. The bank charges the agent interest for every day that the agent is in debt. The winner of the game is the agent with the most money in its bank account at the end of the game.

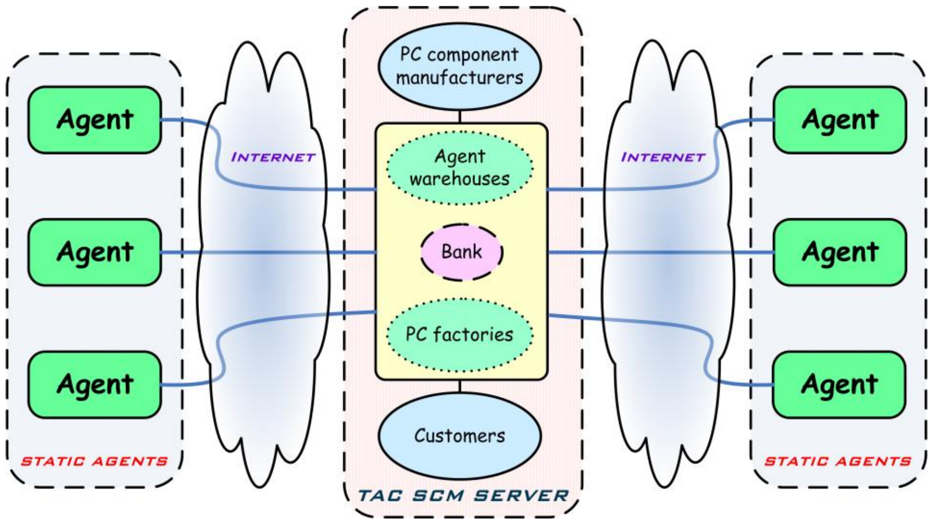

**Fig. 1.** The architecture of the TAC SCM system

## 3 The CrocodileAgent

An intelligent software agent is an autonomous program which acts on behalf of its principal (human or organizational) while conducting complex information and communication actions over the Web. The technology of intelligent software agents enables automated and process execution and coordination, thus creating added value for its principal. Figure 2 presents our model of intelligent software agents [11, 12], which we used while designing our TAC SCM agent.

The features of the technology of intelligent software agents make them perfectly applicable in modern enterprise systems and electronic markets (e-markets) [13]. The accelerated economic globalization trend and rapid development of ICT technologies in the past decade are leading us closer to the existence of just one market - the global one. Consequently, the functions of supply and demand are becoming more and more dynamic and the possibilities of choice have risen to amazing levels. This is a reason why companies today have great difficulties in enhancing the efficiency of their current business processes. Companies are instantly forced to make important decisions while continuously trying to maximize their profits. Keeping in mind the great volatility that characterizes the complex set of market conditions and the vast quantity of available information, a possible solution for improving business efficiency is the automation of business processes and excluding humans from making decisions (where this is possible). Humans simply do not posses the cognitive ability to process such an enormous quantity of information (and to make adequate decisions) in the few moments during which the relevant information does not change. A very logical solution to this problem lies in the technology of intelligent software agents –

i.e. computer programs with the ability to completely autonomously manage a set of tasks.

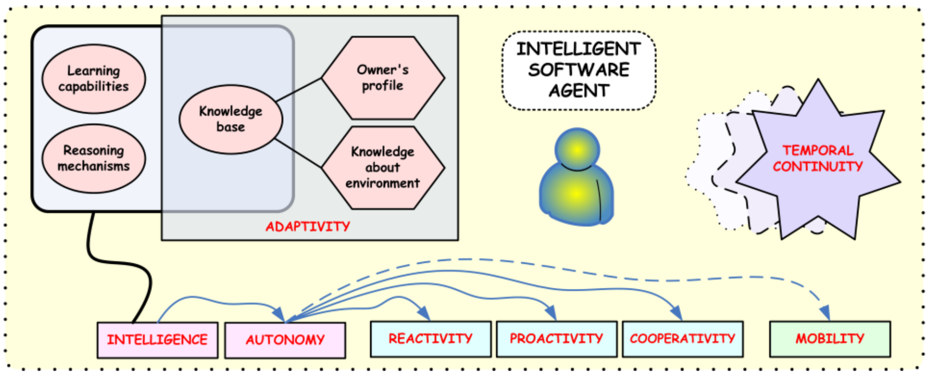

**Fig. 2.** A model of intelligent software agent

The CrocodileAgent [13, 14] is an intelligent agent developed at the Department of Telecommunications, Faculty of Electrical Engineering and Computing in Zagreb, Croatia. Designing the CrocodileAgent was an extension of a project that started in 2004 when the KrokodilAgent [15] was developed to participate in 2004 TAC SCM.

The agent's daily responsibilities can be divided into three logical tasks: negotiating supply contracts, bidding for customers' orders and managing daily assembly activities along with taking care of shipping completed orders to customers. These tasks will be described in the following sections.

## 3.1 Negotiation of supply contracts

In order to sell PCs it is necessary to purchase components and produce PCs from those components. The purchasing protocol is rather simple. The agent sends RFQs (*Requests for Quotes*) to the supplier that produces the needed component. The supplier responds with an offer. If the offer fits the agent's needs the agent replies with an order. A description of the CrocodileAgent's supply tactics follows.

On the day 0 the agent sends RFQs with the following delivery dates: 3, 9, 17, 27 and 69. The reserve prices the agent is willing to pay for the components are 102%, 107%, 92%, 82% and 77% of the nominal price on the respective delivery dates. These parameters were chosen after analyzing the prices and the delivery dates in a series of games. The requested quantities are smaller in short-term RFQs since the reserve prices in short-term RFQs are higher. RFQs are sent to all the suppliers that produce the needed component. The agent accepts the cheapest offer. That causes a temporary droop of the agent's reputation in the eyes of the other supplier. However, since the requested quantities are not high, the agent's reputation fully recovers in 20-30 days. The agent accepts partial offers if the chosen supplier can not deliver the requested quantity on time. In that case the quantities and reserve prices are modified for more aggressive purchasing of that component in the near future.

During the game the CrocodileAgent grounds its procurement strategy on short-term purchasing of smaller quantities of PC components. This strategy prevents the agent from paying large amounts of money to stock PC components in the warehouse.

Minimal required and maximal allowed quantities of components in storage are high during the game. As the game comes to an end the quantities are lowered to reduce potential loss of money. The goal is to sell out all the components that are still in inventory at the end of the game since they are paid upon delivery.

At the beginning of each day, the agent calculates the component quantity ordered, but not delivered, up to that moment for each component separately. The CrocodileAgent's ordered quantities of components are multiplied with a distance factor. This factor shrinks as the delivery date grows. The distance factor was introduced to lower the risk of running out of components since the supplier can cancel the order or deliver the ordered components later than arranged. New components are ordered if the current quantity of components stored in the warehouse increased with the estimated ordered quantity is lower than the maximal allowed quantities of components for that day. In spite of the maximal allowed quantities, in the first part of the game the agent makes long-term orders with small amounts of components to ensure cheap components for the second part of the game.

There are special mechanisms which calculate the reserve prices and exact quantities that need to be ordered. In case there is a very low quantity of a certain component in the warehouse a particular mechanism is activated. It allows short-term procurement of this component where the agent pays a higher price than usual. There is a similar mechanism if the customer demand rises rapidly. In this case the agent uses more components to produce more PCs, so the mechanism is activated to ensure that the agent does not run out of components and as result loses potentially profitable PC orders.

Special attention was paid to the end of the game. The intention was to enable the agent to send offers to customers for as long as possible, but also to maintain a low level of all components in the warehouse. This way, the following scenario was prevented; a large quantity of one component could be left over, not because there was no customer demand, but because the agent had spent all the other components needed to produce a certain type of PC.

## 3.2 Bidding for customer orders

The negotiation protocol between the customers and the agents is the same as the negotiation protocol between the agents and the suppliers. Each day customers send RFQs to the agents. The agents reply by sending offers. The agent that offered the lowest bid price wins the customer order.

### 3.2.1. An algorithm for sending offers

The CrocodileAgent grades each RFQ and within those grades RFQs are sorted in chronological order of their delivery dates. The grade is determined by the difference between the customer's reserve price and the agent's cost of producing that PC. After grading and sorting RFQs the agent starts to send offers if the agent's PC production cost is lower than the customer's reserve PC price and there are enough components to produce the requested PCs. In case the latter condition is not fulfilled, the agent checks if the requested PCs can be delivered from the reserve PCs stored in the warehouse.

This algorithm comes in three versions. The version that is active on a certain day is determined depending on the stage of the game (beginning, middle, end), the number of production cycles needed to produce all the active orders and the algorithm that was used the day before. The basic difference between these three versions is the method of determining the offer prices for the PCs. The most frequently used version during the game is the *Normal version* and it determines the offer price in two ways:

- If the agent offers PCs based on the components currently present in the warehouse, the offer price is calculated as the basic PC price increased by the agent's desired profit;
- If the agent offers PCs that are already produced and stored in the warehouse, the offer price is slightly under the customer's reserve price.

The *High Demand Version* uses a "greedy" algorithm since the offer prices for the PCs are always just slightly under the customer's reserve price. It is used when there is a very high customer demand for PCs since, in those cases, agents usually do not send offers for all RFQs received. The *End Game Version* is used for the finishing stage of the game. What makes this version of the algorithm different from the other two versions is the fact that it sorts RFQs in increasing order of their corresponding penalties. The main purpose of this version is to sell out the whole inventory in the warehouse so the profit that the agent adds to the basic PC price is minimal. All the versions of the selling algorithm implement a mechanism used for preventing late deliveries. Each day, the agent monitors its obligations to customers by calculating the number of factory cycles needed to fulfill its existing orders. On the basis of that information it determines the earliest delivery date for sending new PC offers. This way the agent is prevented from sending offers that cannot be delivered by the requested delivery date.

### 3.2.2. Calculating prices of components in the warehouse and the profit margin

The basic PC price is calculated by summing the average prices of each component incorporated in the PC. The agent always knows the price paid for each component that is in its warehouse. If the current supply of components is higher than the calculated optimal for that day a discount for the components is approved. The agent also gives a discount on components at the end of the game to sell out the components that are still in the warehouse.

The profit margin is calculated every day and the calculating algorithm analyzes the following parameters:

- Percentage of the agent's recent offers which resulted in customer orders
  - If this parameter is decreasing, the profit margin also decreases and vice-versa;
- Distribution of agent's factory occupancy in the next few days
  - High factory utilization causes the increase of profit margin and vice-versa;
- Recent PC prices of other agents compared to CrocodileAgent's.
  - If CrocodileAgent offers cheap PCs the profit margin increases and vice-versa;
- Due date listed in the customer RFQ

  - Earlier due date causes the higher profit margin and vice-versa;
- Demand level in the market segment the wanted PC belongs to
  - If the demand is low then the profit margin is decreased and vice-versa.

### 3.3 Managing factory activities and shipping completed orders to customers

In prior versions the CrocodileAgent produced PCs only after receiving customer orders, but we decided to add the possibility of producing PCs even if nobody ordered them. Since the TAC SCM game is of stochastic nature, customer demand varies during the game. If the agent does not produce PCs and the PC demand is low, a large part of the agent's factory capacities stay unutilized. If the agent produces PC stocks during a period of low PC demand, its factory will be utilized and everything will be prepared for a period of high PC demand. This tactic has it weaknesses since the agent cannot know for sure what the future demand is going to be. Thus, it can produce more PCs than can be sold by the end of the game. We tried to lower this risk by introducing quantity limits which represent the maximum number of PCs which can be available in stock. As the end of the game approaches, these limits are lowered accordingly.

Each day the list of active orders is sorted in chronological order of the delivery dates. The agent first checks if there are enough PCs in the warehouse to fulfill the order. If there are enough PCs, they are added to the delivery schedule. If there are not enough PCs, the agent checks if there are enough components to produce the requested PCs. If so, the components are reserved and the agent tries to add them to the production schedule. When finished with analyzing all the active orders, the CrocodileAgent checks the production schedule for the next day. After determining the amount of free capacity available, the agent checks which PC types can be produced without creating a large stock.

## 4 TAC SCM 2005 competition and controlled experiments

The 2005 TAC SCM competition was divided into three parts: qualifying rounds[1] held from June 13th-24th, seeding rounds[2] held from July 11th -22nd and final round held from August 1st-3rd. There were 32 teams competing in the qualifying and 25 teams in the seeding rounds. 24 teams competed in the finals. The CrocodileAgent took 4th place in the quarterfinals[3] with an average score of 11.64M and ended its participation in 2005 TAC SCM.

To evaluate the performance of our agent, we held a competition with some of the best agents from the TAC SCM 2005 competition. The chosen opponents were: TacTex [16], Mertacor [8], DeepMaize [17], MinneTAC and PhantAgent. All the agents were downloaded from the official TAC Web page (http://www.sics.se/tac).

[1] http://www.sics.se/tac/page.php?id=47
[2] http://www.sics.se/tac/page.php?id=48
[3] http://www.sics.se/tac/page.php?id=49

The competition was held in our laboratory and consisted of 20 games. The final ranking is shown in Table 1. After the competition finished, we conducted a detailed analysis of the games played. We focused on the supply procurement mechanism since the biggest change in the game rules for the TAC SCM 2005 concerned the suppliers and their price determining algorithm.

**Table 1.** Competition results at server `mobility3.labs.tel.fer.hr`.

| Place | Agent | Score | Games Played | Place | Agent | Score | Games Played |
|---|---|---|---|---|---|---|---|
| 1 | TacTex | 5 638 295 | 20 | 4 | PhantAgent | 976 780 | 20 |
| 2 | DeepMaize | 4 581 805 | 20 | 5 | MinneTAC | -426 722 | 20 |
| 3 | Mertacor | 1 364 353 | 20 | 6 | CrocodileAgent | -1 243 212 | 20 |

The majority of the analysis was done with the CMieux Analysis and Instrumentation Toolkit for TAC SCM [18]. The main task was to analyze component purchases. After gathering information regarding the prices the agents paid for each type of component and the quantities they purchased, we calculated the average prices. These prices are shown in Figure 3. Components marked with ID 1xx are CPUs, 2xx are motherboards, 3xx are memory and 4xx are hard disks.

Since the price of a CPU accounts for more than 50 % of the PC price, it is very important to purchase cheap CPUs. It can be noticed from Figure 3 that TacTex bought some of the cheapest CPUs while DeepMaize paid the highest prices for CPUs. The CrocodileAgent bought the third cheapest CPUs. The situation is slightly different with other components: Mertacor bought the cheapest motherboards and memory, while the CrocodileAgent bought the cheapest hard disks. MinneTAC bought the most expensive motherboards, memory and hard disks.

Furthermore, we analyzed the quantity of components bought during the game. From Figure 4, it can be noticed that TacTex, DeepMaize and the CrocodileAgent bought larger amounts then the other agents. The average minimal number of components that stayed in the agents' warehouses after the game has finished can be determined from the same figure, because the maximum number of PCs that can be assembled is equal to the minimal amount of a certain component type. The leftover components represent a direct loss of money since they were paid for upon delivery. The best results regarding leftover component management were obtained by the PhantAgent and DeepMaize. The majority of their leftover components were memory components. This is desirable since memory is the cheapest component type.

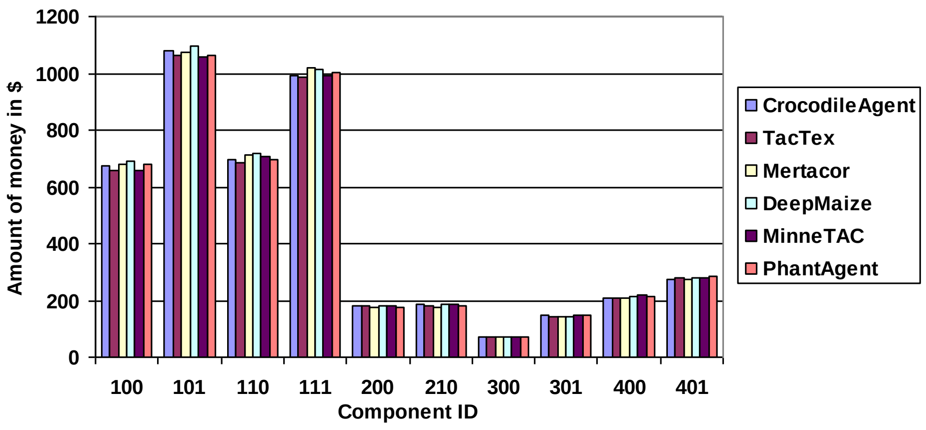

**Fig. 3.** Average component prices

The Mertacor agent also had a relatively small number of leftover components, but they were mostly CPUs. This is not as good since CPUs are at least ten times more expensive than memory components. All three agents had between 500 and 800 leftover components. TacTex and MinneTAC had more than 2600 leftover components on average. The difference is that half of TacTex's leftover components were CPUs, while half of MinneTAC's leftover components were memory components. The CrocodileAgent was in the middle with an average of 1250 leftover components. This is a direct consequence of the ordering algorithm that does not compare the total number of ordered components of each component type.

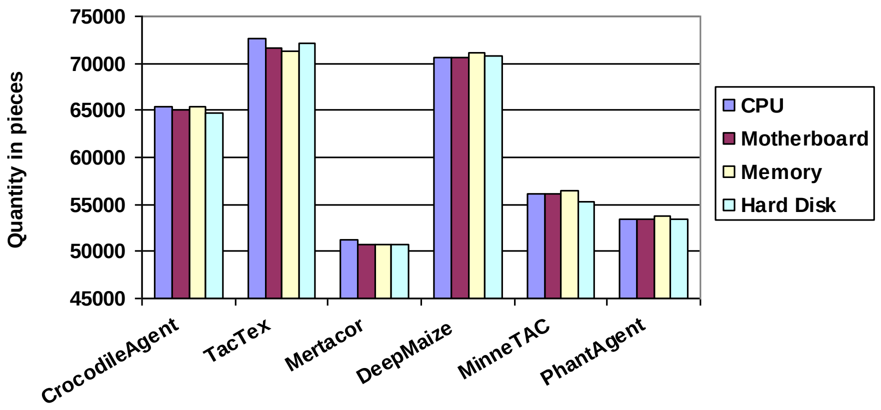

**Fig. 4.** Total quantity of bought components

## 5 Conclusions and Future Work

In this paper, we presented the CrocodileAgent which is a trading agent that participated in TAC SCM 2005. After a short game description, we listed the basic TAC SCM Agent tasks and explained how they were implemented in the CrocodileAgent.

We briefly presented the results of the CrocodileAgent in the TAC SCM 2005. In order to improve the functionalities of the agent, we held a competition with some of the best agents in TAC SCM 2005. We figured that this was a good way to determine the CrocodileAgent's soft spots. A thorough analysis of the competition was conducted. The results were a little discouraging since the CrocodileAgent placed last, but a lot was learned. The main reason for the CrocodileAgent's results lies in its reactive algorithm for selling PCs. This algorithm does not predict the fluctuation of prices on the PC market, but it only reacts to the current state of the PC market. Thus, during further development of our agent special attention needs to be dedicated to the PC selling algorithm with an emphasis on customer demand prediction and the prediction of winning PC prices. The component purchase algorithm and mechanisms for managing factory activities function quite well, but there is always room for improvement.

## References

1. Benish, M., Sardinha, A., Andrews, J., Sadeh, N.: CMieux: Adaptive Strategies for Competitive Supply Chain Trading. In Proceedings of the 8th Int. Conference on Electronic Commerce (ICEC), Fredericton, Canada, 2006. 1-10
2. Wurman, P.R., Wellman, M.P., Walsch, W.E.: Specifying Rules for Electronic Auctions. AI Magazine, Vol. 23 (3). American Association for Artificial Intelligence, Menlo Park (2002). 15-24
3. Podobnik, V., Trzec, K., and Jezic, G.: An Auction-Based Semantic Service Discovery Model for E-Commerce Applications. Lecture Notes in Computer Science, Vol. 4277. Springer-Verlag, Berlin Heidelberg New York (2006). 97-106
4. Fensel, D.: Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag, Berlin Heidelberg New York (2004)
5. Carlson, B.: The Digital Economy: What is New and What is Not?. Structural Change and Economic Dynamics, Vol. 15 (3). Elsevier, North-Holland (2004). 245-264
6. Collins, J., Arunachalam, R., Sadeh, N., Eriksson, J., Finne, N., Janson, S.: The Supply Chain Management Game for the 2005 Trading Agent Competition. http://www.sics.se/tac/tac05scmspec_v157.pdf. Date accessed: Nov 12, 2006.
7. Eriksson, J., Finne, N., Janson, S.: Evolution of a Supply Chain Management Game for the Trading Agent Competition. AI Communications, Vol. 19 (1). IOS Press, Amsterdam (2006). 1-12
8. Kontogounis, I., Chatzidimitriou, K.C., Symeonidis, A.L., Mitkas, P.A.: A Robust Agent Design for Dynamic SCM Environments. In Proceedings of the 4th Hellenic Joint Conference on Artificial Intelligence (SETN), Heraklion, Greece, 2006. 127-136
9. Jordan, P.R., Kiekintveld, C., Miller, J., Wellman, M.P.: Market Efficiency, Sales Competition, and the Bullwhip Effect in the TAC SCM Tournaments. In Proceedings of the

AAMAS Joint International Workshop on the Trading Agent Design and Analysis and Agent Mediated Electronic Commerce (TADA/AMEC) Hakodate, Japan, 2006. 99-111

10. Ketter, W., Collins, J., Gini, M., Gupta, A., Shrater, P.: Identifying and Forecasting Economic Regimes in TAC SCM. In Proceedings of the IJCAI Workshop on Trading Agent Design and Analysis (TADA), Edinburgh, UK, 2005. 53-60
11. Bradshaw, J.M.: Software Agents. MIT Press, Cambridge, Massachusetts, USA (1997)
12. Chorafas, D.N.: Agent Technology Handbook. McGraw-Hill, New York, USA (1998)
13. Podobnik, V., Petric, A., Jezic, G.: The CrocodileAgent: Research for Efficient Agent-Based Cross-Enterprise Processes. Lecture Notes in Computer Science, Vol. 4277. Springer-Verlag, Berlin Heidelberg New York (2006). 752-762
14. Petric, A., Podobnik, V., Jezic, G.: The CrocodileAgent: Analysis and Comparison with Other TAC SCM 2005 Agents. In Proceedings of the AAMAS Joint International Workshop on the Trading Agent Design and Analysis and Agent Mediated Electronic Commerce (TADA/AMEC) Hakodate, Japan, 2006. 202-205
15. Petric, A., Jurasovic, K.: KrokodilAgent: A Supply Chain Management Agent. In Proceedings of the 8th International Conference on Telecommunications (ConTEL), Zagreb, Croatia, 2005. 297-302
16. Pardoe, D., Stone, P.: Bidding for Customer Orders in TAC SCM: A Learning Approach. In Proceedings of the AAMAS International Workshop on Trading Agent Design and Analysis (TADA), New York, USA, 2004.
17. Wellman, M.P., Estelle, J., Singh, S., Vorobeychik, Y, Kiekintveld, C., Soni, V.: Strategic Interactions in a Supply Chain Game. Computational Intelligence, Vol. 21 (1). Blackwell Publishing, Oxford, UK (2005). 1-26
18. Benisch, M., et. al.: CMieux Analysis and Instrumentation Toolkit for TAC SCM. Carnegie Mellon University School of Computer Science TR CMU-ISRI-05-127 (2005)