The CrocodileAgent: A Software Agent for SCM Procurement Gaming

Tvrtko Milicic, Vedran Podobnik, Ana Petric, Gordan Jezic

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

Abstract. The Trading Agent Competition (TAC) is an international forum which promotes high-quality research regarding the trading agent problem. One of the TAC competitive scenarios is Supply Chain Management Procurement Challenge (SCM-PC) where three agents compete by buying components, assembling PCs from these components and selling the manufactured PCs to customers. The idea of SCM-PC is the development of agent-based systems that implement efficient business strategies, with special emphasis on the procurement processes. In this paper, we analyze the SCM-PC environment and describe the main features of the CrocodileAgent, our entry in the SCM-PC Competition.

1 Introduction

In the past, both the markets and choices available were much smaller than today, so the volatility of supply and demand functions was much more inert. Under such market conditions, companies did not need to make important decisions daily but they rather based business transactions on long-term partnerships. On the other hand, the accelerated economic globalization trend in the past decade is 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. Under such market conditions, companies should also implement procurement strategies based on one-off settlements (spot market trading). The SCM-PC (*Supply Chain Management Procurement Challenge*) Game creates such environment which requests from trading agents to efficiently combine procurement based on both long-term contracts and spot market settlements.

Supply chain management (SCM) deals with planning and coordinating activities such as material procurement, product assembly, and the distribution of manufactured products. Dynamic supply chain management improves the competitiveness of companies since it has a direct impact on their capability of adjusting to the changing market demands quickly and efficiently [1]. Since the annual worldwide supply chain transactions are counted in trillions of dollars, even the slightest possibility of improvement cannot be neglected.

In this paper, we analyze the SCM-PC environment and describe the main features of the CrocodileAgent, an intelligent agent we developed to participate in the SCM-PC Competition. The paper is organized as follows. Section 2 describes the basic rules of the SCM-PC Game. Section 3 describes the CrocodileAgent's architecture

Milicic, Tvrtko; Podobnik, Vedran; Petric, Ana; Jezic, Gordan.
The CrocodileAgent: A Software Agent for SCM Procurement Gaming.
Lecture Notes in Computer Science. 5027 (2008); 865-875.

and functionalities. In Section 4 CrocodileAgent's performances in the SCM-PC 2007 Competition are analyzed. Section 5 proposes directions for future work and concludes the paper.

2 The TAC SCM-PC Game

The Trading Agent Competition (TAC, http://www.sics.se/tac) is an international forum that promotes high-quality research on the trading agent problem. In the year 2007 TAC had four competitive scenarios.

The oldest of them, Supply Chain Management Game (TAC SCM), has been running as an annual tournament since 2003. It revolves around a PC (*Personal Computer*) assembly game, where trading agents from different teams compete for both customer orders for PCs and for PC components required to assemble these PCs. In addition to this baseline game, the 2007 tournament featured two new challenges: a *Procurement Challenge* and a *Prediction Challenge*. While the Procurement Challenge scenario requires trading agents to balance both long-term and one-off procurement contracts as they try to maximize profits and meet stochastic customer demand, in the Prediction Challenge scenario agents should make daily predictions over the course of a number of TAC SCM Games about four different types of prices: current and future computer prices, and current and future component prices. All three TAC SCM Games were motivated by the desire to develop automated strategies for buyer and seller trading agent in electronic markets (e-markets). The trading rules are fixed by the TAC SCM organizers, and competition entrants compete with one another by creating agents that seek to trade under these fixed rules. The fourth scenario, which was also introduced in the competition for the first time in 2007, is called the TAC Market Design scenario, or shortly CAT. In contrast to the TAC SCM based scenarios, the CAT environment is populated with the software trading agents created by the organizers of the competition, and as an entrant one must compete by defining rules for matching buyers and sellers and setting commission fees for providing this service. Entrants compete against each other in attracting buyers and sellers and making profits. This is achieved by having effective matching rules and setting appropriate fees that are a good trade-off between making profit and attracting traders.

In the SCM Procurement Challenge [2] scenario (see Figure 1), each of three manufacturer agents included in the game has its own PC manufacturing company. During the 100 SCM-PC days (one virtual day lasts 10 seconds), manufacturer agents compete for supply contracts from 10 different supplier agents. Each supplier produces one of four types of components (i.e., *CPUs*, *motherboards*, *memories* and *hard drives*). Long-term contracts are negotiated just once at the beginning of the game, consequently making agreed terms for long-term procurement valid till the end of the game. Each week, manufacturer agents may decide to make a purchase based on long-term contract. The quantities in such purchase can vary between the minimum quantities they committed to up to pre-specified maximum quantities. Each day, manufacturer agents may also decide to procure additional components outside of their long-term procurement contracts (this daily-based component purchase based on one-off settlements is called spot market procurement). All the manufacturer

agents' factories are characterized by the same limited capacity, while customers' demands are randomly generated and require each manufacturer agent to satisfy an equal part. This allows agents to ignore the customer bidding dimension of their supply chain, making the SCM-PC simpler than the baseline TAC SCM scenario: in the SCM-PC, manufacturer agents are only expected to focus on making procurement decisions. At the end of the game, the manufacturer agent with the most money in the bank is declared the winner.

In order to participate in the game, an agent has to connect to the game server. The SCM-PC Game server has several functionalities. Namely, it simulates suppliers (i.e., PC component manufacturers), customers (i.e., PC buyers) and the bank. The game server also controls manufacturer agents' factories and warehouses. Each SCM-PC 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. For every day that the agent is in debt, the bank charges the agent interest while for every day that its bank account is positive, the bank pays interest to the agent.

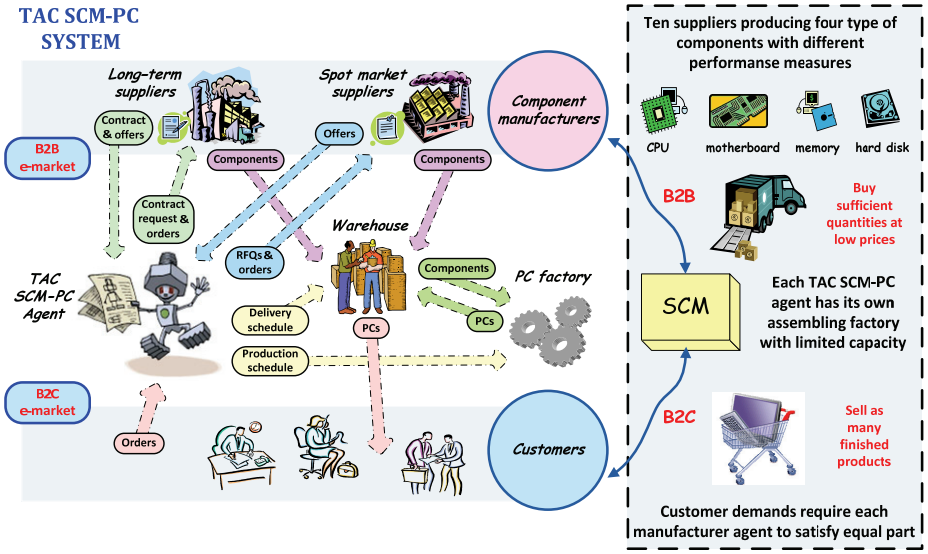

Fig. 1. The relationships in the TAC SCM-PC system

3 CrocodileAgent

The CrocodileAgent is an intelligent trading agent developed at the Department of Telecommunications, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. The CrocodileAgent [3, 4, 5] is long-standing participant in the baseline TAC SCM Competitions [6, 7, 8]. In the 2004 Competition the CrocodileAgent participated in TAC SCM Semi-finals, while in the 2005 and 2006 Competitions our TAC SCM agent took part in the quarter-finals. The CrocodileAgent ended its participation in 2007 Competition as the Second Finals winner, achieving its best result ever.

This section is going to present the CrocodileAgent's version for SCM-PC 2007. The CrocodileAgent's architecture (see Figure 2) is based on incorporating generic intelligent software agent model [9] into the IKB (*Information-Knowledge-Behaviour*) framework [10], a three layered agent-based framework for designing strategies in electronic trading markets.

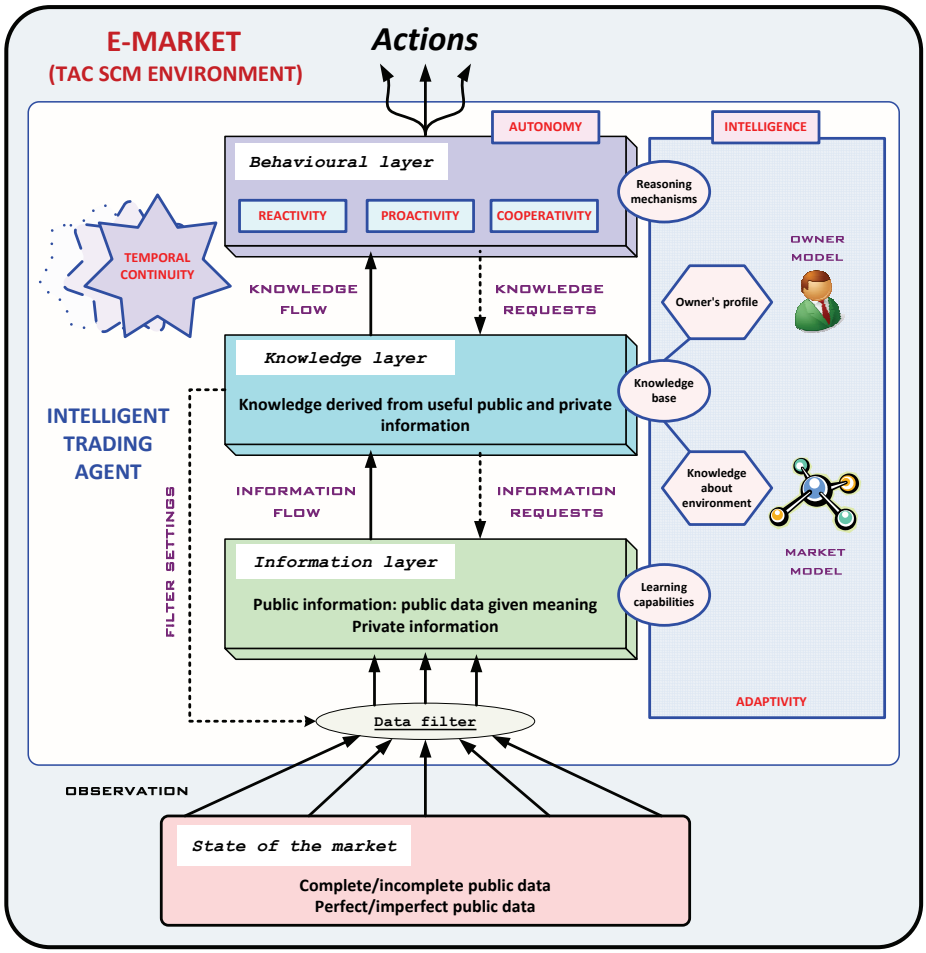

Fig. 2. The CrocodileAgent's architecture

3.1 Negotiating long-term contracts

The most important parameters used in component purchase based on long-term contracts are:

- Q_{max} – the requested maximum weekly component quantity,
- Q_{max}^{real} – the contracted maximum weekly component quantity,

- Q_{min}^{real} – the contracted minimum weekly component quantity: $Q_{min}^{real} \in \lfloor 0.77 \times Q_{max}^{real}, 0.95 \times Q_{max}^{real} \rfloor$ (the level of long-term contract flexibility is randomly determined for each component independently and announced at the beginning of the game),
- p_{nom} – the nominal component prices,
- p_{min} – the minimum price (i.e., the reserve price) for each of the components: $p_{min} \in \lfloor 0.5 \times p_{nom}, 0.75 \times p_{nom} \rfloor$ (the reserve prices are randomly determined for each component independently and announced at the beginning of the game),
- p_{exec} – the maximal execution price (i.e., the maximal price that the trading agent is willing to pay for each component unit it purchases from the supplier),
- p_{exec}^{real} – the actual price every trading agent must pay per component unit purchased from supplier,
- q_{week} – the actually ordered weekly component quantity,
- N_{inv} – the number of components currently stored in the agent's warehouse.

On the game start (*day -1*) the CrocodileAgent negotiates quantity flexible long-term contracts for each component. After each long-term supplier announces p_{min} for each of the components it sells, the CrocodileAgent calculates Q_{max} for submitting a bid to the corresponding long-term supplier. The Q_{max} is determined only according to the p_{min} values: actual quantities linearly decrease from 270 (if $p_{min}=0.5\times p_{nom}$) to 190 (if $p_{min}=0.75\times p_{nom}$) for CPUs and from 370 (if $p_{min}=0.5\times p_{nom}$) to 250 (if $p_{min}=0.75\times p_{nom}$) for other components. After the Q_{max} quantities have been determined independently for all the components, CrocodileAgent submits a single quantity flexible bid for each of the components. In addition to the Q_{max} an agent must also specify the p_{exec}. The CrocodileAgent always sets $p_{exec}=p_{min}$.

Table 1. The relationship between q_{week} and N_{inv}

q_{week} / N_{inv}	CPUs	motherboards	memories	hard disks
$\max(0.77\times Q_{max}^{real},\ Q_{min}^{real})$	≥ 600	≥ 700	≥ 800	≥ 600
Q_{max}^{real}	≤ 300	≤ 400	≤ 500	≤ 400

On the following day (i.e., *day 0*) the CrocodileAgent receives the long-term contracts from the suppliers. Each of the suppliers allocated its resources based on its weekly nominal capacity: quantities were distributed among agents (Q_{max}^{real}) based on requested Q_{max}, starting with the highest bidder. The p_{exec}^{real} is determined according to the rules of the second-price sealed-bid auction. On the *day 0* trading agents also send its first component orders to the long-term suppliers: the CrocodileAgent submits all of its orders with $q_{week}=Q_{max}^{real}$. From this point till the end of the game the CrocodileAgent continues to send weekly orders to long-term suppliers. As the long-term contracts are quantity flexible the CrocodileAgent can every week for every

component choose a different order quantity $q_{week} \in \left\lfloor Q_{min}^{real}, Q_{max}^{real} \right\rfloor$. The CrocodileAgent calculates the q_{week} independently for every component every week, according to the value of N_{inv} parameter. The q_{week} linearly increases from $\max(0.77 \times Q_{max}^{real}, Q_{min}^{real})$ to Q_{max}^{real}, where corresponding marginal N_{inv} values are given in Table 1. The referred N_{inv} values were determined after conducting a series of experiments.

3.2 Negotiating one-off contracts

There are two different aspects of spot market procurement: *day 0* component procurement and ordering components during the game. A close examination of the baseline TAC SCM Game rules [11] (which also define the SCM-PC spot market model) suggests that procurement of components at the very beginning of the game (*day 0* procurement) may provide an agent cheap components throughout the game (because there was no prior component demand). Although the concept of *day 0* procurement strategy has some similarities with long-term component contracting, these two procurement strategies are totally independent because each component is available from two different suppliers: one that only offers long-term contracts and one that only sells in the spot market.

***Day 0* procurement.** The most important parameters used in *day 0* component procurement are:

- $d_{del}[5]$ – requested delivery dates,
- $p_{min}[5]$ – the minimum prices (i.e., the reserve prices),
- $q_{CPU}[5]$ – requested CPU quantities,
- $q_{oth}[5]$ – requested quantities of other components than CPUs,
- p_{nom} – the nominal component prices.

Table 2. The actual parameter values in *day 0* RFQs

d_{del}	7	14	21	52	77
p_{min}	$1.07 \times p_{nom}$	$0.97 \times p_{nom}$	$0.92 \times p_{nom}$	$0.77 \times p_{nom}$	$0.69 \times p_{nom}$
q_{CPU}	300	350	400	450	450
q_{oth}	600	700	800	900	900

The goal of using the *day 0* procurement strategy is to acquire components for the beginning of the game and (if possible) provide the agent with cheap components in the second part of the game. The CrocodileAgent sends the day-maximum of five RFQs with the parameters set as shown in Table 2. The referred parameters were determined after conducting a series of experiments. Unfavourable situation happens when the chosen supplier cannot deliver the requested quantity on time. In that case the agent accepts partial offers.

Component purchase during the game. The most important parameters used in component purchase during the game are:

- N_{min} – the minimal quantity of components required to be in storage (i.e., 400 for CPUs and 700 for other components),
- N_{max} – the maximal quantity of components allowed in storage (i.e., 500 for CPUs and 1000 for other components),
- N_{ord} – the maximal amount of components that can be ordered each day (i.e., 150 for CPUs and 300 for other components),
- N_{tdy} – the quantity of a certain component used in PC production per day,
- N_{inv} – the number of components currently stored in the warehouse,
- p_{min} – the minimum prices (i.e., the reserve prices),
- p_{nom} – the nominal component prices.

It is important to point out that values of the parameters N_{min}, N_{max} and N_{ord} are not fixed values throughout the game. Actually, they are multiplied with *dayFactor*, which firstly increases linearly from 0.3 to 1.5 between days 0 and 60, while afterwards decreases linearly from 1.5 to 0.5 between days 61 and 95.

At the beginning of each day, the agent calculates the component quantity ordered, but not delivered, up to that moment for each component separately. Since the orders with an earlier delivery date will provide components earlier, the agent's ordered quantities of components are multiplied with a distance factor. The distance factor is a value between 0 and 1; the factor shrinks from 1 to 0 as the delivery date grows. When the delivery date reaches 30 days (from the current day) the distance factor becomes 0. The parameter obtained by performing this calculation is referred to as the *evaluatedQuantity*. Similarly, the *evaluatedLongTermQuantity* is calculated, which represents the quantity of all the ordered components that have a delivery date higher than 30 days.

For each component, the agent checks to see if the following condition is fulfilled:

$$N_{inv} + evaluatedQuantity \geq N_{max} . \quad (1)$$

The components are not ordered if condition (1) is fulfilled. However, in spite of condition (1), there are two situations where CrocodileAgent may send some RFQs to spot market suppliers. The first situation is the consequence of fact that, due to volatility of supplier capacity through the game, the prices offered in response to RFQs requesting near-immediate delivery are very unpredictable. To allow for the possibility of making low priced procurement (i.e., $p_{min} \approx 0.6 \times p_{nom}$) the CrocodileAgent sends, if the current date is before day 80 and N_{inv} is less than $1.1 \times N_{max}$, one RFQ with due in 2 days (the minimum possible) for small quantities (40 for all types of components). The *2-day* RFQs enable the agent to be opportunistic in taking advantage of short-term bargains on components without being dependent on the availability of such bargains [12]. The second reason for ignoring fulfilment of condition (1) covers the situation when the CrocodileAgent has not sent any RFQ for certain component during longer period of time (i.e., last 5 days). If the current date is before day 50 and the *evaluatedLongTermQuantity* is lower than its upper limit (i.e, $1.77 \times N_{max}$), the agent sends one RFQ with further delivery date (i.e., around 20 days) to ensure cheap components (i.e., $p_{min} \approx 0.6 \times p_{nom}$ and quantities are set to the N_{ord}) for the second part of the game.

If condition (1) is not fulfilled, the following condition is considered:

$$N_{inv} + N_{tdy} > N_{min}. \quad (2)$$

If condition (2) is not fulfilled the agent purchases components more aggressively with the purpose of getting the number of components in the warehouse above N_{min} as soon as possible (i.e., the agent sends five RFQs requesting near-immediate delivery and relaxes the p_{min} towards higher values). Otherwise, the CrocodileAgent also sends five RFQs, but with the purpose of maintaining the present quantity of components in the warehouse.

It is important to point out that these are only the main characteristics of the algorithm. Additionally, there are special mechanisms which calculate the p_{min} and exact quantities that need to be ordered. A simplified description of some of these mechanisms follows:

- The *lowComponentAlarm* contains several levels and marks the very low quantity of a certain component in the warehouse. In case the alarm is set, the agent is allowed to pay a higher price than usual for the component.
- The *demandPurchaseQuantityFactor* is modified according to customer demand. Sometimes during the game, the customer demand may rise rapidly. When this happens the agent uses more components to produce more PCs, so the parameter is increased to ensure that the agent does not run out of components and consequently looses potentially profitable PC orders.

4 TAC SCM-PC 2007 competition

The SCM-PC competition was held for the first time in the year 2007 [13]. It was divided into two parts: qualifying rounds held from June 26th – 30th and final rounds held on July 23rd.

There were 5 teams competing in the qualifying rounds: **PhantAgent** from "Politehnica" University of Bucharest (Romania), **CMieux** from Carnegie Mellon University (USA), **kshitij** from Centre for Data Engineering, IIIT, Hyderabad (India), **Warrior** from Joint Wayne State University – University of Michigan team (USA) and the **CrocodileAgent**. Because of a small number of participants that were competing in the qualifying rounds, all of them went through to the final rounds. Kshitij team folded, so there were 4 teams competing in the finals. After 9 games played in the final rounds, the average CrocodileAgent's score was 6.399 M which was enough to place at the 3rd place. The rankings of the final rounds are shown in Table 3. Figure 3 shows the progress of average scores for all participants during the final rounds. We can see from this figure how average scores of other agents do not vary significantly during the competition. On the other hand, CrocodileAgent's average score continuously rises from game to game. The agent CMieux, who had the best average score in 8 games, lost the 1st place by not participating in only one game. In the same game, the PhantAgent won with the final score over 20 M and improved his final average score by 2 M which was enough to win in the competition. While the CrocodileAgent began the competition with a negative score in the first two games, afterwards it started to significantly improve its results. It won 2 games in the competition. By taking the 2nd place in the game that cost the CMieux its victory, with the score over 17 M, it came very close to taking the 2nd place in the final order.

Table 3. SCM-PC 2007 final rounds results

Position	Agent	Score	Games Played	Zero Games
1	PhantAgent	8 731 000	9	0
2	CMieux	7 405 000	9	0
3	CrocodileAgent	6 399 000	9	0
4	Warrior	4 200 000	9	0

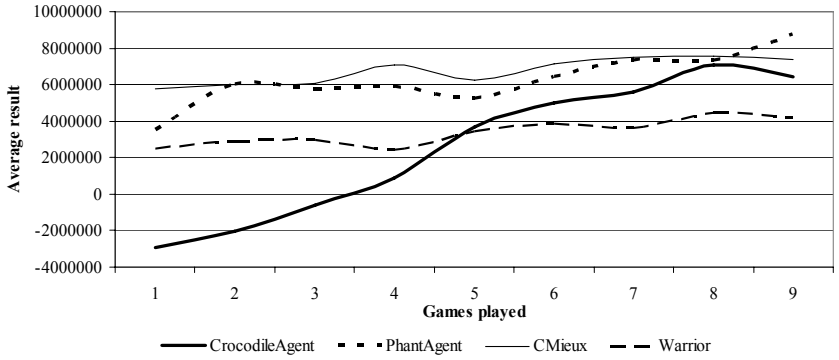

Fig. 3. Average scores of final rounds

The CrocodileAgent had the highest percentage of orders delivered on time and the highest factory utilization. These results are the outcome of boosting the low component management algorithm. The PhantAgent and the CMieux agents had approximately similar results in these segments, with only a small advantage of the PhantAgent, while the Warrior agent's performance in these segments was not very competitive. The problem with the CrocodileAgent was the fact that it had the most unused components left in its warehouse at the end of the game. The Warrior had the best results in the leftover component management, followed by CMieux and PhantAgent. Even though the CrocodileAgent sold the highest number of PCs it the games and implemented very efficient long-term purchase mechanisms, the agent placed 3rd in the final rounds due to very high component purchase prices during the spot market purchase.

5 Conclusions and Future Work

In this paper we presented the SCM-PC Competition and described the main features of the CrocodileAgent, our entry in that competition. The purpose of the SCM-PC Game is to evaluate the performance of mixed procurement strategies. Trading agents which compete in one SCM-PC Game are required to manage supply chain risks through combination of the long-term, quantity flexible procurement contracts and one-off, spot market settlements.

The SCM-PC 2007 was the first SCM-PC Competition. For future work we plan to thoroughly analyze CrocodileAgent's performance through a series of controlled experiments. On the basis of performed analysis improvements of CrocodileAgent's weaknesses are going to be designed and implemented. Moreover, we plan to enhance the CrocodileAgent's spot market trading mechanism with the ability of historical

data analysis, allowing it to efficiently use information from previous games. In such a manner CrocodileAgent could not just react to the current state of the PC market, but also predict the fluctuations of customer demand and, consequently, minimize the quantity of unused components left in its warehouse at the end of the game.

Acknowledgements. This work was carried out within research projects 036-0362027-1639 "Content Delivery and Mobility of Users and Services in New Generation Networks", supported by the Ministry of Science, Education and Sports of the Republic of Croatia, and "Agent-based Service & Telecom Operations Management", supported by Ericsson Nikola Tesla, Croatia.

References

1. Benish, M., Sardinha, A., Andrews, J., Sadeh, N.: CMieux: Adaptive Strategies for Competitive Supply Chain Trading. In: Proc. of the 8th Int. Conf. on Electronic Commerce (ICEC), pp. 1-10. ACM: Fredericton (2006)
2. Sardinha, A., Benish, M., Andrews, J., Sadeh, N.: The 2007 Supply Chain Trading Agent Competition - Procurement Challenge. Technical Report CMU-ISRI-07-106 (2007). http://www.cs.cmu.edu/~alberto/papers/CMU-ISRI-07-106.pdf
3. Podobnik, V., Petric, A., Jezic, G.: The CrocodileAgent: Research for Efficient Agent-Based Cross-Enterprise Processes. In: Meersman, R., Tari, Z., Herrero P., et al. (eds.) OTM 2006. LNCS, vol. 4277, pp. 752-762. Springer, Heidelberg (2006)
4. Petric, A., Podobnik, V., Jezic, G.: The CrocodileAgent 2005: An Overview of TAC SCM Agent. In: Fasli, M., Shehory, O. (eds.) TADA/AMEC 2006. LNCS, vol. 4452, pp. 219-233. Springer, Heidelberg (2007)
5. Petric, A., Podobnik, V., Jezic, G.: The CrocodileAgent: Designing a Robust Trading Agent for Volatile E-Market Conditions. In: Nguyen, N.T. et al. (eds.) KES-AMSTA 2007. LNCS, vol. 4496, pp. 597-606. Springer, Heidelberg (2007)
6. Arunachalam, R., Sadeh, N.: The Supply Chain Trading Agent Competition. Electronic Commerce Research and Applications 4(1), 66-84 (2005)
7. Eriksson, J., Finne, N., Janson, S.: Evolution of a Supply Chain Management Game for the Trading Agent Competition. AI Communications 19(1), 1-12 (2006)
8. Wellman, M. P., Estelle, J., Singh, S., Vorobeychik, Y., Kiekintveld, C., Soni. V.: Strategic Interactions in a Supply Chain Game. Computational Intelligence 21(1), 1-26 (2005)
9. Podobnik, V., Trzec, K., Jezic, G.: Context-Aware Service Provisioning in Next-Generation Networks: An Agent Approach, International Journal of Information Technology and Web Engineering 2(4), 41-62 (2007)
10. Vytelingum, P., Dash, R.K., He, M., Jennings, N.R.: A Framework for Designing Strategies for Trading Agents. In: Proc. of the Int. Workshop on Trading Agent Design and Analysis (TADA), pp. 7-13. IJCAI: Edinburgh (2005)
11. Collins, J., Arunachalam, R., Sadeh, N., Eriksson, J., Finne, N., Janson, S.: The Supply Chain Management Game for the 2007 Trading Agent Competition. Technical Report CMU-ISRI-07-100 (2006). http://www.sics.se/tac/tac07scmspec.pdf
12. Pardoe, D., Stone, P.: An Autonomous Agent for Supply Chain Management. In: Adomavicius, G., Gupta, A. (eds.) Handbooks in Information Systems Series: Business Computing. Elsevier, Amsterdam (2007)
13. Sardinha, A., Benisch, M., Sadeh, N., Ravichandran, R., Podobnik, V., Stan, M.: The 2007 Procurement Challenge: A Competition to Evaluate Mixed Procurement Strategies. Technical Report CMU-ISRI-07-123 (2007). http://www.cs.cmu.edu/~alberto/papers/CMU-ISRI-07-123.pdf