Journal of Universal Computer Science, vol. 14, no. 7 (2008), 1080-1104
submitted: 1/10/07, accepted: 21/1/08, appeared: 1/4/08 © J.UCS

An Agent-Based Solution for Dynamic Supply Chain Management

Vedran Podobnik
(University of Zagreb, Croatia
vedran.podobnik@fer.hr)

Ana Petric
(University of Zagreb, Croatia
ana.petric@fer.hr)

Gordan Jezic
(University of Zagreb, Croatia
gordan.jezic@fer.hr)

Abstract: Supply chain management (SCM) deals with planning and coordinating activities such as material procurement, product assembly, and the distribution of manufactured products. This paper offers an agent-based solution as a potentially adequate approach for the automation of supply chain management. The greatest obstacle in SCM research is obtaining benchmark designed solutions since it is difficult to simulate real business environments, while live testing in real-world systems is not an option. The Trading Agent Competition Supply Chain Management (TAC SCM) scenario provides a unique testbed for studying and prototyping SCM agents by providing a challenging game environment where competing agents engage in complex decision-making activities with the purpose of maximizing their profit. In this paper, we describe the TAC SCM environment and present the main features of the CrocodileAgent, our TAC SCM 2007 entry. Additionally, the CrocodileAgent's performance in the competition, as well as in a series of controlled experiments, is discussed.

Keywords: supply chain management, electronic markets, software agents, trading agents, multi-agent simulation
Categories: I.2.1, I.6.0, J.7, K.4.4

1 Introduction

Supply chain management (SCM) involves several activities, including raw material procurement, and producing, selling, and shipping manufactured goods. In today's economy, supply chains are still based on static long-term relationships between trading partners. These relationships are the main obstacle in realising dynamic supply chains, with the market as a driving force. Dynamic SCM improves the competitiveness of companies since it has a direct impact on their capability of adjusting to changing market demands quickly and efficiently [Benish, 06]. Annual worldwide supply chain transactions are counted in trillions of dollars, making this area of research very interesting, not only to academia, but also to industry since even the slightest improvement can bring a very high profit.

The greatest obstacle in SCM research is obtaining benchmark designed solutions since it is difficult to simulate a real business environment (due to the proprietary nature of such systems), while live testing in real-world systems is not an option (due to the high cost of possible errors). The Trading Agent Competition Supply Chain Management (TAC SCM) scenario provides a unique testbed for studying and prototyping SCM agents by providing a competitive environment in which independently created agents can be tested against each other over the course of many simulations in an open academic setting. In a TAC SCM game, each agent acts as an independent computer manufacturer in a simulated economy [Pardoe, 07]. Since the main purpose of the TAC SCM competition is to explore how to maximize the profit in a stochastic environment of volatile market conditions, it is important to develop an agent capable of reacting quickly to changing market conditions. Furthermore, it is critical to implement predictive mechanisms which enable proactive agent behaviour and provide it with a chance to plan in the face of uncertainty. The idea is to build robust, highly-adaptable and easily-configurable mechanisms that will efficiently deal with all SCM facets [Kontogounis, 06]. Additionally, TAC SCM tournaments provide an opportunity to analyze effects which commonly arise in real-world business transacting, such as the bullwhip effect, and their relationship with companies' profits [Jordan, 06]. Furthermore, the tournament can be helpful in developing methods to identify the current economic regime and forecasting market changes [Ketter, 05].

In this paper, we describe the CrocodileAgent, an intelligent agent developed to participate in the TAC SCM 2007 competition. The paper is organized as follows. Section 2 describes why intelligent software agents are enablers of the digital economy. In Section 3, the TAC SCM game is presented. Section 4 describes the CrocodileAgent's architecture and functionalities. Section 5 comments on the CrocodileAgent's ranking in the TAC SCM 2007 competition, and elaborates upon the results of controlled experiments. In Section 6, related work regarding other TAC SCM agents is presented. Section 7 proposes directions for future work and concludes the paper.

2 Intelligent software agents as enablers of the digital economy

The connection between AI (*Artificial Intelligence*) and economics has received a lot of attention recently [Wurman, 02]. The ideas proposed in this paper are also based on that connection, while the practical implementation of the presented ideas is enabled by the use of the agent-oriented programming (AOP) paradigm and supported by the Internet infrastructure. Although the initial architecture of the Internet was geared towards delivering information visually to humans, currently the Internet is transforming into an environment filled with goal-directed applications which intelligibly and adaptively coordinate information exchanges and actions (Web 3.0) [Podobnik, 06a][Podobnik, 07]. At the same time, computers are evolving from single isolated devices to entry points into a worldwide network of information exchange and business transactions [Fensel, 04]. 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, giving a rise to a vast new range of business combinations [Carlson, 04]. Additionally, by utilizing AOP, the digital economy automates business transactions. AOP can also be used in the realization of emerging virtual organizations or enterprises. Here, agents represent different entities, such as manufacturers, suppliers, service providers, brokers or other partners which interconnect in order to take advantage of rising opportunities and/or changing needs of the global market [Fasli, 07].

An intelligent software agent is an autonomous program which acts on behalf of its owner (human or organizational) while conducting complex information and communication actions over the Internet. Intelligent software agents enable automated process execution and coordination, thus creating added value for its owner. Figure 1 presents a generic model of an intelligent software agent [Bradshaw, 97][Chorafas, 98][Jurasovic, 07][Podobnik, 07], which we used to design our TAC SCM agent.

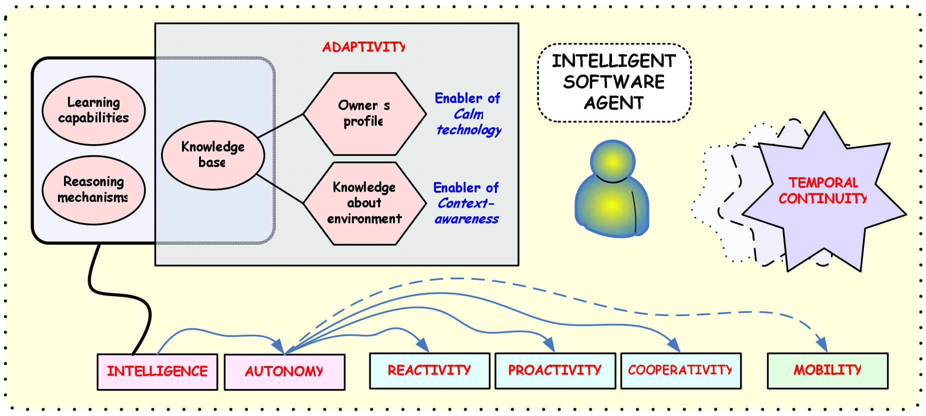

Figure 1: A model of an intelligent software agent

An agent must possess some intelligence grounded on its *knowledge base*, *reasoning mechanisms* and *learning capabilities*. The intelligence of an agent is a prerequisite for all its other characteristics. Depending on the assignment of a particular agent, there are differences in types of information contained in its knowledge base. However, generally this information can be divided into two parts – the *owner's profile* and the agent's *knowledge about its environment*. It is very important to notice that the agent's knowledge base does not contain static information. Adversely, the agent continuously updates its owner's profile according to its owner's latest needs. This allows the agent to efficiently represent its owner, thus realizing the *calm technology* concept. Calm technology is that which serves us, but does not demand our focus or attention [Weiser, 97]. Furthermore, the agent also updates knowledge regarding its environment with the latest events from its ambience and the current state of observed parameters intrinsic to its surroundings, thus realizing *context-awareness*. Context-awareness describes the ability of an agent to provide results that depend on changing context information [Bellavista, 06]. An agent executes tasks *autonomously* without any interventions from its owner, making it an invisible servant, just as Weiser envisioned [Weiser, 97]. An agent must be *reactive*, so it can properly and timely respond to impacts from its environment. An agent not only reacts to excitations from its environment, but also takes initiatives coherent to its tasks. A well-defined objective is an inevitable prerequisite for *proactivity*. An efficient software agent collaborates with other agents from its surrounding: it is *cooperative*. If an agent is capable of migrating between heterogeneous network nodes, this agent is called a *mobile* software agent [Trzec, 07]. An agent has a lifetime throughout which the persistency of its identity and its states should be retained. Thus, it is characterized by *temporal continuity*.

The features of intelligent software agents described above make them perfectly applicable in modern enterprise systems and electronic markets (e-markets). In the past, both markets and the choices available were much smaller than they are today. Consequently, the volatility of supply and demand functions was much more inert. Under such market conditions, companies did not need to make important decisions daily. Instead, they based their business transactions on long-term partnerships. 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 are rising to amazing levels. This is a reason why companies today have great difficulties in enhancing the efficiency of their current business processes, while continuously trying to maximize their profits. Companies are instantly forced to make lots of important decisions, while global competition and perpetually shorter product life cycles are forcing them to explore more agile practices. Keeping in mind the great volatility which characterizes the complex set of market conditions and the vast quantity of available information, a possible solution for improving business efficiency is automating business processes and minimizing human decision-making (where this is possible). Humans simply do not possess the cognitive ability to process such enormous quantities of information (and 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 application of the AOP paradigm – i.e., the creation of computer programs with the ability to completely autonomously manage a set of tasks.

3 The TAC Supply Chain Management 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. One of its game scenarios is the TAC SCM. In the TAC SCM game [Eriksson, 06][Collins, 07] scenario, each of the six agents included in the game has its own PC (*Personal Computer*) manufacturing company. During the 220 TAC SCM days, agents compete in a simulated economy composed of two different markets, as shown in Figure 2. The length of one TAC SCM day is 15 seconds of real time.

In the B2B (Business-to-Business) market, agents compete in buying the raw materials necessary to produce PCs [Sardinha, 07]. Participants in this market are all the agents and eight suppliers which produce four types of components (CPUs, motherboards, memory, hard drives) with different features. In its factory, an agent can manufacture 16 different types of PCs. In the B2C (Business-to-Consumer) market, agents compete in selling all the PCs they produced to customers and, at the same time, trying to earn as much money as possible.

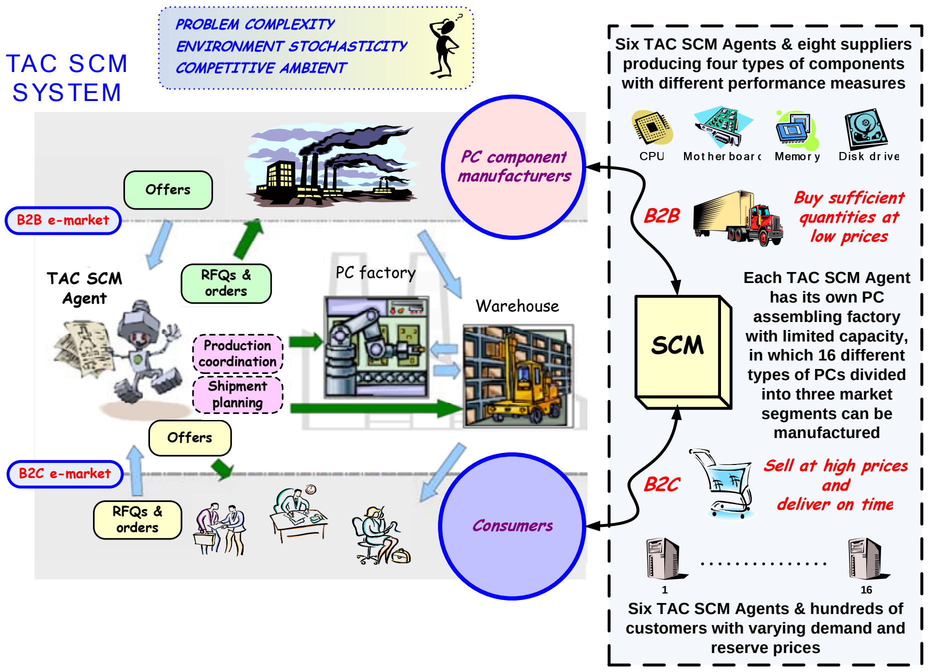
TAC SCM SYSTEM
PROBLEM COMPLEXITY
ENVIRONMENT STOCHASTICITY
COMPETITIVE AMBIENT
Offers
B2B e-market
RFQs & orders
TAC SCM Agent
PC factory
Warehouse
Production coordination
Shipment planning
Offers
B2C e-market
RFQs & orders
PC component manufacturers
Consumers
Six TAC SCM Agents & eight suppliers producing four types of components with different performance measures
CPU
Motherboard
Memory
Disk drive
B2B
Buy sufficient quantities at low prices
SCM
Each TAC SCM Agent has its own PC assembling factory with limited capacity, in which 16 different types of PCs divided into three market segments can be manufactured
B2C
Sell at high prices and deliver on time
1
16
Six TAC SCM Agents & hundreds of customers with varying demand and reserve prices

Figure 2: Relationships in the TAC SCM system

3.1 The architecture of TAC SCM system

The architecture of the TAC SCM system is shown in Figure 3. 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 therefore 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 highest balance on its bank account at the end of the game.

Each day, agents receive messages from the game server with all relevant information concerning the state of the game, customers, suppliers, the bank and their own factory and warehouse. Messages that an agent exchanges with customers and suppliers are not available to other agents and there is no interaction between the agents themselves. Hence, each agent faces strategic uncertainty since the strategies of other agents remain unknown. Agents' responsibilities can be divided into three main tasks, which are described in the following subsections. They are:

- component procurement,
- product sales, and
- production and delivery.

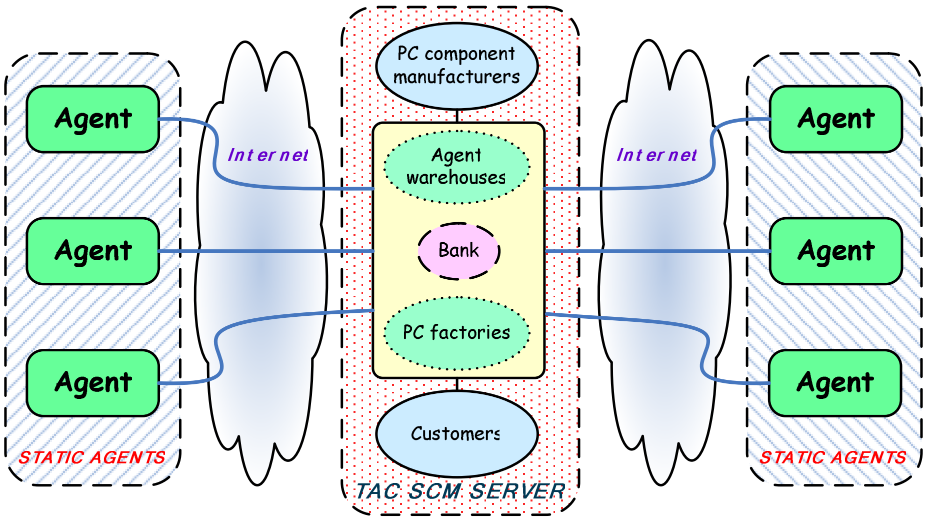

Figure 3: The architecture of the TAC SCM system

3.2 Component Procurement

In order to sell PCs, it is necessary to purchase components and produce PCs from them. There are four different types of components: CPUs, motherboards, memory, hard drives. PCs are assembled from one component of each type and there are two suppliers for each type of component. A supplier has to handle three main daily tasks:

- manage limited production capacity which varies during the game,
- reply to agents' Requests for Quotes (RFQs) by sending offers,
- ship ordered components to agents.

The purchasing protocol is rather simple. The agent sends RFQs to the supplier that produces the needed component. The RFQ contains the requested quantity, delivery date and the price the agent is willing to pay for the certain component type. The supplier responds with an offer. Due to production capacity restrictions, it is possible that the supplier may not be able to deliver the requested quantity by the desired delivery date. If such is the case, it sends the agent two types of offers: a partial offer proposing a smaller quantity of components than requested; and an earliest complete offer proposing a later delivery date by which the requested quantity could be delivered. If one of these offers fits the agent's needs, the agent replies with an order. Suppliers use reputation rating to discourage agents from sending RFQs with no intention of buying in order to raise component prices by creating seemingly high component demand.

3.3 Product Sales

Agents earn money by producing PCs and selling them to customers. The PC purchasing protocol is very similar to the component purchasing protocol. Each day, customers send RFQs to agents. Each RFQ specifies the requested PC type, quantity, due date, the reserve price they are willing to pay for the PCs, and the penalty the agent will have to pay if the PCs are not delivered by the requested due date. Unlike suppliers, agents cannot send different types of offers. The only parameter the agent can negotiate on is the PC price. After receiving offers from all interested agents, the customer sends an order to the agent who offered the cheapest PCs.

Agents whose offer was not the winning one are not informed about the winning price or prices offered by other agents. The only information they receive is the highest and the lowest price at which a certain PC type was sold on the previous day. The number of RFQs and the RFQ parameters themselves vary throughout the game according to a random walk, increasing the risk and uncertainty agents face each day.

3.4 Production and Delivery

Each agent has its own PC assembling factory and a warehouse for storing components and assembled PCs. Each of four component types comes in two different versions based on its characteristics. From these components, a total of 16 types of PCs can be assembled. Depending on the combination of components, PCs require a different number of assembly cycles and can be classified into three market segments: *High range*, *Mid range*, and *Low range*. The factory has a limited number of assembly cycles so an agent has to carefully organize production in order to fulfil all received customer orders. A storage fee is charged for keeping components and PCs in the warehouse. This is meant to discourage agents from piling up a large inventory over a long period of time.

4 Overview of the CrocodileAgent 2007

The CrocodileAgent is an intelligent trading agent developed at the Department of Telecommunications, Faculty of Electrical Engineering and Computing in Zagreb, Croatia. The CrocodileAgent [Petric, 05][Podobnik, 06b][Petric, 07a][Petric, 07b] is a long-standing participant in TAC SCM competitions [Arunachalam, 05][Wellman, 05][Eriksson, 06].

4.1 The CrocodileAgent's Architecture

The CrocodileAgent's architecture, shown in Figure 4, is based on incorporating the generic intelligent software agent model (see Figure 1) into the Information-Knowledge-Behaviour (IKB) framework [Vytelingum, 05]. The IKB framework is a three layered agent-based framework for designing strategies in e-markets.

The first layer is the Information Layer (IL) which gathers data from the ongoing game and assigns that data a meaning. This data can be divided into two parts: data gathered from the market which is available to all agents (i.e., public data) and private data regarding the agent's own states and actions throughout the game. Due to a limited capacity, the IL cannot contain all the data available about the game. The data that gets stored into the IL is determined by the Data Filter (DF).

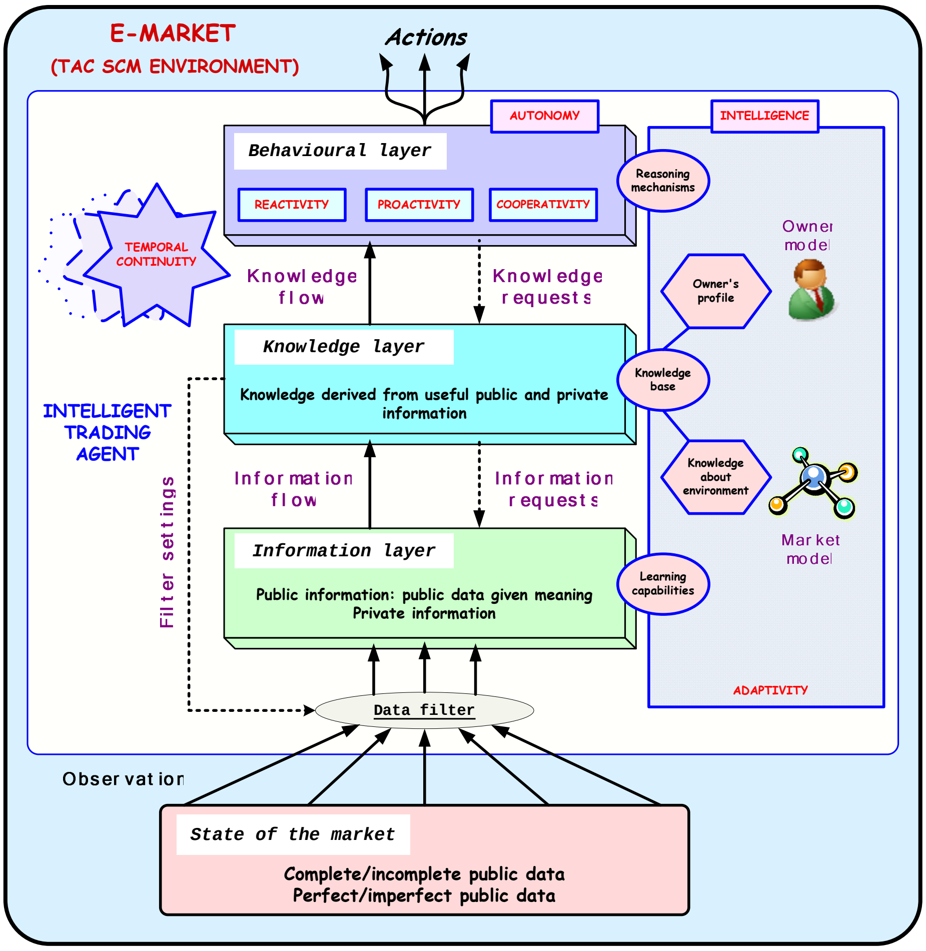

Figure 4: The CrocodileAgent's architecture

The second layer is the Knowledge Layer (KL) which represents knowledge acquired from the information stored in the IL. The KL also contains data obtained from previous games, enabling the agent to learn from past experiences. The KL determines and modifies settings of the DF. Knowledge contained in the KL's *knowledge base* can be divided into the *owner's profile* (i.e., the model of agent's owner) and the *knowledge about its environment* (i.e., the model of the market).

The third layer is the Behavioural Layer (BL) which is a decision-making component that determines the agent's strategic behaviour. The BL uses knowledge from the KL and utilises the agent's *reasoning capabilities* to make important decisions regarding component purchases, the production schedule, PC sales and shipment.

4.2 The CrocodileAgent's Procurement Strategy

Component procurement mechanisms are responsible for assuring that there is always enough components in stock (in order to maintain maximal utilization of production capacity), while aiming to purchase components at the lowest possible price. Since a storage fee is charged for keeping components in the warehouse, there is also motivation to maintain minimal component stocks to reduce storage costs. The uncertainty of future consumer demand, as well as supplier production, makes this problem even more complex. Consequently, TAC SCM agents must balance between short-term and long-term procurement, while constantly making trade-offs between the quantity and price of their orders.

There are two different aspects of component procurement in the TAC SCM game: *day0* component procurement and ordering components during the game. A close examination of the TAC SCM game rules [Collins, 07] suggests that procurement of components at the very beginning of the game (*day0* procurement) may provide an agent with cheap components for the beginning of the game (since there is no prior component demand), while long-term procurement in first few days may ensure an agent have components with decent prices throughout the game.

4.2.1 *Day0* procurement

The most important parameters used in *day0* component procurement are:

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

d_{del}	p_{min}	q_{CPU}	q_{oth}
3	$1.05 \times p_{nom}$	150	300
5	$1.02 \times p_{nom}$	300	600
7	$1.01 \times p_{nom}$	300	600
10	$0.97 \times p_{nom}$	300	600
13	$0.95 \times p_{nom}$	450	900

Table 1: The actual parameter values in day0 RFQs

The goal of using the *day0* procurement strategy is to obtain components for the beginning of the game, when there is a lot of pressure on suppliers due to agents' empty warehouses. The CrocodileAgent sends the day-maximum of five RFQs with the parameters set to those shown in Table 1. The referred parameters were determined after conducting a series of experiments. An unfavourable situation occurs when the chosen supplier cannot deliver the requested quantity on time. In this case, the agent accepts partial offers.

4.2.2 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., 375 for CPUs and 750 for other components),
- N_{max} – the maximal quantity of components allowed in storage (i.e., 550 for CPUs and 1100 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 on the current 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 the values of parameters N_{min}, N_{max} and N_{ord} are not fixed throughout the game. In fact, they are multiplied by a *dayFactor*, which is equal to 1 for the first 30 days, then increases linearly from 1 to 1.5 between days 31 and 140, and finally decreases linearly from 1.5 to 0.7 between days 141 and 200. The *dayFactor* then stays at this level until the end of the game.

A special aspect of component purchase during the game is long-term procurement. Although utilization of the long-term procurement strategy involves certain risks, such as the possibility of creating a huge component stock and potential component overpay (in games with longer periods of low consumer demand), utilization of this strategy plays an important role in games with longer periods of high consumer demand. This stems from the fact that component prices are usually high (when consumer demand is high), while at the same time component availability for short-term orders is usually very poor. In order to ensure an agent have decently-priced components throughout the game, the CrocodileAgent uses the maximum of five RFQs per day for the first 20 days of the game (components needed during those days are purchased through *day0* procurement, if possible). This gives a total of one hundred RFQs in which the CrocodileAgent sends offers for long-term purchases of all types of components. Delivery dates are uniformly distributed between days 15 and 210, making components arrive regularly every other day. The exact quantities (i.e., $N \in [105, 155]$ for the CPUs and $N \in [180, 315]$ for other components) and reserve prices (i.e., $p_{min} \in [0.62, 0.69] \times p_{nom}$ for the RFQs with later delivery due dates and $p_{min} \in [0.82, 0.85] \times p_{nom}$ for the RFQs with earlier delivery due dates) in these long-term RFQs depend only on the requested delivery due date. The referred values were determined after conducting a series of experiments.

After day 20, the CrocodileAgent begins to apply short-term (lead time less than 7) and medium-term (lead time between 7 and 30) purchasing. At the start 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 met:

$$N_{inv} + \textit{evaluatedQuantity} \geq N_{max} \qquad \textbf{(1)}$$

If so, the components are not ordered. However, in spite of condition (1), there are two situations in which the CrocodileAgent may send some RFQs to component suppliers. The first situation is a consequence of the fact that, due to the volatility of supplier capacities throughout the game, the prices offered in response to RFQs requesting near-immediate delivery are very unpredictable. To allow for the possibility of achieving low-priced procurement (i.e., $p_{min} \in [0.6, 0.65] \times p_{nom}$), the CrocodileAgent sends two RFQs requesting small quantities due within 2 days (the minimum delivery timeframe possible). The exact quantities and reserve prices in *2-day* RFQs depend on the current date and N_{inv}. *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 [Pardoe, 06]. The second situation in which the CrocadileAgent can ignore fulfilment of condition (1) is when it has not sent any RFQs for a certain component over a longer period of time (at least 10 days). If the current date is before day 130 and the *evaluatedLongTermQuantity* is lower than its upper limit (i.e, $1.77 \times N_{max}$), the agent sends one short-term RFQ to ensure cheap components (i.e., $p_{min} \in [0.62, 0.67] \times p_{nom}$ and quantities are set to the $0.8 \times N_{ord}$) for the later stage of the game.

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

$$N_{inv} + N_{tdy} > N_{min} \qquad \textbf{(2)}$$

If condition (2) is not fulfilled, the agent purchases components more aggressively with the aim 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 aim 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 p_{min} and the exact quantities which need to be ordered. A simplified description of some of these mechanisms follows:

- The *lowComponentAlarm* contains several levels and marks a 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 corresponding component.
- The *demandPurchaseQuantityFactor* is modified according to customer demand. Sometimes during the game, customer demand may rise rapidly. When this happens, the agent uses more components to produce more PCs.

In this case, the parameter is increased to ensure that the agent does not run out of components and consequently loose potentially profitable PC orders.

Special attention was paid to the end of the game. The intention was to maintain at least a minimal level of all components in the warehouse up until the game's end to enable the CrocodileAgent to fulfil orders from customers for as long as possible. It is very important to allow the agent to send RFQs requesting near-immediate delivery of any type of component to prevent the situation where a large quantity of one component is left over because the agent had to use all the other components to produce a certain type of PC.

4.3 The CrocodileAgent's Sales Strategy

Component procurement mechanisms are responsible for deciding which computers to offer at what prices so that the available resources are used as efficiently as possible. This usually leads to computing the highest offer price that can maintain maximal factory utilization, while still obtaining marginal profit [Stan, 06].

4.3.1 An algorithm for sending offers

The CrocodileAgent sorts RFQs in decreasing order of their reserve prices, for every PC type separately. After sorting RFQs, the agent starts to send offers if the agent's PC production cost (increased for the current days' minimal profit percentage) is lower than the customer's reserve PC price and if the requested PCs can be delivered from the already produced PC stock stored in the warehouse. In case the latter condition is not fulfilled, the agent checks whether there are enough components available to produce the requested PCs.

This algorithm comes in four versions. The version that is active on a certain day depends on the stage of the game (beginning, middle, end), the number of production cycles needed to produce all active orders, and the version of the algorithm that was used the day before. These four versions mainly differ with respect to the method of determining offer prices for PCs. The version most frequently used during the game is the *Normal version* which determines the offer price using the method described in subsection 4.3.2. The *High Demand Version* uses a "greedy" algorithm since offer prices for PCs are always slightly lower than the customer's reserve price. This version is used when there is a very high customer demand for PCs since, in such cases, agents do not usually send offers for all the RFQs received. The *Game Start Version* is used in the beginning stage of the game, where only a few offers with very high offer prices are sent to consumers since the CrocodileAgent's component stocks are not yet created. The *End Game Version* is used in the finishing stage of the game. This version differs from the other three in the fact that it sorts RFQs in increasing order of their corresponding penalties. The main aim of this version is to sell out the whole inventory in the warehouse so that the profit the agent adds to the basic PC price is minimal. All versions of the selling algorithm implement a mechanism for preventing late deliveries. Each day, the agent monitors its obligations to customers by calculating the number of factory cycles needed to fulfil its existing orders. Based on this information, it determines the earliest possible delivery date for sending new PC offers. This way the agent is prevented from sending offers which cannot be delivered by the requested delivery date. Additionally, to avoid great variations in the number of orders won daily, special attention is paid to limiting the maximal number of offers sent per day. This number depends on the number of RFQs issued by consumers on that day, factory cycles won that day as a consequence of consumers accepting offers sent yesterday, and the total number of factory cycles needed to produce all active consumer orders.

4.3.2 Calculating the prices of components in the warehouse and the profit margin

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

The offer price (used in the *Normal version* of the CrocodileAgent's algorithm for sending offers to consumers) is based on the previous day's price report, which is delivered to the CrocodileAgent on a daily basis and contains the lowest and highest winning prices for each computer type. The CrocodileAgent first calculates the average of the highest winning prices over the last three days of the game (for each computer type separately). This number is then multiplied by the *ordersWonFactor* (i.e., $ordersWonFactor \in [0.92, 1.07]$). This factor which depends on factory cycles won that day as a consequence of consumers accepting offers sent yesterday, and the total number of factory cycles needed to produce all active consumer orders. Basically, this factor increases when the factory is too crowded since this is a sign of too many won bids, and decreases when the factory is not utilized well enough [Stan, 06]. After the *ordersWonFactor* is applied, the final offer price is calculated in accordance with two additional parameters:

- The due date listed in the customer RFQ
    - An earlier due date causes a higher offer price and vice-versa;
- The demand level in the market segment the requested PC belongs to
    - If demand is low, the offer price decreases and vice-versa.

4.4 The CrocodileAgent's Production and Delivery Strategies

Initially, the CrocodileAgent produced PCs only after receiving customer orders, i.e., the PCs were not manufactured in advance. Later, we added the possibility of producing PCs even if nobody ordered them. Due to the stochastic nature of the TAC SCM game, customer demand varies during the game. If the agent does not produce PCs and there is a low demand on the PC market, a large part of the agent's factory capacities stay unutilized. If the agent produces PC stock during a period of low PC demand, its factory will be utilized and the agent will be prepared for a period of high PC demand. However, in some cases, the agent may produce more PCs than it can sell by the end of the game since future demands always have some degree of uncertainty. We tried to lower this risk by introducing quantity limits which represent the maximum number of PCs which can be available in stock. These limits are modified during of the game. As the end of the game approaches, they are lowered accordingly. The limits also differ for each PC market segment. For example, if the CrocodileAgent predicts that the demand for Mid Range PCs will be high, it increases the limit for PCs in that range.

Each day, the CrocodileAgent sorts its list of active orders in chronological order of their delivery dates after which the PC production and delivery algorithm is executed. The algorithm runs as follows:

- If there are enough PCs in the warehouse to fulfil the order, they are reserved and added to the delivery schedule,
    - If there are not enough PCs, but there are enough components to produce the requested PCs, the components are reserved and the agent tries to add PCs to the production schedule,
        - The production demand will be successfully fulfilled only if there is enough free factory capacity for the next day,
- After analyzing all active orders, the agent makes plans for creating PC stock,
    - In order to create PC stock, the agent checks the amount of free capacity available for the following day, whether there are enough components to produce the PCs, and which PC types can be produced without creating a larger stock than allowed.

5 The CrocodileAgent's performance

The CrocodileAgent has participated in TAC SCM competitions since 2004. In this section, we analyze the results of controlled experiments designed to evaluate the impact of the changes we have made on the CrocodileAgent over the years. Since there were significant rule changes after the TAC SCM 2004 competition, CrocodileAgent 2004 is excluded from these experiments. Before the experiment overview, we present a brief summary of the last TAC SCM competition.

5.1 The TAC Supply Chain Management Competition 2007

The TAC SCM 2007 competition was divided into three parts: qualifying rounds held from June 14th-22nd, seeding rounds held from July 9th -17th and final rounds held from July 23rd-25th. There were 18 teams competing in the 2007 TAC SCM. The CrocodileAgent took 3rd place in the quarterfinals with an average score of 6.775 M and 5th place with an average score 5.116 M in the semi-finals. The CrocodileAgent ended its participation in TAC SCM 2007 as the Second Finals winner with an average score of 24.43 M.

5.2 Experiments

We held three competitions with some of the best agents from the TAC SCM 2006 competition. There were six agents in each competition: five fixed opponents (DeepMaize, Maxon, PhantAgent, Southampton and TacTex) and one version of the CrocodileAgent (from TAC SCM 2005, 2006 or 2007 Final rounds). All opponent agents were downloaded from the Agent Repository accessible from the official TAC

Web page (http://www.sics.se/tac/showagents.php). Competitions were held in our laboratory, each consisting of 25 games.

5.2.1 CrocodileAgent 2005 Performance

Place	Agent	Score	Games played
1.	**DeepMaize**	11 476 435	25
2.	**PhantAgent**	11 164 750	25
3.	**Southampton**	5 189 560	25
4.	**Maxon**	4 517 666	25
5.	**TacTex**	549 976	25
6.	**CrocodileAgent 2005**	-143 117	25

Table 2: Competition 1 results at server pocahontas.zavod.tel.fer.hr

The final ranking of the first competition is shown in Table 2. After the competition finished, we conducted a detailed analysis of the games played. The majority of the analysis was done using the CMieux Analysis and Instrumentation Toolkit for TAC SCM [Benisch, 05]. If we take a closer look at the overall results of the first competition, we can see that the agents can be divided into three groups with respect to their final scores and the differences between them. Since the final score is a result of trading in both the B2B and B2C markets, we will discuss both component procurement and PC sales for each agent.

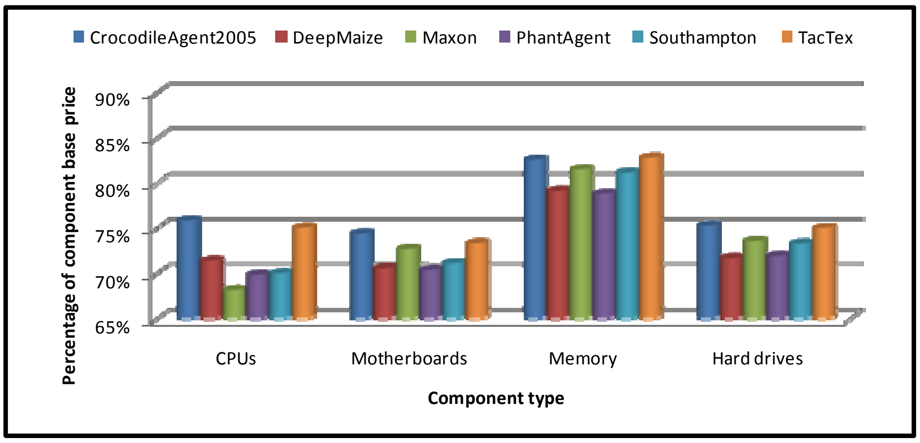

Figure 5: The average prices of components purchased during Competition 1

The first task was to analyze component purchases. After gathering information regarding the prices agents paid for each component type and the quantities they purchased, we calculated the average prices. Since components with different performance characteristics have different prices, we used the ratio between the actual prices agents paid for given components and the component base prices in order to display the results. The average prices are shown in Figure 5.

Since the price of a CPU accounts for more than 50% of the total PC price, it is very important to purchase cheap CPUs. We can see from Figure 5 that Maxon bought some of the cheapest CPUs, while CrocodileAgent 2005 and TacTex paid the highest prices for their CPUs. The situation is quite similar with non-CPU components: CrocodileAgent 2005 and TacTex again paid the highest prices, while DeepMaize and PhantAgent had the best purchasing algorithms and bought relatively cheap components. Maxon's purchasing algorithm functioned much better for CPUs, while it bought non-CPU components at rather high prices in comparison with the prices paid by other agents.

If we look at the average PC selling prices shown in Figure 6, we can see the obvious reason for DeepMaize winning the competition. Namely, DeepMaize sold the most expensive PCs (in addition to buying cheap components). TacTex also had a very good PC selling algorithm, but its procurement strategies considerably lowered its ranking. PhantAgent had a significantly better selling strategy than Southampton, Maxon or the CrocodileAgent, whose results were all similar. The CrocodileAgent's cheap PC sales, in combination with purchasing some of the most expensive components, are the main reasons the agent finished last in the competition.

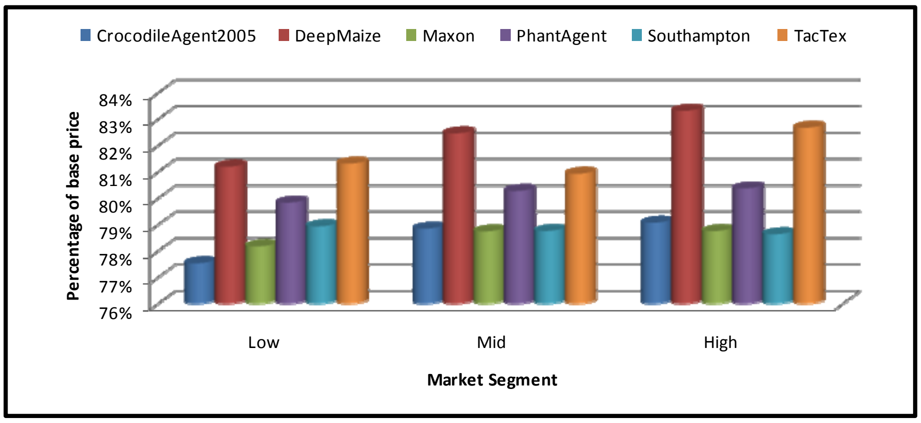

Figure 6: The average prices of PCs sold during Competition 1

5.2.2 CrocodileAgent 2006 Performance

The performance of the CrocodileAgent which participated in TAC SCM 2006 was analyzed from the results obtained in the second competition. In Table 3, we can see a significant difference between the average results of the two leading agents which changed places with respect to ranking from the first competition. The ranking of the remaining agents did not change.

Place	Agent	Score	Games played
1.	**PhantAgent**	14 215 509	25
2.	**DeepMaize**	11 989 889	25
3.	**Southampton**	7 202 768	25
4.	**Maxon**	6 625 170	25
5.	**TacTex**	897 503	25
6.	**CrocodileAgent 2006**	-35 244	25

Table 3: Competition 2 results at server pocahontas.zavod.tel.fer.hr

The situation with respect to component procurement (see Figure 7) also remained the same. Maxon bought the cheapest CPUs, PhantAgent and DeepMaize bough the cheapest non-CPU components, while the CrocodileAgent and TacTex bought the most expensive components.

When we compare the average selling prices (see Figure 8), we can see that the CrocodileAgent significantly improved its selling algorithm, while the performance of other agents was similar to their performance in the first competition. The improvement of the CrocodileAgent's selling policy was most likely caused by the introduction of the "greedy" algorithm which is used in case there is a high consumer demand. Another improvement is the increase of the CrocodileAgent's competitiveness in case there is a low consumer demand. It was obtained through the modification of the mechanism for calculating PC prices by introducing a discount for components which were present in the warehouse for a longer period of time.

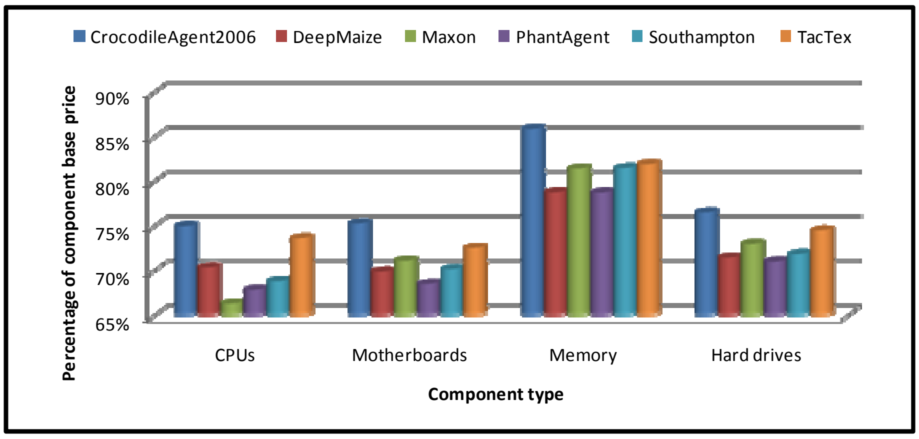

Figure 7: The average prices of components purchased during Competition 2

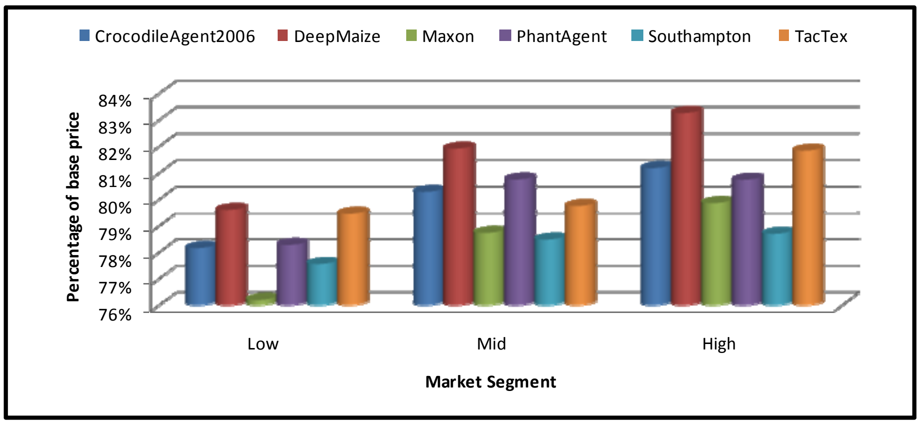

Figure 8: The average prices of PCs sold during Competition 2

5.2.3 CrocodileAgent 2007 Performance

The third competition was carried out in order to analyze the performance of the CrocodileAgent which participated in TAC SCM 2007. The two leading agents again changed places, while the difference between their final scores was similar to their difference in the first competition. The ranking and the difference between Southampton and Maxon was also similar to that of prior competitions. We can see a significant improvement in the latest version of the CrocodileAgent by looking at Table 4.

According to the average component purchase prices (see Figure 9), Maxon still bought the cheapest CPUs, although the CrocodileAgent's were just slightly more expensive. Furthermore, the CrocodileAgent bought the second cheapest memory and the cheapest motherboards and hard drives. The ratios of the other agents were the same as those in prior competitions.

Place	Agent	Score	Games played
1.	**DeepMaize**	8 916 689	25
2.	**PhantAgent**	8 661 415	25
3.	**CrocodileAgent 2007**	4 357 205	25
4.	**Maxon**	2 224 996	25
5.	**Southampton**	2 091 718	25
6.	**TacTex**	-3 137 101	25

Table 4: Competition 3 results at server pocahontas.zavod.tel.fer.hr

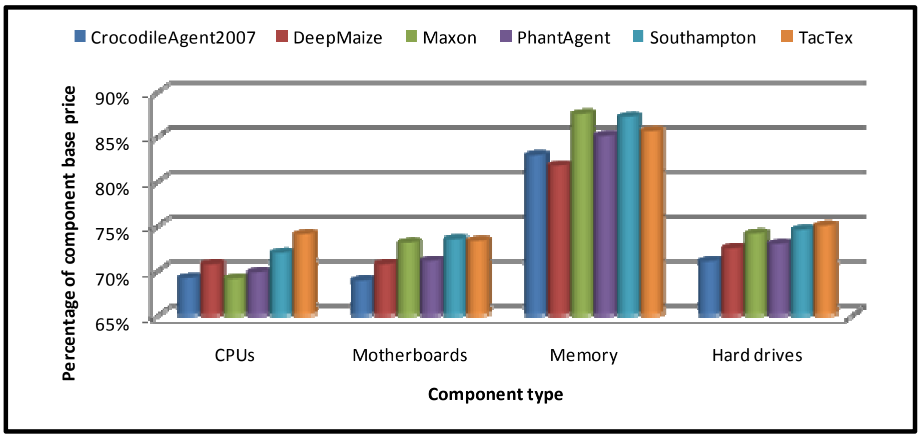

Figure 9: The average prices of components purchased during Competition 3

When observing the average PC selling prices (see Figure 10), we can see that the CrocodileAgent sold the cheapest PCs and that its new selling algorithm obtained worse results than its previous one. Comparing with prior competitions, we can see that DeepMaize maintained the best PC selling strategy, while Southampton improved its results in comparison with Maxon and came closer to PhantAgent's selling PC prices.

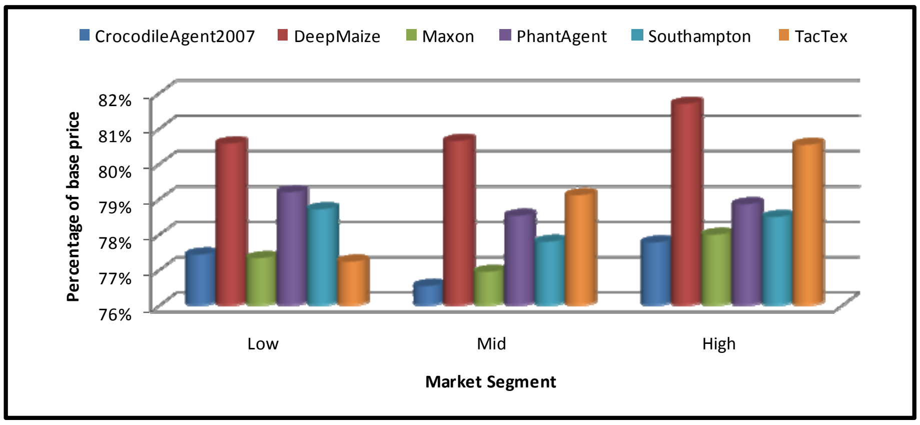

Figure 10: The average prices of PCs sold during Competition 3

5.2.4 Old CrocodileAgent vs New CrocodileAgent

The purpose of the conducted experiments was to analyze the CrocodileAgent's performance and compare the results of the old versions with the new one. We can see a significant improvement in the component purchasing mechanism (see Figure 11). The average price paid by the agent for CPUs was more than 6% lower in the newest 2007 version than in the 2005 version.

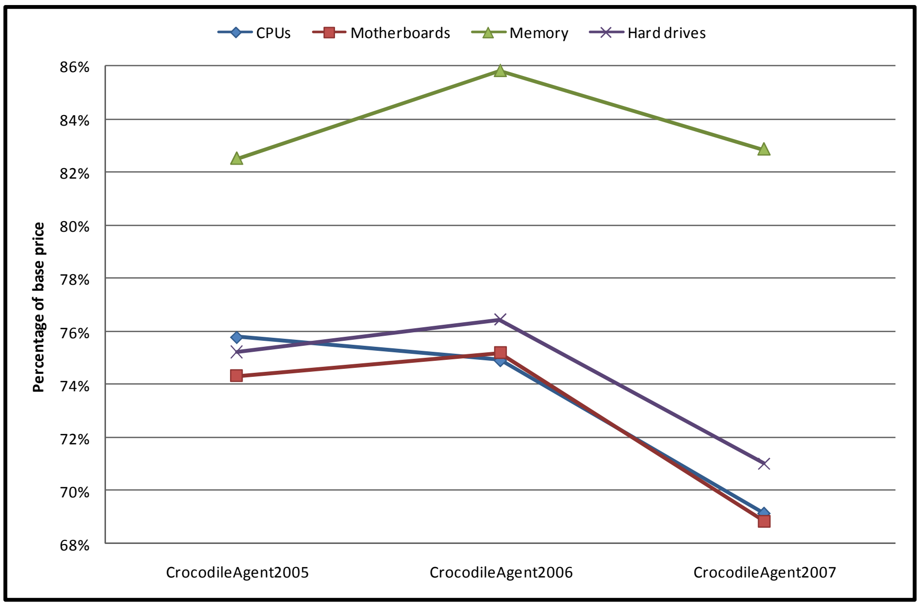

Figure 11: The average prices the CrocodileAgent paid for components

The best improvement among the components was achieved for CPUs which is very desirable since CPUs account for more than 50% of the total PC price. The improvement was achieved by introducing different purchasing policies for CPU and non-CPU components in order to prevent the agent from buying CPUs on significantly higher prices than usual while the *lowComponentAlarm* is set. The average prices paid for motherboards and hard drives were also several percent lower than in the previous versions. The price paid for memory stayed the same. This does not represent a significant drawback since memory is the cheapest of all the components involved.

Unlike the purchasing mechanism, the modifications of the PC selling algorithm did not bring the expected improvements. From Figure 12, we can see that the latest version of the CrocodileAgent gave the lowest average PC selling price. This fact does not necessary mean that CrocodileAgent 2005 implemented a better selling mechanism than CrocodileAgent 2007 since a comparison of Figures 6, 8 and 10 reveals that the *overall* average selling PC prices were lowest in Competition 3. This controversy, where CrocodileAgent 2007 has a significantly better total score when compared to CrocodileAgent 2005, but has a statistically less efficient selling mechanism, is a great example of SCM complexity. It is direct proof that advances in SCM research cannot be made by isolating one SCM task (e.g., final product selling) and optimizing its solution, but rather the SCM problem must be approached as integral and indivisible.

Although five of six agents in Competitions 1 and 3 remained unchanged, different strategies utilized by CrocodileAgents 2005 and 2007 caused perceptibly different competition environments resulting in diverse efficiency levels and varying final results of the other agents.

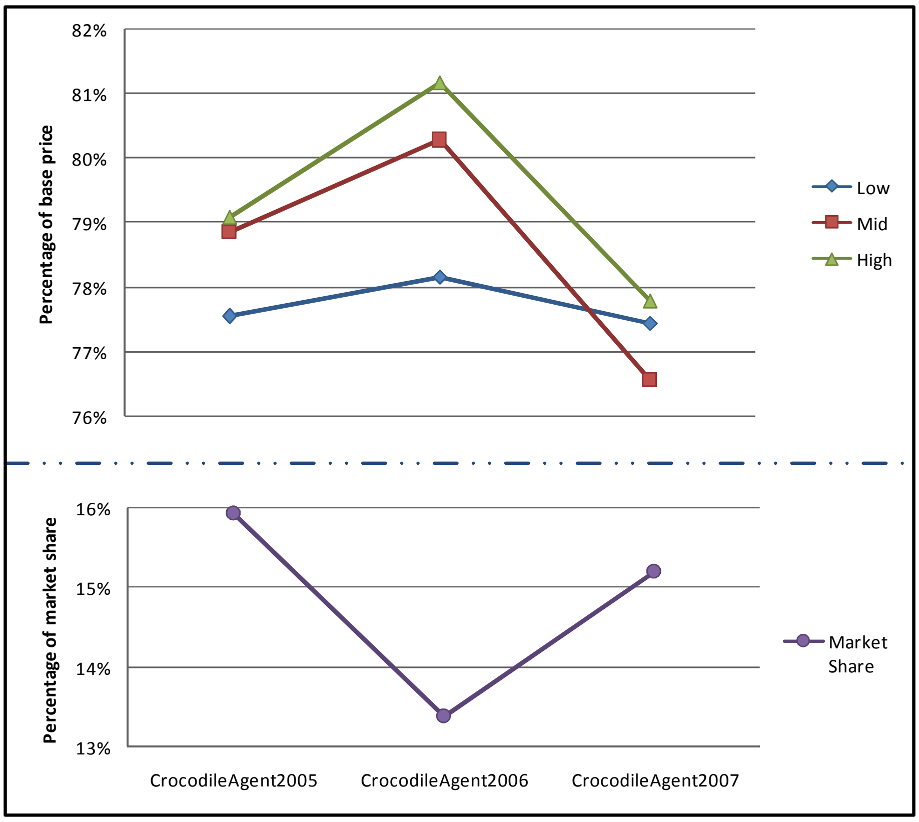

Figure 12: The average PC selling prices of the CrocodileAgent and its average market share

6 Related Work

The TAC SCM game was created in 2003 in collaboration with Carnegie Mellon University (CMU) and the Swedish Institute of Computer Science (SICS). Since then, most of the agent development teams which have been competing in TAC SCM competitions have published several papers describing their entries in the game. Most of these papers can be found at http://tac.eecs.umich.edu/researchreport.html.

There are several approaches aimed at solving supply chain management problems. In this section, we will address only those approaches which are used by the agents we chose for our experiments.

SouthamptonSCM employs fuzzy reasoning for determining which customer RFQs to bid on and for adapting PC offer prices to prevailing market conditions, agent’s inventory level and the elapsed time of the game [He, 06]. On the B2B market, the agent uses a mixed component procurement strategy to balance long and short-term component orders.

PhantAgent's architecture consists of three correlated modules that are in charge of the agent's main daily tasks [Stan, 06]. Since the modules are strategically interdependent, finding the optimal solutions for sub-problems does not necessarily lead to the agent's best overall performance. Thus, the PhantAgent uses heuristic approximations instead of optimization algorithms.

The TacTex agent bases its actions on its predictions regarding the future of the economy in the game, with an emphasis on component supplier offer prices and customer demand [Pardoe, 04][Pardoe, 07]. The agent can adapt these predictions based on the observed behaviour of other agents in the current game. TacTex also adapts its strategies according to observations from past games, particularly focusing on other agents' strategies with respect to buying components at the beginning of the game and selling computers at the end of the game.

Deep Maize is designed to estimate the marginal values for each component and PC as accurately as possible, given predictions regarding market conditions and production constraints [Wellman, 05][Kiekintveld, 06]. The agent first performs centralized, high-level optimization to create a long-term projected production schedule. In second-stage optimization, it forms a modified objective function from the obtained resource values in order to make low-level decisions which affect individual sub problems.

7 Conclusions and Future Work

In this paper, we presented the CrocodileAgent, a trading agent which represents proof-of-concept of the proposed agent-based dynamic supply chain management paradigm. After a brief game description, we listed the basic TAC SCM Agent tasks and explained how they were implemented in CrocodileAgent 2007. Furthermore, we presented the performance of the CrocodileAgent in TAC SCM 2007, and analyzed a competition we held with CrocodileAgents 2005, 2006, and 2007, and five of the best agents from TAC SCM 2006. We think that this is a good way to evaluate the impact of the changes we have made in the CrocodileAgent over the years, and to determine the CrocodileAgent's soft spots.

A thorough analysis of the competitions was conducted. Special attention was given to the performance of the last version of our agent, CrocodileAgent 2007. We found evidence that component purchase mechanisms and algorithms for managing factory activities function quite well. However, the CrocodileAgent's reactive algorithm for selling PCs did not perform as well as the selling algorithms employed by the top TAC SCM agents. This algorithm does not predict the fluctuation of prices on the PC market, but only reacts to its current state. Thus, further development of our agent will focus on improving the PC selling algorithm, with an emphasis on customer demand prediction and the prediction of winning PC prices. This does not mean that the component procurement algorithms are going to be neglected. We plan to improve our procurement algorithm by introducing multi-attribute decision making techniques for evaluating suppliers' offers and deciding which offer to accept [Sadeh, 03].

This paper describes the shift from traditional SCM management to agile supply chain practices. This includes adopting more flexible contractual relationships and taking advantage of new levels of business interoperability across the supply chain. We are all witnessing the emergence of similar trends and strategic interactions in the service sector as well. Therefore, another important direction of our future work will be broadening our supply chain research from the product to the service domain (in the service domain, relationships across the supply chain are far more complex, so here we speak of supply networks). Preparation for adapting our TAC SCM agent to the service domain has already been done by changing our agent's architecture from functional [Petric, 07a] to IKB-based. An advantage of an IKB-based architecture is the separation of the Information Layer (IL), the Knowledge Layer (KL) and the Behaviour Layer (BL), which enables the physical distribution of layers on multiple computers. Although the current version of the CrocodileAgent is run on only one computer, for future work we plan to transform the CrocodileAgent into a JADE [Bellifemine, 07] multi-agent system. In this system, the IL, KL and BL layers will be distributed across multiple computers and various JADE agents will be allocated to individual layers. An additional advantage of the IKB-based design model is system scalability. Although such architecture is not imperative for the TAC SCM environment, it is crucial for environments with a huge number of entities on both the consumer and business sides of the supply chain/network (i.e., the telecom environment).

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

[Arunachalam, 05] Arunachalam, R., Sadeh, N. M.: "The Supply Chain Trading Agent Competition"; Electronic Commerce Research and Applications, 4, 1 (2005), 66-84.

[Bellavista, 06] Bellavista, P., Corradi, A., Montanari, R., Tonin, A.: " Context-Aware Semantic Discovery for Next Generation Mobile Systems"; IEEE Communications, 44, 9 (2006), 62-71.

[Bellifemine (07)] Bellifemine, F. L., Caire, G., Greenwood, D.: "Developing Multi-Agent Systems with JADE"; John Wiley Sons, Chichester (2007)

[Benisch, 05] Benisch, M., et. al.: "CMieux Analysis and Instrumentation Toolkit for TAC SCM"; Carnegie Mellon University Technical Report CMU-ISRI-05-127, Pittsburgh, USA (2005).

[Benisch, 06] Benish, M., Sardinha, A., Andrews, J., Sadeh, N.: "CMieux: Adaptive Strategies for Competitive Supply Chain Trading"; Proceedings of the 8th International Conference on Electronic Commerce (ICEC'06), Fredericton (2006), 1-10.

[Bradshaw (97)] Bradshaw, J. M.: "Software Agents"; MIT Press, Cambridge (1997)

[Carlson, 04] Carlson, B.: “The Digital Economy: What is New and What is Not?”; Structural Change and Economic Dynamics, 15, 3 (2004), 245-264.

[Chorafas (98)] Chorafas, D. N.: “Agent Technology Handbook”; McGraw-Hill, New York (1998)

[Collins, 07] Collins, J., Arunachalam, R., Sadeh, N., Eriksson, J., Finne, N., Janson, S.: "The Supply Chain Management Game for the 2007 Trading Agent Competition"; Carnegie Mellon University Technical Report CMU-ISRI-07-100, Pittsburgh, USA (2007).

[Eriksson, 06] Eriksson, J., Finne, N., Janson, S.: "Evolution of a Supply Chain Management Game for the Trading Agent Competition”; AI Communications, 19, 1 (2006), 1-12.

[Fasli (07)] Fasli, M.: “Agent Technology for E-Commerce”; Wiley Sons, Chichester (2007)

[Fensel (04)] Fensel, D.: “Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce”; Springer, Berlin (2004)

[He, 06] He, M., Rogers, A., Luo, X., Jennings, N. R.: "Designing a Successful Trading Agent for Supply Chain Management”; Proceedings of the 5th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'06), Hakodate (2006), 1159-1166.

[Jordan, 06] Jordan, P. R., Kiekintveld, C., Miller, J., Wellman, M. P.: "Market Efficiency, Sales Competition, and the Bullwhip Effect in the TAC SCM Tournaments"; Proceedings of the AAMAS Joint International Workshop on the Trading Agent Design and Analysis and Agent Mediated Electronic Commerce (TADA/AMEC'06), Hakodate (2006), 99-111.

[Ketter, 05] Ketter, W., Collins, J., Gini, M., Gupta, A., Shrater, P.: "Identifying and Forecasting Economic Regimes in TAC SCM"; Proceedings of the IJCAI Workshop on Trading Agent Design and Analysis (TADA'05), Edinburgh (2005), 53-60.

[Kiekintveld, 06] Kiekintveld, C., Miller, J., Jordan, P. R., Wellman, M. P.: "Controlling a Supply Chain Agent Using Value-Based Decomposition"; Proceedings of the 7th ACM conference on Electronic commerce (EC'06), Ann Arbor (2006), 208-217.

[Kontogounis, 06] Kontogounis, I., Chatzidimitriou, K. C., Symeonidis, A. L., Mitkas, P. A.: "A Robust Agent Design for Dynamic SCM Environments"; Proceedings of the 4th Hellenic Joint Conference on Artificial Intelligence (SETN'06), Heraklion (2006), 127-136.

[Jurasovic, 07] Jurasovic, K., Kusek., M.: “Verification of Mobile Agent Network Simulator”; Lecture Notes in Computer Science, 3053 (2007), 520-529.

[Pardoe, 04] Pardoe, D., Stone, P.: "Bidding for Customer Orders in TAC SCM: A Learning Approach"; Proceedings of the AAMAS International Workshop on Trading Agent Design and Analysis (TADA'04), New York (2004), 52-58.

[Pardoe, 06] Pardoe, D., Stone, P.: "Predictive Planning for Supply Chain Management"; Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS’06), The English Lake District (2006) .

[Pardoe (07)] 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)

[Petric, 05] Petric, A., Jurasovic, K.: "KrokodilAgent: A Supply Chain Management Agent"; Proceedings of the 8th International Conference on Telecommunications (ConTEL'05), Zagreb (2005), 297-302.

[Petric, 07a] Petric, A., Podobnik, V., Jezic, G.: "The CrocodileAgent 2005: An Overview of TAC SCM Agent"; Lecture Notes in Computer Science, 4452 (2007), 219-233.

[Petric, 07b] Petric, A., Podobnik, V., Jezic, G.: "The CrocodileAgent: Designing a Robust Trading Agent for Volatile E-Market Conditions"; Lecture Notes in Computer Science, 4496 (2007), 597-606.

[Podobnik, 06a] Podobnik, V., Trzec, K., Jezic, G.: "An Auction-Based Semantic Service Discovery Model for E-Commerce Applications"; Lecture Notes in Computer Science, 4277 (2006), 97-106.

[Podobnik, 06b] Podobnik, V., Petric, A., Jezic, G.: "The CrocodileAgent: Research for Efficient Agent-Based Cross-Enterprise Processes"; Lecture Notes in Computer Science, 4277 (2006), 752-762.

[Podobnik, 07] 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 (2007), 41-62.

[Sadeh, 03] Sadeh, N. Sun, J.: "Multi-Attribute Supply Chain Negotiation: Coordinating Reverse Auctions Subject to Finite Capacity Considerations"; Proceedings of the 5th International Conference on Electronic Commerce (ICEC '03), Pittsburgh (2003), 53-60.

[Sardinha, 07] Sardinha, A., Benisch, M., Sadeh, N., Ravichandran, R., Podobnik, V., Stan, M.: "The 2007 Procurement Challenge: A Competition to Evaluate Mixed Procurement Strategies"; Carnegie Mellon University Technical Report CMU-ISRI-07-123, Pittsburgh, USA (2007).

[Stan, 06] Stan, M., Stan, B., Florea, A. M.: "A Dynamic Strategy Agent for Supply Chain Management"; Proceedings of the 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '06), Washington (2006), 227-232.

[Trzec, 07] Trzec, K., Lovrek, I.: "Field-based Coordination of Mobile Intelligent Agents: An Evolutionary Game Theoretic Analysis"; Lecture Notes in Computer Science, 4692 (2007), 198-205.

[Vytelingum, 05] Vytelingum, P., Dash, R. K., He, M., Jennings, N. R.: "A Framework for Designing Strategies for Trading Agents"; Proceedings of the IJCAI Workshop on Trading Agent Design and Analysis (TADA'05), Edinburgh (2005), 7-13.

[Weiser (97)] Weiser, M., Brown, J. S.: "The Coming Age of Calm Technology"; In Dening, P. J., Metcalfe, R. M., Burke, J. (eds.), "Beyond Calculation: The Next Fifty Years of Computing"; Springer, New York (1997)

[Wellman, 05] Wellman, M. P., Estelle, J., Singh, S., Vorobeychik, Y., Kiekintveld, C., Soni, V.: "Strategic Interactions in a Supply Chain Game"; Computational Intelligence, 21, 1 (2005), 1-26.

[Wurman, 02] Wurman, P. R., Wellman, M. P., Walsch, W. E.: "Specifying Rules for Electronic Auctions"; AI Magazine, 23, 3 (2002), 15-24.