# StormFront: A High-Volume Agent for TAC SCM

Todd Hester and David DeAngelis

Agent-Based Electronic Commerce
CS395T: Fall 2006, Professor Stone
The University of Texas at Austin
todd@cs.utexas.edu, dave@lips.utexas.edu

**Abstract.** Supply chain management can be greatly improved using autonomous agents. The Trading Agent Competition Supply Chain Management (TAC SCM) competition is a forum for developing such agents. This paper introduces StormFront, a moderately successful agent that attempts to produce a high volume of products at a low profit margin, capturing the majority of customer demand. The underlying components and design rationale of StormFront are described, followed by empirical testing, and finally the results and discussion of our class tournament are presented.

## 1 Introduction

Supply chains are currently in use by countless corporations worldwide for producing consumer goods. These corporations may be small home-based operations or large multinational conglomerates, and the end products run the entire spectrum of complexity. Raw materials can be obtained from competing suppliers, production methods and efficiency vary widely, and many sales strategies and target product markets exist. The common requirement of supply chains is an agent, software or human, that can make rational, timely decisions in a wildly dynamic operating environment. Often supply chains are well suited for management by software agents. Complex decisions must be made, and the market effects of such decisions must be anticipated for satisfactory performance. It has been shown that agents can consistently obtain larger gains from trade than humans in a continuous double auction [1]. If the operating environment for a supply chain manager can be sufficiently modeled, the computation ability of an agent can be leveraged to reap substantial gains over a human supply chain manager.

We present and evaluate our TAC SCM agent, StormFront. StormFront breaks the supply chain management problem into two main sub-tasks, supply management and demand management. We describe the methods we use for each of these modules. In addition, we provide results showing that the methods we implemented in our agent are significant improvements over the TacTex starter agent [2]. We discuss the effects of these changes and why they are successful as well as the results of the class competition.

## 2 TAC SCM

TAC SCM is a competitive game designed jointly by the e-Supply Chain Management Lab at Carnegie Mellon University and the Swedish Institute of Computer Science (SICS). Six software agents compete in a supply chain scenario where the goal is to achieve the highest total profit over a fixed time period. The supply chain problem is framed in the context of a personal computer producer. The role of the agent is to procure resources, interact with clients, and manage production. The full game details are provided in [3], but the agent can be decomposed into two main responsibilities: interacting with suppliers and manufacturers to efficiently produce computers, and interacting with customers to sell computers at a reasonable profit. These duties are essentially production and sales.

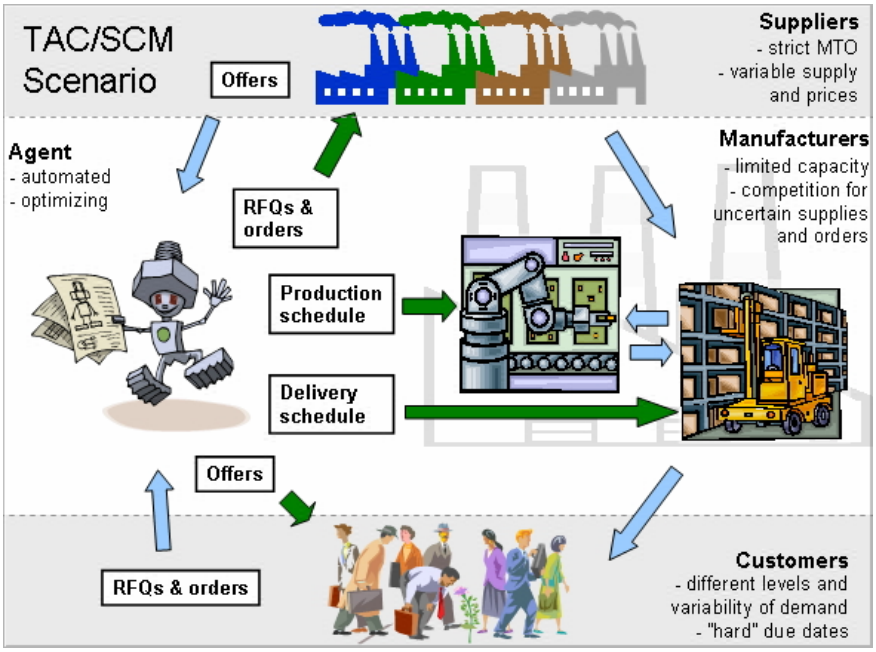


**Figure 1: TAC SCM Scenario [3]**

### 2.1 Supplies and Production

Sixteen different computer types can be built from four different components. These components are provided by a fixed number of suppliers. Each supplier behaves in a strict made to order manner, and they have a limited number of components that they can produce. This limitation follows a random walk, and the component price offered to the agent is dependent on the supplier's free capacity. The agent

must send a request for quote (RFQ) to a supplier that includes the component type, quantity, and due date. The supplier then responds with an offer price to fulfill the RFQ if possible, and the agent must choose to accept or reject the offer. Rejecting too many supplier offers can cause the supplier to lose trust in the agent, meaning it will not respond favorably to future RFQs.

Manufacturers are responsible for producing computers from components in inventory and delivering them to customers on time. The manufacturers are limited by a fixed number of manufacturing cycles within each simulation day and the number of components in inventory. Production and delivery schedules must be determined by the agent one day in advance.

### 2.2 Sales

Potential customers send an RFQ to every SCM agent indicating the type and quantity of computers needed, along with a due date, a penalty for late delivery, and a reserve price. Agents then respond to the customers' RFQs with offers and the customer accepts the lowest price offer if it is less than or equal to the reserve price in the RFQ.

There are many interdependencies among these different agent responsibilities. For example, enough components must be kept in stock in order to produce enough computers to fulfill customers' orders. Also, components must be in stock at the right time to fully utilize the production cycles. Environmental effects such as a change in customer demand or component availability should also affect computer production and pricing schemes.

## 3 Related Work

Our agent is built using the framework provided by the TacTex starter agent [2]. The TacTex starter agent is a stripped-down version of the TacTex agent [4], which won the 2005 and 2006 TAC SCM competitions. TacTex and the TacTex starter agent both split the problem of building an SCM agent into two tasks, supply management and demand management. TacTex uses a greedy scheduler to control computer production as well as determine the customer RFQs to make offers on. For component purchasing, the TacTex starter agent builds a model of supplier capacities based on RFQ probes (RFQs for zero quantity), which can be used to predict component prices. In addition to optimizing the production and component purchasing processes, TacTex is able to adapt to the set of opponents it is playing over a series of games.

Ideas from many other agents also went into the design of StormFront. SouthamptonSCM [5] focuses on the purchase of components, splitting the supply side into far and near future component procurement. Southampton projects component usage by predicting the customer demand for each computer type over the next 35 days. If the expected inventory level over the 35 days falls below a set minimum threshold, Southampton will order enough computers to maintain its minimum threshold.

DeepMaize [6] is another agent that we used in our work. DeepMaize utilizes a customer demand predictor that is based on the specifications of the game. It is able to predict the number of customer RFQs for each demand segment for any future day to optimal accuracy. The DeepMaize customer demand predictor is freely available on the web from the SCM agent repository [7] and is used in our agent.

Kephart, Hanson, and Greenwald [8] show the effects of pricing in the domain of online pricebots. The most successful agent in their experiments is MY, a myopically optimal agent that sets is prices to maximize profits in the short term. This agent undercuts its opponents by a minimal amount, winning 80% of the market share. In this domain, undercutting the opponent prices is the optimal strategy until lowering the sales price any further becomes unprofitable. The authors also show that a population of agents all using the MY strategy would continually lower their prices to undercut the other agents, resulting in an escalating price war. The MY agent that is able to reprice the fastest (thus keeping its price lower than its competitors) will win the most profit.

## 4 Agent Architecture

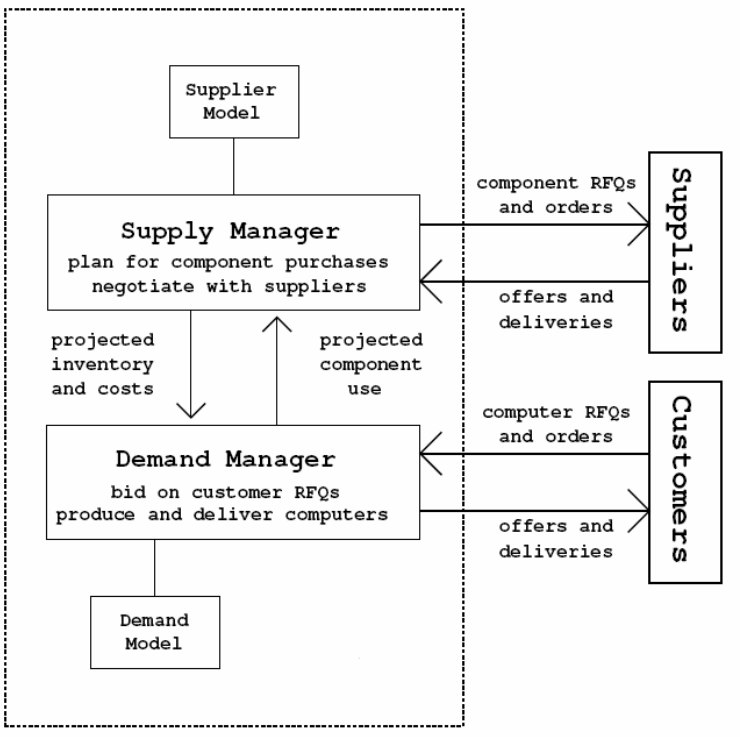


**Figure 2: Agent Overview [4]**

StormFront's architecture is based on the architecture of the TacTex starter agent [2]. Figure 2 shows an overview of the agent architecture. The agent must perform the following tasks: select which customer RFQs to bid on and how much to bid, schedule the production of computers for customer RFQs, schedule the delivery of computers for customer RFQs, send RFQs to suppliers for required components, and decide which supplier RFQ offers to accept. Our agent breaks these tasks into two main modules: the supply manager and the demand manager. The supply manager handles all component related tasks, including sending RFQs to suppliers for components and deciding which supplier RFQs to accept. The demand manager handles all tasks including computer sales and production. It determines which customer RFQs to make offers on and what price to offer for each one. In addition, it must determine the schedules for computer production and delivery. The supply and demand manager interact in two ways: the demand manager must provide the supply manager with projections of future component use based on its current production schedule and the supply manager must provide the demand manager with projections of future component deliveries so the demand manager knows which customer RFQs it will be able to build.

### 4.1 Supply Manager

The goal of the Supply Manager is to obtain the components required for computer production at the lowest possible prices. The agent must buy components at low cost while maintaining enough components in inventory for the demand manager to build computers for all of its orders. The supply manager also tries to keep the component inventory levels at a minimum to avoid excessive storage costs. The Supply Manager's task is divided into two main parts: *projecting* what components will be needed in the coming days, and *purchasing* these components at the lowest possible prices.

#### 4.1.1 Component Projection

The projection of future component needs is very similar to the approach taken by Southampton [5]. The projection is based on the components needed for currently scheduled production and the components required for anticipated future RFQs. The components required for currently scheduled production can be obtained from the production scheduler and the expected deliveries are determined from current outstanding orders. The components required for anticipated future RFQs are determined using the method described below.

Customer demand in the SCM game is split into three segments: low, medium, and high. The number of customer RFQs in each market segment is taken from a Poisson distribution around a target number of RFQs for that day. The target number of RFQs is determined for each market segment via a random walk [3]. The DeepMaize customer demand predictor utilizes this information to predict future RFQs for each market segment [6]. StormFront uses the predictions of the number of RFQs from the DeepMaize customer demand predictor when making its projection of future component use.

StormFront's use of the predicted number of customer RFQs is different from the way the predictions are used in Southampton. StormFront develops a model predicting the percentage of each RFQ type the agent will win. For each computer type, StormFront calculates the percentage of RFQs the agent won from the total number received for that type over the last four days. The agent assumes it will win a similar percentage of RFQs in the future. In this way, StormFront is able to adapt to different game situations, projecting less computer production in more competitive markets when it has been winning few RFQs and more production when it has been winning many RFQs.

Using the number of RFQs for each computer type projected by DeepMaize and the acceptance rate predicted by StormFront, the agent can estimate the number of each computer type it will need to produce for each future day. StormFront can then project the components needed for each of these computers to be produced.

In very competitive games, it may not be profitable to sell some types of computers. It is important for the agent not to purchase components that cannot be utilized for a profit. To prevent this from occurring, StormFront does not project any future RFQs being accepted for unprofitable computers. This reduces the number of unprofitable components the agent will purchase.

#### 4.1.2 Component Purchasing

The component projection module provides the component purchasing module with the number of components required for each future day. The number of components currently in stock plus any components expected in pending deliveries minus the number projected for use provides the number of components that the agent needs for any given day. In addition the required components, the supply manager seeks to maintain a minimum threshold of components. Since there is assumed to be some error in the projections of component use, this buffer is reduced when looking more days ahead. The minimum buffer is set to 500 of each component (250 for CPUs) when looking five days ahead and is reduced linearly to zero when looking 25 days ahead. This buffer is also reduced linearly to 0 between days 205 and 215 to reduce the stock of components at the end of the game.

The supply manager purchases components up to 25 days ahead of when they are required. It has to determine how many components to purchase and when they are needed. The agent looks at the projected component requirements for the next 25 days in 5 day increments (5, 10, 15, 20, and 25 days ahead). If the number of components in stock is projected to fall below the minimum threshold for that day, the agent attempts to purchase components to arrive by that day to maintain the threshold. The agent determines how to make the orders to obtain the components for the lowest cost by using price predictions from its supplier model.

A model of each supplier is maintained using the supplier model included in the TacTex starter agent. The costs of components from each supplier are based on the amount of free capacity the supplier has before the due date of the order. Using the offer prices received from RFQs, the free and committed capacity of each supplier can be estimated. This capacity can then be used to predict the sales prices that would be offered for any quantity and due date. On each day, five RFQs are allowed to each

supplier. The agent sends up to five RFQ probes (zero quantity RFQs) each day to each supplier to keep its model of supplier capacity up to date. When submitting RFQs for actual orders, a reserve price of 110% of the estimated price is set to ensure that the resulting price is not far off from what was expected. All offers received on RFQs are accepted in order to maintain a perfect reputation with each supplier.

If there are components required at some future day, it is the supply manager's job to acquire the needed components at the lowest possible cost. The supply manager uses price estimates from the supplier model to determine the best supplier and lead time to purchase the components. If it is not cheapest to purchase the components now, then the supply manager waits. For example, if components are required in 20 days, but the cheapest orders have a lead time of 5 days, the agent will wait until 5 days before the components are due to place the order. This assumes that the optimal lead times for orders will be similar in 15 days. This delay also allows the agent obtain better information on the components it will need.

Since supplier prices are determined by the amount of free capacity they have, the quantity being ordered has a significant effect on the price received. Reducing the size of the order being placed can drastically reduce the price offered. For this reason, the supply manager checks to see if splitting the order between two suppliers would result in a lower cost. It is usually much cheaper to split component orders between two suppliers when possible.

### 4.2 Demand Manager

The demand manager has two main tasks: dealing with customer RFQs by responding with profitable offers, and scheduling the production and delivery of computers. The demand manager sets sales prices for each customer RFQ received and uses these prices to calculate the profit for each RFQ. It then makes offers on the most profitable set of RFQs that it can produce. Computers are produced in such a way as to guarantee production of the most profitable orders and to keep the stock of computers low. The two main components of the demand manager are described below.

#### 4.2.1 Customer RFQs

The agent responds to customer RFQs with an offer to satisfy the RFQ for a particular cost. This cost is based primarily on the cost at which the identical computer type sold for on the previous day. If such information is unavailable, the computer price in the offer is based on the cost of the components that are used to create the computer. StormFront uses the previous day's lowest sales price and then subtracts one dollar when making offer prices. The intent of this strategy is to undercut the competition and therefore generate more revenue by selling more computers. Over the course of one game (219 days) the selling price of each computer type would then fall. Because the game is limited in duration, we hypothesize that StormFront will still be able to make a profit even though sales prices are low. The direct goal is not to hurt other agents, but simply to keep our factory utilization at full capacity and sell a

large volume of computers at a small profit margin. As shown by Kephart et. al [7], undercutting the competition by a minimal amount can help capture up to 80% of the market share. This would leave other agents with more profitable pricing schemes only a small market in which to compete. The success of this pricing strategy against others requires that the customer demand is fixed, and that our manufacturing capacity is a substantial portion of that customer demand.

The demand manager uses the sales price along with the cost of components for the RFQs to determine the profit for each RFQ. The component costs are determined by getting component price estimates from the supplier model in the supply manager. The agent iterates through the RFQs in order of profit and checks to see if each one can be produced. If it is possible to produce the RFQ or take the required computers from current inventory, an offer is made on the RFQ. In this way, the agent makes offers on the most profitable RFQs and ensures that it should be able to produce all the computers that are ordered.

#### 4.2.2 Production Scheduler

Customer RFQs come with a due date of 3 to 12 days after the RFQ was received. Since the production and delivery of the computer each take one day, computers for an order can be produced at most 10 days from the day it is received. For this reason, the next ten days of production are always scheduled each day.

StormFront schedules computer production based on profitability. The most profitable orders are scheduled first, preventing them from being delivered late or missed entirely. In addition to sorting by profitability, orders that need to be produced immediately to be delivered on time are placed at the front of the production queue. This attempts to avoid late fees while maintaining the benefits of sorting by profitability.

When producing a computer, StormFront tries to keep its surplus of computers low by taking computers from inventory if possible. When asked to produce a computer, the production scheduler first checks if there are more than 100 surplus computers of that type in stock. If there are excess computers in stock, then the order will be taken directly from inventory, otherwise the production scheduler will attempt to produce the computer at the latest possible day. If it is not possible to produce the computer, then the scheduler will once again look to take the computer from inventory, reducing the number of remaining computers below the 100 computer buffer level. The 100 computer buffer is decreased linearly to 0 between days 205 and 215 to reduce the number of computers left in stock at the end of the game.

Computers are always delivered to customers on the day that their RFQ is due if possible. If there are not any computers available, the computers may be delivered up to 4 days late with a penalty. Our production scheduler attempts to prevent late deliveries by prioritizing deliveries that are due. Computers are never delivered before their due date in case the computers are needed for a new RFQ that comes in with an earlier due date.

## 5 Experimental Results

The first experiments run with StormFront placed our agent against the TAC Starter agent in the presence of four other real agents from the repository (TacTex06, Maxon06, Mertacor05, and PhantAgent06)[7]. This is a reasonable benchmark to show that our changes improve upon the starter agent because they are both competing in an environment with more sophisticated agents. The difficult competition from the most successful agents keeps the profits low, but Figure 3 shows that StormFront earns an average of 7.5 million dollars more profit than the starter agent. A paired t-test for means shows that StormFront earns a higher net profit that the starter agent with 99.9% confidence (P-Value = 0.0008).

One of the main areas that our agent design focused on is in procuring components at the lowest possible cost. It does this by projecting component usage through the use of models of customer demand and the percentage of RFQs it has been winning. The agent then uses supplier models to predict the best suppliers and lead times to purchase the required components cheaply. Figure 3 also shows the total expenditure on materials (components plus storage) for StormFront versus the starter agent. The starter agent spends on average 20.7 million dollars more than StormFront on materials. A paired t-test for means shows that StormFront spends less on materials with 99% confidence (P-Value = 0.001).

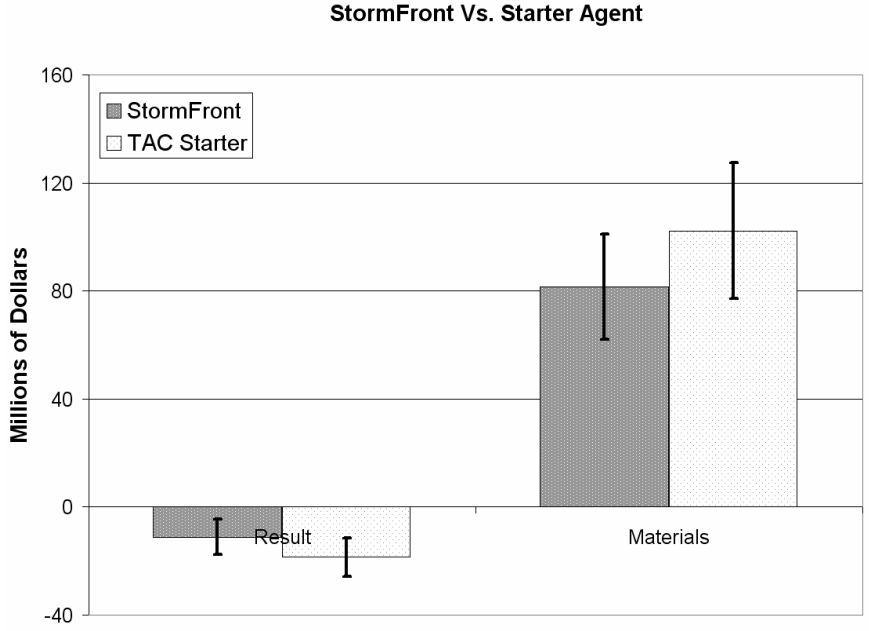


**Figure 3: Net profit and materials expenditure**

We tested the StormFront agent with these supply manager modifications against an identical agent with the supply manager modifications disabled in a series of

games with four successful agents from previous TAC SCM competitions. The four agents we tested with were TacTex06, Maxon06, Mertacor05, and PhantAgent06. This was done to show that the materials savings is a direct consequence of the supply manager changes, not an artifact of something else in the StormFront agent. The agent with the supply manger included earned an average of three million dollars more than the agent without it and spent an average of 21.3 million dollars less on components than the other agent. Using a paired two sample t-test for means, these results show that StormFront with the supply manager gains more net profit than the agent without the supply manager with 98% confidence (P-Value = 0.016). Also, using the supply manager causes a drop in materials expenditure (purchasing & storage) with 99.9% confidence (P-Value = 0.000028).

One of the other changes we made was to set the sales prices to be the previous day's low price minus one dollar. This change made our agent win many more RFQs than other agents by undercutting their prices. We theorized that since the other class agents are based on the TacTex starter agent, testing against a population of these agents would provide a good test of how well our agent would perform in the class. In tests between an agent with price undercutting, an agent with the default pricing scheme, and four TacTex starter agents, the agent with price undercutting won 7733 RFQs while the next highest agent only won 3973 RFQs. Although the modified agent was selling its computers for less money than the other agents, it was only a dollar less, so the agent made nearly double the revenue of the other agents. Overall, it finished with a loss of 21,168 dollars, the agent with default pricing lost 12.1 million dollars, and the starter agents lost between 70.6 and 77.3 million dollars each.

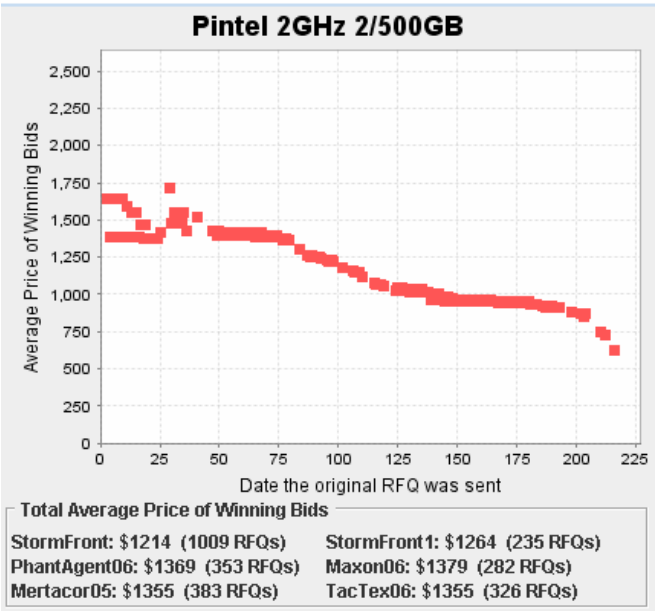


**Figure 4: StormFront winning bid price [9]**

Figure 4 shows StormFront's winning bid prices in a typical game [9]. The clear downward trend is expected because the sale price is fixed to the previous day's low sale price minus one dollar. When competing against real agents (including TacTex) this strategy is capable of capturing nearly 3 times the market share of each of the other agents.

Another aspect of the agent that we tested was the modification to project component use for only profitable computers. The goal of this change was to prevent the agent from producing and selling unprofitable computers. In a test game with the agent with the modifications, the agent without them and the same four TAC SCM agents as in previous tests, the agent with the modifications lost 6.6 million dollars during the game while the agent without them lost 13.4 million dollars. This is a difference of 6.8 million dollars. In addition, the agent with the change had a margin of -4% while the unmodified agent had a margin of -13%. This means that the modified agent was selling its computers for much higher profit that the unmodified agent.

## 6 Competition Results

The class SCM competition involved 16 games between the same six agents. Five agents were submitted from other class members and the sixth agent was a "mystery agent" selected from the available binaries in the agent repository on the SCM website. The average results over all sixteen rounds are shown in Table 1. StormFront came in fifth in the tournament, losing an average of nine thousand dollars per game. StormFront did make more in revenue than all but one other agent, but also spent the most on components.

**Table 1: Average Results (in millions of dollars)**

| Agent | Revenue | Interest | Material Costs | Storage Costs | Penalty Costs | Result |
|---|---|---|---|---|---|---|
| redbull | 101.454 | -0.040 | 83.29 | 1.819 | 0.629 | 15.670 |
| Simplicity | 114.003 | -0.201 | 96.18 | 3.703 | 1.009 | 12.903 |
| garfield | 66.829 | -0.240 | 58.97 | 2.328 | 1.287 | 4.004 |
| MysteryAgent | 73.720 | -0.382 | 67.83 | 1.178 | 3.040 | 1.291 |
| StormFront | 102.012 | -0.259 | 98.535 | 1.557 | 1.670 | -0.009 |
| JAgent | 27.881 | -0.048 | 27.14 | 0.313 | 1.130 | -0.756 |

Table 2 shows the total number of orders that each agent was able to deliver during the sixteen games of the competition. The average component cost, storage cost, revenue, and profit per order are also shown. The positive effects of StormFront's supply manager can be seen in Table 2. Although StormFront was producing far more computers than any of its competitors, it was able to maintain one of the lowest costs per order, spending only 12,932 dollars per order, over four thousand dollars per order better than the winning agent. Although StormFront had to maintain enough compo-

nents in stock to build a huge number of computers, it was able to attain lower storage costs than nearly any other agent. The only agent that achieved a lower storage cost per order than StormFront was JAgent, but it was also producing a third as many orders as StormFront.

**Table 2: Statistics Per Order**

| Agent | Delivered Orders | Revenue Per Order | Cost Per Order | Storage Cost Per Order | Profit Per Order |
|---|---|---|---|---|---|
| redbull | 78129 | 20777 | 17058 | 373 | 3719 |
| Simplicity | 97092 | 18787 | 15851 | 610 | 2936 |
| garfield | 53943 | 19822 | 17491 | 690 | 2331 |
| MysteryAgent | 86129 | 13695 | 12601 | 219 | 1094 |
| StormFront | 121910 | 13388 | 12932 | 204 | 456 |
| JAgent | 43131 | 10343 | 10071 | 116 | 272 |

The results in Table 2 are also very indicative of the effects of the undercutting StormFront did in selling computers. StormFront's price undercutting allowed it to win many more orders than the other agents. StormFront's 121,910 orders delivered accounted for over 25% of all deliveries made during the competition. StormFront was selling its computers for the second smallest amount in the game however, preventing it from making much profit. StormFront was only able to make an average of 456 dollars per order, much less than the 3,719 dollars per order that the winning agent, redbull, won during the competition.

To show the effects of variations in the level of customer demand on StormFront's pricing scheme, we examine game 226 from the competition. In game 226, there was low customer demand for low-end computers (98,152 computers requested) and high customer demand for mid-range computers (148,649 computers requested). Figure 5 (created by the CMieux viewing tool [9]) shows the sales prices for a low-end and mid-range computer during this game. For the low-end computer, nearly all the computers were sold at the low price that StormFront was offering. This prevented other agents from selling computers at higher prices. With the higher customer demand for mid-range computers, there was enough demand to buy all the computers offered at StormFront's low prices as well as many more computers at the higher prices offered by other agents. Thus StormFront's sales strategy was most successful when there was low customer demand and it was able to prevent any computers from being sold at higher prices.

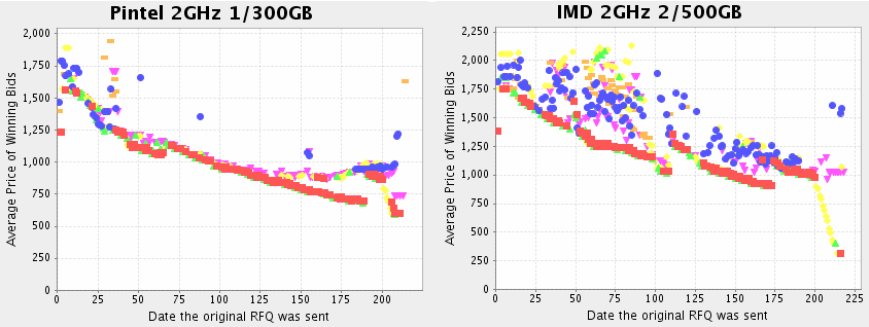


**Figure 5: Sales Prices of low-end (left) and mid-range (right) computers (StormFront in red) [9]**

## 7 Discussion

All the agents in the class competition were based on the TacTex starter agent. Every team changed some parts of the starter agent, but there remained many similarities between the final agents in the competition.

One particular area where the effects of the similarity between agents can be seen is in the sales prices of computers. The default setting of the starter agent was to sell computers at 75% of a default computer price the first day. On subsequent days, agents would sell computers for the lowest price that was accepted on the previous day.

The knowledge of the schemes other agents would likely use to set their sales prices provided a great opportunity. One key strategy of an agent in the SCM scenario is to barely undercut your opponent's sales prices, allowing you to sell many more computers for slightly less than your opponents. Knowing that many other agents would sell their computers for yesterday's low price, our agent sold its computers for one dollar less than that price. This allowed our agent to undercut the other agents by exactly one dollar, the minimum amount we could undercut them by. This simple change of subtracting one dollar from the starter agent's sales prices allowed us to win more than 25% of all RFQs in the competition. Since our orders were selling for such low prices, the high number of orders our agent received did not result in a lot of profit. The agents that sold many fewer computers at much high prices were much more successful in the competition than StormFront was. Selling computers for the lowest price is only effective when there is low customer demand. In these cases, StormFront is able to win nearly all the orders at the low price and prevent other agents from selling their computers for higher prices. In markets with high demand, StormFront is unable to win enough of the orders to prevent other agents from selling computers at higher prices

A key aspect of our agent is its supply manager. The agent projects component usage and uses this information to procure components ahead of time for the lowest

possible price. The component projection using the DeepMaize customer demand model allows us to purchase components ahead of many other agents and for a lower price. During testing, our agent paid 21.3 million dollars less for components than the same agent without our supply manager implementation. In the competition, our agent paid one of the lowest component costs per order even though it was producing many more computers than any other agent. The agent was able to maintain enough components for the demand manager to produce all computers for all of its orders, keeping the factory active 96% of the time. It also obtained its other goal of keeping component stock and storage costs low, having the second lowest component cost of any agent in the game.

In very competitive games, it is often the case that it is not profitable to produce and sell certain types of computers. Purchasing components for and producing these unprofitable computers can have a large negative effect on the final revenues of the agent. Our agent tackles this problem from the supply side by not projecting any future RFQs to be accepted for unprofitable computers. Then the agent does not purchase any components for the production of these computers. The agent also could have been stopped from selling unprofitable computers in the demand manager by not making any offers on computers that would not be profitable. This is not a good solution to this problem because once the agent is already in possession of the components or computers, it is better off selling them than keeping them in stock. At the end of the game, components and computers still in stock provide no value to the agent. For this reason, it is best to cut off the sale of unprofitable computers from the supply side, never buying the components of unprofitable computers in the first place. In testing, this change resulted in a 7 million dollar improvement over an agent that projected component use for all computers whether they were profitable or not.

Overall, StormFront was very good at high-volume low-profit production. It was able to win a large share of the RFQs and produce a massive number of computers efficiently. At the same time, it was able to purchase components at a low price and maintain a low but sufficient inventory of components. StormFront also did a very good job of reducing its inventory of computers and components at the end of the game. If StormFront's computer sales prices were increased so as to increase its profit per computer, its existing abilities to buy components cheaply could make it a very successful agent.

## Contributions

Todd Hester and David DeAngelis worked together to develop general agent strategies and decide which areas to focus on. Todd worked on the component modules as well as the production schemes while David worked on the production scheduling and computer pricing. In addition, we worked together on writing the paper and collecting empirical results.

## References


[1] R. Das, J. Hanson, J. Kephart, and G. Tesauro. "Agent-Human Interactions in the Continuous Double Auction," In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, Seattle, USA August 2001.

[2] David Pardoe. "TAC SCM Starter Agent," accessed October 15th, 2006. http://www.cs.utexas.edu/~TacTex/starterAgent/

[3] John Collins, Raghu Arunachalam, Norman Sadeh, Joakim Eriksson, Niclas Finne, Sverker Janson. "The Supply Chain Management Game for the 2006 Trading Agent Competition," Technical Report CMU-ISRI-05-132, School of Computer Science, Carnegie Mellon University.

[4] David Pardoe, Peter Stone, and Mark VanMiddlesworth. "TacTex-05: An Adaptive Agent for TAC SCM," In *AAMAS 2006 Joint workshop on Trading Agent Design and Analysis & Agent Mediated Electronic Commerce VIII*, Hakodate, Japan, May 2006.

[5] Minghua He, Alex Rogers, Xudong Luo, and Nicholas R. Jennings. "Designing a Successful Trading Agent for Supply Chain Management," *Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1159-1166. Hakodate, Japan, 2006.

[6] Christopher Kiekintveld, Jason Miller, Patrick Jordan, and Michael P. Wellman. "Controlling a Supply Chain Agent Using Value-based Decomposition," *Seventh ACM Conference on Electronic Commerce*, Ann Arbor, MI, 2006.

[7] Jeffrey O. Kephart, James E. Hanson, and Amy R. Greenwald. "Dynamic Pricing By Software Agents," *Computer Networks*, Amsterdam, NL, 2000.

[8] Benisch et. al. 2005. "CMieux analysis and instrumentation toolkit for TAC SCM," Technical Report CMU-ISRI-05-127, School of Computer Science, Carnegie Mellon University.

[9] Swedish Institute of Computer Science (SICS). "Trading Agent Competition – TAC Agent Repository," accessed December 6th, 2006. http://www.sics.se/tac/showagents.php