6 A Aplicação LearnAgentsSCM

Supply Chain Management (SCM) é o planejamento e coordenação das atividades de uma cadeia de suprimentos (Chopra et al., 2004). Essas atividades podem ter vários participantes e organizações, e a coordenação desses participantes é um ponto chave para uma cadeia eficiente. O *Supply Chain* não está relacionado apenas com os produtores e fornecedores, mas também com transportadoras, centros de distribuição, e clientes. Essa cadeia é extremamente dinâmica e envolve um grande fluxo de informação, produtos e recursos financeiros entre diferentes estágios.

O TAC *Supply Chain Management* (Arunachalam et al., 2004) foi modelado para capturar a complexidade de uma cadeia de suprimentos dinâmica, porém com regras simples para que muitas equipes possam participar dessa competição. O jogo foi desenvolvido por pesquisadores do laboratório de *e-Supply Chain Management* da *Carnegie Mellon University* e *Swedish Institute of Computer Science (SICS).*

O *LearnAgentsSCM* é um sistema multi-agente para a gestão automatizada de uma cadeia de suprimentos. O sistema é utilizado em ambientes competitivos onde os participantes competem por clientes, um número limitado de peças, e precisam gerenciar a produção com uma limitação na capacidade produtiva. A sua arquitetura define agentes que resolvem subproblemas de gestão da cadeia de suprimentos, e apresenta uma técnica para combinar todas essas soluções. Os subproblemas típicos são seleção de clientes, planejamento da produção, negociação de peças para o estoque, entre outros. O *LearnAgentsSCM* utilizou o ambiente do TAC para testar o sistema desenvolvido, pois esse ambiente simula um ambiente competitivo e permite mensurar a performance do sistema contra um *benchmark* internacional.

6.1. O TAC Supply Chain Management

Seis agentes "produtores de PC" participam em cada jogo do TAC SCM. Esses participantes competem por clientes com incerteza na demanda e por peças de um numero limitado de fornecedores. Todo dia, os clientes enviam pedidos de cotações e selecionam as melhores ofertas baseadas no preço e na data de entrega. Os agentes são limitados pela capacidade da fabrica, e tem que gerenciar a compra de peças de oito fornecedores. Quatro componentes são precisos para montar um PC: CPU, Placa Mãe, Memória, e Disco Rígido. Cada componente está disponível em diferentes modelos. A demanda dos clientes pode ser medida pelo número de cotações de PCs. A figura 58 resume o cenário do jogo TAC SCM.

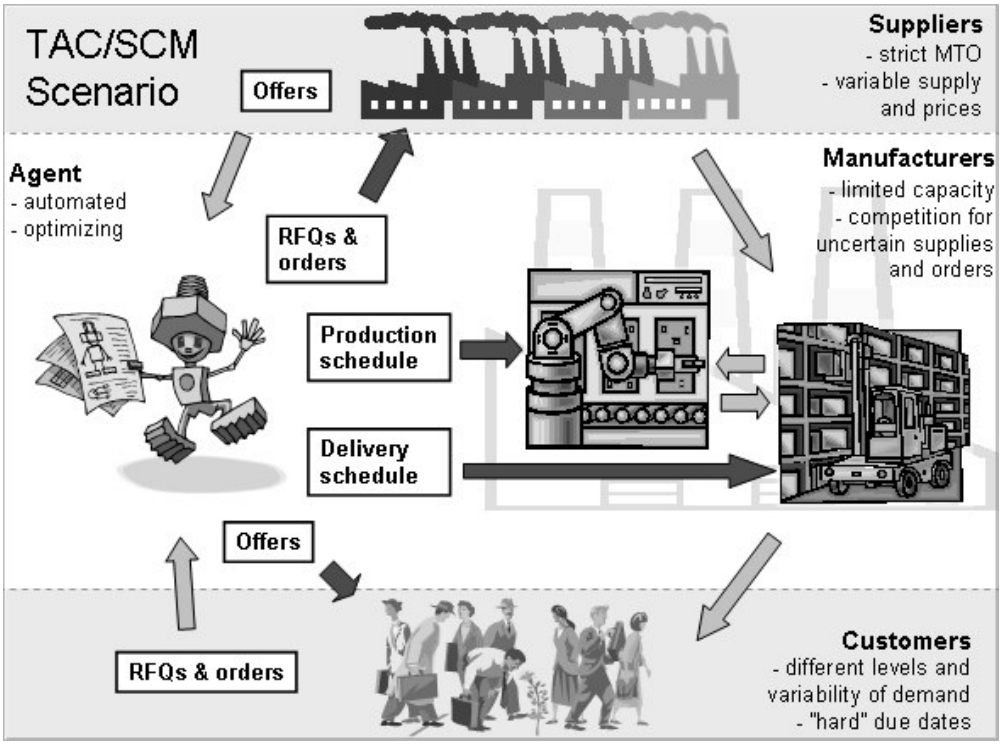


Figura 58: O Jogo TAC SCM.

O jogo começa quando um ou mais agentes se conectam ao servidor do TAC SCM. O servidor simula o comportamento dos fornecedores e clientes, e oferece um ambiente para cada participante com um banco para sua conta corrente, uma fábrica, e estoques de peças e PCs. No final do jogo, o agente que possuir o maior valor de dinheiro na conta corrente é declarado o vencedor.

O jogo apresenta um bom nível de complexidade, pois cada agente deve competir em vários mercados por diferentes componentes do lado dos fornecedores, e em mais outros mercados por diferentes tipos de PCs no lado dos clientes. Além disso, todos esses mercados possuem interdependências e incertezas. Esse cenário permite a criação de várias estratégias diferentes, e um agente precisa ser adaptativo para estar pronto para vários cenários imprevisíveis.

6.2. Evolução do TAC SCM

A primeira versão do ambiente do jogo utilizado no TAC SCM foi projetada em 2003 por um grupo do Laboratório de *e-Supply Chain Management* da *Carnegie Mellon University* e do *Swedish Institute of Computer Science (SICS)*. A versão do ano seguinte passou por mudanças significativas para que o ambiente simulasse um mercado mais competitivo.

Posição	Agente	Pontuação Média
1	RedAgent	11.61 M
2	deepmaize	9.473 M
3	TacTex	5.017 M
4	Botticelli	3.330 M
5	PackaTAC	-1.680 M
6	whitebear	-3.453 M

Tabela 17: Resultado da Final do TAC SCM de 2003.

A primeira edição do TAC SCM foi sediada dentro da *Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)* em 2003. A tabela 17 apresenta os resultados desta edição. A média dos três primeiros concorrentes é de 8,7 milhões, o que representa a existência de soluções robustas para esse complexo cenário.

As estratégias dos agentes do TAC SCM podem ser classificadas em duas grandes categorias: (i) *buy-to-build*, onde o agente estoca componentes e começa a produzir PCs sem ter necessariamente ordens para programar a produção; e (ii) *build-to-order*, o agente espera receber primeiro as ordens dos clientes, e depois produz e entrega PCs baseado nessas ordens. O agente que adota a estratégia *buy-to-build* é mais agressivo e pode aceitar qualquer oportunidade que for altamente lucrativa. Porém, o agente corre o risco de ter estoque encalhado, e pagar custos

altos de estocagem. A estratégia *build-to-order* é mais conservadora, pois só aceita oportunidades que possam ser produzidos no futuro. Porém, se o cliente enviar um pedido de cotação altamente lucrativo para ser entregue imediatamente, o agente não pode aceitar esse tipo de oportunidade.

Em 2003, o vencedor *RedAgent* (Keller et al., 2004) utiliza um sistema multi-agente com várias heurísticas simples, e utiliza uma estratégia *buy-to-build* junto com o mecanismo de leilões para determinar a alocação ótima de recursos. Nessa estratégia, a primeira preocupação do produtor é estocar componentes e produzir PCs sem necessariamente ter ordens de clientes para cumprir. O *RedAgent* se beneficiava do fato de não haver um custo de estocagem nessa versão do jogo. Além disso, ele poderia vender seus PCs a qualquer momento, mesmo abaixo do custo, para se ver livre do estoque. Os agentes dentro do sistema *RedAgent* utilizam o mecanismo dos leilões para determinar as alocações dos recursos. Por exemplo, o *Assembler Agent* é um agente de produção e ele tem que negociar através de leilões o uso de matéria prima e ciclos de fábrica.

Posição	Agente	Pontuação Média
1	FreeAgent	10.28 M
2	Mr.UMBC	8.652 M
3	UMTac-04	6.518 M
4	Botticelli	441 711
5	deepmaize	-5.108 M
6	SouthamptonSCM	-10.41 M

Tabela 18: Resultado da Final do TAC SCM de 2004.

A segunda versão do TAC SCM, em julho de 2004, teve a sua realização dentro da *Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS'04)*, na cidade de Nova York. Algumas regras foram modificadas para aumentar a competitividade do jogo. Por exemplo, o custo de estocagem foi incluído nesta versão. A tabela 18 apresenta o resultado final da competição. Os agentes *FreeAgent*, *Mr.UMBC*, *UMTac-04* não apresentaram, até janeiro de 2005, qualquer relatório de pesquisa sobre as estratégias utilizadas. A média dos três primeiros concorrentes nesse ano é de 8,48 milhões. Três agentes novos ocuparam os três primeiros lugares em relação ao ano 2003, porém a média desses três se manteve quase constante.

Os criadores do agente *Botticelli* (Benisch et al., 2004), quarto lugar em 2003 e 2004, modelaram o problema de responder a RFQs dos clientes como um modelo estocástico. O agente utiliza algumas heurística e programação inteira para decidir que RFQs aceitar. A programação da produção é feita com um algoritmo guloso. Esse agente utiliza a estratégia mais próxima do *build-to-order*, pois produz PCs baseado nas ordens atuais e ofertas (RFQs) com alta probabilidade de se tornarem ordens.

O agente *Deep Maize* (Kiekintveld et al., 2004) aceita os RFQs baseado nos estoques de PCs, disponibilidade de peças e ociosidade na produção. Ele possui uma estratégia de sempre manter estoque de PCs prontos. A programação da produção é feita com um horizonte de três dias, e utiliza programação linear para escolher os PCs a serem produzidos. Esse agente utiliza a estratégia mais próxima do *buy-to-build*.

6.3. Arquitetura LearnAgentsSCM

O *LearnAgentsSCM* também utiliza a tecnologia de agentes, e cria entidades autônomas e inteligentes para atuarem em complexos ambientes distribuídos, competitivos e abertos. A técnica de modelar um problema computacionalmente difícil do *LearnAgents* foi utilizada também nesse novo domínio. O método de decomposição de subproblemas foi utilizado para diminuir a complexidade do problema. Assim, cada agente se torna responsável por um ou mais desses subproblemas.

A figura 59 apresenta a arquitetura do *LearnAgentsSCM*, e esta seção descreve a modelagem do sistema multi-agente usando uma linguagem de modelagem chamada ANote (Choren, 2002).

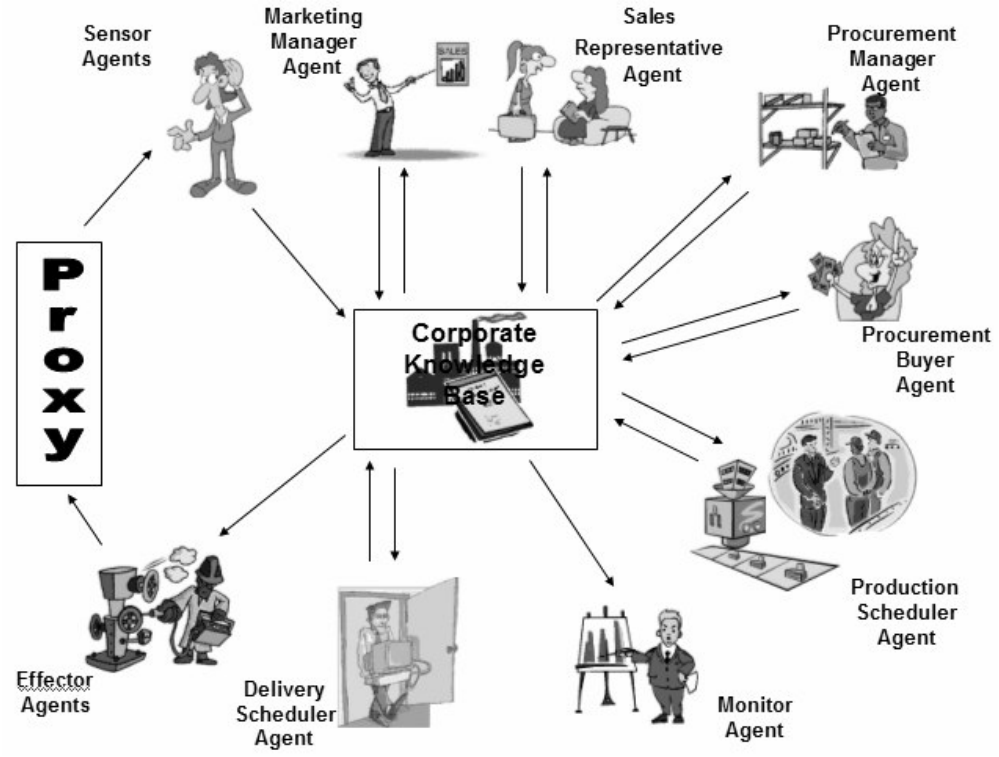


Figura 59: A Arquitetura do LearnAgentsSCM.

Esse projeto de software também utiliza a técnica orientada a objetivos para a fase de requisitos. Esta fase utiliza o processo recursivo de decomposição de um objetivo do problema principal em vários objetivos de subproblemas mais simples. No caso do *LearnAgentsSCM*, esse processo recursivo definiu todos os objetivos dos subproblemas relacionados com a produção, compra de matéria prima, e comercialização de bens.

O objetivo maior desse sistema é produzir PCs e obter o maior lucro possível. Essa produção envolve os seguintes processos: (i) gerenciar as vendas para o cliente, (ii) gerenciar a produção das ordens, (iii) gerenciar a compra de matéria prima, e (iv) gerenciar a entrega dos PCs. A figura 60 mostra esse processo de decomposição dos objetivos.

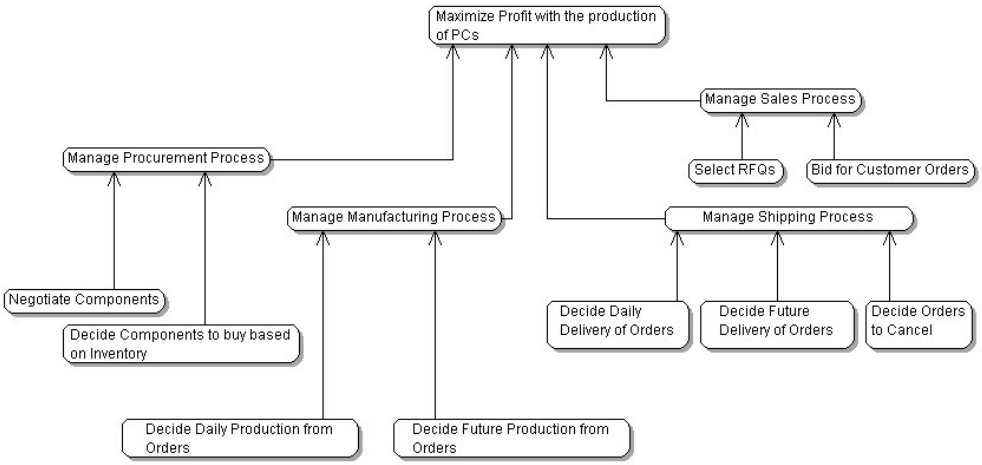


Figura 60: A decomposição dos objetivos relacionados com a produção, compra de matéria prima, e comercialização de bens.

O processo de vendas pode ser decomposto em dois objetivos: (i) selecionar os pedidos de cotações mais interessantes, por exemplo, em relação à lucratividade ou a oportunidade de explorar um novo mercado, e (ii) escolher o preço para as cotações selecionadas.

O processo de produção pode ser decomposto em: (i) escolher a melhor alocação das ordens a serem produzidas no dia corrente, e (ii) decidir a produção das ordens no futuro para poder calcular a ociosidade da fábrica.

O processo de entrega pode ser decomposto em três objetivos: (i) decidir a entrega diária, (ii) decidir a entrega no futuro para saber quais ordens serão canceladas, e (iii) decidir cancelar o menor número de ordens, caso seja inevitável.

A compra da matéria prima pode ser decomposta em dois objetivos: (i) decidir as peças a serem compradas escolhendo o tamanho ótimo do lote de compra, e tamanho ótimo do estoque de segurança, e (ii) negociar a compra das peças selecionadas e quantidades ordenadas.

A segunda fase do processo de modelagem é criar os tipos de agentes que sejam capazes de atingir os objetivos da figura 60. Essa criação começa com processo de relacionamento de todos os objetivos de nível mais baixo com os respectivos tipos de agentes. A tabela 19 apresenta um mapeamento entre esses objetivos e tipos de agentes. A figura 61 apresenta os vários tipos de agentes e

seus relacionamentos. Esta figura permite uma visão estática do projeto do nosso sistema multi-agente.

Goal	Agent
Select RFQs	Marketing Manager
Bid for Customer Orders	Sales Representative
Decide Daily Production from Orders	Production Scheduler
Decide Future Production from Orders	Production Scheduler
Decide Daily Delivery of Orders	Delivery Scheduler
Decide Future Delivery of Orders	Delivery Scheduler
Decide Orders to Cancel	Delivery Scheduler
Decide Components to buy based on Inventory	Procurement Manager
Negotiate Components	Procurement Buyer

Tabela 19: A mapeamento entre subproblemas e tipos de agentes.

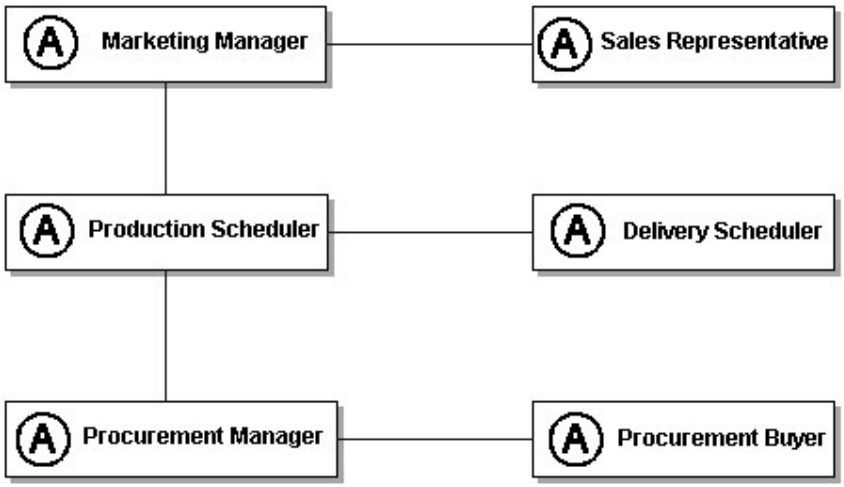


Figura 61: O Diagrama de Agentes do LearnAgentsSCM.

Os Sensores do ambiente não foram incluídos na figura 61 para manter a especificação mais simples e de fácil compreensão. Os objetivo dos Sensores nessa arquitetura são: (i) atualizar a base de conhecimento corporativa com

informações sobre clientes, RFQs, Ordens, informações sobre a fábrica, e informações sobre os fornecedores; e, (ii) enviar mensagens para outros agentes sobre eventos importantes do ambiente externo.

Os Efetuadores do ambiente também não foram incluídos na figura 61 pelo mesmo motivo dos sensores. O objetivo dos Efetuadores nessa arquitetura é enviar informações com respostas a RFQs, pedido de cotação aos fornecedores para o ambiente externo. Esse agente é puramente reativo, pois envia as informações ao ambiente externo apenas quando é solicitado por outro agente do sistema.

Pelo mesmo motivo dos Sensores e Efetuadores, o Monitor não foi incluído na figura 61. Esse agente é responsável por armazenar informações da base de conhecimento corporativa em um banco de dados. Esses dados são utilizados por uma ferramenta da CA (Computer Associates) chamada *Forest and Trees* para monitorar a performance de cada agente individualmente e o sistema multi-agente como um todo. Uma tela da ferramenta de monitoração pode ser vista na figura 62.

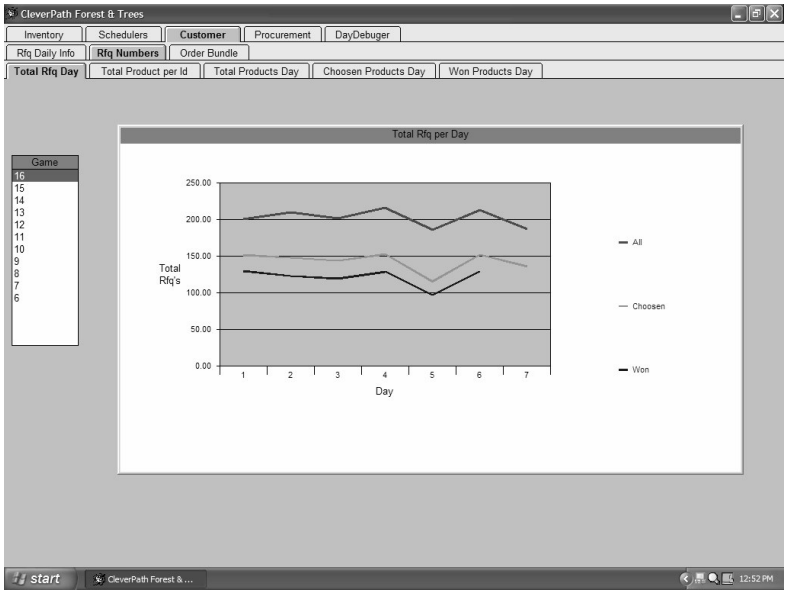


Figura 62: Interface do Monitor.

Após a criação dos vários tipos de agentes, cada agente deve possuir uma especificação de cenários. Esses cenários possuem informações sobre o contexto

de como cada agente deverá atingir seus objetivos. Segue abaixo, uma descrição resumida dos agentes criados e seus cenários:

Gerente de Marketing – Esse agente é responsável por selecionar quais RFQs, ou pedido por cotações dos clientes, serão respondidos. O Gerente de Marketing deve selecionar os RFQs lucrativos, e deve utilizar estratégias de segmentação de mercado para melhorar a performance da escolha. A figura 63 apresenta o digrama de cenário do Gerente de Marketing.

Select RFQs	
Main Agent	Marketing Manager
Preconditions	Message from Sensor
Main Action Plan	while true Read in CKB the RFQs received Select RFQs to send bids Write in CKB the list of RFQs selected Send message to Sales Representative end while
Interaction	Sales Representative
Variant Action Plan	

Figura 63: Cenário da Seleção dos Pedidos por Cotações.

Vendedor – Esse agente é responsável por definir o preço dos RFQs selecionados pelo Gerente de Marketing. O agente pode adotar diferentes estratégias de preços dependendo do mercado. A figura 64 apresenta o digrama de cenário do Vendedor.

Bid for Customer Orders	
Main Agent	Sales Representative
Preconditions	Message from Marketing Manager
Main Action Plan	while true Read in CKB the RFQs received Read in CKB the list of RFQs selected Decide bids for each selected RFQs Write in CKB the bids for each selected RFQs Send message to Effector end while
Interaction	Effector
Variant Action Plan	

Figura 64: Cenário da Decisão dos Lances.

Programador da Produção – Este agente decide a programação da produção do dia corrente para todas as ordens dos clientes. O Gerente da Produção também calcula a programação da produção para os próximos dias para que possa ser determinada a ociosidade da fábrica. A figura 65 apresenta o digrama de cenário do Gerente de Produção.

Decide Daily and Future Production from Orders	
Main Agent	Production Scheduler
Preconditions	Message from Sensor
Main Action Plan	while true Read in CKB the list of Active Orders and Inventory Decide Orders to Produce (current day) Decide Orders to Produce (future) Write in CKB the production schedule Write in CKB the production schedule (future) Send message to Effector end while
Interaction	Effector
Variant Action Plan	

Figura 65: Cenário da Programação da Produção.

Programador da Entrega – Este agente é responsável por decidir quais ordens serão entregues no dia corrente e no futuro. A decisão das ordens entregues no futuro permite a decisão das ordens a serem canceladas. A figura 66 apresenta o digrama de cenário do Gerente de Entrega.

Decide Delivery of Orders and Orders to Cancel	
Main Agent	Delivery Scheduler
Preconditions	Message from Sensor
Main Action Plan	while true Read in CKB the list of Orders Decide Orders to Deliver (completed orders) Decide Orders to Deliver in the future Decide Orders to Cancel Write in CKB the delivery schedule Write in CKB the delivery schedule (future) Send message to Effector end while
Interaction	Effector
Variant Action Plan	

Figura 66: Cenário da Programação da Entrega e Ordens a Cancelar.

Gerente de Compras – Este agente decide quais peças devem ser compradas. Este decisão dever ser baseada no nível de estoque atual. A figura 67 apresenta o digrama de cenário do Gerente de Entrega.

Decide Components to Buy	
Main Agent	Procurement Manager
Preconditions	Message from Sensor
Main Action Plan	while true Read in CKB the current inventory Decide components to buy Write in CKB the components to buy Send message to Procurement Buyer end while
Interaction	Procurement Buyer
Variant Action Plan	

Figura 67: Cenário da Decisão de Compra dos Componentes.

Comprador – Este agente negocia com os fornecedores as pecas definidas pelo Gerente de Compras. Esta decisão se baseia no preço mais baixo e fidelidade dos fornecedores em relação à entrega. A figura 68 apresenta o digrama de cenário do Gerente de Entrega.

Negotiate Components - Part I	
Main Agent	Procurement Buyer
Preconditions	Message from Procurement Manager
Main Action Plan	while true Read in CKB the components to buy Select Suppliers to Request Quotes Write in CKB the RFQs to send to Suppliers Send message to Effector end while
Interaction	Effector
Variant Action Plan	

Negotiate Components - Part II	
Main Agent	Procurement Buyer
Preconditions	Message from Sensor
Main Action Plan	while true Read in CKB the Offers from Suppliers Decide Suppliers based on Offers received Write in CKB the Offers to accept Send message to Effector end while
Interaction	Effector
Variant Action Plan	

Figura 68: Cenário da Negociação dos Componentes com os Fornecedores.

Um diagrama interessante que deriva dos diagramas de cenário é o diagrama de interação. Esse diagrama deixa explícita a comunicação entre os agentes, e permite uma visão dinâmica do sistema. A figura 69 apresenta o processo de vendas, e a interação entre o Gerente de Marketing e o Vendedor.

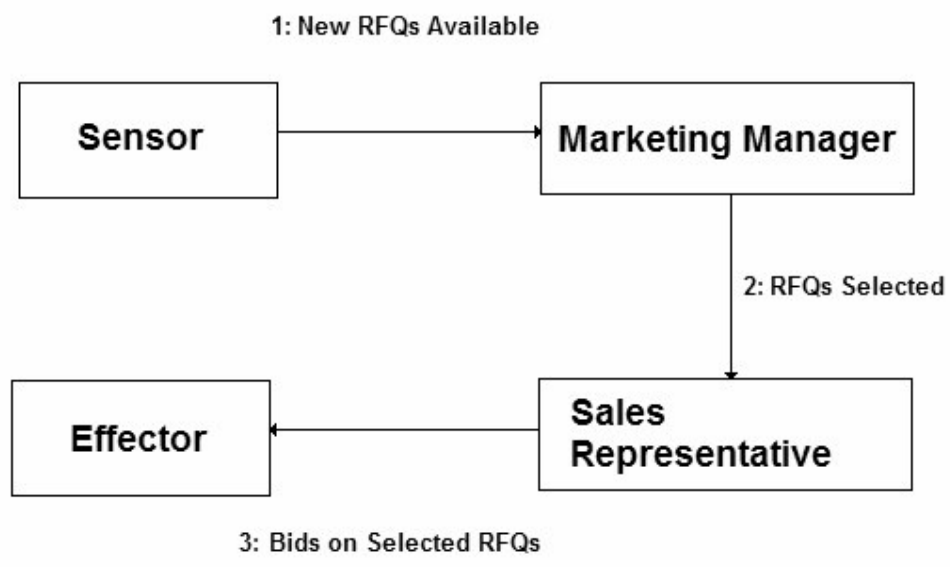


Figura 69: Diagrama de Interação do processo de vendas.

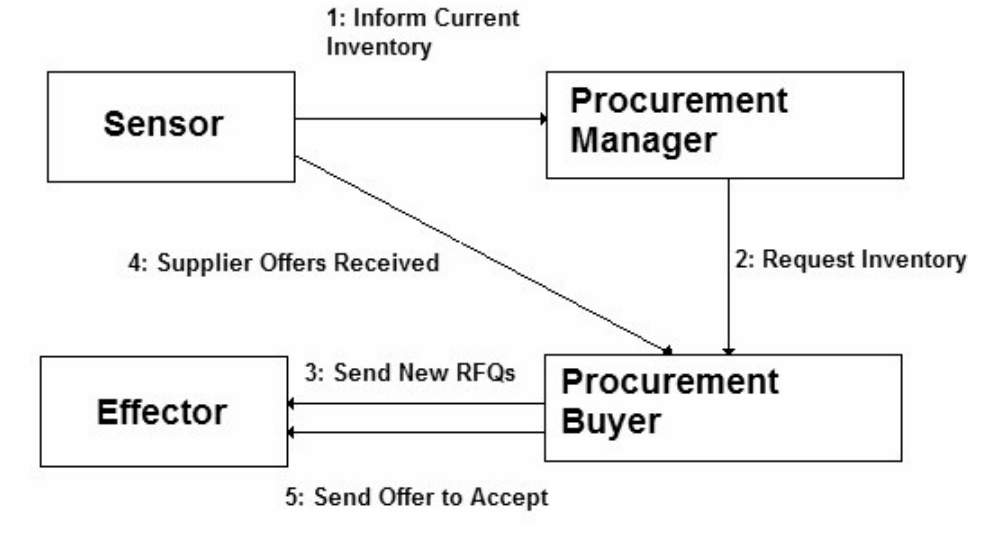


Figura 70: Diagrama de Interação do Processo de Compra de Matéria Prima.

A figura 70 apresenta o processo de compra da matéria prima. O Gerente de Compras decide quais peças serão compradas, e envia para o Comprador pedidos

de compra. O comprador negocia a compra das peças com os fornecedores disponíveis. O processo de produção e entrega está presente na figura 71 e 72.

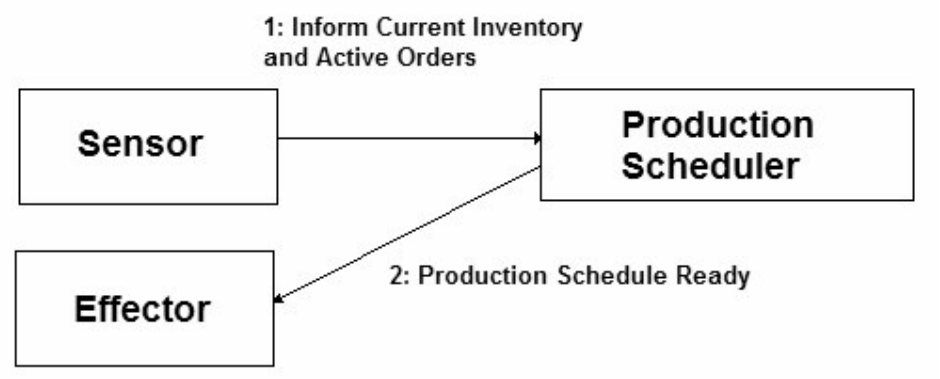


Figura 71: Diagrama de Interação do Processo de Produção.

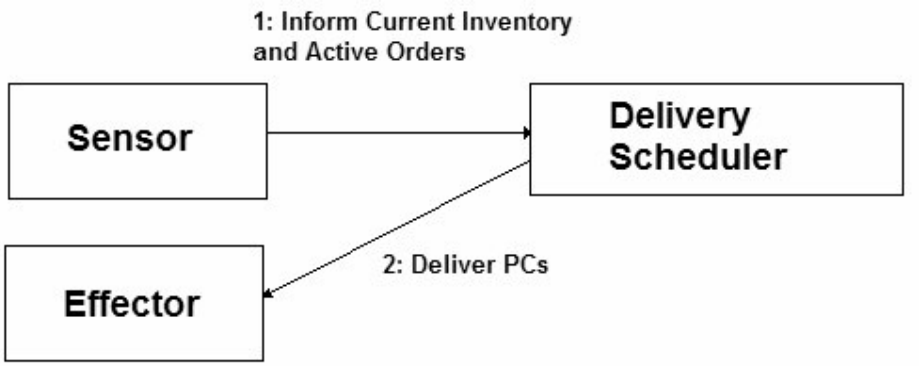


Figura 72: Diagrama de Interação do Processo de Entrega.

6.4.A Engenharia de Algoritmos dos Agentes do LearnAgentsSCM

Esta seção apresenta os principais algoritmos utilizados no Gerente de Compras, Programador da Produção, Programador da Entrega, e Gerente de Marketing. O restante dos agentes utiliza uma lógica simples que são muito parecidas com o código apresentado nos cenários da seção 6.3.

6.4.1.Gerente de Compras

O Gerente de Compra utiliza o conceito de comprar matéria prima em lotes (Horngren et al., 1997; Chopra et al., 2004) para satisfazer a demanda por peças. A figura 73 mostra o gráfico da demanda por peça para uma demanda constante.

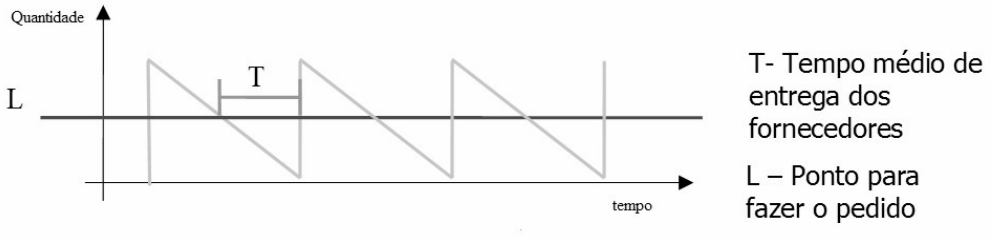
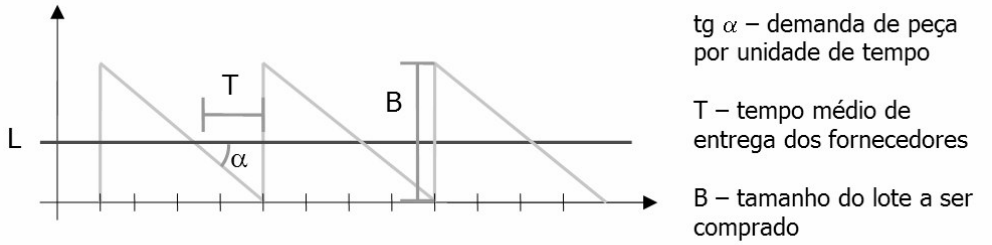


Figura 73: Nível de Estoque por tempo.

Se o fornecedor de uma peça leva um tempo *T* para fazer a entrega de um pedido, então o agente deve fazer esse pedido quando o nível de estoque atingir o valor *L*. Uma decisão importante é calcular esse valor *L* por peça:



$$tg\ \alpha = L / T$$

$$L = tg\ \alpha * T \Rightarrow$$

$$L = demanda * T$$

Figura 74: Decisão de Compra dos Componentes.

No ambiente do TAC SCM, a demanda por peça não é constante, pois a demanda dos clientes por PCs é um passeio aleatório. Além disso, o tempo de entrega das peças pelos fornecedores também não é constante. Conseqüentemente, o Gerente de Compras re-calcula a demanda por peça e o tempo médio de entrega dos fornecedores utilizando uma média móvel. O algoritmo abaixo resume a decisão diária do Gerente de Compras:

Para cada peça:

1. Calcular demanda *d* por unidade de tempo utilizando uma média móvel.
2. Calcular o tempo médio *T* de entrega dos fornecedores utilizando uma média móvel.
3. Calcular o $L = d * T$, representando o nível de estoque para se fazer o pedido.
4. Se o nível atual de peças em estoque for menor que *L*, então fazer o pedido para um fornecedor de tamanho *B* do lote.

6.4.2. Programador da Produção e Programador da Entrega

A decisão de produzir e entregar as ordens é problema de alocação de recursos para gerar o maior lucro possível. Porém, o problema possui algumas restrições: (i) toda ordem de cliente tem que ser produzida e entregue até uma data limite, (ii) a fábrica possui uma capacidade limitada de ciclos de produção por dia, e (iii) necessidade de estoques de peças para produzir os PCs. Uma descrição de alto nível para o problema é feita abaixo:

Objetivo da Alocação de Recursos:

Maximizar o Lucro Esperado.

Entradas:

Ordens Ativas;

Capacidade da fábrica por dia;

Componentes em estoque.

Saídas:

As ordens produzidas em cada dia;

As entregas das ordens. Para simplificar o problema, as entregas são sempre feitas um dia depois da produção.

Um algoritmo guloso é proposto para resolver esse problema. O algoritmo procura por ordens que geram muita receita líquida por ciclo de fábrica e não leva em consideração os custos de peça, pois as peças já foram compradas. Esse algoritmo não garante uma alocação ótima, mas já produz bons resultados.

Seja:

N – número de ordens ativas

O_i – A i-ésima ordem ativa ($i < N$)

Onde, cada O_i possui os seguintes atributos:

P_i – Penalidades por Ordem i

TC_i – Ciclos de Fábrica para produzir a Ordem i

R_i – A Receita da Ordem i

D_i – Data limite para produção (depois dessa data a ordem recebe a penalidade, e 5 dias depois o cliente cancela a ordem automaticamente)

Assim, podemos definir a Receita Líquida Média por Ciclo gerada pela produção da ordem i, com início no dia j, como:

$$C_{ij} = (R_i - P_{ij}) / TC_i$$

Então, O algoritmo procura por ordens que geram muita receita líquida por ciclo de fábrica, e executa o algoritmo abaixo até alocar todos as ordens ativas:

1. Calcular C_{ij} e C_i* para todos os O_i's a serem produzidos

   P_{ij} – Penalidade por Ordem i no dia j ($1 \leq j \leq 8$) de Produção

   $$P_{ij} = (j - D_i)^+ . P_i$$

   C_{ij} – Receita por ciclo (incluindo a penalidade) por Ordem i no dia j de Produção

$C_{ij} = (R_i - P_{ij}) / TC_i$

$C_{ij} = -\infty$, se capacidade de produção na fábrica no dia j está esgotado, ou se não há componentes em estoque para produzir a Ordem i

$C_i^* = \max\{ C_{ij} \}$, para cada Ordem i

2. Encontrar o maior C_i^* correspondente a Ordem i (se $C_i^* < 0$ então a ordem não é alocada para a produção)
3. Produzir a Ordem i (encontrada no passo 2) no dia d

For $j = (D_i + 5)$ to 0
If ($C_{ij} \neq -\infty$) and ($C_{ij} = C_i^*$)
$d = j$
Break For
End if
End for

4. Atualizar níveis de estoque e ciclos de fábricas disponíveis

A estratégia adotado pelo algoritmo é gulosa na Receita Líquida com agendamento no último período viável livre.

6.4.3.Gerente de Marketing

O Gerente de Marketing utiliza o algoritmo guloso da seção 6.4.2, com algumas pequenas modificações, para decidir a quais RFQs serão enviados lances. Os novos RFQs são considerados como Ordens Ativas. Porém, a receita das Ordens Ativas é multiplicada por um fator k, pois Ordens Ativas "verdadeiras" são mais importantes. Além disso, o C_{ij} agora inclui o custo de peça para produzir a ordem i. Conseqüentemente:

C_{ij} – Lucro por ciclo (incluindo a penalidade) por Ordem i no dia j de Produção

CP_i – Custo de Peças para a ordem i

$C_{ij} = (k.R_i - P_{ij} - CP_i) / TC_i$, se for uma Ordem Ativa

$C_{ij} = (R_i - P_{ij} - CP_i) / TC_i$, se for um RFQ

$C_{ij} = -\infty$, se capacidade de produção na fábrica no dia j está esgotado, ou se não há componentes em estoque para produzir a Ordem i

Depois de executar o algoritmo guloso com as modificações acima, os RFQs selecionados para a produção são enviados para o Vendedor para que possa enviar os lances.

6.5. Utilizando o MAS-School no Processo de Desenvolvimento do LearnAgentsSCM

Na primeira fase do MAS-School, o engenheiro de software deve definir dois elementos centrais que estão relacionados com aprendizado: (i) *Objetivo Sistêmico*, SG, e (ii) *Medida de Performance Sistêmica*, SP, que mede o ganho de performance do sistema. No *LearnAgentsSCM*, o *Objetivo Sistêmico* foi definido utilizando a técnica orientada a objetivos da figura 60. Conseqüentemente, (i) SG: produzir PCs e obter o maior lucro possível, e (ii) SP: A pontuação média representando o lucro médio obtido do sistema no ambiente TAC SCM.

Na segunda fase, os tipos de agentes definidos na tabela 19 são selecionados para utilizar técnicas de *machine learning*. O objetivo é estabelecer um *Problema de Aprendizado do Agente* bem definido. Conseqüentemente, três características são definidas: (i) *Objetivo do Aprendizado*, G; (ii) *Medida de Performance*, P, que mede a melhora da performance no agente individualmente; e, (iii) uma *Experiência de Treinamento*, E, que define o processo de aquisição do conhecimento no agente com o aprendizado. No *LearnAgentsSCM*, o Gerente de Compras foi um dos agentes selecionados para utilizar um técnica de *machine learning*. O *Problema de Aprendizado do Agente* para o Gerente de Compras pode ser definido por:

(vii) G: Prever o consumo de matéria prima;

(viii) P: O erro entre a previsão da peça e o real; e

(ix) E: Utilizar um histórico de utilização de peças.

Na terceira fase, um bom *design* do agente é criado para permitir a inclusão ou reuso de várias técnicas de *machine learning*. Além disso, o *design* deve proporcionar uma boa manutenção de código. O Gerente de Compras também utilizou o padrão de projeto orientado a objetos da seção 4.2.

As decisões mais importantes na fase de implementação de uma técnica de *machine learning*, nos agentes selecionados são: (i) a representação do conhecimento; (ii) o algoritmo de aprendizado; (iii) o conjunto de treinamento utilizado pelo algoritmo de aprendizado. O conhecimento do Gerente de Compras foi modelado como uma função chamada *ComponentRequired*. Essa função recebe uma demanda passada de matéria prima *P* e gera a futura demanda *F* (*ComponentRequired*: $P \rightarrow F$). A representação aproximada da função *ComponentRequired* utiliza uma função para essa demanda futura: $ComponentRequired(n) = (1/\alpha)*(AskPrice(n-1) + ...+ PredictedAskPrice(n-\alpha))$, onde α é o tamanho da janela da média; e *n* é o *n*-ésimo jogo. O algoritmo de aprendizado Least Mean Squares (LMS) é utilizado para adaptar o α: $\alpha(n) = \alpha(n-1) + \beta*(ComponentRequired(n-1)- ComponentUsed(n-1))$, onde β é a taxa de aprendizado.

Ao invés de desenvolver o sistema multi-agente com todos os agentes inteligentes em uma única fase, a fase de implementação utiliza uma proposta incremental para facilitar a integração e análise da performance dos agentes. A primeira versão do *LearnAgentsSCM* foi desenvolvido apenas com agentes de software reativos. A pontuação média foi medida para poder avaliar a evolução do *benchmark*, mesmo sabendo que o sistema não apresentaria boa performance. A tabela 20 mostra o *benchmark* feito com as várias versões do *LearnAgentsSCM*. Cada teste para avaliar a performance do sistema era feito em um simulador da competição real contra seis jogadores chamados *dummies* (TAC). O sistema jogava dez partidas no *benchmark* para avaliar a performance média, e minimizar tanto o efeito da variação da demanda dos clientes quanto à variação na produção dos fornecedores.

LearnAgentsSCM Version	Agent Selected / Intelligent Module	Average Score	Games Played
0.0	---	-633.291.661	10
1.0	Procurement Manager / Lot Size and When to Order	-248.466.127	10
2.0	Production Scheduler & Delivery Scheduler / Greedy Algorithm Marketing Manager / Greedy Algorithm	-28.901.477	10
3.0	Procurement Manager / Safety Inventory	28.865.996	10

Tabela 20: Resultados da Evolução do Benchmark.

A versão 1.0 do *LearnAgentsSCM* implementou a técnica de comprar matéria prima em lotes e decisão ótima de quando comprar (Chopra et al., 2004) no Gerente de Compra. Essa técnica permite uma boa redução de custos, e conseqüentemente um prejuízo menor em relação ao primeiro *benchmark*. Esse resultado gerou uma pontuação média de -248 milhões.

A versão 2.0 implementou, no Programador da Produção e Programador da Entrega, o algoritmo guloso para programar a produção e entrega de produtos. O Gerente de Marketing também utilizou uma versão modificada desse algoritmo guloso para decidir quais RFQs serão aceitos. Essa técnica tornou a fábrica mais eficiente e lucrativa, e conseqüentemente melhorou o resultado para -28 milhões.

A versão 3.0 do sistema implementou a técnica de manter estoque de segurança (Chopra et al., 2004) e compra de matéria prima em pequenos lotes. Essa técnica permitiu uma redução de custos ainda maior, e conseqüentemente a pontuação média subiu para 28 milhões.

A figura 75 apresenta um gráfico com a evolução da inteligência do *LearnAgentsSCM*. O ganho constante na performance da inteligência permite constatar novamente a eficácia do método MAS-School.

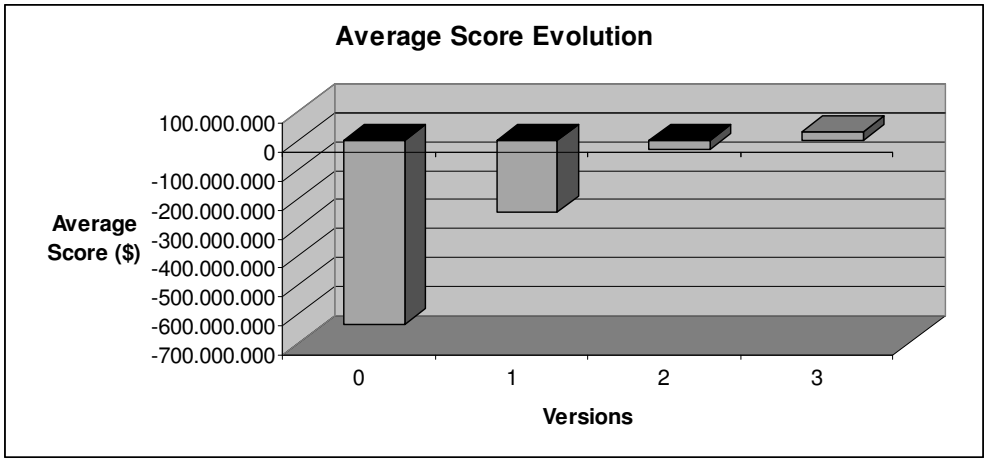


Figura 75: Gráfico da Evolução do Benchmark.