5
Estudo de Caso

Este capítulo destina-se a uma pequena introdução ao sistema *LearnAgents*, e a comparação entre a modelagem de um agente do sistema com a linguagem ANote e a modelagem deste mesmo agente de modo concomitante com o ANote e o método GP-MVAP.

Desta forma, será possível dar uma idéia de como o método GP-MVAP pode ser utilizado em um sistema complexo e também mostrar as vantagens da representação com o método GP-MVAP.

Nas seções subseqüentes serão apresentadas: a arquitetura do sistema, a modelagem com o ANote, os pontos de flexibilização de um agente do sistema e a nova modelagem utilizando o método GP-MVAP.

5.1.
Sistema LearnAgents

O sistema *LearnAgents* servirá para dar uma idéia de como a solução pode trazer benefícios para o desenvolvimento do sistema e ao próprio desenvolvedor, o sistema pode ser visto com mais detalhes em (Sardinha, 2005a).

O sistema *LearnAgents* foi escolhido por ser suficientemente complexo e por ter passado por diversos testes durante a competição TAC que faz parte do calendário da conferencia AAMAS.

Em 2004 o sistema *LearnAgents* obteve a terceira colocação na competição, além de ser um sistema complexo e ter passado por vários testes, ele foi modelado com uma linguagem de modelagem de agentes (ANote) o que ajudou na aplicação da solução.

Na competição TAC uma das regras é que não pode haver intervenção humana, ou seja, os agentes devem trabalhar sozinhos em leilões de bens, onde os bens podem ser passagens aéreas, hospedagens em hotel e ingressos para museus, parques ou luta livre. Cada sistema recebe oito clientes com suas respectivas preferências, a pontuação de cada agente é calculada pela diferença de utilidade

dos pacotes montados e o custo dos bens adquiridos nos leilões, refletindo o lucro total obtido pelo agente (Sardinha, 2005a, b).

O objetivo maior desse sistema era comprar pacotes de viagens para os clientes e obter o maior lucro possível.

Para tal competição foi desenvolvido um sistema com 48 agentes seguindo a seguinte arquitetura: "agentes sensores" para passagens aéreas, hospedagens em hotel e ingressos para eventos; "agentes negociadores" para passagens aéreas, hospedagens em hotel e ingressos para eventos; "agentes de predição" para passagens aéreas, hospedagens em hotel e ingressos para eventos; "agentes ordenadores" de passagens aéreas, hospedagens em hotel e ingressos para eventos; "agente alocador mestre e escravo"; "agente monitor"; base de conhecimento comum e venda. Tal arquitetura pode ser vista na figura a seguir.

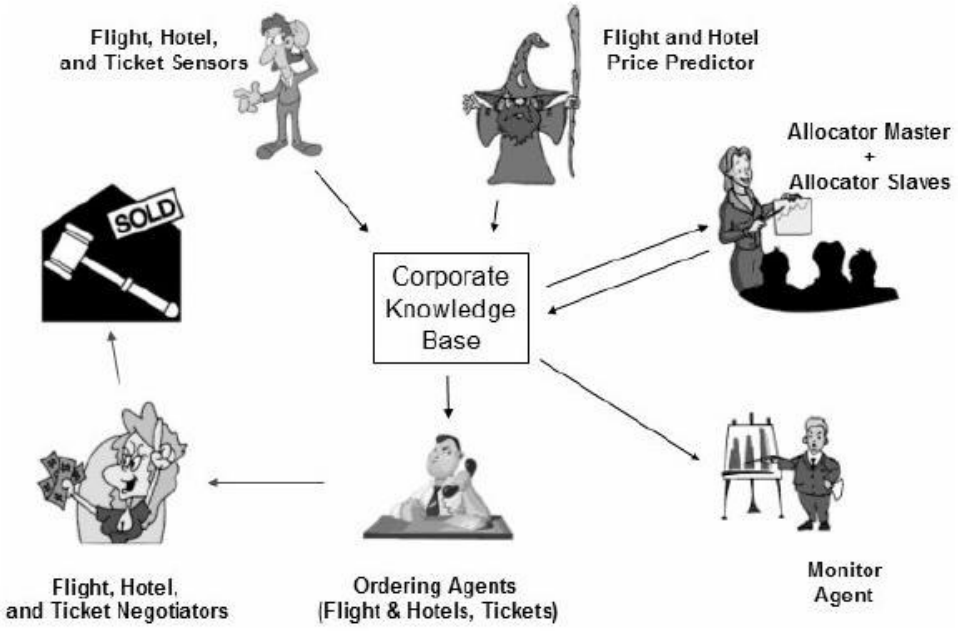

Figura 26. Arquitetura do *LearnAgents*.

O sistema foi desenvolvido de forma distribuída o qual executava em quatro CPUs dois Windows e dois Linux (Sardinha, 2005a). Após esta breve apresentação do sistema *LearnAgents* será apresentada na seção seguinte a modelagem do sistema utilizando a linguagem ANote.

5.2.
Modelando LearnAgents com o ANote

Nesta seção será apresentada toda a modelagem do sistema que foi feita com a linguagem ANote em (Sardinha, 2005a) e a seção subseqüente apresentará a modelagem de apenas um agente com o método GP-MVAP.

Na seção 5.1. foi observado que o objetivo principal do sistema era comprar pacotes de viagens para os clientes e obter o maior lucro possível.

Para alcançar o seu objetivo principal foram identificados: a negociação de pacotes de viagem tendo em vista que o mercado é bastante competitivo e uma base de conhecimento do mercado para auxiliar no processo de negociação. A figura seguinte mostra o diagrama de objetivo do sistema com a decomposição dos objetivos do sistema, tendo no topo o seu objetivo principal (Sardinha, 2005a).

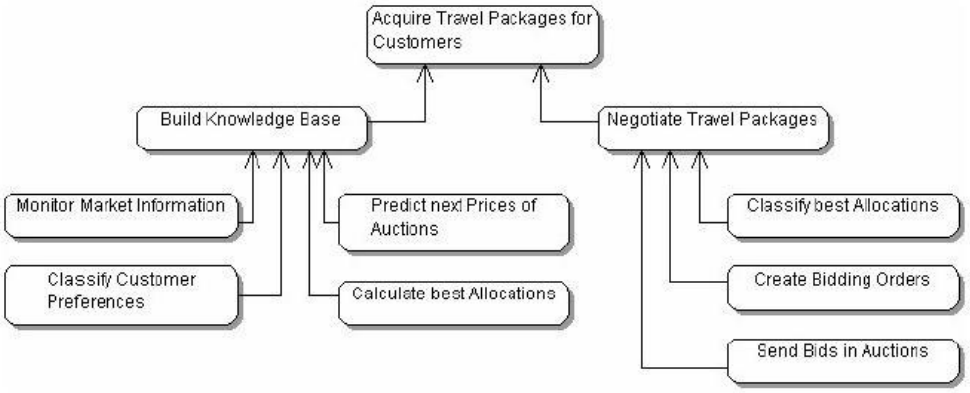

Figura 27. Decomposição dos objetivos relacionados com o comércio de bens.

A tabela a seguir (Sardinha, 2005a) pode ser utilizada como um complemento da modelagem, onde tal tabela faz o mapeamento entre o tipo de agente o sub-problema que fica sobre sua responsabilidade.

Tabela 17. Mapeamento entre sub-problemas e tipos de agentes.

Goal	**Agent**
Monitor Market Information	Hotel Sensor, Flight Sensor, Ticket Sensor
Classify Customer Information	Hotel Sensor
Predict Next Prices of Auctions	Price Predictor

Calculate Best Allocations	Allocation Master, Allocation Slave
Classify Best Allocations	Ordering
Create Bidding Orders	Ordering
Send Bids in Auctions	Hotel Negotiator, Flight Negotiator, Ticket Negotiator

A figura a seguir mostra o diagrama de agente do ANote que é uma visão estática do sistema multi-agente, onde vários tipos de agentes se relacionam, a fim de prover a melhor performance para o sistema *LearnAgents* (Sardinha, 2005a, b).

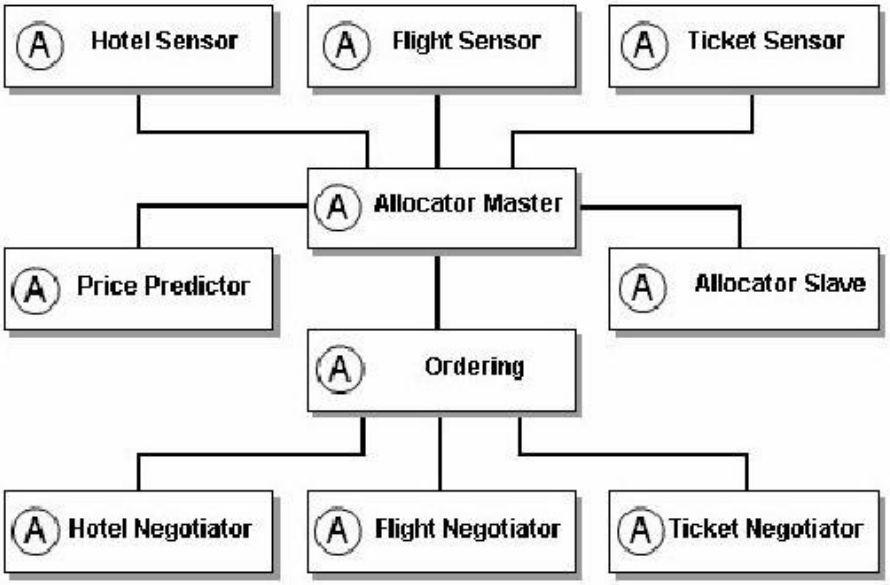

Figura 28. Arquitetura formal do *LearnAgents*.

Seguindo a modelagem com o ANote, será apresentado o diagrama *scenario view* dos vários agentes do sistema.

"Agentes sensores" responsáveis por atualizar a base do conhecimento com os preços correntes dos leilões e envia mensagens para outros agentes.

Tabela 18. Diagrama *scenario view* dos agentes sensores.

Monitor Market Information	
Main Agent	Sensor Agent
Preconditions	Event from environment
Main Action Plan	While true Collect current prices of auctions Update CKB

	Send message to Price Predictor End while
Interaction	Price Predictor
Variant Action Plan	-

"Agentes preditores" através de uma série histórica dos preços de vôos e hotéis, os preditores prevêem o valor que o leilão deve caminhar.

Tabela 19. Diagrama *scenario view* dos agentes preditores.

Predict Prices	
Main Agent	Price Predictor Agent
Preconditions	Message from Sensors
Main Action Plan	While true Collect current prices in CKB Predict Auction Prices Update CKB Send message to Allocator Master End while
Interaction	Allocator Master
Variant Action Plan	-

"Agentes alocadores" mestre e escravos têm como objetivo calcular vários cenários diferentes onde esses agentes utilizam os preços previstos e correntes para montar tal cenário.

Tabela 20. Diagrama *scenario view* de interação do agente alocador mestre com os alocadores escravos.

Distribute Allocation Problem	
Main Agent	Allocator Master Agent
Preconditions	Message from Price Predictor
Main Action Plan	While true Collect current prices in CKB Divide allocation problem to slaves Insert allocation problems in CKB Send message to Allocator Slaves End while

Interaction	Allocator Slaves
Variant Action Plan	-

Tabela 21. Diagrama *scenario view* do agente alocador escravo.

Solve Allocation Problem	
Main Agent	Allocator Slave Agent
Preconditions	Message from Allocator Master
Main Action Plan	While true Collect allocation problem in CKB Solve allocation problem Update solution in CKB Send message to Allocator Master End while
Interaction	Allocator Master
Variant Action Plan	-

Tabela 22. Diagrama *scenario view* de interação do agente alocador mestre o agente ordenador.

Combine Allocation Problem	
Main Agent	Allocator Master Agent
Preconditions	Message from Allocator Slave
Main Action Plan	While true Collect slave solution in CKB Combine Solutions Update CKB Send message to Ordering End while
Interaction	Ordering
Variant Action Plan	-

"Agente ordenador" assume o papel de chefe dos negociadores e utiliza os cenários calculados para tomar suas decisões.

Tabela 23. Diagrama *scenario view* do agente ordenador.

Classify Best Allocations	
Main Agent	Ordering Agent
Preconditions	Message from Allocator Master

Main Action Plan	While true
	Collect allocations in CKB
	For each Negotiator Agent do
	Decide goods to buy
	Decide goods amounts
	Calculate good high prices
	Send message to Negotiator Agent
	End do
	End while

Interaction	Ordering
Variant Action Plan	-

"Agentes negociadores" para cada leilão a um negociador especializado na compra de um tipo de bem. Os negociadores só começam a operar nos leilões depois que recebem as ordens de compra do agente ordenador.

Tabela 24. Diagrama *scenario view* dos agentes negociadores.

Negotiate Goods	
Main Agent	Negotiator Agent
Preconditions	Message from Ordering
Main Action Plan	While true Collect good orders in CKB Negotiate goods in auctions End while
Interaction	-
Variant Action Plan	-

"Agente monitor" é responsável por armazenar informações da base de conhecimento, esses dados são utilizados por uma ferramenta a fim de monitorar

o desempenho de cada agente e do sistema multi-agente como um todo (Sardinha, 2005a).

Como é possível notar a modelagem pura e simplesmente com o ANote não é tão simples de perceber em que plano será utilizado qual tipo de leilão.

Deste ponto da modelagem em diante será utilizado o método GP-MVAP juntamente com a linguagem ANote.

5.3. Pontos de Flexibilização do Sistema LearnAgents

O *LearnAgents* é composto de 48 agentes cada qual trabalhando para alcançar seus objetivos e chegar a um objetivo comum de ambos no sistema. Dentro dos agentes deste sistema podem ser observados vários pontos de flexibilização, porém para esta seção será apresentado apenas o "agente negociador" que é o que apresenta pontos de flexibilização mais claros e bem definidos.

O ponto que evidencia o uso do método GP-MVAP no "agente negociador" é que neste agente é necessário executar a ação de negociar bons leilões, porém a abrangência da teoria de leilões é bastante grande. Para o jogo foi bem definido que tipos de leilões seriam utilizados, as descrições do jogo podem ser vistas em TAC - *Trading Agent Competition* "*Game Description*" (2004).

Porém pensando numa aplicação comercial o "agente negociador" traz neste ponto uma variabilidade que poderá ser representado através do método GP-MVAP e com isso será fácil representar as formas de se obter uma boa negociação para os diferentes leilões. Com isto será possível promover o reuso e o aumento na qualidade do software, já que, será possível concentrar os esforços no foco da aplicação.

Isto implica que o "agente negociador" terá um plano abstrado que contém a implementação da ação de obter boas ordens da base de conhecimento e a assinatura da ação de obter uma boa negociação nos leilões. Os planos concretos existirão conforme a necessidade da aplicação, onde tais planos irão implementar a forma de fazer uma boa negociação para os diferentes leilões.

Na seção seguinte será modelado o "agente negociador" utilizando a solução proposta, com isso será possível comparar com a modelagem anterior e

apresentar as vantagens do método GP-MVAP para o desenvolvimento do sistema.

5.4. Modelando LearnAgents com o Método GP-MVAP e o ANote

Aqui será apresentada a modelagem do agente negociador do sistema *LearnAgents* utilizando o método GP-MVAP, a solução foi utilizada de forma concomitante com a linguagem ANote. Como pode ser visto na seção 5.2. a modelagem atual do sistema *LearnAgents* apresenta uma fulga do meta-modelo original do ANote para tentar representar de forma correta o que ocorre com o sistema e além disso não expressa as possíveis instâncias que podem ser implementadas para tal sistema.

O agente negociador apresenta uma flexibilização de planos e ações a qual não é possível representar através da modelagem ANote tradicional. Em função disto o agente negociador foi escolhido e com o uso da linguagem ANote juntamente com o método GP-MVAP será possível obter a vantagem de poder representar através dos pontos de flexibilização um *framework* para uma futura linha de produtos.

Tabela 25. Diagrama *scenario view* estendido.

Negotiate Goods	
Main Agent	Negotiator Agent
Precondition	Message for ordering
Main Action Plan	Collect good orders in CKB() Flexibility (Plan Creation)
Interaction	-
Variant Action Plan	-

Tendo em mãos toda a analise de requisitos feita para o projeto é possível perceber que no escopo da aplicação são abordados vários tipos de leilões (TAC - *Trading Agent Competition* "*Game Description*", 2004). Esta característica, de ter vários tipos de leilões no escopo do sistema, mostra que o plano do agente terá uma variação incondicional possibilitando uma abordagem de flexibilização em planos e mostra que a aplicação pode ter instâncias distintas em função do plano do agente.

A partir daí, tendo em mãos os planos que deverão ser instanciados, podem ser criados os diagramas *flexibility view* e *action note view*, a partir do *flexibility view* será possível observar quais planos podem ser instanciados e com o *action note view* é possível determinar quais as ações que serão implementadas e em função das *tags* é possível determinar em quais planos serão utilizadas tais ações.

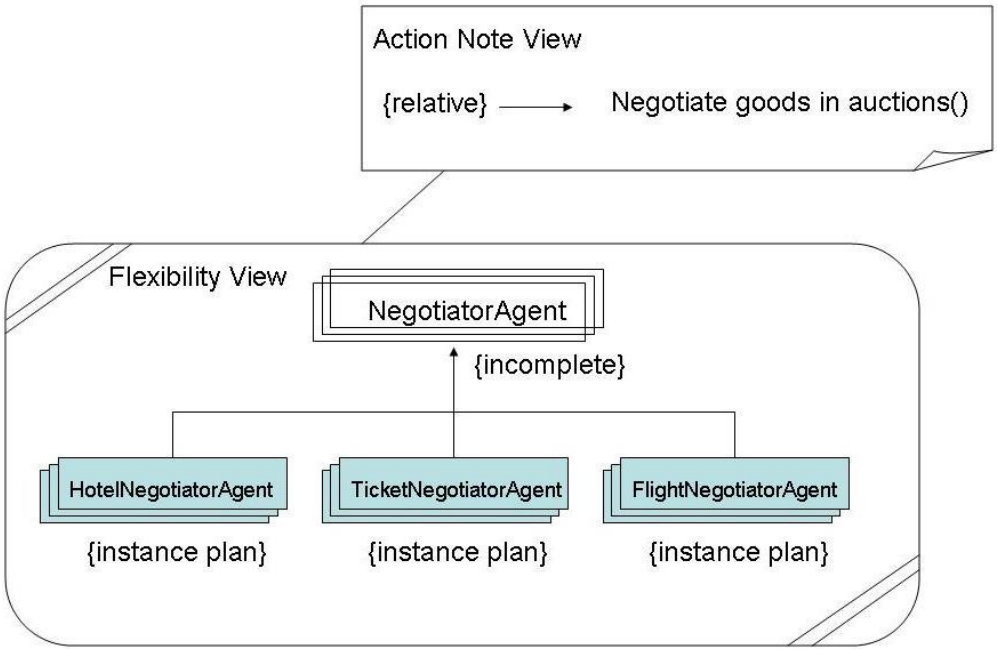

Figura 29. *Flexibility view* para o agente negociador.

A figura 34 mostra o *flexibility view* e o *action note view*, neste ultimo pode-se observar que a ação "*Negotiate goods in auctions( )*" está marcada com a *tag {relative}* que indica que tal ação será implementada em função do plano que está sendo instanciado, ou seja, no caso do sistema *LearnAgent* são utilizadas três formas distintas de se fazer um leilão e em função destas três maneiras distintas de leilão é que a ação "*Negotiate goods in auctions( )*" será implementada.

Um ponto que também é de grande importância diz respeito à técnica para servir de guia para as instâncias de possíveis aplicações.

Para guiar o desenvolvedor no processo de instanciação será utilizada uma técnica baseada em RDL (com a visão de agente) e desta forma, o analista terá todo um mecanismo para desenvolver um guia dedicado ao desenvolvedor, mostrando a forma com que deve ser feita uma instância para tal aplicação. Juntamente com os demais diagramas o analista poderá modelar todas as instâncias sem infringir as restrições da linguagem.

Tabela 26. Técnica baseada em RDL utilizada para guiar a instanciação de planos.

Instantiation View	
Description:	Loop indica que o código (PLAN EXTENTION(Negotiator Agent Plan, ?)) é executado um dado numero de vezes para se obter a quantidade de extensões necessárias. Retorna cada plano com suas ações específicas e executa cada plano até que a condição seja falsa. ADD_ASSIGN(SELECT_PLAN_EXTENSION(NegotiatorAgent_Plan), *Action Note View*); mostra que os planos que serão instanciados receberão as assinaturas das ações de acordo com o que foi definido no diagrama *Action Note View*.
Code:	**Coobook:** Negociar e obter bons preços no leilão **Recipe main** **loop** PLAN EXTENTION(Negotiator Agent Plan, ?); **End loop** ADD_ASSIGN(SELECT_PLAN_EXTENSION(NegotiatorAgent_Plan), *Action Note View*); //SELECT_PLAN_EXTENSION(NegotiatorAgent_Plan); **End_Recipe** **End_Coobook**

Caso o responsável pela modelagem queira ser ainda mais detalhista pode adicionar o comando ADD_CODE(AgentClass1, action1, "{//código que deve ser adicionado}"), o qual indica em que agente e qual ação deste agente deverá ser adicionado o código. Desta forma, o desenvolvedor terá tudo o que será necessário para realizar a instanciação.

Uma das vantagens da solução proposta é a minimização do erro por parte do desenvolvimento, já que todas as possíveis aplicações já foram mapeadas e a forma de instância-las também, uma segunda vantagem é a possibilidade de re-uso

da modelagem já que as possíveis alternativas de aplicações já foram mapeadas e dessa forma para uma aplicação com a mesma base (*kernel*) basta que seja alterado o ponto de flexibilização (planos e ações).