UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JOÃO PAULO DELGADO PRETI

USO DE AGENTES INTELIGENTES BASEADOS EM LÓGICA FUZZY PARA O PROCESSO DE NEGOCIAÇÃO DE COMPRA E VENDA EM SISTEMAS ERP

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: PROF. DR. JOÃO BOSCO MANGUEIRA SOBRAL

Cuiabá – MT, Novembro de 2002.

COLEÇÃO DE AGENTES INTELIGENTES BASEADOS EM LÓGICA FUZZY PARA O PROCESSO DE NEGOCIAÇÃO DE COMPRA E VENDA DE SISTEMAS ERP

JOÃO PAULO DELGADO PRETI

Esta Dissertação foi julgada adequada para a obtenção do título de **Mestre em Ciência da Computação** na área de **Sistema de Computação** e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____
Prof. Fernando Álvaro Ostumi Gauthier, Dr.
Coordenador do Curso

Banca Examinadora

_____
Prof. João Bosco Mangueira Sobral, Dr.
Orientador

_____
Prof. Mário Antônio Ribeiro Dantas, Dr.
Membro da Banca

_____
Prof. Maria Elenita Menezes do Nascimento, Phd.
Membro da Banca

_____
Prof. Gilson Alberto Rosa Lima, Dr.
Membro da Banca

“Somos o que repetidamente fazemos.
A excelência, portanto, não é um feito,
mas um hábito”.
ARISTÓTELES

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ACL	Agents Communication Language
API	Application Program Interface
B2b	Business to Business
B2c	Business to Consumer
CBB	Consumer Buying Behavior
CORBA	Common Object Request Broker Architecture
CPU	Central Process Unit
DCOM	Distributed Component Object Model
EDI	Eletronic Data Interchange
ERP	Enterprise Resource Planning
FAQ	Frequently Asked Question
IA	Inteligência Artificial
IBM	International Business Machine
JVM	Java Virtual Machine
KIF	Knowledge Interchange Format
KQML	Knowledge Query and Manipulation Language
MRP	Manufacturing Resource Planning
OLE	Object Link Editor
ORB	Object Request Broker
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SQL	Structured Query Language
WAN	Wide Area Network
WWW	World Wide Web

RESUMO

O objetivo deste trabalho é mostrar, através do paradigma de Agentes e da lógica nebulosa, uma integração de aplicações de Comércio Eletrônico para sistemas ERP (*Enterprise Resource Planning*), automatizando a negociação da venda de produtos sobre o critério de preço.

Utilizar-se há páginas em ASP para configuração de um agente remoto, a fim de estabelecer comunicação *Business to Consumer* (b2c) com o sistema ERP, e agentes móveis e estáticos para comunicação *Business to Business* (b2b) para a automação de diversas tarefas da relação compra e venda do sistema, principalmente a negociação, para tal, será utilizado o modelo CBB (*Consumer Buying Behaviour*), que se baseia no comportamento de compra do consumidor.

PALAVRAS CHAVE: Agentes, ERP, lógica nebulosa, negociação.

ABSTRACT

The proposal of this work is to implement, beyond software agents, an integration of applications for Eletronic Commerce to ERP (Enterprise Resource Planning) systems; personalizing the sell of the product under price criteria.

ASP pages will be used to configure a remote agent (negotiator) to establish communication Business to Consumer (b2c) with the ERP system, and mobile and static agents to communication Business to Business (b2b) for the automation of different tasks involved in buying and selling behavior, mainly the negotiation, to reach that, the CBB (Consumer Buying Behaviour) model, which is based on consumer behavior, will be used.

KEY WORDS: Agents, ERP, fuzzy logic, negotiation.

1 INTRODUÇÃO

Este trabalho versa sobre a implementação da integração de comércio eletrônico e ERP (*Enterprise Resource Planning*), utilizando a tecnologia de agentes inteligentes baseados em lógica fuzzy.

Um grande número de usuários está em contato com os computadores, para as mais diferentes finalidades, e esse número continuará a crescer como a tecnologia que os cerca, assim como foi com a televisão. Essas máquinas se tornaram veículos para o incremento do alcance nas atividades de cada dia. A aquisição de notícias e informação, correio, negociações e até interações sociais e entretenimento tem se tornado mais e mais baseadas em computador.

O mundo dos programas é cercado de grandes riquezas e diversidade. Milhares de programas estão disponíveis para os usuários e empresas hoje em dia, disponibilizando assim uma grande variedade de informação e serviços dos mais variados tipos. Enquanto a maioria desses programas supri seus usuários de forma significativa, há entretanto uma grande demanda para programas que possam interoperar, ou seja, trocar informações e serviços com outros programas possibilitando assim a resolução de problemas que não poderiam ser resolvidos pelo programa isolado.

Com o processo de globalização mundial, essa interoperabilidade se faz mais necessária ainda, empresas querem expandir-se por todo o globo, assim como os usuários poderão escolher os produtos que mais lhe agradam, as empresas poderão ter uma seletividade maior em relação aos possíveis fornecedores que se encontram em qualquer região do planeta.

Agentes têm o potencial de participar ativamente para completar tarefas, mais do que servir como ferramentas passivas. Contudo, as pessoas não querem agentes genéricos, elas querem agentes especializados, para ajudar em seu trabalho, suas tarefas, seus objetivos.

São construídos agentes colaborativos para o processo de negociação de compra e venda de um sistema ERP, para integração de aplicações de comércio eletrônico no processo de compra e venda, tanto para o usuário comum (*business to consumer)*, como para as empresas fornecedoras (*business to business*). Uma coleção de agentes podem estar colaborando para a realização de uma tarefa, no qual pode-se assumir uma certa

base de entendimento entre eles. A metáfora usada é de que há um assistente pessoal que colabora com o usuário no mesmo ambiente de trabalho, reduzindo o trabalho e a sobrecarga de informação.

Todas as denominações nos dicionários descrevem a palavra agente como sendo algo que tem algum propósito especializado, neste caso, nosso propósito é estabelecer a integração usuário – ERP – fornecedor, através da Internet (b2c / b2b).

O problema que torna a interoperabilidade difícil é a heterogeneidade. Os programas são escritos por pessoas diferentes, em tempos diferentes, e em diferentes linguagens. Consequentemente, todos eles oferecem interfaces diferentes. Através da tecnologia de agentes se encontrou uma solução para a realização deste trabalho.

1.1 Trabalhos Correlatos

Da literatura pesquisada, um trabalho sobre Comércio Eletrônico com Agentes Móveis foi tomado como base para esta pesquisa: CUBe. [VALDO, 1999]

CUBe é um sistema que utiliza sistemas distribuídos como abordagem para integração de aplicações em Comércio Eletrônico e ERP, através da mediação dos Agentes Móveis. É um trabalho que serve de base e que referencia outros trabalhos de Comércio Eletrônico como AuctionBot e V-Market. Trata-se de um sistema diferenciado por dar suporte a múltiplos produtos, automação de várias etapas do modelo CBB e por integrar múltiplos processos de estratégia e comportamento.

O presente trabalho é uma complementação ao projeto CUBe que será visto com maiores detalhes no capítulo 8, adotando sua abordagem para integração de agentes negociadores para sistemas ERP a atividades de Comércio Eletrônico via Internet.

1.2 Definição do Problema

Antigamente o ato de venda era realizado de forma mais artesanal, onde os fabricantes confeccionavam seus produtos e os vendiam às pessoas com as quais tinham amizade Robinson e Hall (1971).

No comércio moderno, com a globalização do mercado, esse tipo de prática desapareceu, pois a maioria dos fabricantes de hoje produz mercadorias para mercados regionais, nacionais ou internacionais.

Com o surgimento de novos fabricantes e novos mercados, ocorreu um aumento significativo nas opções de onde comprar e para quem vender.

A abertura da economia juntamente com o desenvolvimento da Internet e outras tecnologias, fizeram com que o número de empresas concorrentes aumentasse significativamente, provocando a queda dos preços e o aumento da qualidade dos produtos e serviços oferecidos.

De acordo com Stup apud Schneider (1997), o futuro pertence às grandes redes e grandes marcas, e às lojas pequenas que tenham como diferencial o preço e o atendimento ao mercado de vendas eletrônicas.

A globalização fez com que, tanto os vendedores quanto os compradores, tivessem que adotar processos mais elaborados ao realizar as tarefas de compra ou venda de produtos. O objetivo é atingir um conjunto de requisitos (preço, qualidade, entre outros.) necessários para se obter êxito nestas tarefas.

Quanto aos canais de distribuição dos produtos, houve um avanço significante. Muitas mercadorias passaram a ser vendidas por telefone, máquinas e computadores, sendo a utilização da Internet o canal mais promissor, onde é estabelecido o comércio eletrônico.

Mesmo com todos esses avanços que estão surgindo nos processos de compra e venda, o tratamento personalizado de cada cliente, como ocorre no comércio tradicional, está distante da realidade das empresas que fazem comércio eletrônico. Os consumidores e fornecedores, necessitam de uma comunicação mais ágil e processos mais eficientes e com menos custos, a fim de aproximar às relações de compra e venda entre ambos.

A principal preocupação de empresas, ao menos no setor privado, tem recentemente se voltado para como gerenciar suas áreas funcionais de modo a obter, manter e ampliar seu poder competitivo. Podemos definir, então, as estratégias funcionais como ferramentas cujo objetivo principal é o aumento da competitividade da organização. Para tal, buscam organizar recursos da área funcional da empresa e conformar um padrão de decisões coerente, para que esses recursos possam prover uma

composição adequada de características de desempenho que possibilite à organização competir eficazmente no futuro. O desafio torna-se portanto, aproximar os laços entre empresa e consumidor, buscando estabelecer um tratamento personalizado. Neste trabalho, a negociação automatizada do desconto, torna-se um passo para alcançar este objetivo.

1.3 Solução Proposta

Como passo para a solução do problema em questão, a criação de uma coleção de agentes colaborativos baseados na lógica nebulosa é proposta.

Tais agentes, implementados na linguagem Java, utilizando uma coleção de classes dos Aglets da IBM, tem por finalidade, analisar um conjunto de variáveis que traçam um perfil do cliente (gastos acumulados e periodicidade de compra) e do produto (quantidade, longevidade, compras efetivadas e demanda) para chegar a um valor de desconto a ser aplicado aos produtos que serão adquiridos, buscando assim, valorizar o cliente participativo ou incentivar a conquista de novos clientes.

O objetivo é criar o que denominamos inteligência de mercado, algo que diferencie a empresa de sua concorrência e a torne menos dependente do chamado *feeling* de seu pessoal de vendas.

Essa personalização do serviço ao cliente evolui para a criação de um maior número de nichos de mercado, possível pela obtenção de informações mais precisas sobre esses grupos de clientes ou nichos. O limite dessa tendência é o tratamento tão personalizado a ponto de cada cliente ser considerado um nicho de mercado.

1.4 Estrutura do Trabalho

No que segue, o trabalho se encontra dividido em 10 capítulos:

No capítulo 2, uma visão geral sobre comércio eletrônico, suas implicações e alguns conceitos importantes são apresentados.

O capítulo 3, explana sobre sistemas ERP, sua definição, suas funcionalidades e alguns termos relacionados ao assunto.

No capítulo 4, a tecnologia *Workflow* é apresentada por ser a base de construção de sistemas ERP, e portanto, é vista em maiores detalhes, com suas definições, seus níveis, formas de implantação e realidade com relação à absorção dessa tecnologia.

No capítulo 5, a lógica nebulosa (*fuzzy logic*), é apresentada sob seus aspectos funcionais, comparando-a com a lógica clássica e expondo suas características únicas para a definição do problema abordado.

O capítulo 6 apresenta o paradigma de agentes, conceituando, abordando suas definições e sua tipologia, justificando seu uso no presente trabalho como importante método na abstração do problema.

No capítulo 7, apresenta um conjunto de classes que permite a criação dos agentes, mostrando algum de seus elementos, seu modelo de eventos e de comunicação, e abordando um pouco sobre seu pacote.

No capítulo 8, o modelo CBB de comportamento de compra do consumidor é explicado, explanando sobre o processo de negociação.

No capítulo 9, é apresentada como solução para o problema, uma implementação de 5 agentes colaborativos que possuem como estratégia a definição do desconto, analisando o perfil de compra do cliente e o perfil do produto a ser adquirido pelo mesmo.

No capítulo 10, as considerações finais são apresentadas, expondo as conclusões, os problemas enfrentados e propostas de trabalhos futuros que venham a enriquecer esta pesquisa.

O anexo A, apresenta o modelo de dados modificado para a execução dos agentes.

O anexo B exibe os diagramas de caso de uso da solução proposta.

O diagrama de classe das classes desenvolvidas é exibido no anexo C.

Finalmente, no anexo D, está o código Java da classe Fuzzy utilizada para criação de todos os agentes colaborativos do sistema que se baseiam na lógica nebulosa.

2 COMÉRCIO ELETRÔNICO

Já ocorre a algum tempo o comércio eletrônico, onde grandes companhias o têm usado para conduzir suas transações empresariais. Várias empresas têm investido nesse ramo, apostando alto nessa nova forma de fazer comércio. Muitas empresas têm perdido dinheiro, são várias as razões para tal ocorrência, insegurança, incrédulo do usuário, falta de conhecimento dessa nova forma de comercializar, sistemas empresariais que não suportam essa tecnologia, enfim, estão ocorrendo mudanças e elas são dolorosas, mas não há dúvida que ela se efetivará, é uma questão de tempo.

Muitas companhias estão mudando sua forma de administrar por causa da Internet. Hoje a Internet já se expandiu consideravelmente a nível mundial, e com o aumento de sua qualidade (velocidade, segurança, etc.), sua influência nos negócios irá aumentar, fazendo com que o comércio empresarial se expanda como a própria Internet.

O comércio eletrônico vem crescendo e se expandindo de forma reveladora, incluindo negociação de transações de compra e transferência de fundos sobre redes de computadores, compra e venda de novos artigos (*commodities*) tais como a informação eletrônica, leilões, assinaturas das mais variadas revistas, prestação de serviços em geral, e etc. Essa nova forma de fazer comércio não se baseia apenas na compra e venda de produtos como é visto por muitos.

As empresas estão acordando para uma nova forma de administrar seu negócio, perceberam a possibilidade de competir com suas concorrentes maiores, visto que não é preciso um excelente arquiteto e construções físicas maravilhosas e onerosas para chamar a atenção do cliente. As páginas virtuais são criadas com algumas ferramentas e imaginação, podendo ficar tão criativas e chamativas como a de uma empresa de grande porte.

A Internet não é apenas um meio para realizar as mesmas transações já efetuadas pela empresa só que de forma digital, é um meio de se pensar novas possibilidades a serem incorporadas no negócio, onde não existem fronteiras para conduzir seu comércio *on-line*.

Para encontrar as necessidades do mercado, empresas projetam e fabricam novos produtos, comercializam estes produtos, distribuem-nos, e fornecem suporte ao cliente, gerando rendimentos para si próprias ao longo do processo. Clientes têm que identificar

primeiro as suas necessidades de compra: um produto físico, um serviço ou informação. Depois procuram informações sobre aquele produto ou serviço; encontram locais que os vendem, e comparam as opções que encontraram (preços, serviços, reputação, dentre outros) antes de comprarem o produto efetivamente. Além disso, fazer comércio deve também envolver **negociação** de preço, quantidade, termos de entrega, e algumas questões legais. E o ciclo de venda não termina com a entrega do produto ou serviço. O suporte aos clientes traz inúmeros benefícios a ambas as partes.

O comércio eletrônico possui muitas vantagens que não se tinham no comércio tradicional. Empresas podem disponibilizar seus serviços e produtos 24 horas por dia, eliminando assim a questão do tempo, e por ser um comércio numa estrutura Internet, elimina-se também a questão do local.

Fora isso, o comércio eletrônico permite novas formas de comércio, assim como novas formas de se fazer comércio, como pode ser visto na *Amazon.com*, que é uma livraria baseada em Seattle, Washington. A companhia não tem lojas físicas, vende todos seus livros através da Internet, e coordena as entregas diretamente com os editores, eliminando a necessidade de inventários. Estas definições de comércio eletrônico não são estáticas; desde que novas oportunidades oferecidas pelas correntes tecnológicas não tenham sido completamente exploradas, novas tecnologias de redes ou aplicações de *software* podem aparecer a qualquer momento.

2.1 Comércio Tradicional.

Considere as tarefas que uma companhia tem que realizar quando seu empregado quer adquirir um item. Primeiro, gera-se uma requisição, depois ganhar aprovação, em seguida, selecionar um fornecedor apropriado, finalmente, determinar a disponibilidade e remeter a ordem de compra. O vendedor, por sua vez, deve verificar o crédito e histórico de vendas, analisar o inventário, agendar a entrega, notificar o depósito e remeter uma fatura.

No modelo do comércio eletrônico, o comprador pode selecionar produtos de um endereço na *Web*, requisitar aprovação e remeter a ordem de compra via processo eletrônico. O vendedor pode adicionar a ordem ao seu banco de dados, analisar o

depósito e o histórico do cliente, combinar a entrega e manipular as comunicações, tudo através do comércio eletrônico.

2.2 Ciclo de Vida

Todas as empresas que fornecem informação sobre produtos e serviços, podem armazená-las e exibi-las na forma digital, o que torna o produto mais versátil em uma nova mídia cada vez mais aceita. Esses dados podem ser armazenados em um banco de dados e apresentados eletronicamente através de páginas *Web*.

Analisemos agora os 5 processos: distribuição da informação, ordem, pagamento, execução, serviço e suporte. Todas essas etapas fazem parte do novo ciclo de vida do comércio eletrônico VALDO (1999).

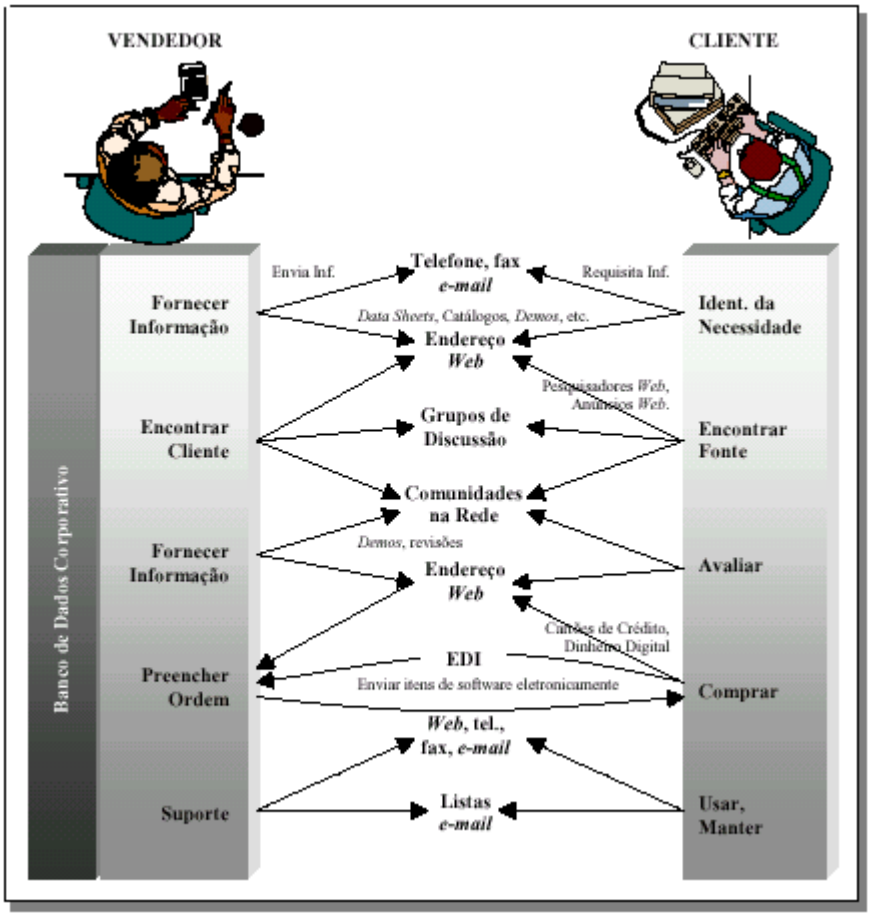

Figura 2.1 – Ciclo de Vida do Comércio Eletrônico / Processos Comerciais

2.2.1 Distribuição da informação

O fornecimento de dados para o processo de pesquisa de seus clientes faz-se necessário, pois eles precisam ter conhecimento dos seus produtos e serviços para que possam ser vendidos. A Internet é um meio efetivo de comunicação com os clientes; os clientes podem encontrar informações sobre seus produtos e serviços através de páginas *Web*, formulários de preenchimento, FAQs (*Frequently Asked Questions*), *e-mail*, grupos de discussão e etc.

2.2.2 Ordem

A realização de uma ordem de pedido é fácil e rápida, visto que os formulários eletrônicos se espelham nos formulários usuais de pedidos da empresa. Por causa desse fator, muitas empresas, por mais simples que sejam, estão migrando seus sistemas para a *Web*.

2.2.3 Pagamento

Pela facilidade do dinheiro digital, cheque eletrônico, cartões de crédito, boleto impresso, nada impede que o processo de recebimento do dinheiro seja efetuado sem complicações. Em vista dessa popularidade da *Web*, as empresas estão começando a usar EDI (*Eletronic Data Interchange*) para transações sobre a Internet com seus fornecedores e, usando os formulários baseados na *Web* ou *e-mail* seguro para repassar transações EDI a seus parceiros comerciais.

2.2.4 Execução

A Internet pode ser usada para transferência de informação de produtos para clientes. Além dos formulários padronizados, tais como: *newsletter*, periódicos, relatórios de análise, e preços de produtos, pode-se incluir nesta lista de dados

eletrônicos o *software*. Documentações, aperfeiçoamento de programas e atualizações estão também integrados aos processos de distribuição sobre a Internet.

Pode-se utilizar EDI para informar os entregadores (transportadoras) sobre produtos que precisam ser entregues, e a Internet pode ser usada para comunicação com fornecedores e distribuidores, através do *e-mail*, sobre questões como, por exemplo, *status* da entrega.

Uma empresa precisa de meios para informar seus clientes atuais e potenciais sobre um produto, mesmo que este produto seja uma informação ou um produto real; os serviços da Internet (*e-mail*, *Web Site*) podem disponibilizar de forma clara e de fácil acesso informações sobre o lançamento deste produto.

2.2.5 Suporte e serviços

A venda pode ser somente o começo, o cliente pode precisar de algum tipo de assistência, assim como a empresa pode fornecer a seus clientes uma melhoria dos produtos e serviços que ele virá a oferecer no futuro.

FAQs (perguntas freqüentemente realizadas), atualizações de software, correções de problemas, formulário de perguntas ou simplesmente o recebimento de *e-mails*, são formas de assistência que podem estar disponíveis aos clientes.

2.3 Benefícios

O comércio eletrônico pode abrir novos mercados para um negócio, alcançar novos clientes em qualquer parte do globo, facilitar o comércio para os clientes já existentes, disponibilizando, por exemplo, os pedidos durante 24 horas por dia e sete dias na semana, facilita a administração, visto que, reduz consideravelmente o número de papéis envolvidos, abre portas para novas formas de se fazer comércio, como ordem, pagamento, suporte, etc. Esses benefícios podem ser de curto a longo prazo.

2.4 e-business e e-commerce

Estes dois termos são às vezes ditados como se tivessem o mesmo significado, e estão em evidência no Comércio Eletrônico. Essa utilização gerou muita discussão em relação a definição desses termos; grandes empresas, como a IBM, têm investido milhões de dólares para que suas idéias se consolidem. As definições aqui adotadas vão de encontro ao exposto por Jason Smith, um dos fundadores da *Columbus Group Communications*, uma companhia que trabalha com soluções e estratégias na Internet.

O termo *e-Commerce* está relacionado a transação; tipicamente, a transação monetária, mas pode também ser a transação que acontece quando se faz um *download* de *software* ou atualiza-se o registro de serviços de um cliente. O foco está na transação em si. Já o termo *e-business* é mais amplo, cobrindo todos os tipos relacionados de troca de informação eletrônica.

O conceito de *e-business* está voltado para a redefinição de modelos comerciais e fundamentalmente a mudança do modo pelo qual as companhias operam. O foco está em fazer todas as operações comerciais de uma empresa eletronicamente, não somente a transação, mas o *status* da ordem, faturamento, ou uma parte do inventário, por exemplo.

Percebe-se que *e-commerce* é apenas uma parte do *e-business*, e por isso estes dois termos não podem ser usados de forma idêntica.

2.4.1 Business-to-Business x Business-to-Consumer

O termo business-to-business (b2b) está relacionado às transações comerciais entre empresas, ou entre uma empresa e fornecedores, vendedores ou sua cadeia de abastecimento (*suply-chain*), enquanto o termo *business-to-consumer* (b2c) está relacionado às transações entre empresas e clientes, um nicho de mercado que está em plena ascensão.

Milhões de dólares podem ser economizados com a digitalização de relatórios e processos comerciais, isso pelo resultado direto do aumento da produtividade, redução de papel e agilidade nas aprovações.

Canais importantes e valiosos podem ser abertos com a Internet, por exemplo, empresas podem abrir canais com seus fornecedores (b2b) permitindo um retorno de investimento quase imediato. Essas características fazem uma companhia incorporar o negócio eletrônico.

A Internet oferece um modo barato e potencialmente mais satisfatório para certos tipos de transações comerciais, visto que o atributo preço é sempre considerado, claro que outros aspectos também são considerados importantes, como segurança, reputação e etc. Porém deve-se assegurar que a ordem é de fácil uso e incorpora várias características possíveis *off-line* (fax, boleto, telefone, depósito).

O b2c na Internet está longe de causar impactos significativos nas empresas, mas é um processo irreversível para todos, que mais tarde revelará seu lado lucrativo tanto para o consumidor quanto para empresas.

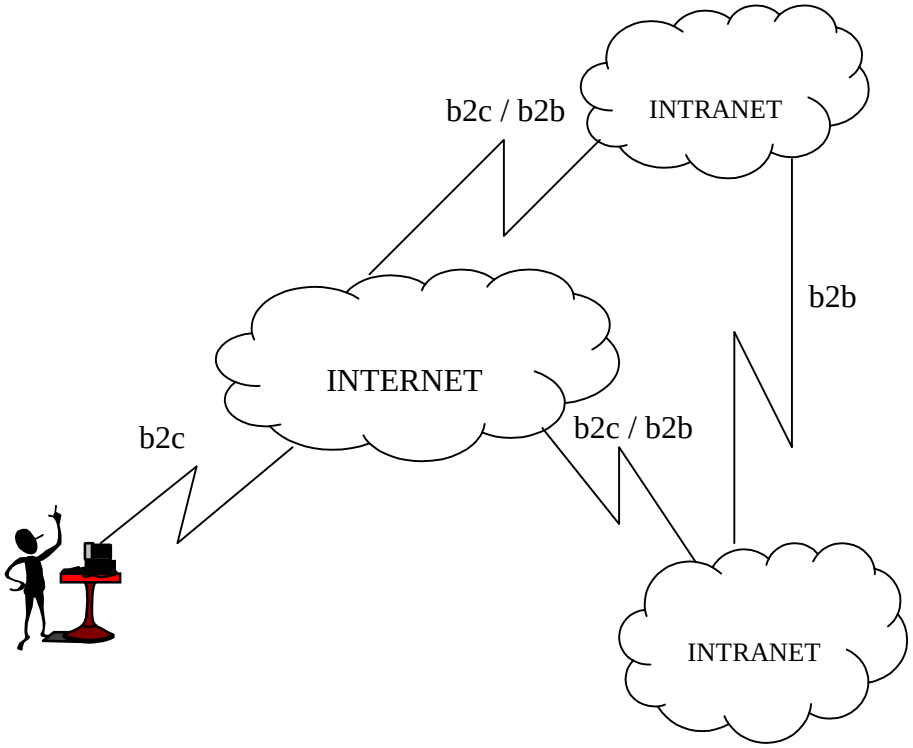

Figura 2.2 – Business to Consumer (b2c) / Business to Business (b2b)

3 SISTEMAS ERP

A competitividade entre as empresas no mercado atual é grande, procura-se produzir com qualidade, preço e diferenciação / inovação, características difíceis de se conseguir, para isso, as empresas tentam reduzir ao máximo seus custos, mas sem prejudicar a qualidade. Ferramentas e filosofias auxiliam as empresas nesse processo e muitas reconhecem a necessidade de passar a gerenciar a empresa como um conjunto de processos e não apenas como uma série de departamentos isolados.

Os sistemas ERP (*Enterprise Resource Planning*) trabalham com essa filosofia; a empresa como um conjunto de processos. São sistemas integrados que surgiram por essa rápida necessidade de desenvolvimento, ao mesmo tempo em que as empresas são pressionadas para terceirizar todas as atividades que não pertençam ao seu foco principal de negócios. Além de ser um sistema integrado desenvolvido por empresas especializadas, ERP abrange a maioria ou a totalidade dos processos empresariais, por isso sua denominação antiga de sistemas integrados de gestão empresarial CRUZ (1998).

Os sistemas ERP são geralmente divididos em módulos, onde as informações alimentadas em um módulo são instantaneamente disponibilizadas para os demais módulos que delas dependem, são portanto, módulos que se comunicam entre si e que geralmente atualizam uma mesma base de dados. Esses sistemas são tão integrados que permitem a utilização de ferramentas de planejamento que podem analisar o impacto de decisões de manufatura, suprimentos, finanças ou recursos humanos em toda a empresa.

A sigla ERP foi cunhada com a intenção de definir esses sistemas integrados como uma evolução dos sistemas MRP II (*Manufacturing Resource Planning* ou Planejamento dos Recursos de Produção). De acordo com CORRÊA (1993), *"O princípio básico do MRP II é o princípio do cálculo de necessidades, uma técnica de gestão que permite o cálculo, viabilizado pelo uso de computador, das quantidades e dos momentos em que são necessários os recursos de manufatura (materiais, pessoas, equipamentos, entre outros), para que se cumpram os programas de entrega de produtos com um mínimo de formação de estoques"*. Os sistemas ERP podem, então, ser considerados uma evolução do modelo MRP II à medida em que permitem controlar os demais recursos empresariais. Embora os conceitos utilizados em sistemas ERP

possam ser usados por empresas que queiram desenvolver internamente os seus aplicativos, a terminologia sistemas ERP refere-se essencialmente a pacotes comerciais.

Exemplos de sistemas ERP existentes no mercado são: R/3 da alemã SAP, Baan IV da holandesa Baan, OneWorld da americana JD Edwards, Oracle Financials da americana Oracle, Magnus e Microsiga da brasileira Datasul, e Logix da empresa Logocenter.

3.1 Funcionalidades

De forma geral, os sistemas ERP fornecem suporte às atividades administrativas (finanças, recursos humanos, contabilidade e tributação), comerciais (pedidos, faturamento, logística e distribuição) e produtivas (projeto, manufatura, controle de estoques e custos). Essa grande gama de funcionalidades e processos empresariais variam de acordo com o fornecedor do *software* ERP.

Davenport, apresenta as funcionalidades dos sistemas ERP separando-as em funções de *back-office*, compostos por recursos humanos, manufatura e finanças, *front-office*, compostos por vendas e serviços, além da tecnologia e do chamado *supply-chain management* ou administração da cadeia de suprimentos. O enfoque deste trabalho se dá em funções de *front-office*, na personalização da venda de um produto ou serviço.

3.2 Termos Relacionados

Existem alguns termos relacionados aos sistemas ERP que são importantes para a compreensão dos aspectos envolvidos, embora não os definam VALDO (1999).

Funcionalidade: é o conjunto total de funções embutidas em um sistema ERP, suas características e suas diferentes possibilidades de uso. A composição destas funções forma o sistema de informações transacional que dá suporte aos processos de negócio.

Módulos: são os menores conjuntos de funções que podem ser adquiridos e implementados separadamente em um sistema ERP. Normalmente, tais conjuntos de funções correspondem a divisões departamentais de empresas (vendas, finanças, produção, planejamento da produção, dentre outros).

Parametrização: é o processo de adequação da funcionalidade de um sistema ERP a uma determinada empresa, através da definição dos valores de parâmetros já disponibilizados no próprio sistema. Parâmetros são variáveis internas do sistema que determinam, de acordo com seu valor, o comportamento do sistema. Segundo MARTIN (1983), "*uma boa possibilidade de parametrização é a chave para (1) fazer pacotes se adaptarem às organizações com um mínimo de mudanças e (2) evitar custos de manutenção*".

Customização: é a modificação de um sistema ERP para que este possa se adequar a uma determinada situação empresarial impossível de ser reproduzida através de parâmetros já existentes. Apesar da customização poder ser feita para adaptar um sistema ERP às necessidades imediatas do cliente, quanto maior for a quantidade de customizações realizadas, mais o sistema se afastará do modelo ERP e mais próximo ficará do modelo interno de aplicações.

Localização: é a adaptação (através de parâmetros e customizações) de sistemas ERP desenvolvidos em outros países e trazidos para utilização dentro da realidade brasileira (impostos, leis, procedimentos).

4 *WORKFLOW*

ERP está intimamente ligado a *Workflow*, para que os processos, as atividades e os procedimentos possam corresponder à expectativa que o planejamento e a organização com que eles foram criados, é preciso automatizar.

A tecnologia *Workflow* foi desenvolvida para automatizar e facilitar o fluxo de trabalho de um processo, a fim de permitir a uma organização atingir seus objetivos.

Segundo CRUZ (1998), "*Workflow é a tecnologia que possibilita automatizar processos, racionalizando-os e potencializando-os por meio de dois componentes implícitos: organização e tecnologia"*. Quando citado no capítulo 3 que ERP passa a "*gerenciar a empresa como um conjunto de processos e não apenas como uma série de departamentos isolados*", estava-se falando em *Workflow*.

Por isso um capítulo a parte para explanar sobre essa tecnologia que vai revolucionar processos e para entender melhor o funcionamento dos sistemas ERP que se baseiam nessa tecnologia.

4.1 Outras Definições

Em português, podemos traduzir por um substantivo composto, para que não se perca de vista seu real significado: Fluxo de trabalho! Entretanto, por uma questão de internacionalização da tecnologia, vamos chamá-la de *Workflow*.

Uma definição de *Workflow* já foi citada no início do capítulo, mas cabem aqui outras definições:

"*Workflow é o fluxo de controle e informação num processo de negócio*".

James G. Kobielus

"*Workflow é um conjunto de ferramentas que possibilita análise proativa, compressão e automação de atividades e tarefas baseadas em informação*".

Thomas M. Koulopoulos

"*Workflow é a tecnologia que ajuda a automatizar políticas e procedimentos numa organização*".

Setrag Khoshafian

4.2 Evento – Uma Abordagem para *Workflow*

O evento é a menor parte de um sistema *Workflow*. Na verdade, é a menor parte de qualquer processo, conhecido, também, como tarefa. O evento pode ser de um dos seis tipos listados a seguir CRUZ (1998):

- Inicialização;
- Notificação;
- Interação;
- Duração;
- Dependência;
- Finalização;

4.2.1 Inicialização

É o evento que dá inicio à viagem que um objeto fará dentro do fluxo de trabalho. Ele é também conhecido como o evento que dispara (*trigger*) uma seqüência de eventos, ou que dá origem ao processo de *Workflow*. Esse evento pode ser simples, como quando um usuário se conecta ao sistema, ou complexo, quando dispara inúmeros outros eventos ao mesmo tempo.

4.2.2 Notificação

É um evento que existe em decorrência de outro evento de *Workflow*. É, geralmente, uma mensagem eletrônica que indica a ocorrência de fim de processo, por exemplo, embora possa disparar outro evento.

4.2.3 Interação ou Negociação

É a execução repetitiva de um objeto por várias tarefas, seguindo as mesmas regras todo o tempo. Um documento que necessite ser assinado três vezes é um bom exemplo de interação.

4.2.4 Duração

É o tempo necessário para completar um evento, ou postergá-lo até que determinada condição tenha sido satisfeita. Convém salientar que o tempo de duração não inclui o tempo de transferência.

4.2.5 Dependência

É um evento de espera. Quando um objeto aguarda por outro objeto que é pré-requisito de seu processamento, ele é dependente desse objeto.

4.2.6 Finalização

É o último evento de um objeto dentro de um sistema *Workflow*.

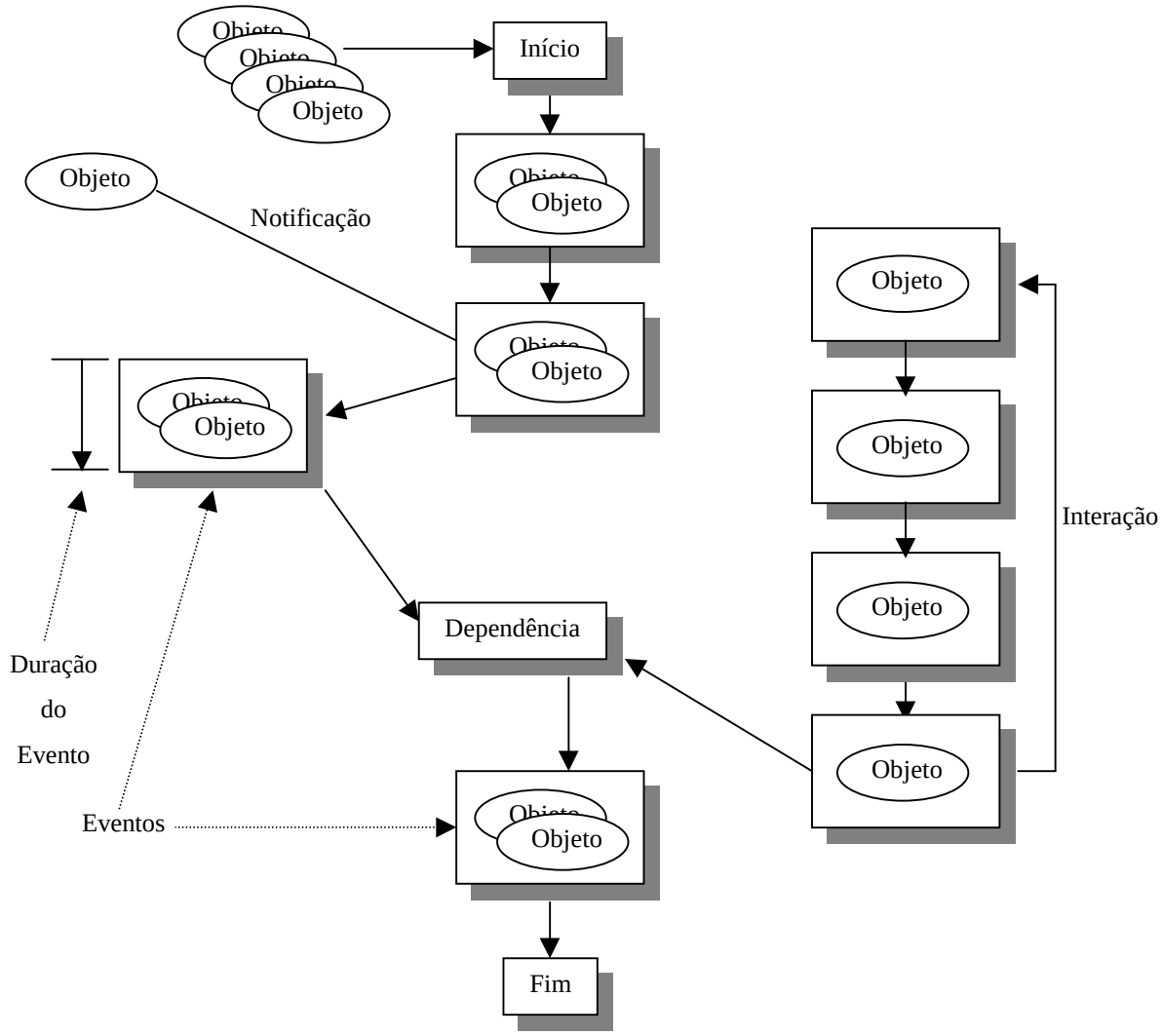

Figura 4.1 – Seis tipos de eventos

4.3 *Workflow* e Equipes Virtuais

Os agentes exercem função de uma equipe virtual, e segundo Aurélio Buarque de Holanda a palavra virtual significa:

"*Diz-se do que está predeterminado e contém todas as condições essenciais à sua realização"*.

Para que essa equipe virtual tome forma e exista é necessário que se cumpram certas condições:

Objetivo – é a essência da equipe, a razão de sua existência;

Tempo – assume proporções dramáticas, a tecnologia *Workflow* pode resolver quase todos os problemas de sincronismo existentes entre as diversas atividades que compõem um processo desse tipo. Pela primeira vez, uma tecnologia é capaz de juntar toda a informação que equipes virtuais necessitam para trabalhar, e faz isso dentro de um contexto de tempo e regras precisas de processamento. A tecnologia *Workflow* pode contribuir para manter os prazos dentro do programado, fazer com que todos os documentos sejam processados segundo as regras preestabelecidas e manter as rotas do fluxo de trabalho de acordo com o mapa criado para o processo;

Tecnologia – é a forma de como essas equipes virtuais se comunicarão entre si e com o sistema propriamente dito de forma consistente e precisa seguindo os conceitos acima. Essa tecnologia será representada pela Internet e pelos Agentes.

"*É a tecnologia Internet que mais impacto vai trazer para qualquer sistema Workflow, pois além de permitir a qualquer empresa operar seus processos onde quer que esteja, vai permitir que qualquer estrutura organizacional possa ser dinamicamente suportada por esta tecnologia"* [CRUZ, 1998].

4.4 Níveis do *Workflow*

Existe um número até que razoável de fabricantes de produtos para *Workflow*, com os mais variados níveis e abrangências. Muitos fabricantes, entretanto, insistem em dizer que seus produtos são "*full*" *Workflow* quando, na verdade, preenchem apenas um ou outro nível da essência *Workflow*.

A Figura 4.2 mostra os possíveis níveis de *Workflow*:

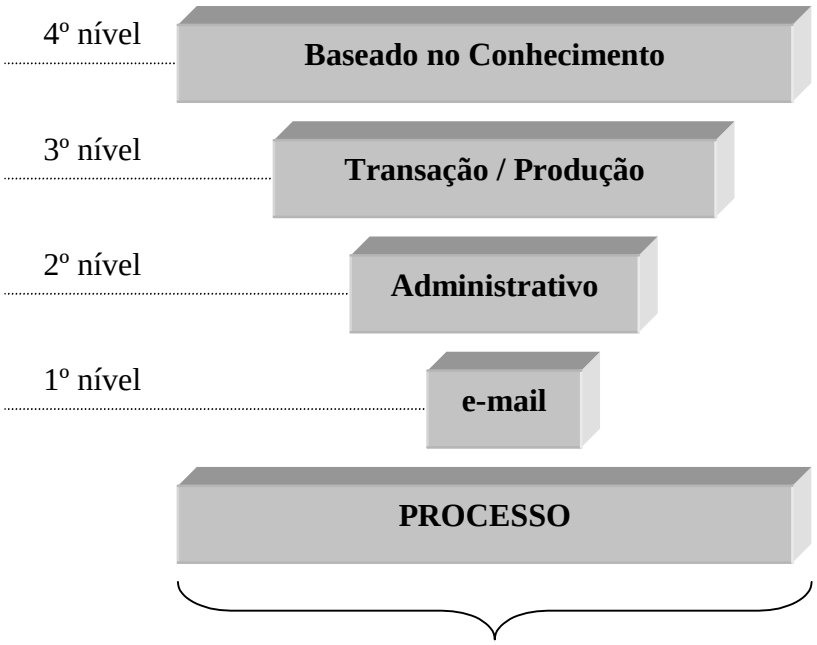

Figura 4.2 – Os 4 níveis de um Sistema Workflow

4.4.1 e-mail

O nível mais elementar, que mostra a abrangência do *Workflow* ao processo produtivo de qualquer empresa, é o nível *e-mail*. O tipo *e-mail* é caracterizado somente pelo fluxo de informações através de sistemas de correio eletrônico (agendamentos, distribuição da informação para grupos, pedidos, dentre outros). Nesse nível existem muitos e bons *softwares*, mas o mesmo não se pode dizer de produtos mais avançados.

4.4.2 Administrativo

Esse nível é um pouco mais complexo que o nível orientado para *e-mail* e mais simples que o orientado para transações. Este nível tem características de sistemas de correio eletrônico, mas com algumas capacidades a mais, o que o torna ideal para tratamento de documentos e formulários que servem de suporte para rotinas que, embora repetitivas, e aparentemente sem complexidade, precisam ser realizadas corretamente. Em princípio, todos os *softwares* que tratem do fluxo de documentos e formulários podem ser classificados como *Workflow* Administrativo.

4.4.3 Transação / Produção

Um *software* de 3º nível para *Workflow* deve conter os seguintes módulos:

- *Toolkit* (caixa de ferramentas);
- Administração;
- Linguagem de Programação SQL-GUI;
- Desenvolvimento de Formulários;
- Agenda;
- *e-mail*;
- Organizador eletrônico de documentos.

Uma das características mais importantes dos *softwares* de *Workflow* de terceiro nível é a sua capacidade de ser *aplication-independent* (independente de aplicação). Isso quer dizer que um *software* de *Workflow* tem a capacidade de interagir com qualquer aplicação existente na empresa sem estar amarrado a ela.

Este é o nível que será trabalhado em sistemas ERP, para integração de ferramentas para Comércio Eletrônico.

4.4.4 Baseado no Conhecimento

O último tipo de *Workflow* é baseado no conhecimento. Isso significa dizer que ele tem características e ferramentas que permitem aprender com seus próprios erros e acertos. Qualquer sistema baseado no conhecimento tem tecnologia para ir além da execução pura e simples das regras preestabelecidas e incorporar exceções a seus procedimentos.

Inteligência Artificial é uma das tecnologias que permitem esse tipo de sistema. Comercialmente não existe um sistema *Workflow* deste nível. No dia que surgir um sistema deste nível, estará quebrado mais um paradigma na gestão de processos.

4.5 Implantação de uma Ferramenta

Para implantar uma ferramenta em sistemas ERP (que são baseados na tecnologia *Workflow*), um software de *Workflow* tem que ser construído dentro das especificações cliente-servidor.

Uma preocupação com o gerenciamento das rotas e a verificação do andamento dos trabalhos permite um gerenciamento proativo de todas as ocorrências, casos ou instâncias, tratadas dentro do *Workflow*.

4.5.1 1ª Etapa: análise do fluxo de trabalho atual

Para projetar um modelo de informação é preciso partir de alguma realidade. Isto pode ser feito analisando-se o modelo atual a fim de conseguir-se os elementos necessários para projetar o novo fluxo de trabalho. Em síntese pode-se dizer que são necessários três grandes momentos nessa análise:

Início – Como o processo é executado hoje;
Meio – O que precisa ser implantado, ou aperfeiçoamentos a implantar;
Fim – Modelo conceitual do novo processo.

4.5.2 2ª Etapa: projetar o modelo de informação do que se quer automatizar

Primeiramente é necessário entender alguns conceitos de *Workflow* para melhor compreensão das etapas que serão descritas posteriormente:

Modelo de Informação

"É o conjunto de objetos que fazem parte de um fluxo de trabalho e que têm por função dar-lhe vida por meio da automatização dos procedimentos. O modelo de informação descreve a estrutura e os atributos dos vários tipos de

objetos, tais como formulários, documentos, pastas e tudo que se relacione com eles, envolvidos no procedimento que a empresa quer automatizar" CRUZ (1998).

Procedimento

"*é o modelo computadorizado de um fluxo de trabalho desenvolvido para aprimorar os controles usados no gerenciamento de tempos e responsabilidades envolvidos na obtenção de um objetivo*" CRUZ (1998).

Como ajuda, o roteiro de trabalho a seguir descreve as etapas descritas que devem ser cuidadosamente levantadas.

- Início – sob quais condições o processo se inicia;
- Programação – descrever os tempos máximo e mínimo permitidos para cada atividade;
- Pré-atividade – estabelecer o montante em que determinada pessoa pode iniciar atividade certa;
- Execução – descrever tipos de ferramentas, aplicações, metodologias e técnicas usadas para processar um item de trabalho;
- Notificação – descrever as condições sob as quais as pessoas serão notificadas sobre um evento do processo;
- Pós-atividade – estabelecer o momento em que determinada pessoa pode completar certa atividade e o que deve ocorrer quando ela for terminada;
- Segurança – descrever quem estará autorizado a participar do processo, que funções será autorizado a executar e que informações poderão ser manuseadas;
- Auditoria – descrever quais eventos serão auditados e em que nível de detalhamento;
- Término – descrever as condições em que o processo deverá terminar ou ser interrompido.

Para projetar o modelo de informação, as maiorias dos *softwares* de *Workflow* necessitam da definição dos elementos descritos a seguir.

4.5.2.1 Objetivo do Procedimento

Qual o objetivo a ser atingido? Esse é o objetivo que deve servir de guia para o projeto de implantação. É em resumo, o porquê da existência do procedimento que vai ser automatizado.

O que é um objetivo para *Workflow*? Para *Workflow*, o objetivo é a gestão do fluxo de trabalho por meio de seus documentos a fim de torná-lo mais ágil, seguro, eficiente e eficaz. Um objetivo pode ser simples ou múltiplo.

4.5.2.2 Papéis do *Workflow*

Depois de definido o objetivo, ou os objetivos, do procedimento, faz-se necessário definir quem participará do ambiente do fluxo de trabalho, isto é, quem serão os "atores" que executarão as tarefas necessárias para que cada etapa do procedimento seja cumprida. Os papéis podem ser tanto individuais como grupais.

4.5.2.3 Rota do Procedimento

É a rota que os documentos, formulários, instruções, a informação, enfim, devem percorrer para que o procedimento tenha vida. Existem vários tipos de rota, seqüencial, paralela, condicional, dinâmica, mas, no geral, elas podem ser divididas em dois grandes grupos, simples e compostas.

4.5.2.4 Documentos e Formulários

No ambiente *Workflow*, documentos e formulários são veículos para os dados do caso, ou instância, que devem ser processados para que o trabalho seja realizado. Esses elementos podem conter vários tipos de dados, textos, imagens, planilhas, som, sendo o tipo multimídia mais utilizado atualmente por ser o mais sofisticado, permitindo

englobar num único caso, ou instância, os vários tipos de dados necessários a um procedimento.

4.6 Realidade

A conexão entre *Workflow* e Internet é uma realidade. Os *softwares* mais sofisticados permitem tanto integrar os processos de negócio de uma empresa com o mundo e seus clientes, como realizar o “roteamento” integral entre as atividades que estejam fora e dentro da Internet com segurança e eficiência.

A partir de qualquer ponto do planeta, sem necessidade sequer de estar usando um *software Workflow*, um usuário pode ativar um processo de negócio desenhado com a tecnologia *Workflow*. A interação desse usuário pode se dar com um Serviço de Atendimento ao Consumidor, com atividades que programem Ordens de Produção, Ordens de Compra ou, literalmente, qualquer atividade que faça parte do fluxo automatizado por meio de um *software* desse tipo. Essa característica permite às empresas abrirem suas portas ao mundo, para oferecer um melhor serviço a seus clientes, sem descuidar da segurança que uma abertura desse tipo necessita ter.

5 LÓGICA NEBULOSA

Parece que no mundo real, tudo é uma questão de ponto de vista ou de graduação, ou seja, tudo depende. O mundo real não é bivalente, é na realidade multivalente com um infinito espectro de opções em vez de duas.

Verdade absoluta e precisão existem apenas como “casos extremos”, a insistência nos extremos como válidas deve-se apenas a uma polarização cultural, nada mais. Assim, o objetivo da lógica nebulosa, também chamada de lógica *fuzzy*, é o de capturar esses tons de cinza e graus de verdade. A lógica nebulosa, trabalha, com tal incerteza e verdade parcial os fenômenos naturais, de uma maneira sistemática e rigorosa.

A lógica *fuzzy* é uma técnica que incorpora a forma humana de pensar em um sistema de controle. Um controlador *fuzzy* típico pode ser projetado para comportar-se conforme o raciocínio dedutivo, isto é, o processo que as pessoas utilizam para inferir conclusões baseadas em informações que elas já conhecem.

Embora as estratégias de controle inteligente possam ser implementadas por outros meios, as implementações por lógica *fuzzy* freqüentemente se tornam muito mais eficientes, devido aos seguintes pontos:

- Estratégias de controle *fuzzy* nascem da experiência e de experimentos, em vez de modelos matemáticos. Portanto, uma implementação lingüística é muito mais rápida de se implementar.
- Estratégias de controle *fuzzy* envolvem um número muito grande de entradas, a maioria das quais relevantes apenas para condições especiais. Tais entradas são ativadas apenas quando condições específicas prevalecem. Dessa forma, algumas condições raras ou excepcionais podem ser incorporadas com pouquíssimo *overhead* computacional, e ainda assim a estrutura de software permanece transparente e compreensível.
- Estratégias de lógica *fuzzy* implementadas em produtos comerciais, voltadas para o mercado de massa, devem ser de custo baixo. Comparadas com soluções convencionais de controle, a lógica *fuzzy* é freqüentemente mais eficiente, ao se comparar a codificação e o tempo computacional de execução.

Em Princípio da Incompatibilidade, Zadeh afirma que "*Conforme a complexidade de um sistema aumenta, nossa habilidade de fazer afirmações precisas e significativas sobre seu comportamento diminui, até um limiar em que a precisão e relevância tornam-se praticamente características mutuamente exclusivas*."

A lógica clássica aristotélica é bivalente, isto é, reconhece apenas dois valores: verdadeiro ou falso. A lógica *fuzzy* é multivalente, isto é, reconhece uma multitude de valores e, como a operação das atividades humanas, requer uma aproximação de dados e informações sensoriais, através de termos vagos ou imprecisos. O cérebro humano codifica tais imprecisões naturais através de conjuntos e números *fuzzy*. Um vendedor não precisa de um valor exato e definido para uma variável, por exemplo, classificação do cliente. Tal vendedor consegue classificar a informação de classificação do cliente em **conjuntos**, tipo RUIM, BOM, ÓTIMO. Esses valores representam valores "fuzzyficados" dos valores exatos da classificação do cliente. A seguir, o vendedor formula e executa uma estratégia de venda baseada na compreensão de cada variável de entrada e de saída, o fluxo de dados no cérebro fica reduzido apenas ao que é necessário para se executar a tarefa requerida com a precisão e resolução necessárias; assim o operador humano processa as quantidades *fuzzy*, chegando a uma variável *fuzzy* com sua ação de controle. O ser humano naturalmente trabalha com características incertas, mas as máquinas, equipamentos e controles industriais precisam de um número real que represente o valor de referência necessário. Dessa maneira, é necessário um processo de conversão do valor *fuzzy* – resultante da saída da inferência – para um número real; esse processo é chamado por defuzzyficação.

Expressões verbais, imprecisas, qualitativas, inerentes da comunicação humana, que possuem vários graus de incerteza, são perfeitamente manuseáveis através da lógica nebulosa.

Um conjunto *fuzzy* é um agrupamento impreciso e indefinido, onde a transição de não-pertinência para pertinência é gradual, não abrupta. A incerteza de um elemento, isto é, seu grau fracionário de pertinência, pode ser concebido como uma medida de possibilidade, ou seja, a possibilidade de que um elemento seja membro do conjunto.

A agregação e combinação de todas as regras *fuzzy* constituem a chamada base de regras do controlador, que é depositária de toda a inteligência que executará as desejadas ações de controle, sob as condições especificadas.

A construção da base de conhecimentos é sempre a primeira fase no projeto de um sistema inteligente. A segunda fase contém o uso da base de conhecimentos, onde através da aplicação de várias entradas pode-se produzir uma saída.

Um controlador *fuzzy* é composto dos seguintes blocos funcionais:

- interface de fuzzyficação;
- base de conhecimento;
- lógica de tomada de decisões;
- interface de defuzzyficação.

5.1 Fuzzyficação

Os valores discretos das variáveis de entrada geralmente são provenientes de sensores das grandezas físicas ou de dispositivos de entrada computadorizados. Um fator de escala pode ser usado para converter os valores reais de entrada para outros que sejam cobertos pelos universos de discurso pré-definidos para cada variável de entrada. Ademais, a interface de fuzzyficação usa funções de pertinência contidas na base de conhecimento, convertendo os sinais de entrada em um intervalo [0,1] que pode estar associado a rótulos lingüísticos.

5.2 Base de conhecimento

A base de conhecimento representa o modelo do sistema a ser controlado. Consistindo de uma base de dados (funções de pertinência lingüísticas) e uma base de regras *fuzzy* lingüísticas. A base de dados fornece as definições numéricas necessárias às funções de pertinência usadas no conjunto de regras *fuzzy*. A base de regras caracteriza os objetivos de controle e a estratégia de controle utilizadas por especialistas na área, por meio de um conjunto de regras de controle em geral lingüísticas.

5.3 Lógica de tomada de decisões

A lógica de tomada de decisões, incorporada na estrutura de inferência da base de regras, usa implicações *fuzzy* para simular tomadas de decisões humanas. Ela gera ações de controle – conseqüentes – inferidas a partir de um conjunto de condições de entrada – antecedentes.

5.4 Defuzzyficação

A defuzzyficação consiste em obter-se um único valor discreto, utilizável numa ação de controle concreta no mundo real, a partir de valores *fuzzy* de saída obtidos. Este único valor discreto representa um compromisso entre os diferentes valores *fuzzy* contidos na saída do controlador.

Esta função é necessária apenas quando a saída do controlador tiver de ser interpretada como uma ação de controle discreta. Existem sistemas que não exigem defuzzyficação porque a saída *fuzzy* é interpretada de modo qualitativo.

Funções de pertinência *fuzzy* representam os aspectos fundamentais de todas as ações teóricas e práticas de sistemas *fuzzy*. Uma função de pertinência é uma função numérica gráfica ou tabulada que atribui valores de pertinência *fuzzy* para valores discretos de uma variável, em seu universo de discurso.

Embora as funções de pertinência utilizadas sejam triangulares e trapezoidais, isto não é obrigatório. A quantidade de funções em um universo de discurso e seu formato são escolhidos com base na experiência, ou na natureza do processo a ser controlado. De modo geral, esta não é uma tarefa trivial.

Um número prático de conjuntos *fuzzy* lingüísticos (funções de pertinência) é algo entre 2 e 7. Quanto maior o número de conjuntos, maior a precisão, mas a demanda computacional também é mais significativa. Experiências mostraram que a partir de valores maiores que 7 conjuntos não há melhorias extremamente significativas.

Os formatos mais freqüentemente encontrados são triângulos e trapezóides, pois são gerados com facilidade. Em casos onde um desempenho suave é de importância crítica, funções do tipo $\cos^2(x)$, gaussiana, sigmóide e spline cúbico (S-shape) podem ser usadas.

Outro fator que afeta a precisão é o grau de superposição entra as funções de pertinência *fuzzy*. Um mínimo de 25% e uma máximo de 75% foram determinados experimentalmente como adequados, sendo 50% um compromisso razoável, pelo menos para os primeiros testes num sistema em malha fechada. Certamente, essa superposição surge como resultado, não como parâmetro de entrada. A propriedade inerente de interpolação da lógica nebulosa se dá em parte à superposição entre as funções de pertinência *fuzzy*. Funções trapezoidais com patamares largos e laterais superpostas bastante íngremes podem ser usadas, onde a saída *fuzzy* não é sensível a mudanças nos valores de entrada que recaiam sob as porções constantes dos trapezóides.

Devido ao fato de que projetar funções de pertinência é uma tarefa trabalhosa, técnicas especiais usando redes neurais e/ou algoritmos genéticos estão disponíveis para sua geração automática. Tais sistemas são dotados de uma capacidade de aprendizagem, a partir dos conjuntos de dados de entrada, com a qual identificam a posição e formato das funções de pertinência.

Na defuzzyficação, o valor da variável lingüística de saída inferida pelas regras *fuzzy* será traduzido num valor discreto. O objetivo é obter um único valor numérico discreto que melhor represente os valores *fuzzy* inferidos da variável lingüística de saída, ou seja, a distribuição de possibilidades. Assim, a defuzzyficação é uma transformação inversa que traduz a saída do domínio *fuzzy* para o domínio discreto. Para selecionar o método apropriado de defuzzyficação, pode-se utilizar um enfoque baseado no centróide ou nos valores máximos que ocorrem da função de pertinência resultante. Os seguintes métodos são muito utilizados: (1) Centro-da-Área (C-o-A), (2) Centro-do-Máximo (C-o-M), e (3) Média-do-Máximo (M-o-M).

5.4.1 Centro-da-Área (C-o-A)

O método Centro-da-Área é freqüentemente chamado de método do Centro-de-Gravidade, pois calcula o centróide da área composta que representa o termo de saída *fuzzy* (μ_{out}), esse termo de saída *fuzzy* é composto pela união de todas as contribuições de regras. O centróide é um ponto que divide a área de μ_{out} em duas partes iguais.

O método de defuzzyficação C-o-A apresenta pequenos problemas, um deles ocorre quando as funções de pertinência não possuem sobreposição, onde o centro

geométrico da figura na realidade não deveria ter significado físico, outro fator é que se mais de uma regra tiver a mesma saída *fuzzy* há uma sobreposição de áreas que não é devidamente contabilizada, além disso a necessidade de integração numérica toma esforço computacional para cálculo.

5.4.2 Centro-do-Máximo (C-o-M)

Neste método os picos das funções de pertinência representados no universo de discurso da variável de saída são usados, enquanto ignora-se as áreas das funções de pertinência; as contribuições múltiplas de regras são consideradas por esse método. Os valores não-nulos do vetor de possibilidades de saída são posicionados nos picos correspondentes. Assumindo que representam pesos, o valor de saída defuzzyficado, discreto, é determinado achando-se o ponto de apoio onde os pesos ficam equilibrados. Assim, as áreas das funções de pertinência não desempenham nenhum papel e apenas os máximos (pertinências singleton) são usados. A saída discreta é calculada como uma média ponderada dos máximos, cujos pesos são os resultados da inferência.

Onde $\mu_{O,k}(ui)$ indicam os pontos em que ocorrem os máximos (alturas) das funções de pertinência de saída. Esse método também é chamado por método de defuzzyficação pelas alturas.

Como pode-se perceber, as equações C-o-A e C-o-M são extremamente semelhantes, exceto que a equação C-o-A usa as áreas de cada função de pertinência, enquanto a equação C-o-M usa apenas seus máximos. Naturalmente, os resultados defuzzyficados obtidos usando-se os dois métodos serão ligeiramente diferentes. Essa abordagem representa um melhor compromisso entre possíveis saídas com multiplicidade de disparo de conjunto *fuzzy*. Ou seja, se três regras forem acionadas, duas impondo saída X e uma impondo saída Y, a saída X fica reforçada nesse método de defuzzyficação pelas alturas.

5.4.3 Média-do-Máximo (M-o-M)

Uma abordagem para defuzzyficação poderia ser a de utilizar a saída cujo valor tenha o maior valor de pertinência $\mu_{out}(ui)$. Em casos onde a função de pertinência tenha mais de um máximo essa idéia não poderia ser utilizada. A abordagem C-o-M também não funcionaria bem, devido à necessidade de se escolher qual máximo utilizar. Pode-se então tomar-se a média de todos os máximos:

Onde u_m é o m-ésimo elemento no universo de discurso, onde a função $\mu_{out}(ui)$ tenha um máximo e M é o número total desses elementos. A abordagem M-o-M é também chamada de solução mais plausível; por desconsiderar o formato da funções de pertinência de saída.

5.4.4 Método de Defuzzyficação a ser Utilizado

Aplicações em malha fechada: a propriedade de continuidade, é importante, pois se a saída de um controlador fuzzy controla uma variável do processo, saltos na saída do controlador podem causar instabilidade e oscilações, logo é prudente optar pela defuzzyficação C-o-M.

Reconhecimento de padrões: pode-se usar o método M-o-M, porque se deseja-se identificar objetos pela classificação do sinal de um sensor, o resultado mais plausível é interessante. O vetor de possibilidades de saída é o resultado da classificação, pois ele contém informações sobre a similaridade entre o sinal e os objetos-padrão.

Suporte à decisão: a escolha do método de defuzzyficação depende do contexto da decisão. Decisões quantitativas, como alocação de recursos, priorização de projetos, ou cálculo do desconto a ser dado em um determinado produto, podem usar o C-o-M, enquanto o M-o-M é recomendado para decisões qualitativas, como detecção de fraude em cartões de crédito e avaliação de crédito.

Um método de defuzzyficação é dito contínuo se uma mudança infinitamente pequena numa variável de entrada não causa nunca uma mudança abrupta em nenhuma das variáveis de saída.

Os métodos de defuzzyficação C-o-M e C-o-A são ambos contínuos, já o método M-o-M é descontínuo. Isso se deve ao fato de que o melhor compromisso jamais pode saltar para um valor diferente com uma pequena mudança nas entradas. Por outro lado, a solução mais plausível não é única e pode assim saltar para um valor diferente.

É importante lembrar que funções de pertinência complexas não apresentam resultados melhores para valores de saída. Como sugestão prática utilize sempre que possível funções triangulares. Afinal, os métodos de defuzzyficação C-o-M e M-o-M usam apenas os máximos das funções de pertinência.

6 AGENTES

Uma força que ajuda o desenvolvimento desta pesquisa é que as pessoas querem vários tipos de assistência ou assistentes (agentes). Note que um agente não é necessariamente uma interface entre o computador e o usuário.

Os agentes podem assistir o usuário de diversas formas: eles ocultam a complexidade e a dificuldade das tarefas, eles executam tarefas em cima do comportamento humano, eles podem treinar ou até mesmo ensinar o usuário, e monitoram eventos e procedimentos.

6.1 Definição

Pela perspectiva do usuário final, um agente é um programa que o ajuda a realizar um determinado serviço ou recebe tarefas que lhe são delegadas, já para o sistema, um agente é um objeto de software que está situado em um ambiente de execução e possui obrigatoriamente propriedades que serão vistas ao longo deste capítulo.

Existem vários sinônimos para essa terminologia agentes: *knowbots*, *softbots*, *taskbots*, *userbots*, *robots*, *personal agents*, *autonomous agents*, *personal assistants*.

Pela variedade de funções que os agentes podem exercer, existem também uma série de adjetivos para diferenciá-los: *search agents*, *report agents*, *presentation agents*, *navigation agents*, *role-playing agents*, *management agents*, *search and retrieval agents*, *domain-specific agents*, *development agents*, *analysis and design agents*, *testing agents*, *packaging agents* e *help agents* SOFTWARE AGENTS (1999).

Essa variedade de terminologias e funções acabam por causar muita confusão por ser um tema tão vasto e complexo, que mesmo alguns estudiosos, usam várias definições para este assunto. Abaixo seguem algumas definições de importantes pesquisadores da área de agentes.

“Agentes de software são programas que ajudam pessoas e atuam do seu lado. A função dos agentes é permitir que as pessoas deleguem tarefas a eles” LANGE (1998).

"É um componente de software ou hardware capaz de agir em ordem para a realização de tarefas em favor do usuário" SOFTWARE AGENTS (1999).

"Uma entidade de software com um programa contido em si próprio que tem a capacidade de controle de sua própria criação e ação, baseado na percepção do seu ambiente, e com isso perseguir um ou mais objetivos" JENNINGS (1996).

"Agentes são programas de computadores que empregam técnicas de Inteligência Artificial para fornecer assistência ativa para usuários em tarefas baseadas no computador" MAES (1994).

"Agentes autônomos são sistemas computacionais que residem em ambientes dinâmicos e complexos, sentem e agem autonomamente neste ambiente, e fazendo isto executam um conjunto de metas ou tarefas para os quais são projetados" MAES (1995).

"Tudo o que opera, o que trata de negócio por conta alheia; causa; o promotor; o que pratica a ação" BUENO (1996).

"Um agente autônomo é um sistema situado dentro de um ambiente e uma parte deste sente aquele ambiente e atua sobre ele, todo tempo, perseguindo sua agenda e assim afetando sua percepção futura" FRANKLIN 96.

Os conceitos de agentes discutidos neste trabalho vão de encontro aos estudos e definições de LANGE (1998). Não existe uma definição única e correta, todas as definições aqui apresentadas se referem a área de estudo que se concentram seus pesquisadores, em vista disso, analisemos essas peculiaridades.

6.2 Tipologia dos Agente

Primeiramente, os agentes podem ser classificados pela sua mobilidade, i.e. sua habilidade de mover na rede. Desta forma podemos defini-los em duas classes: estacionários ou móveis.

Segundo, eles podem ser classificados em deliberativos ou reativos. Agentes deliberativos possuem simbolismo interno, modelo racional e dedicam-se ao alcance da coordenação com outros agentes. Agentes reativos, ao contrário, não tem nenhum modelo simbólico do seu ambiente, e agem usando estímulos em resposta ao estado do ambiente em que se encontram.

Terceiro, os agentes podem ser classificados segundo as características que exibem. Na BT Labs, três foram identificados: autônomos, aprendizes e cooperativos. Autônomo se refere ao princípio de que o agente pode operar sozinho sem interferência humana. Aprendizes têm a característica de "aprender" conforme eles reagem ou interagem com o ambiente externo. E Cooperativos têm como princípio cooperar com outros agentes, os agentes precisam de uma habilidade "social", i.e. a habilidade de interagir com outros agentes e possivelmente usuários através de alguma linguagem de comunicação SOFTWARE AGENTS (1999).

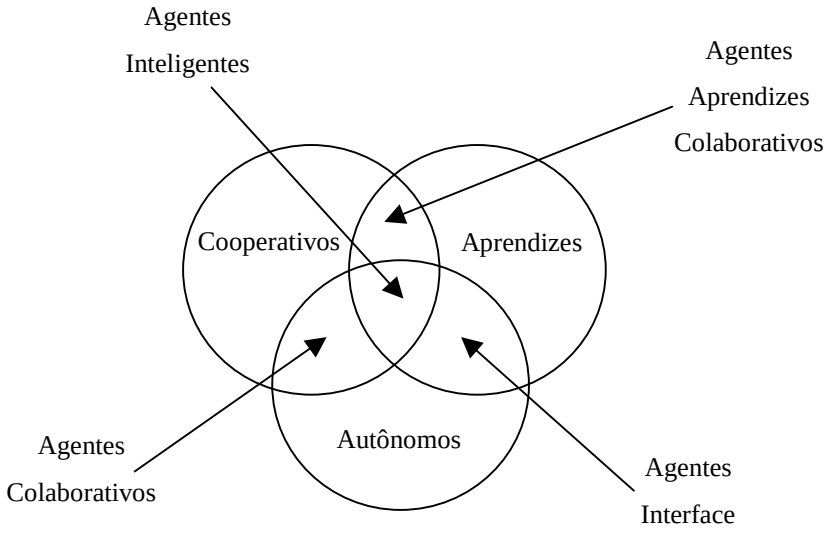

Figura 5.1 – Visão genérica de uma Tipologia de Agentes

Vale ressaltar que essas distinções não são definitivas. Por exemplo, com agentes colaborativos, há mais ênfase na cooperação e autonomia que no aprendizado; portanto, não implica que agentes colaborativos nunca aprendem.

Uma visão ideal, seria em que os agentes demonstrassem as três características igualmente bem, mas isso é mais uma aspiração do que uma realidade. Agentes realmente inteligentes não existem ainda.

Quarto, agentes podem ser classificados pelas suas regras, e.g. *world wide web* (WWW) agentes Internet ou de informação. Essa categoria de agentes usualmente exploram os sites de pesquisa como *WebCrawlers*, *Lycos* e *Spiders*.

Quinto, pode ser incluída também a categoria de agentes híbridos, os quais combinam duas ou mais filosofias de agentes em um único agente.

Na verdade, os agentes existem em um espaço multi-dimensional, mas para não ficar incompleto ou impreciso, foram identificados sete tipos de agentes SOFTWARE AGENTS (1999):

- Agentes Colaborativos;
- Agentes Interface;
- Agentes Móveis;
- Agentes Internet/Informativos;
- Agentes Reativos;
- Agentes Híbridos;
- Agentes Heterogêneos.

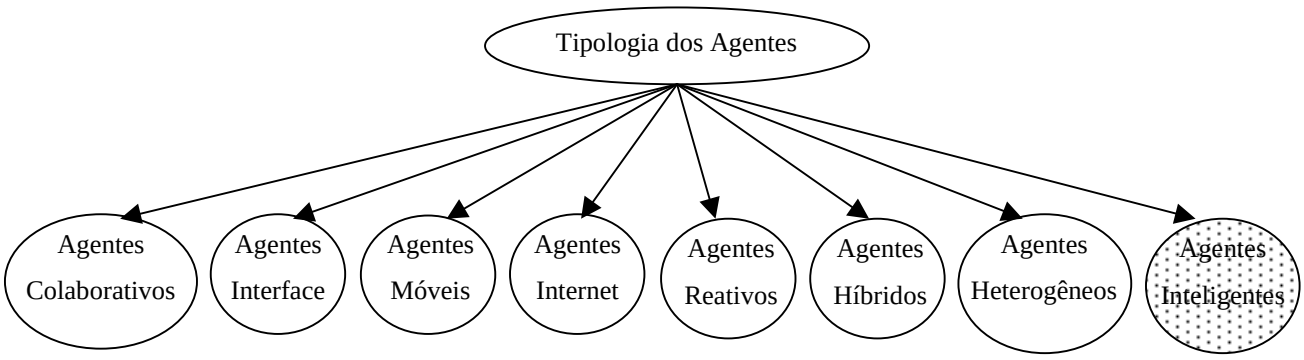

Figura 5.2 – Classificação dos Agentes

6.2.1 Agentes Colaborativos

Enfatizam autonomia e cooperação com outros agentes para a realização de uma ou mais tarefas. Eles podem aprender, mas não é um aspecto típico de suas operações. Para haver uma coordenação entre os agentes colaborativos, eles precisam negociar entre eles, para juntos, realizarem determinada tarefa.

Resumindo, a principal característica desses agentes incluem autonomia, habilidade social, sensibilidade e pró-atividade. Portanto, eles são hábeis em agir racionalmente e autonomamente em ambientes com múltiplos agentes. Tendem a ser estáticos, podem ser benevolentes, racionais, confiáveis, algumas combinações dessas características ou nenhuma delas.

6.2.1.1 Motivação

- São bons para resolver problemas que são muito grandes para um único agente;
- Permitem interconexão e interoperabilidade de múltiplos sistemas;
- Provêem soluções para problemas distribuídos;
- Utilizados para alcançar modularidade (reduz complexidade), velocidade (paralelismo), confiabilidade (redundância), flexibilidade (novas tarefas são criadas mais facilmente por causa da organização modular) e reusabilidade.

6.2.2 Agentes Interface

Enfatizam autonomia e aprendizado para a realização de uma ou mais tarefas. É um agente que opera como “assistente pessoal”, colaborando com o usuário no mesmo ambiente de trabalho.

Agentes interface dão assistência e suporte, tipicamente para usuários que estão aprendendo uma aplicação em particular. O agente observa e monitora as ações realizadas pelo usuário, aprende novos caminhos, e os sugere melhores formas de se

realizar determinada tarefa. Para o aprendizado, o agente pode assistir o usuário de 4 formas:

- Observando e imitando o usuário;
- Recebendo confirmações positivas ou negativas do usuário;
- Recebendo instruções explícitas do usuário;
- Comunicando-se com outros agentes, pedindo conselhos.

6.2.2.1 Motivação

- Tarefas repetitivas podem ser delegadas ao agente;
- Eliminam o tédio humano em executar várias operações manuais;
- Reduzem o trabalho e sobrecarga de informação;
- Se adapta as preferências do usuário;
- Informações entre diferentes usuários podem ser trocadas (quando os agentes aprendem com outros agentes).

Como os computadores estão sendo utilizados para executar mais tarefas e estão se tornando cada vez mais integrados a mais serviços, os usuários precisarão de ajuda para lidar com a informação e o trabalho sobrecarregado. Agentes Interface mudam radicalmente o estilo de interação usuário-computador MAES (1994).

6.2.3 Agentes Móveis

Agentes móveis são capazes de navegar em redes de grandes distâncias (WAN) como a Internet, interagindo com agentes estrangeiros, coletando informações e retornando tendo realizado as operações definidas pelo usuário. Contudo mobilidade não é uma condição suficiente, agentes móveis são agentes porque são autônomos e cooperativos.

6.2.3.1 Evolução

Se comparar os paradigmas da computação em rede (cliente-servidor, código sob demanda e agentes móveis), pode-se perceber uma escala evolutiva numa tendência de maior flexibilidade, onde o cliente e o servidor se fundem em uma única estação.

No paradigma cliente-servidor, este último possui o *know-how*, os recursos e o processamento, ou seja, o código que implementa os serviços requisitados pelo cliente estão armazenados localmente no servidor. São exemplos de tecnologias que utilizam esse paradigma: *Object Request Broker* (ORB), *Remote Method Invocation* (RMI) e *Remote Procedure Call* (RPC).

Já no paradigma código sob demanda, o cliente obtém o *know-how* apenas quando necessário, ou seja, o know-how se encontra no servidor, quando um cliente precisa realizar uma tarefa ele faz o *download* do *know-how* (código) necessário para a realização da mesma. Dessa forma, o cliente possui os recursos e o processamento enquanto o servidor possui todo o *know-how*. São exemplos de tecnologias que utilizam esse paradigma: *Applets* e *Servlets* Java.

E por último o paradigma dos agentes móveis, onde cada estação da rede possui um alto grau de flexibilidade, visto que há uma mistura de *know-how*, recursos e processamento em cada uma delas. Neste paradigma o *know-how* não se encontra concentrado como nos outros, mas sim distribuído entre as estações da rede. *Aglet* é um exemplo de tecnologia que utiliza este paradigma, e por isso será vista com mais detalhes no capítulo 6.

6.2.3.2 Estado da Arte

a) *Voyager* da ObjectSpace

Da mesma forma que aplicações Java precisam executar em uma máquina virtual Java (JVM), agentes precisam executar em cima de algum ambiente. *Voyager* é uma plataforma da ObjectSpace para computação distribuída acrescida de agentes em Java. Para se ter uma idéia melhor de como o ambiente interage com a sua aplicação veja Figura 5.3 adiante:

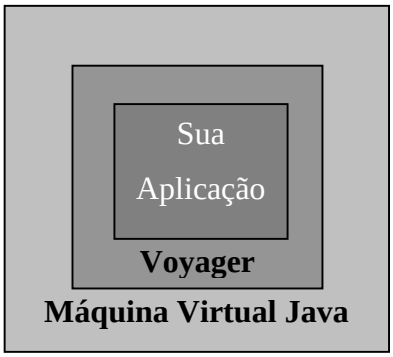

Figura 5.3 – Relação entre a JVM, o Voyager e sua Aplicação

É uma plataforma que possui um grande conjunto de objetos com capacidade de mensagem, permite também que os objetos movam-se (*com.objectspace.voyager.mobility*) como agentes em uma rede. Pode-se criar aplicações usando as técnicas de programação tradicional e distribuída acrescida de agentes, visto que *Voyager* combina as propriedades de um ORB baseado em Java com o sistema de agentes móveis MCCARTY (1999).

b) *Aglets* da IBM

São agentes Java móveis que suportam o conceito de autonomia de execução e roteamento dinâmico de itinerário. São armazenados por um servidor *Aglet* da mesma forma que os *applets* o são em um navegador *Web*. Enquanto o servidor *Aglet* fornece um ambiente para os *aglets* executarem, a máquina virtual Java (JVM) e o gerente de segurança *Aglet* o tornam seguro para receber e hospedar *aglets*. O termo *aglet* (*lightweight agent*) é uma combinação de *agent* e *applet* (*lightweight application*), por isso pode-se pensar no *aglet* como uma generalização e extensão dos *applets* e *servlets* Java VALDO (1999).

6.2.3.3 Aplicações

Os agentes móveis por possuírem essa autonomia de execução e roteamento dinâmico podem ser bem aplicados em áreas de grande interesse, dentre elas o comércio eletrônico, redes de telecomunicação e *Workflow*.

Uma transação comercial necessita de acesso em tempo-real a recursos remotos, diferentes agentes teriam diferentes metas e poderiam implementar e exercer diferentes estratégias para alcançar estes objetivos.

O tamanho físico das redes de telecomunicações e as necessidades restritas sobre os quais eles operam, faz com que o suporte e gerenciamento de serviços de telecomunicações avançada se tornem árduos e difíceis, caracterizados pela reconfiguração de redes dinamicamente e padronização de usuários. Os agentes móveis formam base para que tais sistemas ainda sejam efetivos.

Graças a mobilidade dos agentes móveis, ele fornece um grau de autonomia para o *Workflow* que necessita de suporte aos fluxos de informação na empresa. Possuindo a informação e características necessárias para se movimentarem dentro da organização, independente de alguma aplicação, os agentes móveis exercem a função de "*Workflows* individuais".

6.2.3.4 Elementos do Sistema

Quando se pensa em agentes, deve-se ter bem definido dois conceitos: os agentes em si e seus ambientes ou locais de execução.

Abstraindo a nível de agentes móveis 5 atributos são visíveis: estado, implementação, interface, identificador e mandante. Esses atributos estão sempre com o agente quando em movimento na rede. Vejamos um detalhamento maior desses atributos:

- Estado: geralmente dividido em estado de execução ou estado de objeto, que é o valor das variáveis de instância no objeto. Servem para que o agente possa retomar a execução no computador destino após sua viagem.

- Implementação: garante a independência de execução do agente independente de sua localização. O agente pode ir ao destino e analisar o código existente, recuperando somente o código necessário, ou carregar todo o código consigo.
- Interface: conjunto de assinaturas de métodos que garantem a comunicação do agente.
- Identificador: único durante toda a vida do agente, garantindo assim, reconhecimento e localização de agentes em trânsito na rede através de serviços de diretórios, por exemplo.
- Mandante: é uma entidade cuja identidade pode ser autenticada por algum sistema cujo agente esteja tentando acessar, garantindo assim, a determinação de responsabilidades morais e legais.

Com relação ao contexto onde um agente é executado, 4 conceitos são importantes:

- Máquina: máquina virtual Java (JVM), um ambiente virtual para servir de sistema operacional para os agentes poderem executar suas funções;
- Recursos: redes, banco de dados, processadores, memória, demais *hardrwares* e *softwares* necessários para a execução do agente.
- Localização: é a identificação de onde o agente se encontra, é uma combinação do nome do local de execução e do endereço (IP e porta da máquina) de rede da máquina no qual o local existe.
- Mandantes: tem-se dois em um mesmo local: o construtor, autor do local de implementação, e o seu administrador, mandante que tem a responsabilidade pela operação do ambiente de execução dos agentes.

6.2.3.5 Motivação

- Redução do tráfego da rede: sistemas distribuídos freqüentemente trabalham com protocolos que envolvem múltiplas interações para cumprir uma determinada tarefa. Com os agentes móveis, isto não ocorre, pois estes são despachados para a estação de destino e lá (localmente) executam suas tarefas.

Com isso movemos o processamento para os dados e não os dados para o processamento;

- Diminuição do tempo de atraso na rede: sistemas críticos, muitas vezes necessitam de respostas em tempo-real para mudanças ocorridas em seus ambientes. Controlar tal sistema em uma rede de uma fábrica de um tamanho considerável envolve atrasos significativos. Agentes móveis podem ser despachados de um computador central e executar localmente um controle sobre este sistema, ocasionando um tempo de atraso praticamente nulo;
- Encapsulamento de protocolos: quando dados são trocados em um sistema distribuído, cada estação tem o seu próprio código para implementar os protocolos necessários para codificar os dados que saem e interpretar corretamente os dados que entram. Com a falta de uma política segura de atualizações, estes protocolos freqüentemente tornam-se um problema legado. Agentes móveis, podem mover-se para uma estação e estabelecer “canais de comunicação” baseados em protocolos proprietários;
- Execução assíncrona e autônoma: freqüentemente a comunicação entre dispositivos móveis baseiam-se em conexões de redes muito frágeis; tarefas que necessitam de uma contínua interação podem tornar-se impossíveis. Para solucionar isto, estas tarefas podem ser embutidas em agentes móveis, que são despachados na rede. Depois de transmitido, o agente torna-se independente do processo que o criou e pode operar de forma assíncrona e autônoma. O sistema pode conectar depois de um tempo e coletar o agente com o resultado da tarefa que lhe foi delegada;
- Adaptação dinâmica: agentes móveis tem a habilidade de sentir o ambiente de execução e reagir de forma autônoma e automática a estas mudanças. Caso haja muita urgência para se cumprir uma determinada tarefa, o agente pode multiplicar-se entre as estações de uma rede para manter uma configuração ótima e solucionar mais rapidamente sua tarefa;
- Naturalmente heterogêneos: uma rede de computadores é fundamentalmente heterogênea (tanto *hardware* quanto *software*). Agentes móveis são geralmente independentes da camada de transporte e do computador e

dependentes somente do seu ambiente de execução, o que fornece condições ótimas para integração de sistemas não compatíveis;

- Robustez e tolerância a faltas: a habilidade dos agentes móveis em reagir dinamicamente a situações e eventos desfavoráveis o torna favorável a construção de sistemas distribuídos robustos e tolerantes a faltas;
- Fácil coordenação: pode ser simples coordenar um número de requisições independentes e remotas;
- Alto nível de abstração, tornando fácil o ato de delegar funções e dar manutenções em sistemas já existentes.

6.2.3.6 Algumas Questões

A lista abaixo levanta as maiores questões em agentes móveis, mas que não são exaustivas. Algumas delas já foram respondidas em sub-itens anteriores e outras serão respondidas ao longo do trabalho.

- Transporte: Como um agente se move de um lugar para outro?
- Autenticação: Como garantir que o agente diz quem ele realmente é, e que ele representa quem ele diz estar representando? Como saber se depois de ter navegado em várias redes ele não foi infectado por um vírus?
- Sigilo: Como saber se seus agentes mantêm sua privacidade? Como saber que alguém não usou seu agente para seus próprios ganhos? Como saber se seu agente não foi destruído e seu conteúdo perdido?
- Segurança: Como protegê-lo contra vírus? Como prevenir a u m agente não entrar num laço sem fim e consumir todos os ciclos da CPU?
- Performance: Qual seria o efeito de ter centenas, milhares ou milhões de agentes numa WAN?
- Interoperabilidade / Comunicação: Como localizar serviços específicos? Como se comunicar com agentes escritos em outras linguagens?

6.2.4 Agentes Internet / Informativos

Eles existem para nos ajudar a gerenciar e administrar esse crescimento explosivo da informação. Esses agentes gerenciam, comparam ou manipulam informações de diferentes origens.

Os agentes Internet ou Informativos são caracterizados usando um critério diferente, eles são definidos pelo que fazem, em contraste com os agentes colaborativos ou de interface que são definidos pelo que são.

6.2.4.1 Motivação

- Necessidade de ferramentas para administrar essa explosão de informações;
- Benefício econômico (empresas de *browsers* investem milhões nessa área).

6.2.5 Agentes Reativos

Representam uma categoria especial de agentes que não possuem um modelo simbólico do ambiente; no entanto, respondem em estímulos para o estado presente do ambiente no qual eles se encontram. Contudo, o ponto mais importante a ser notado nesses agentes não é esse (i.e. linguagem, teoria ou arquitetura), mas o fato deles serem relativamente simples e interagirem com outros agentes de forma básica.

Agentes reativos não possuem uma representação simbólica explícita, não possuem uma razão explícita e uma funcionalidade emergente. São simples, fáceis de entender, e sua “cognição” é baixa; isso porque eles precisam “lembrar” pouco. Eles são situados, i.e. não fazem projeções ou revisam algum modelo, suas ações dependem no que acontece no presente momento.

6.2.5.1 Motivação

- Simples de entender;
- Podem ser usados para simular muitos mundos artificiais.

6.2.6 Agentes Híbridos

Já foram vistos 5 tipos de agentes: colaborativos, interface, móveis, internet e reativos. Debates para descobrir qual deles é o melhor são inconsistentes, já que cada tipo tem suas qualidades e deficiências, o objetivo é maximizar as qualidades e minimizar as deficiências para o propósito da sua aplicação. Uma forma de se conseguir isso é através da abordagem híbrida.

Agentes híbridos são constituídos de uma combinação de duas ou mais filosofias de agentes em um único agente. Essas filosofias incluem a filosofia móvel, a interface, a colaborativa, etc.

São agentes que serão usados também na implantação deste trabalho, visto que técnicas de IA deverão ser utilizadas e inseridas nos agentes móveis para que os agentes de negociação possam negociar. Esses agentes serão programas de computadores que simularão um relacionamento humano, fazendo algo que outra pessoa poderia estar fazendo por você GUHA (1994).

6.2.6.1 Motivação

- Benefício da união de duas ou mais filosofias quando uma sozinha não satisfaz.

6.2.7 Agentes Heterogêneos

Ao contrário dos agentes híbridos descritos anteriormente, referem-se a uma configuração integrada de pelo menos dois agentes de pelo menos duas classes distintas. Um sistema de agentes heterogêneos podem conter um ou mais agentes híbridos.

6.2.7.1 Motivação

O argumento principal é de que o mundo possui uma grande e rica diversidade de produtos de *software*, provendo uma grande quantidade de serviços para um largo

domínio similar. Pensando nesses programas trabalhando de forma isolada, há uma grande demanda para vê-los interoperando de uma maneira a adicionar valores.

Uma chave para esse requerimento de interoperabilidade entre agentes heterogêneos é ter uma linguagem de comunicação de agentes (ACL – *Agent Communication Language*), onde agentes de diferentes programas podem comunicar entre si.

6.3 Engenharia de Software

A engenharia de software baseada em agentes foi inventada para facilitar a criação de programas capazes de interoperar nessas circunstâncias. Nesta abordagem de desenvolvimento, os programas são escritos como agentes de programas.

Enquanto os agentes podem ser tão simples como as sub-rotinas, tipicamente eles são grandes entidades com algum tipo de controle persistente.

Engenharia de software baseada em agentes é comumente comparada à programação orientada a objeto. Como um objeto, um agente provê uma interface de comunicação independente da sua estrutura interna de dados e algoritmos. A primeira diferença entre as duas abordagens se engana na linguagem da interface. Em geral na programação orientada a objeto, o significado da mensagem pode variar de um objeto para outro. Em agentes usa-se uma linguagem comum, independente de semântica.

Essa linguagem é representada pela ACL (*Agent Communication Language*), ela pode ser mais bem entendida consistindo de três partes – seu vocabulário, uma linguagem interna chamada KIF (*Knowledge Interchange Format*), e uma linguagem externa chamada KQML (*Knowledge Query and Manipulation Language*). Uma mensagem ACL é uma expressão KQML onde os “argumentos” são termos ou sentenças no formato KIF formado por palavras do vocabulário ACL. A eficiência da comunicação pode ser alcançada disponibilizando uma camada lingüística, na qual o contexto está de acordo. Esta é a função da KQML GENE (1994).

Caso necessite estabelecer comunicação entre agentes e programas que já foram escritos, existem abordagens que podem ser adotadas a fim de estabelecer a comunicação.

Uma abordagem é implementar um *tradutor* que media entre programas existentes e outros agentes. O tradutor aceita mensagens de outros agentes, as traduz para o protocolo de comunicação nativo do programa, e passa as mensagens para o programa. O tradutor aceita também as respostas do programa, as traduz para ACL, e envia as mensagens resultantes para outros agentes.

Esta abordagem tem a vantagem da não necessidade de conhecimento do programa ou do comportamento de suas comunicações. É uma abordagem usada em situações em que o código do programa não está disponível ou quando é muito delicado para se modificar.

Uma segunda abordagem para lidar com esses *softwares* é implementar um *wrapper*, i.e., inserir código no programa para permitir sua comunicação em ACL. O *wrapper* pode examinar diretamente a estrutura de dados do programa e modificar essa estrutura. Esta abordagem tem a vantagem de melhor eficiência em relação à abordagem do tradutor, já que há menos comunicação serial.

A terceira e mais drástica abordagem é rescrever (reengenharia) o programa original. A vantagem é que a eficiência pode ser melhor que as outras duas abordagens GENE (1994).

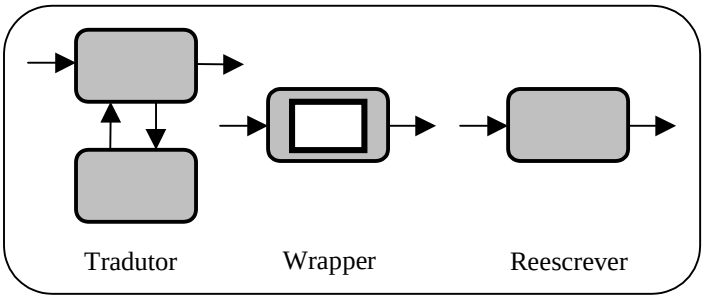

Figura 5.4 – Abordagens de implementação para comunicação entre agentes

6.3.1 Uma Questão de Organização

Mesmo tendo a linguagem e a habilidade de construir agentes, resta ainda a pergunta de como esses agentes devem estar organizados para alcançar a colaboração. Duas abordagens muito diferentes tem sido exploradas: comunicação direta (*direct communication*), nas qual os agentes tem controle sobre os caminhos a percorrer, e

caminhos assistidos (*assisted coordination*), no qual os agentes respondem a programas especiais para determinar os caminhos a serem percorridos GENE (1994).

A vantagem da comunicação direta é a não necessidade de responder pela sua existência, capacidade, ou direção a algum programa. Duas arquiteturas populares da comunicação direta são: a abordagem *contract-net* e a *specification sharing*.

Na abordagem *contract-net*, para haver interoperabilidade, os agentes, na necessidade de determinado serviço, distribuem requisições para outros agentes. Os agentes recipientes dessas mensagens avaliam a requisição e enviam respostas ao agente originário, o qual, utiliza dessas respostas para avaliar qual dos agentes irá contratar para execução de determinado serviço.

Na abordagem *specification sharing*, para haver interoperabilidade, os agentes enviam a outros agentes informações sobre suas capacidades e necessidades. Esses agentes podem então usar essas informações para coordenar suas atividades. Esta abordagem é mais eficiente do que a *contract-net* porque diminui o número de comunicações necessárias efetivamente.

Uma desvantagem da comunicação direta é o custo. Enquanto o número de agentes for pequeno não há problemas. Mas, em um ambiente como a Internet, com milhões de programas, o custo de comunicação seria proibitivo. Outra desvantagem é a complexidade de implantação. Na comunicação direta, cada agente é responsável por negociar com outros agentes e precisa conter todo o código necessário para suportar essa negociação.

Uma alternativa popular para a comunicação direta que elimina ambas desvantagens é organizar os agentes no que é chamado de sistemas federativos (*federated system*). Os agentes não se comunicam diretamente uns com os outros. Contudo, eles comunicam somente com os programas do sistema chamados facilitadores (*facilitators*), e os facilitadores comunicam entre si.

Os facilitadores usam a documentação provida pelos agentes para roteá-los aos lugares apropriados. De fato, os agentes formam uma federação na qual eles entregam sua autonomia para seus facilitadores, os quais são responsáveis por preencher suas necessidades.

Em sistemas federativos, os agentes usam ACL para documentar suas necessidades e habilidades para seus facilitadores locais.

Gerenciadores de objetos distribuídos como CORBA, OLE, DCOM provêem transparência de localização para sistemas orientados a objeto, roteando mensagens para os objetos sem a necessidade de identificação da localização desses objetos.

Usando ACL, os agentes podem expressar suas necessidades e capacidades de forma mais precisa do que em metalinguagens padrões, e os facilitadores podem usar essa informação adicional para ser mais discriminativos no roteamento das mensagens. Para lidar com notações incompatíveis, os facilitadores podem traduzir mensagens de um vocabulário para outro usando definições supridas pelos agentes ou retornadas do dicionário ACL. Portanto, eles podem decompor a mensagem em submensagens e enviá-las para agentes diferentes. Quando necessariamente, eles podem combinar múltiplas mensagens GENE (1994).

7 *AGLETS*

São objetos Java que podem mover-se de uma estação em uma rede para outra. É dirigido por eventos e comunica-se por passagem de mensagens, além de ter sua própria *thread* de execução.

7.1 Alguns de seus Elementos

Vamos agora analisar o modelo dos *aglets*. Este modelo define um conjunto de abstrações, e comportamento necessário para integrar a tecnologia de agentes móveis em redes WAN, como a Internet por exemplo VALDO (1999). As chaves desta abstração são:

a) Aglet

É um objeto Java móvel que visita estações habilitadas para receber os *Aglets* em uma rede de computadores. É autônomo, porque roda em sua própria *thread* de execução depois de chegar a uma estação, e reativo, porque responde a mensagens recebidas.

b) Proxy

É uma representação de um *aglet*. Serve como um escudo protegendo o *aglet* de acessos diretos a seus métodos públicos. O *proxy* também fornece transparência de localização para o *aglet*.

c) Contexto

É um ambiente de execução dos *aglets*; corresponde ao local definido anteriormente. É um objeto estacionário que fornece um meio para manutenção e gerenciamento de *aglets* em execução, em um ambiente uniforme onde o sistema da estação está seguro contra acessos indevidos de outros *aglets*. Um ponto em uma rede de computadores pode conter múltiplos contextos; além disso, contextos são nomeados e podem ser localizados pela combinação do endereço do servidor e do seu nome.

d) Identificador

Um identificador está associados a cada *aglet*. Todo *aglet* possui um identificador global único que é inalterado durante toda a sua vida.

Com as abstrações acima descritas, pode-se controlar um objeto *aglet* da sua criação até a sua destruição; vamos descrever as principais operações de um *aglet* VALDO (1999).

- Criação (*creation*): sempre realizada em um contexto. Ao novo *aglet* é atribuído um identificador, inserido em um contexto e inicializado. O *aglet* inicia sua execução logo que sua inicialização seja bem sucedida. Este é um dos modos de se trazer um *aglet* à vida.
- Clonagem (*cloning*): produz uma cópia idêntica do *aglet* naquele contexto. As únicas diferenças são seu identificador e o fato da execução ser reiniciada no novo *aglet*, isto se deve ao fato das *threads* de execução não serem duplicadas. Esta é a segunda forma de se trazer um *aglet* à vida.
- Liquidação (*disposal*): depois de criado, um *aglet* pode ser destruído. A liquidação de um *aglet* para sua execução corrente e remove este objeto do contexto atual.
- Emissão (*dispatching*): a emissão é uma mobilidade ativa realizada pelos *aglets*; ou seja, um *aglet* "empurra" a si próprio de uma estação corrente para uma estação remota.
- Retração (*retracting*): os *aglets* também podem realizar uma mobilidade passiva; consiste em uma estação remota puxar um *aglet* da estação corrente. Movendo um *aglet* de um contexto para outro irá remover seu contexto atual e inseri-lo no contexto destino, onde reiniciará sua execução.
- Ativação/desativação (*activation/deactivation*): a desativação de um *aglet* é a habilidade do mesmo em temporariamente suspender sua execução e armazenar seu estado em um meio secundário. A ativação irá fazer o processo inverso. Isto é útil quando se deseja que um *aglet* que está

consumindo recursos, entre em um estado de hibernação para liberar aquele recurso e retornar seu uso posteriormente.

- Comunicação (*messaging*): múltiplos agentes podem trocar informação de acordo com uma determinada tarefa.

7.2 Seu Modelo de Eventos

O modelo de programação dos *aglets* é baseado em eventos. Neste modelo existem *listeners* que disparam as devidas ações quando um determinado evento ocorre. Existem três *listeners* dentro do modo *aglet*:

- *Clone listener*: é disparado quando ocorrem eventos de duplicação. Pode-se desta forma programar tarefas antes do *aglet* ser duplicado, quando a cópia é criada e depois da duplicação ter ocorrido.
- *Mobility listener*: é disparado quando um evento de mobilidade ocorre; com isso pode-se preparar ações antes do *aglet* viajar, antes de ser retraído ou quando chega em um novo contexto (local).
- *Persistence listener*: espera por eventos de persistência. Permite ao programador trabalhar ações quando um *aglet* está sendo desativado e depois de ter sido ativado.

7.3 Seu Modelo de Comunicação

Toda a comunicação dos *aglets* é por passagem de mensagens. A seguir são exibidos alguns componentes deste modelo de comunicação:

- *Message*: uma mensagem é um objeto trocado entre os *aglets*. Este modelo permite passagem de mensagens síncronas e assíncronas entre os *aglets*.
- *Future reply*: é usado em mensagens assíncronas enviadas como um *handle* e permite ao emissor da mensagem receber uma resposta assíncrona.

- *Reply set*: pode conter múltiplos objetos com respostas futures e é usado para buscar resultados assim que eles estejam disponíveis. É possível também receber uma resposta e ignorar as subseqüentes.

7.4 O Pacote *Aglet*

A API do *aglet* é um pacote Java contendo classes e interfaces. Esta API define os fundamentos e funcionalidades dos agentes móveis. A seguir serão descritas as principais classes e interfaces desta API.

7.4.1 Classe com.ibm.aglet.Aglet

A classe abstrata *Aglet* define os métodos fundamentais para que os agentes móveis possam controlar a mobilidade e seu ciclo de vida. Todos os agentes móveis definidos nos *Aglets* tem que herdar esta classe abstrata. A classe *Aglet* também pode ser usada para obter os atributos associados a um determinado *aglet*.

7.4.2 Interface com.ibm.aglet.AgletProxy

A interface *AgletProxy* atua como um *handle* de um *aglet* e fornece um modo comum de acesso a este agente. Uma classe *Aglet* tem muitos métodos públicos que não devem ser acessados diretamente por outros *aglets* por razões de segurança, caso algum *aglet* queira se comunicar com outro *aglet* tem que primeiro obter o *proxy* deste objeto, e então interagir através desta interface. Outra regra importante da interface do *AgletProxy* é fornecer ao *aglet* transparência de localização.

7.4.3 Interface com.ibm.aglet.AgletContext

Fornece uma interface para o ambiente de execução ocupado pelos *aglets*. Qualquer *aglet* pode obter uma referência para o objeto *AgletContext* e usar a informação local obtida ou mesmo criar novos *aglets* neste contexto. Uma vez

despachado um *aglet*, o objeto contexto correntemente ocupado é desvinculado e o novo contexto é então vinculado.

7.4.4 Classe com.ibm.aglet.Message

Objetos *aglet* comunicam-se pela troca de objetos da classe *Message*. Um objeto *Message* pode ter um objeto *String* para especificar o tipo de mensagem e outros objetos como argumentos.

7.4.5 Interface com.ibm.aglet.FutureReply

O objeto definido pela interface *FutureReply* é retornado quando do envio de mensagens assíncronas e usado como um *handle* para receber os resultados passados mais tarde de forma assíncrona. Com esta interface o receptor pode determinar se uma resposta está disponível, e pode esperar pelo resultado por um período específico de tempo, continuando a sua execução caso a resposta não tenha sido retornada.

8 MEDIAÇÃO DE AGENTES EM COMÉRCIO ELETRÔNICO

Com um crescimento significativo dos recursos de informações disponíveis na Internet, os agentes de *software* têm se destacado pela habilidade de automatizar tarefas repetitivas e que consomem um tempo considerável como: pesquisa, compra e venda de produtos na Internet. Muitas das tarefas envolvidas em um comportamento de compra podem ser automatizadas. Tarefas como: pesquisa de produtos, pesquisa de fornecedores e negociação podem ser automatizadas ou assistidos por sistemas baseados em agentes.

8.1 O Modelo CBB

O modelo CBB (*Consumer-Buying Behavior* ou Comportamento de Compra de Consumidores) define os processos de decisão que as pessoas utilizam na compra de produtos. Vários modelos tentam capturar este processo em um conjunto de etapas. Representam a simplificação de um comportamento muito complexo, no qual as etapas não são entidades discretas. Normalmente, as etapas podem sobrepor-se e serem concorrentes e iterativas. Mesmo sendo simplista, estes modelos fornecem uma importante ferramenta para elucidar em quais circunstâncias os sistemas de mediação de agentes em comércio eletrônico aplicam às experiências de compra dos consumidores.

As etapas definidas no CBB são:

1. Necessidade de Identificação: também chamado reconhecimento do problema, esta etapa representa o estado da necessidade do consumidor.
2. Pesquisa de Produto: nesta etapa o consumidor decide o que comprar. Avalia uma série de diferentes produtos e tenta identificar quais seriam as necessidades satisfatórias.
3. Pesquisa de Mercado: nesta etapa o consumidor já sabe o que quer e decide de quem vai comprar o produto. A decisão é baseada em um

conjunto de critérios como: preço, disponibilidade, reputação, prazo de entrega, etc.

4. Negociação: esta etapa determina como a transação irá ocorrer. Muito dos modelos tradicionais não identificam esta etapa explicitamente, mas a separação deste processo em uma nova etapa é muito útil na determinação das regras dos agentes.
5. Compra e Entrega: esta etapa pode algumas vezes sinalizar o fim da etapa de negociação. O processo de pagamento e de entrega acontecem nesta etapa.
6. Suporte ao Produto e Avaliação: esta etapa inclui o suporte ao produto, serviços fornecidos aos clientes e avaliação de satisfação do produto adquirido.

8.2 Negociação entre Agentes

Negociação é o processo de se determinar o preço ou outros termos de uma transação. Exemplos de negociação usada no comércio em geral incluem bolsas de valores (NYSE, NASDAQ), leilões de arte, e presente em várias outras situações.

O benefício da negociação dinâmica do preço de um produto ao invés do seu valor fixo é que permite ao negociador determinar o valor do bem a princípio; evitando-se com isto que esse seja empurrado para o mercado. O resultado disto é que recursos limitados são alocados satisfatoriamente para aqueles que pagam mais.

Entretanto existem impedimentos para se usar negociação. No mundo físico, certos tipos de leilões requerem que todas as partes estejam geograficamente situadas, por exemplo, em casas de leilões. O processo de negociação também pode ser frustrante para o consumidor médio que não está acostumado com este tipo de procedimento. E finalmente, alguns protocolos de negociação ocorrem sobre um período de tempo muito extenso o que torna inviável para consumidores impacientes ou limitados pelo horário. Em geral, as negociações no mundo real resultam em custos que podem ser muito altos para consumidores ou negociantes.

No mundo digital, muito destes impedimentos desaparecem. Por exemplo, EggHead (http://www.egghead.com), eBay´s (http://www.ebays.com), Arremate

(http://www.arremate.com.br) e iBazar (http://www.ibazar.com.br) são quatro *sites* populares que vendem produtos usando uma escolha de protocolos de leilão. Ao contrário das casas de leilão, estes *sites* não necessitam que as partes estejam geograficamente localizadas (somente virtualmente localizadas). Entretanto, ainda necessitam que seus consumidores administrem suas próprias estratégias de negociação em um determinado período de tempo VALDO (1999).

Para automatizar ainda mais os leilões e o processo de negociação existente neles, estudos vêm sendo realizados para a mediação de agentes nesta etapa do modelo CBB; desta forma, além dos consumidores não precisarem de uma presença física, também não precisarão administrar toda a etapa de negociação, compra e venda existente entre compradores e vendedores.

Os agentes adequam-se perfeitamente tanto à etapa 4 (negociação) do modelo CBB, quanto a outras etapas também discutidas anteriormente. Vários sistemas que implementam soluções na área de comércio eletrônico com mediação de agentes estão espalhados em diversos centros: Kasbah (http://ecommerce.media.mit.edu/Kasbah), Tete-a-Tete (http://ecommerce.media.mit.edu/Tete-aTete), Jango (Excite – http://www.jango.com), AuctionBot (http://auction.eecs.umich.edu), V-Market (http://harper.les.inf.puc-rio.br/vbookmarket), dentre outros.

Vejamos o sistema CUBe que utiliza a Mediação de Agentes e que serve como base para a análise deste trabalho.

8.3 CUBe

CUBe é um sistema para integrar Comércio Eletrônico e ERP através de Agentes Móveis, desenvolvido na Universidade Federal de Santa Catarina por Clayton Augusto Valdo, onde os agentes móveis atuam como mediadores das transações do sistema.

CUBe integra *e-Commerce* e *e-Business*, o acesso dos clientes é feito pela Internet através de páginas ASP que possuem itens a serem selecionados através de um formulário eletrônico padrão. Após o pedido, o usuário definirá o tipo de agente remoto que o representará nas negociações com o sistema ERP, sendo este parametrizado com características de comportamento previamente definidas. Dentre os possíveis agentes a serem escolhidos pelo usuário tem-se: Generoso, Prudente e Rígido, cada qual define

um algoritmo de negociação e também uma função de variação de preços durante a elaboração da proposta no ato da negociação.

CUBe é um sistema que busca uma generalização, suportando múltiplos produtos, e a automação de várias etapas do modelo CBB graças aos agentes móveis, onde foi possível automatizar as etapas 1, 4, 5 e 6 do modelo CBB. Tem também suporte a múltiplos processos de negociação e comportamento como já foi citado.

8.3.1 e-Commerce

Dos valores parametrizados para a negociação do sistema de *e-Commerce*, tem-se: *Valor Mínimo* e *Valor Máximo de Desconto* e *Número de Propostas Máximo* que o agente realizará. Depois de preenchido todas as informações necessárias, os dados são enviados ao servidor *Web* que irá analisá-las e redirecioná-las a um servidor de agentes, e este criará o agente remoto, que por sua vez, irá movimentar-se até o sistema ERP da empresa responsável pelos produtos adquiridos e entrar em negociação com o Agente Administrador daquele sistema.

Fechada a etapa de negociação, o agente irá retornar ao Servidor de Agentes, ficando o pedido no sistema ERP, caso este tenha sido aceito; o administrador do sistema ERP ficará então responsável em atualizar informações sobre o pedido e dar andamento ao mesmo dentro do sistema.

8.3.2 ERP

Dentro do sistema ERP, agentes integrantes deste sistema, trabalharão automatizando várias etapas descritas no modelo CBB, antes controladas manualmente por funcionários da empresa.

Após a etapa de negociação – mediada pelo Agente Administrador do sistema ERP – todas as etapas seguintes, da venda até a entrega do produto, serão controladas por agentes específicos, espalhados pela Intranet da empresa responsável pela divulgação do produto.

8.3.3 e-Business

Os fornecedores (bancos, transportadoras ou fornecedores) serão empresas que poderão ser acionadas por Agentes Compradores de outras empresas – seus clientes. Neste caso, para automatizar ainda mais o processo, agentes do sistema interno de cada fornecedor poderão recepcionar os agentes compradores de outras empresas e proceder com uma entrega do pedido.

8.3.4 Arquitetura

Na figura abaixo tem-se a arquitetura geral do sistema CUBe, e algumas tecnologias empregadas na sua construção.

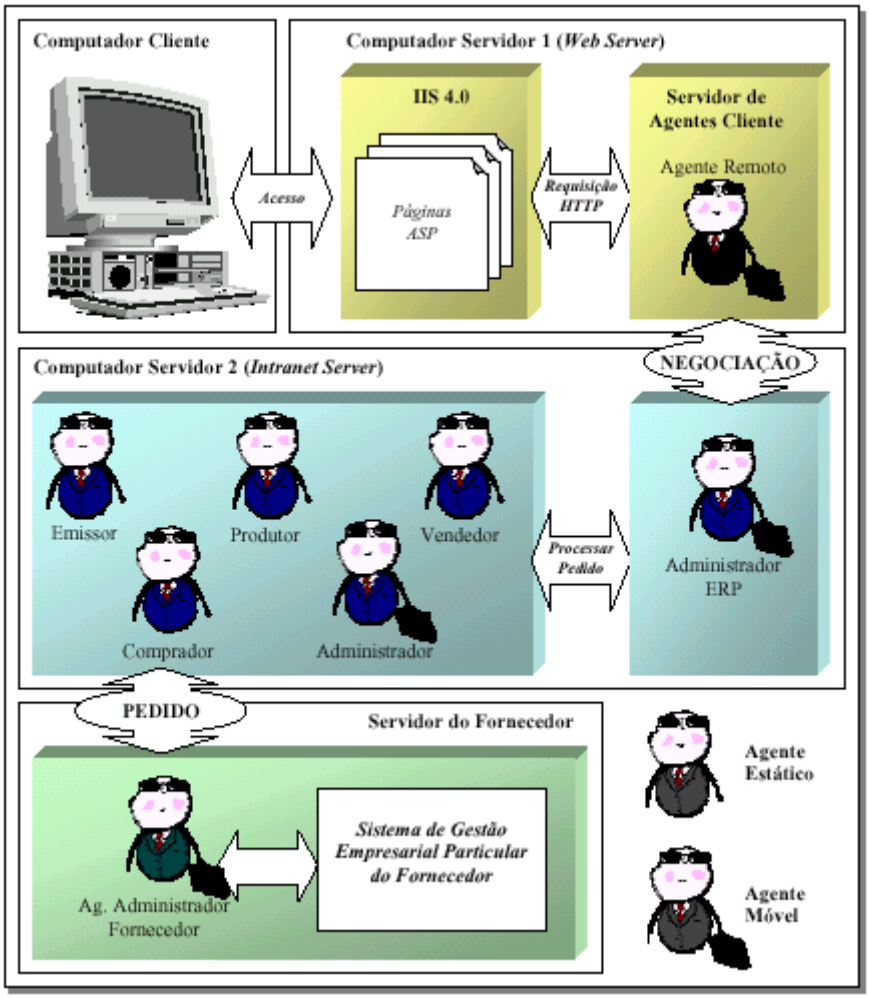

Figura 7.1 – Arquitetura geral do sistema CUBe

9 DESENVOLVIMENTO

O desenvolvimento deste trabalho se deu em uma aplicação já existente, CUBe. Este trabalho veio contribuir ao modelo de negociação adotado por esse sistema ERP, buscando identificar de forma mais precisa o cliente e o produto que está sendo adquirido, estabelecendo assim, uma negociação mais adequada do desconto.

Ao invés de se utilizar os agentes negociadores do CUBe, buscou-se criar uma equipe de agentes colaborativos baseados em lógica nebulosa, onde cada agente é responsável por estar avaliando as etapas definidas no processo de negociação, etapas estas que serão descritas no decorrer deste capítulo.

9.1 Metodologia

A abordagem para o desenvolvimento do trabalho partiu da identificação de um problema relativo à falta de consideração dada ao cliente no comércio eletrônico ao fazer negócio. Um cliente que compra freqüentemente a 2 (dois) anos em uma livraria paga o mesmo, ou está sob as mesmas condições de atendimento, que um cliente que irá efetuar a sua primeira compra, ou que as efetua de forma esporádica.

Sobre essa abordagem é apresentada uma proposta de solução utilizando recursos de TI.

9.1.1 Métodos

Para identificar a definição do escopo deste trabalho, uma pesquisa acerca do sistema CUBe foi realizada, identificando desta forma que o processo de negociação adotado pelo mesmo não estabelece uma personalização adequada da venda do produto, visto não analisar suas características.

Foi observado que o objetivo do sistema CUBe, no que tange ao tópico de múltiplos processos de estratégia e comportamento, é deixar um determinado número de estratégias previamente definidas. Com isso o cliente pode escolher um determinado comportamento para que seu agente possa desempenhar uma determinada negociação,

enquanto que a empresa pode definir comportamentos variados, de acordo com o histórico de compras de seus clientes ou outras características trabalhadas dentro do sistema.

Foram definidos nesse sistema 3 agentes distintos com comportamentos previamente definidos; são eles: Generoso, Prudente e Rígido, sendo que cada um desses comportamentos identifica o agente dentro do sistema, o algoritmo de negociação adotado e a função de variação de cálculo durante a etapa de negociação, como pode ser visto na Tabela abaixo.

Comportamento do Agente	**Função para cálculo da variação**	**Algoritmo adotado durante a negociação**
Generoso	log(x)	negociar até obter uma resposta afirmativa ou receber uma contra-proposta válida.
Prudente	X	negociar até obter uma resposta afirmativa, caso não haja nenhuma, verifica a última proposta realizada pelo agente administrador.
Rígido	x^3	negociar até obter uma resposta definitiva.

Tabela 9.1 – Parâmetros dos Agentes no Sistema CUBe

Os agentes descritos mais adiante neste capítulo, analisam características do cliente e do produto que está sendo adquirido, buscando traçar/identificar o perfil de ambos, efetuando uma real personalização da venda e utilizando uma política distributiva mais justa, visto que o procedimento anteriormente exposto não releva o cliente, justamente quem movimenta o negócio da empresa.

Ao invés de se utilizar os agentes negociadores do CUBe, a solução adotada foi a criação uma equipe virtual, onde cada agente é responsável por estar avaliando as etapas definidas no processo de negociação, etapas estas que serão descritas no decorrer deste capítulo.

9.1.2 Técnicas

As técnicas para a implementação dessa equipe virtual (agentes colaborativos) envolveram:

- O uso de um mecanismo de inferência baseado em lógica nebulosa;
- A adequação da solução proposta aos agentes móveis e estacionários do sistema CUBe;
- A modelagem das variáveis utilizadas no mecanismo de inteligência artificial;
- A modelagem dos agentes desenvolvidos, envolvendo a especificação orientada a objetos e a orientada a agentes;
- A modelagem da base de dados para que flexibilizasse o cadastro dos componentes *fuzzy*.

9.1.3 Ferramentas

Ferramentas de modelagem e desenvolvimento disponíveis foram utilizadas para criar os modelos de análise e projeto UML, de lógica nebulosa e de dados, além da codificação em linguagem de programação. Estas são apresentadas a seguir.

Modelagem em UML

A ferramenta utilizada foi o *software* Rational Rose 2000, versão de demonstração, da empresa Rational Software Corporation.

Modelagem de Dados

O modelo de dados apresentado no trabalho foi desenvolvido com a ferramenta ERWin, versão ERX 3.5.2, da empresa Platinum *Technology*. A implementação do modelo foi feita através do *software* Microsoft Access 97.

Linguagem de Programação

Os agentes colaborativos foram implementados na linguagem Java, utilizando o *software* JDK versão 1.1.8, o editor de programas TextPad versão 4.4.2, da empresa Helios *Software Solutions*, o *software* Aglets 1.0.3 da IBM. O programa UnFuzzy gerou o código do mecanismo de inferência de cada agente, em linguagem C.

Modelagem em Lógica Nebulosa

O desenvolvimento da modelagem em lógica nebulosa utilizou o programa UnFuzzy, versão 1.1, de distribuição gratuita, desenvolvido na Universidade Nacional da Colômbia e o sistema fuzzyTech versão 5.31 Professional Demo, da empresa alemã INFORM GmbH.

O primeiro foi escolhido pela capacidade de geração de código em uma linguagem de programação já conhecida e o segundo, devido aos recursos para modelagem visual interativa. As regras de inferência, resultantes da modelagem com o fuzzyTECH foram transcritas para o UnFuzzy. Ambos demonstraram o mesmo comportamento no momento da avaliação, mas a análise gráfica do fuzzyTECH permitiu um ajuste mais refinado do comportamento das variáveis e uma visão global da influência de cada variável no resultado final.

9.2 Modelos dos Agentes Colaborativos

A ação dos agentes colaborativos dentro do sistema CUBe compreende, de modo geral, ao recebimento dos dados de um determinado pedido pelo agente que representa o gerente dentro do sistema, repassando-lhe os dados com os descontos calculados para cada item do pedido. Nesta seção são detalhados os modelos desenvolvidos para o sistema e para o mecanismo de inteligência artificial em lógica nebulosa.

9.2.1 Modelo de Dados

O modelo de dados reformulado do sistema CUBe para as implementações em lógica nebulosa é exibido no Anexo A. O modelo diz respeito à definição anterior do sistema CUBe com algumas tabelas adicionais (Regras, Variáveis, Variáveis_Produto, Conjuntos, Conjuntos_Específicos) para possibilitar o cadastro dos componentes de um controlador *fuzzy*, como as funções de pertinência para as variáveis de entrada e saída e as regras de inferência.

O modelo de dados desenvolvido permite uma flexibilização na manutenção dos agentes. Qualquer alteração que se fizer necessária nas regras do negócio, deve ser feita na base de dados, visto que o agente só possui os métodos para os cálculos *fuzzy* e manipulação de mensagens, não possuindo em seu código fonte nenhuma especificação de regras ou funções de pertinência.

9.2.2 Especificação Orientada a Objetos em UML

A especificação utilizada para representar os agentes colaborativos dentro do sistema abrange aspectos estáticos com diagramas de caso de uso e de classe.

Para representar o início do processo de negociação o diagrama de caso de uso da Figura abaixo foi abstraído. O *RemoteAgent* é responsável por representar o cliente durante todo o processo no sistema, é ele que monta o pedido e encaminha ao *ERPManagerAgent*, agente responsável pela gerência do sistema ERP e pela solicitação do cálculo de desconto aos agentes colaborativos incrementados no CUBe.

O cálculo do desconto foi abstraído conforme diagrama de caso de uso representado na Figura 9.1.

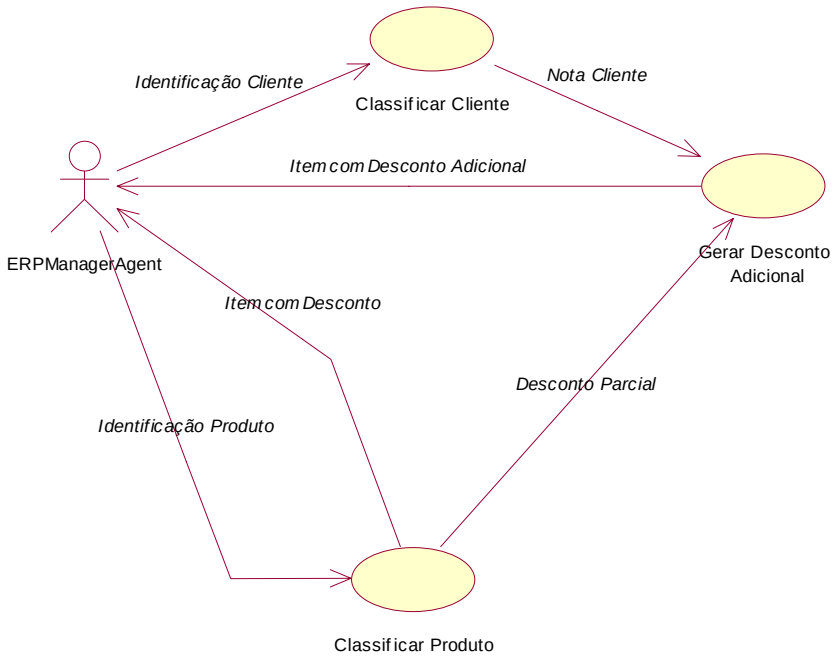

Figura 9.1 – Diagrama de Caso de Uso Expandido – Gerar Desconto

O *ClientClassificationAgent* e o *LevelOfInterestAgent* dão início ao processo ao receber do *ERPManagerAgent*, para o primeiro agente, a identificação do cliente, e para o segundo, o item do pedido a ser analisado.

Os agentes *LevelOf InterestAgent, ProductClassificationAgent* e *DiscountAgent*, são responsáveis por analisar um item do pedido, definindo assim, o desconto. O *ClientClassificationAgent* e o *AdditionalDiscountAgent* apenas são invocados quando o produto não se encontra em promoção, fator este que impede a atuação deles. Estes dois últimos são responsáveis pelo cálculo de um desconto adicional a ser dado sob o desconto do item do pedido analisando as características do cliente.

Diagrama de Classes

O diagrama de classes representado pela Figura 9.2, assim foi abstraído para que qualquer incremento de um novo agente colaborativo se torne tarefa fácil, visto que, como pode ser observado, os agentes possuem apenas os métodos necessários para a

manipulação das mensagens que lhes são encaminhadas e os métodos representativos do motor de inferência, responsável por efetuar os cálculos *fuzzy*.

Dessa forma, faz-se necessário garantir apenas que o novo agente herde da classe Fuzzy criada e configure as regras e funções de pertinência na base de dados abstraída no Anexo A.

Figura 9.2 – Diagrama de Classes dos Agentes Modificados para Negociação

9.3 Negociação do Desconto

Um dos fatores determinantes ao efetuar uma compra com certeza é o preço, certamente o mercado é elemento determinante na formação de conceitos, fatores econômicos, políticos, enfim, de mercado que, ao fim e ao cabo, introduzem efetivos vetores de novas necessidades tanto expressivas, quanto de consumo da comunidade pertinente.

Além disso, sabe-se que os estados de coisas não são estanques, longe disso, interagem e provocam, enquanto fatores atuantes, reações do sistema. Entretanto, os analistas, tanto econômicos, quanto sociais, quanto políticos e, mesmo jurídicos precisam, para viabilizar uma prospecção propensiva, reduzir as variáveis sob análise e circunscrever o conflito a contornos mais próximos sem pretender, normalmente, alcançar todos os fatores intervenientes, nem sequer antecipar a conduta dos agentes interessados PUGLIESI (2001).

Portanto, esses estados favorecem a inexistência de critérios únicos para estabelecer o preço de um produto ou serviço, em vista disso, buscou-se analisar os aspectos mais comumente utilizados no mercado real.

Faz-se um recorte (artificial) que exclui certos aspectos e inclui outros tidos como influências preponderantes para a situação em exame. A rigor, o corte funciona porque os aspectos estudados relacionam-se por interdependência, i.e., os atores, dos fatos sob observação, comportam-se como se tivessem total independência, mas as ações recíprocas interferem no resultado PUGLIESI (2001).

Aspectos como a periodicidade de compra e gasto acumulado do cliente, o número de compras efetivadas de um produto, sua demanda, longevidade e quantidade desejada, são os aspectos analisados nesta pesquisa para avaliação do cliente e do produto, estabelecendo assim uma política de desconto.

O estabelecimento de estratégias adequadas, sempre com alternativas viáveis e sua implementação constituem, em nossa conjectura, o cerne de condutas que possam efetivamente conduzir a uma satisfatória decisão dos estados conflitivos. A palavra estratégia não se refere, consoante o sentido que daremos à palavra neste texto, à aplicação eficiente e atual da força, mas à exploração de uma força potencial em que os mecanismos citados sejam aplicados não apenas à parte contrária, mas a eventuais litisconsortes que causem desconfiança. Relaciona-se não apenas com a distribuição de lucros/perdas entre dois litigantes, mas, também, com a possibilidade de que decisões particulares sejam melhores que quaisquer outras para todos os litigantes ao mesmo tempo, isto é, (em termos da teoria dos jogos, no caso em que o preço da decisão acaba sendo um custo social). Constitui-se portanto em um interesse comum de encontrar soluções ou decisões, que sejam mutuamente vantajosas, ou no pior dos casos, não desvantajosas, visto que, sob o ponto de vista de um enfoque negocial, uma decisão não

é verdadeira ou falsa, nem correta ou incorreta. Uma decisão será conveniente ou inconveniente, pressupondo um quadro situacional (estado de coisas) e os estados de conhecimento, no sentido adotado nesta conjectura, de todos os envolvidos PUGLIESI (2001).

Essa concepção vincula-se, embora não o pareça, à idéia de uma justiça distributiva implicando, consequentemente, o princípio da justa divisão.

Mapear adequadamente o estado de coisas e, valendo-se do estado de conhecimento possível, decidir de forma racional visando a fins, segundo todos os princípios de otimização, deverá ser a conduta capaz de minimizar o prejuízo e, em conseqüência, ampliar o lucro.

Neste trabalho foi definida uma estrutura de agentes colaborativos determinantes de uma política da justa divisão, conforme será abordado a seguir.

9.4 Agentes Colaborativos

O estabelecimento desses agentes, como sendo de influência preponderante no cálculo do desconto a ser dado a cada item do pedido, se deu sob aspectos utilizados comumente na avaliação de um cliente e de um produto.

Outras variáveis poderiam ser levadas em consideração, mas o objetivo deste trabalho é mostrar a possibilidade da automação de um processo de personalização da venda sob o critério preço, visto que cada empresa pode definir quais são as variáveis consideradas influentes em seu meio comercial. Em vista da preocupação de não definir as variáveis sob análise como verdades absolutas, é que deve-se ter em mente uma empresa imaginária.

A comunicação entre os agentes, representada na Figura 9.3, demonstra cada variável de entrada analisada e sua respectiva variável de saída.

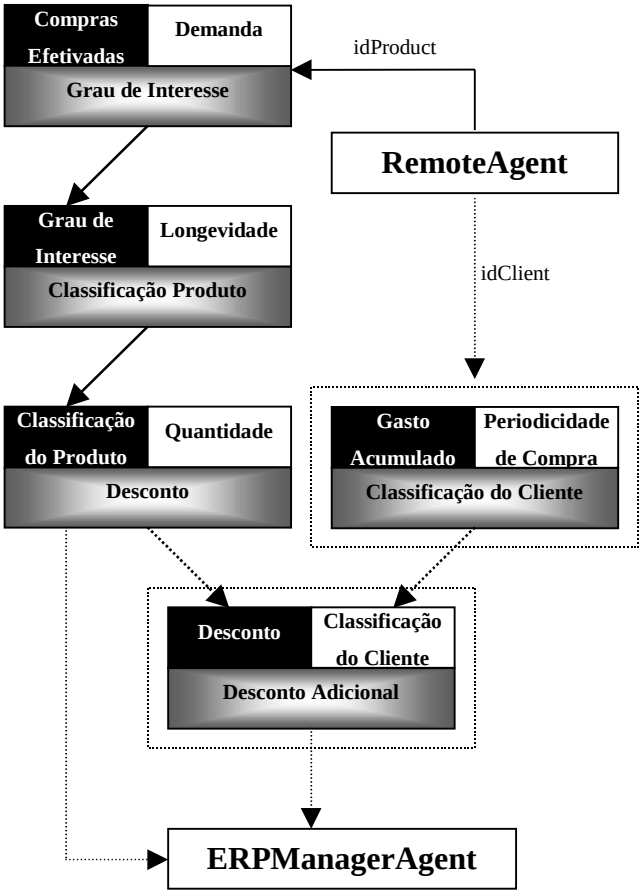

Figura 9.3 – Comunicação dos Agentes Colaborativos

Cada quadro representa um agente colaborador no processo da negociação do desconto, podendo ele ser *fuzzy* ou não, sendo que os agentes envoltos por um traço podem ou não ser ativados, neste caso, somente se o produto não estiver em promoção os agentes em questão serão solicitados para participarem do processo.

As setas indicam as mensagens trocadas, as tracejadas indicam que a mensagem pode ou não ocorrer dependendo da situação do produto (em promoção ou não).

O valor discreto calculado é entregue ao *ERPManagerAgent* para que possa dar andamento ao processo de compra/venda do produto solicitado.

Cada agente será descrito com maiores detalhes em seqüência, notar-se-á que algumas variáveis em análise não possuirão um universo de discurso previamente

definido, visto que pode ocorrer variação do mesmo dependendo do produto em análise. Para solucionar essa questão, cada variável, para um produto em específico, possui seu valor mínimo e máximo cadastrados na base de dados, bem como suas funções de pertinência.

9.4.1 ClientClassificationAgent

A classificação do cliente é a relação entre o Gasto Acumulado e a Periodicidade de Compra do cliente, resultando em um valor no intervalo de [0,100]. Conforme pode ser observado na Figura 9.4, a periodicidade de compra tem maior peso que o gasto acumulado, visto que funciona como índice de fidelidade do cliente. Para definir metas e avaliar dados estatísticos, o cliente fiel é um facilitador, contrário àquele que compra de forma esporádica, dificultando qualquer planejamento.

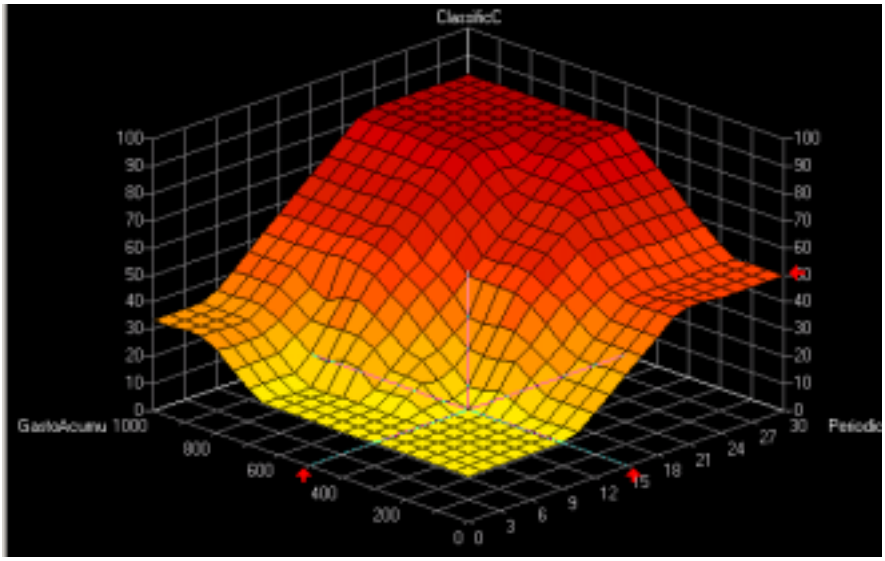

Figura 9.4 – Classificação do Cliente

Gasto Acumulado

O gasto acumulado representa o montante dos valores adquiridos pelo cliente em compras ou participações de outra natureza. Foi determinado um índice atemporal para avaliação desse montante, traduzido em uma moeda própria da empresa, com vistas a evitar que alterações monetárias advindas de inflação ou planos econômicos, afetem valores de compras efetuadas em um período passado.

Periodicidade Compra

A periodicidade de compra é o índice que representa a frequência que um cliente adquire um novo produto da empresa. Consiste em conhecer o totalidade das compras efetivadas pelo cliente junto à empresa e determinar a média destas compras no período decorrido em meses, desde a primeira.

9.4.2 LevelOfInterestAgent

O Grau de Interesse pelo produto é a relação entre as Compras Efetivadas e a sua Demanda, resultando em um valor no intervalo de [0,100]. Visto que neste último caso não se pode comprovar que o acesso a um produto era motivado por interesse de compra, as Compras Efetivadas e a Demanda são consideradas de mesma relevância, a simetria notada na Figura 9.5 demonstra a equivalência do peso das variáveis de entrada.

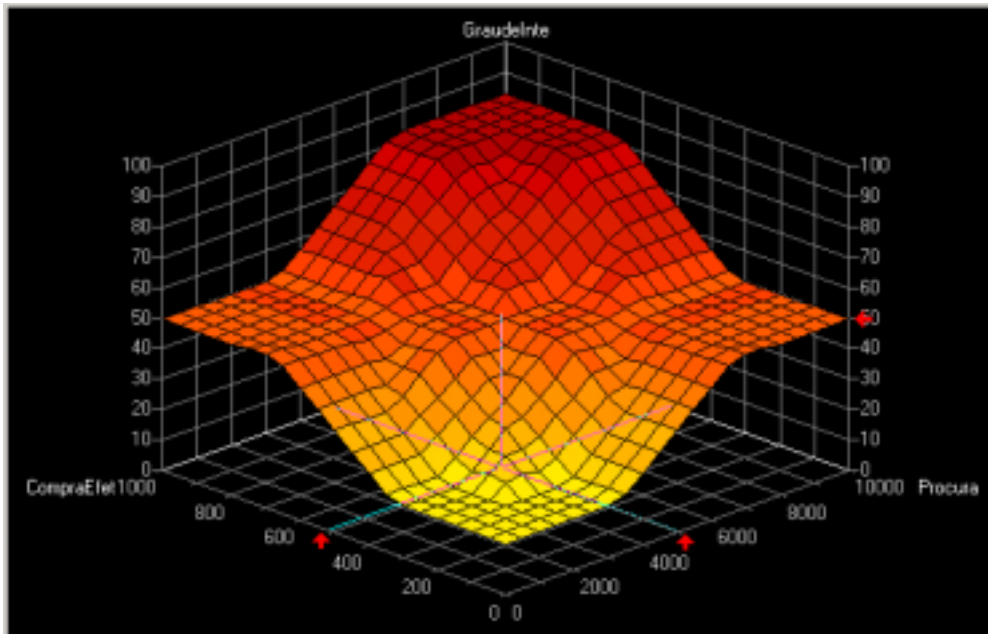

Figura 9.5 – Grau de Interesse

Compras Efetivadas

Consiste na média de venda diária do produto, determinada pelos registros de venda dos últimos 30 dias. Foi atribuída essa quantidade temporal visto a necesidade de se trabalhar com dados que reflitam o atual estado de venda do produto em questão, caso contrário, dados passados poderiam influenciar em uma resultante que não estaria espelhando a real média de venda diária aproximada da data em análise.

Produtos demarcados como sendo de grande importância a avaliação da sazonalidade, buscam dados de compras efetivadas no mês do ano anterior, como uma previsão de vendas, por exemplo, produtos que só vendem na estação de inverno, para esses produtos, não interessa saber os dados de venda dos últimos 30 dias, mas sim como foram as vendas nesse período no ano anterior.

Demanda

Esta variável consiste em um índice de procura por um produto específico, determinado pelos acessos feitos por clientes nos últimos 30 dias, com interesse de compra ou simples curiosidade. Devido à informalidade desses acessos, não se pode determinar com exatidão a motivação pela procura do produto, consequentemente não se pode atribuir à esta variável um peso tão relevante.

9.4.3 ProductClassification

É a relação entre o Grau de Interesse por um determinado produto e a sua Longevidade, resultando em um valor no intervalo de [0,100]. Quanto menor o grau de interesse e a longevidade, menor a classificação do produto, isso porque em ambos os casos faz-se favorável a liberação do produto em questão, conforme pode ser observado na Figura 9.6.

O Grau de Interesse pelo produto tem maior peso que sua longevidade, visto que as compras efetivadas representam a vazão do produto, analisando os últimos 30 dias de venda do mesmo. Se o Grau de Interesse pelo produto é alto, mesmo a Longevidade sendo de valor crítico não desvaloriza o produto em análise.

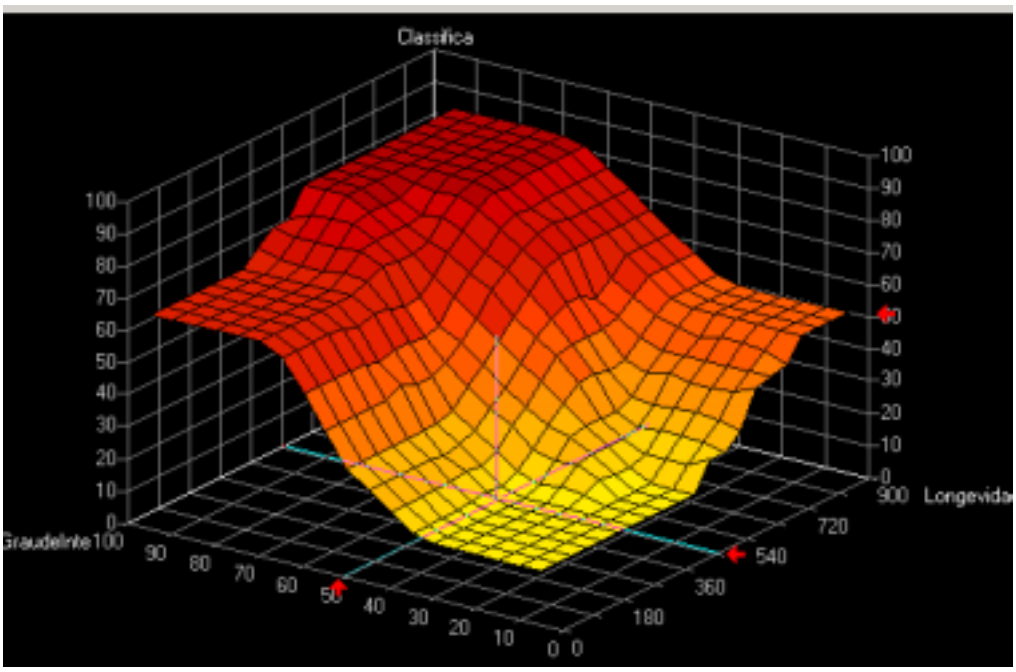

Figura 9.6 – Classificação do Produto

Longevidade

A longevidade do produto representa sua validade para efeito de comercialização, podendo ser o prazo de validade, no caso de produtos perecíveis, ou prazo de demanda significativa, até alcançar sua obsolescência ou a perda de um interesse motivado por moda.

Esta variável tem significativa importância, posto que, determina o grau de necessidade da liberação do produto. Produtos com longevidade pequena devem ser rapidamente liberados, enquanto que no caso inverso, sua rápida liberação pode ou não ser decidida, visando em um caso aumentar o interesse pela compra, ou em outro, aguardar por uma melhor reação do mercado consumidor. O fator que auxilia o agente a determinar os dois possíveis casos citados é o grau de interesse, visto que, sendo alto o grau de interesse pelo produto, essa variável é determinada como de maior importância, caso contrário, o desconto pode ser majorado visando despertar o interesse da população consumidora.

9.4.4 DiscountAgent

É a relação entre a classificação do produto e a quantidade desejada, em uma compra. Quanto menor a Classificação do Produto, maior o desconto. Quanto maior a quantidade, maior o desconto. A quantidade tem maior influência no resultado do que a classificação do produto, conforme Figura 9.7, visto aumentar proporcionalmente a geração de receita.

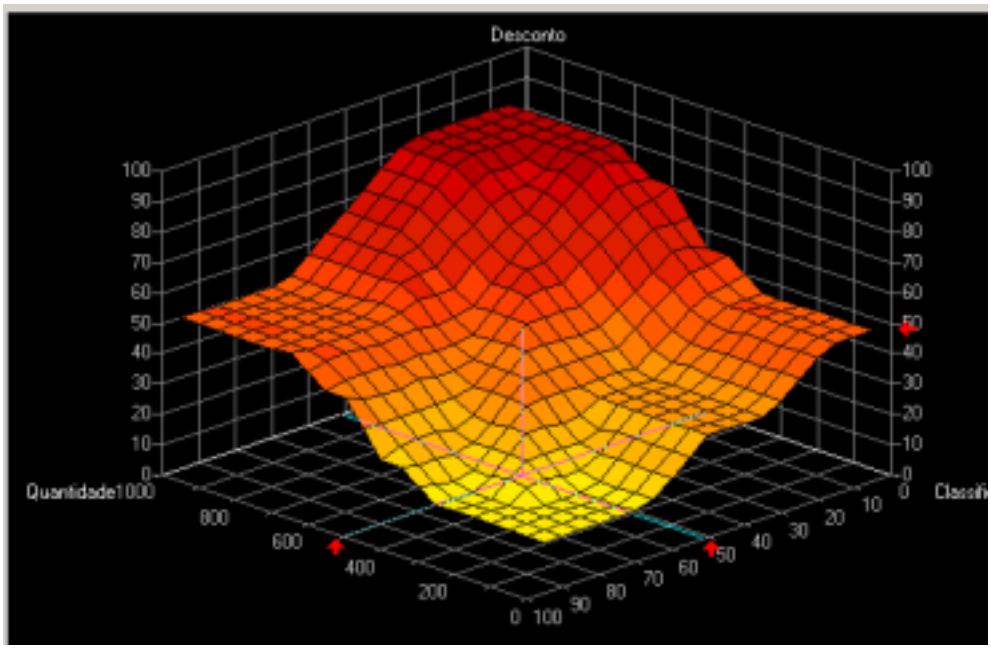

Figura 9.7 – Desconto

9.4.5 AdditionalDiscountAgent

Representa um percentual de desconto adicional a ser aplicado sobre o percentual de desconto do produto, com base na classificação do cliente.

Para evitar descontos irreais do tipo acima do valor de venda do produto, um flag que identifica se o produto está em promoção restringe a aplicação do desconto adicional, função essa que considera a atuação do cliente.

Como pode ser observado na Figura 9.8, o novo desconto é uma função linearmente crescente, visto que, conforme for o crescimento da classificação do cliente,

proporcionalmente igual é o crescimento do desconto adicional a ser dado sobre o desconto já estipulado para o produto.

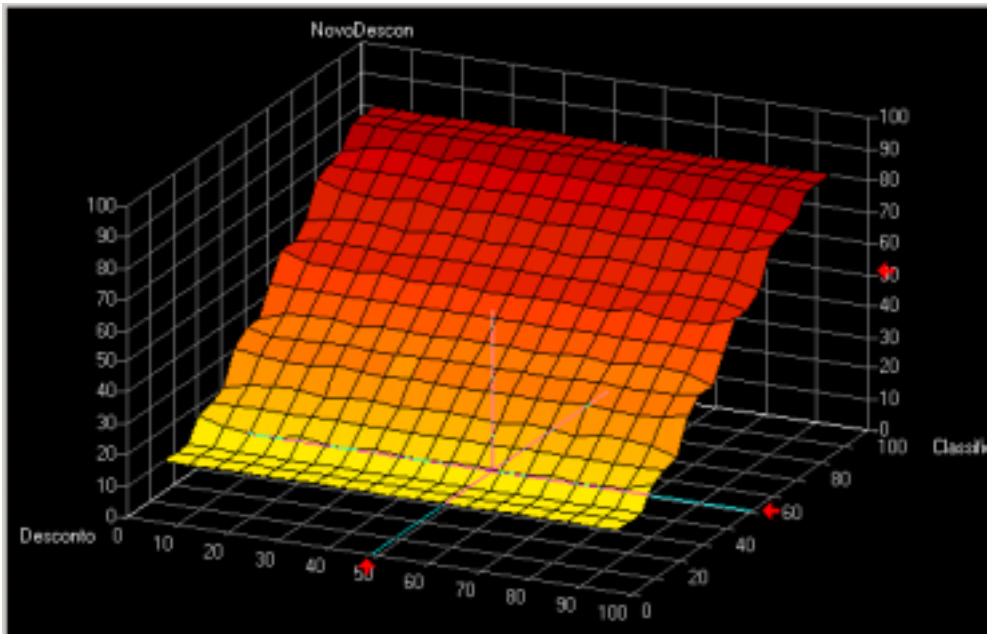

Figura 9.8 – Desconto Adicional

9.5 Algumas Considerações sobre os Agentes Colaborativos

A comunicação entre os agentes é feita através de um objeto *Message*, cuja definição de classe se encontra no pacote dos Aglets. Apesar de no capítulo 6 serem apresentadas algumas linguagens de comunicação para agentes, a classe Message dos Aglets foi suficiente para implementar a comunicação necessária entre os agentes colaborativos.

Dentre os agentes criados, apenas o último, o que estipula o desconto, necessitaria de uma interface de defuzzyficação, mas todos eles possuem, prevendo a extensibilidade da aplicação, visto que outros agentes podem estar interessados em obter informações que se encontram com um agente intermediário, como por exemplo, o grau de interesse por um determinado produto, um agente poderia estar utilizando essa informação para fazer um marketing direcionado.

10 CONSIDERAÇÕES FINAIS

10.1 Conclusões

O desenvolvimento deste trabalho abordou um estudo sobre agentes colaborativos inteligentes. Como exemplo de aplicação, estudou-se um sistema de compra e venda implementado na linguagem Java, sobre o sistema Aglet, os quais proporcionam portabilidade, mobilidade e robustez aos agentes do sistema.

A utilização de sistemas multi-agentes, para viabilizar um sistema de compra e venda, mostra na prática que é possível automatizar boa parte das tarefas que compreendem os processos de compra e venda. Diferentes decisões demandam diferentes tempos para tomar efeito, dados por suas diferentes inércias. A inércia intrínseca dos processos decisórios sobre o critério estudado é diminuída pelo sistema proposto.

O paradigma de programação orientada a agentes mostrou alto nível de abstração o que facilitou a manutenção do sistema durante seu desenvolvimento, reforçando seu uso para sistemas ERP, além de tornar mais fácil o ato de delegar funções.

O equilíbrio do sistema é dinâmico, posto que, a incorporação das informações do meio e a evocação histórica dos fatos fazem com que o ponto de equilíbrio seja deslocado permanentemente, isto é conseqüência da busca de novas metas compatíveis com o novo estado de coisas e da constante retroalimentação de informações externas e internas com o conseqüente aprimoramento da estrutura.

A crescente importância dos agentes móveis aliada a já difundida linguagem Java, está trazendo muitos benefícios para construção de sistemas distribuídos em geral. O Sistema CUBe é apenas uma solução inicial de um destes aspectos. Com a sua estrutura, adicionada deste trabalho e propostas futuras, expostas mais adiante neste capítulo, faz com que este projeto traga benefícios e soluções a muitos problemas encontrados atualmente.

Segundo CRUZ (1998), "*Workflows de 4º nível (baseado no conhecimento) não existem ainda*", este trabalho vêm mostrar essa tendência e uma proposta de caminhar na busca desse nível.

10.2Dificuldades Encontradas

Que tipo de informação as empresas desejam coletar e manter, e como essas informações serão fornecidas ao funcionário que atende o cliente e de que forma serão utilizadas? Em outras palavras, que tipo de “memória organizacional” a empresa pretende desenvolver. Cada empresa possui sua “memória organizacional”, variáveis consideradas relevantes a vários administradores foram as definidas neste trabalho.

Dois grandes problemas foram encontrados e precisam ser solucionados na construção dos agentes de negociação. O primeiro problema é o da competência: como um agente adquiri conhecimento necessário para decidir sobre as decisões a tomar? Neste caso pode-se utilizar um especialista como interface para solucionar o problema, é a abordagem que está sendo utilizada, visto que modelar as funções de pertinência *fuzzy* para cada produto não é uma tarefa fácil.

O segundo problema é o da confiança: como garantir para a empresa a tranqüilidade de delegar tarefas, principalmente com relação ao item preço, a um agente?

10.3Trabalhos Futuros

Ampliar o número de variáveis sob análise, como por exemplo, a inadimplência do cliente e prazos de pagamento para clientes que não optarem pelo desconto, ou optarem por uma mesclagem das duas variáveis.

Isso nos leva a um problema de projeto. De alguma forma, então, é preciso encontrar um meio que reconheça as características centrais da complexidade e ainda assim considere pontos fortes e fracos da cognição humana.

Uma grande contribuição a ser dada para a continuidade desta pesquisa é que devido ao fato de que projetar funções de pertinência é uma tarefa trabalhosa, técnicas especiais usando redes neurais e/ou algoritmos genéticos poderiam ser utilizadas para sua geração automática. Tais agentes seriam dotados de uma capacidade de aprendizagem, a partir dos conjuntos de dados de entrada, com a qual identificariam a posição e formato das funções de pertinência, demonstrando dessa forma oportunismo com o passar do tempo.

Atualmente essa informação sobre as preferências individuais do cliente reside apenas na memória de cada vendedor, mas é preciso que essa informação seja convertida, passando da memória pessoal a memória organizacional.

GLOSSÁRIO

Activation	Ativação.
Agentes Móveis	
aplication-independent	Independente de aplicação.
Back-office	O que está por trás do escritório ou empresa; o que trabalha na estrutura interna da empresa.
Business-to-business	Transação comercial entre empresas.
Business-to-consumer	Transação comercial entre empresas e clientes.
Bytecodes	Arquivo pré-compilado gerado após a compilação de um programa em Java; portável para vários sistemas operacionais diferentes arquiteturas de computadores.
Casting	Mudança de tipo, podendo ocasionar perda de valor ou significado.
Cloning	Clonagem; duplicata exata de um original.
Commodities	Artigos ou objetos de utilidade; mercadorias.
Creation	Criação.
Deactivation	Desativação
Dispatching	Expedir.
Disposal	Liquidação.
e-Business	Comércio eletrônico realizado entre empresas (transações comerciais complexas envolvendo muitas vezes regras de negócio).
e-Commerce	Comércio eletrônico realizado entre uma empresa e um cliente (transação comercial simples).
e-mail	Correio eletrônico utilizado para troca de informações.
Framework	Estrutura, arquitetura; estrutura básica de uma base de dados ou processo ou programa.
Front-office	Parte da empresa que lida diretamente com o cliente, vendendo, prestando suporte ou qualquer outro tipo de serviços.

Future reply Resposta futura, geralmente usado em mecanismos de comunicação assíncrona.

Fuzzy Também chamada de lógica nebulosa, consegue tratar valores veritativos no intervalo [0,1].

Handle Driver ou manipulador; parte do sistema operacional ou programa que controla um periférico.

Hardware Unidades físicas, componentes, circuitos integrados, discos e mecanismos que compõe um computador ou seus periféricos.

Home-page Página eletrônica com informações diversas e acessível a qualquer pessoa que esteja conectada à Internet.

Interface (i) Ponto no qual um sistema de computação termina e outro começa; (ii) circuito, dispositivo ou porta que permite que duas ou mais unidades incompatíveis sejam interligadas em um sistema padrão de comunicação, permitindo que se transfiram dados entre eles; (iii) parte de um programa que permite a transmissão de dados para um outro programa.

Internet Rede de computadores internacional com informações acessíveis ao público através de um link de modem.

Intranet Funcionamento parecido com o da Internet, porém para troca de informações em um ambiente fechado e com um número pequeno de computadores.

Know-how Conhecimento prático; habilidade.

Listeners Ouvintes, ouvinte de algum evento.

Management Administração; controle.

Message Mensagem; recado.

Messaging Comunicação (por mensagens).

Multithread Multitransação; programa que usa mais de um passo lógico, com cada passo sendo executado concorrentemente.

Newsletter Relatório informativo.

Off-line Fora de linha; (i) (processador ou impressora ou terminal) que não está conectado a uma rede ou computador central; (ii) (periférico) conectado a uma rede, mas não disponível para uso.

On-line Em linha; (terminal ou dispositivo) conectado e sob o controle de um

processador central; no exato momento.

Persistence Persistência; característica definida a um arquivo ou objeto que fica armazenado em um meio permanente e pode ser recuperado futuramente.

Proxy Procurador; localizador.

Reply set Conjunto de resposta.

Retracting Retração.

Runtime Tempo de execução ou duração da execução; (i) intervalo de tempo que um programa leva para executar; (ii) tempo durante o qual um computador está executando em um programa; (iii) (operação) executada apenas quando um programa está sendo executado.

Site Endereço onde pode se localizar uma determinada *home-page*.

Software Qualquer programa ou grupo de programas que instrui o hardware sobre a maneira como ele deve executar uma tarefa, inclusive sistemas operacionais, processadores de texto e programas de aplicação.

Status Estado; importância ou posição.

String Cadeia ou seqüência; quaisquer séries de caracteres alfanuméricos ou palavras consecutivas que são manipuladas e tratadas como uma unidade pelo computador.

Supply-chain Cadeia de abastecimento.

Thread Em cadeia; programa que consiste de várias seções menores independentes.

Web Teia; também representa a Internet (WWW – *World Wide Web* ou Grande Teia Mundial).

Web browser Programa navegador; utilizado na Internet para visualização das *Home-Pages*.

Web server Servidor de Internet; utilizado para prover acesso de clientes a páginas (*home-pages*) e outros serviços disponíveis na Internet.

Workflow Fluxo de trabalho; programa que controla o fluxo de informações e trabalho de um grupo de usuários.

REFERENCIAS BIBLIOGRÁFICAS

Software Agents: an overview. URL: http://www.labs.bt.com\project\agents\publish\papers\review1.htm. Novembro, 1999.

BUENO, Francisco da Silveira. **Minidicionário da Língua Portuguesa**. Ed. Rev. e atual, por Helena Bonito C. Pereira, Rena Signer. São Paulo: FTD: LISA, 1996.

CORRÊA, Henrrique L.; GIANESI, Irineu G. N. **Just in Time, MRPII e OPT: um enfoque estratégico**. São Paulo: Atlas, 1993.

CORRÊA, Henrique L.; GIANESI, Irineu G. N.; CAON, Mauro. **Planejamento, Programação e Controle da Produção: MRP II/ERP: conceitos, uso e implantação**. São Paulo: Atlas, 1997.

CRUZ, Tadeu. ***Workflow*: a tecnologia que vai revolucionar processos**. São Paulo: Atlas, 1998.

DEITEL, Harvey M., Paul J. **Java: como programar**. 3ª ed., Porto Alegre: Bookman, 2001.

EDMONDS, Ernest A.; CANDY, Linda. **Support for Collaborative Design: agents and emergence**. Communications of the ACM, julho 1994/vol.37, nº 7, p.41-47.

GENESERETH, Michael R.; KETCHPEL, Steven P. **Software Agents**. Communications of the ACM, julho 1994/vol.37, nº 7, p.48-53.

GUHA, R. V.; LENAT, Douglas B. **Enabling Agents to Work Together**. Communications of the ACM, julho 1994/vol.37, nº 7, p.127-141.

JENNINGS, N. R.; WOOLDRIDGE, M. J. **Software Agents**. IEEE Review, January, 1996, p.17-20.

LANGE, Danny B.; OSHIMA, Mitsuru. **Programming and Deploying Java Mobile Agents with Aglets**. Massachusetts: Addison Wesley Longman, Inc, 1998, 225p.

MAES, Pattie. **Agents that Reduce Work and Information Overload**. Communications of the ACM, julho 1994/vol.37, nº 7, p.31-40.

MAES, Pattie. **Artificial Life Meets Entertainment: life like autonomous agents**. Special Issue on New Horizons of Commercial and Industrial AI, Vol. 38, No. 11, pp. 108-114, Communications of the ACM, ACM Press November, 1995.

MARTIN, James e MCCLURE, Carma. **Buying Software off the Rack**. Harvard Business Review, November/December 1983, p.32-60.

MCCARTY, Bill; DORION, Luke C. **Java: Distributed Objects**. Indianapolis, Indiana,

1999.

MCGEE, James; PRUSAK, Laurence. **Gerenciamento Estratégico da Informação**. 7ª ed., Rio de Janeiro: Campus, 1994.

MINSKY, Marvin. **A Conversation with Marvin Minsky About Agents**. Communications of the ACM, julho 1994/vol.37, nº 7, p.23-29.

NORMAN, Donald A. **How Might People Interact with Agents**. Communications of the ACM, julho 1994/vol.37, nº 7, p.68-71.

PUGLIESI, Márcio. **Conflito, Estratégia, Negociação: O Direito e sua Teoria**. WVC Editora: São Paulo, 2001.

SHAW, Ian S.; SIMÕES, Marcelo Godoy. **Controle e Modelagem Fuzzy**. 1ª ed., São Paulo: Edgard Blücher Ltda, 1999.

ORACLE Press. **Oracle 8i: programação de componentes Java com EJB, CORBA e JSP**. Rio de Janeiro: Campus, 2001.

VALDO, Clayton A. **CUBe: um sistema para integrar comércio eletrônico e ERP através de agentes móveis**. Dissertação de Mestrado, Universidade Federal de Santa Catarina (UFSC), 1999.

ANEXOS

Anexo A – Modelo de Dados do CUBe

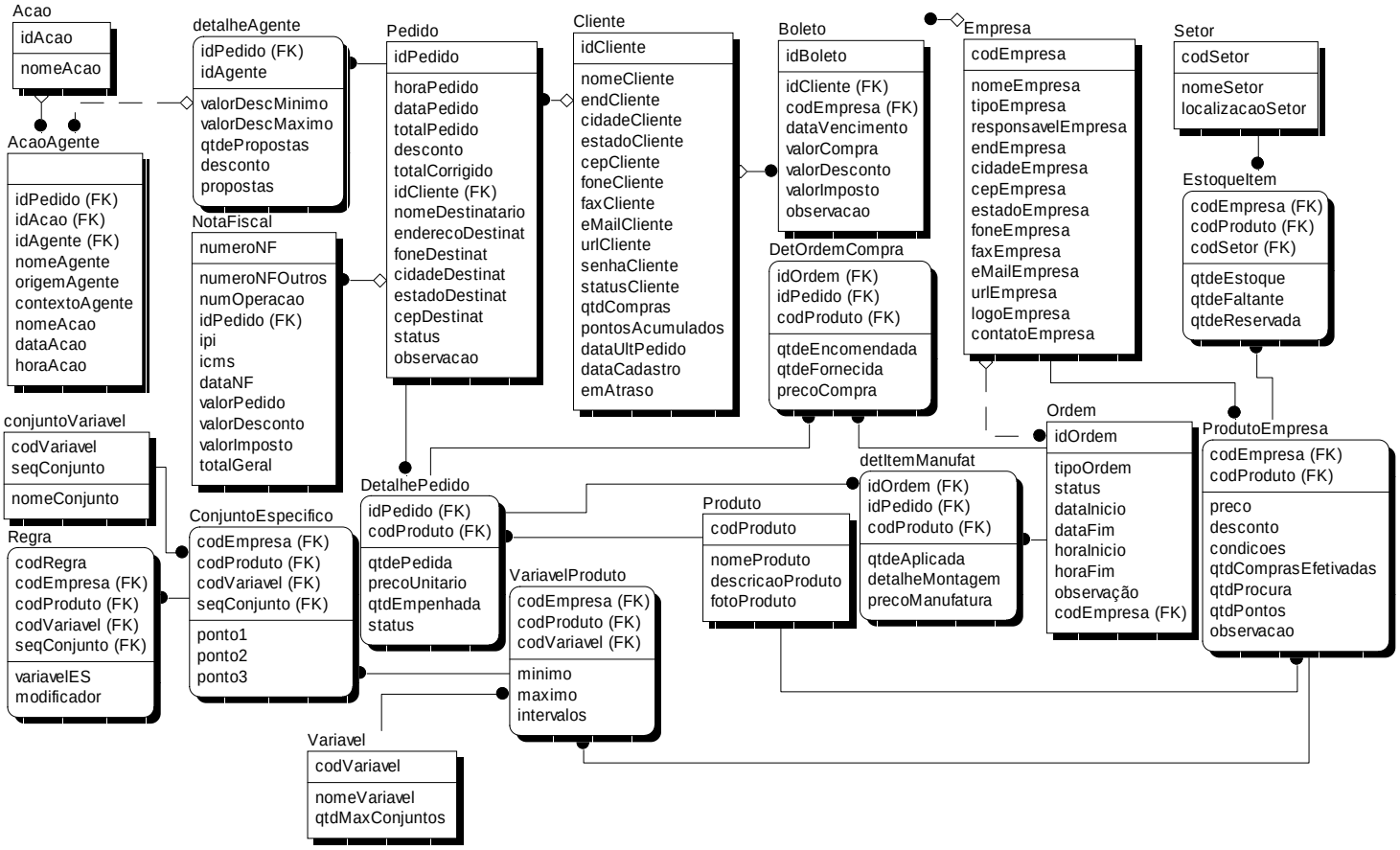

Figura A.1 – Diagrama Entidade-Relacionamento (CUBe)

Anexo B – Diagramas de Caso de Uso

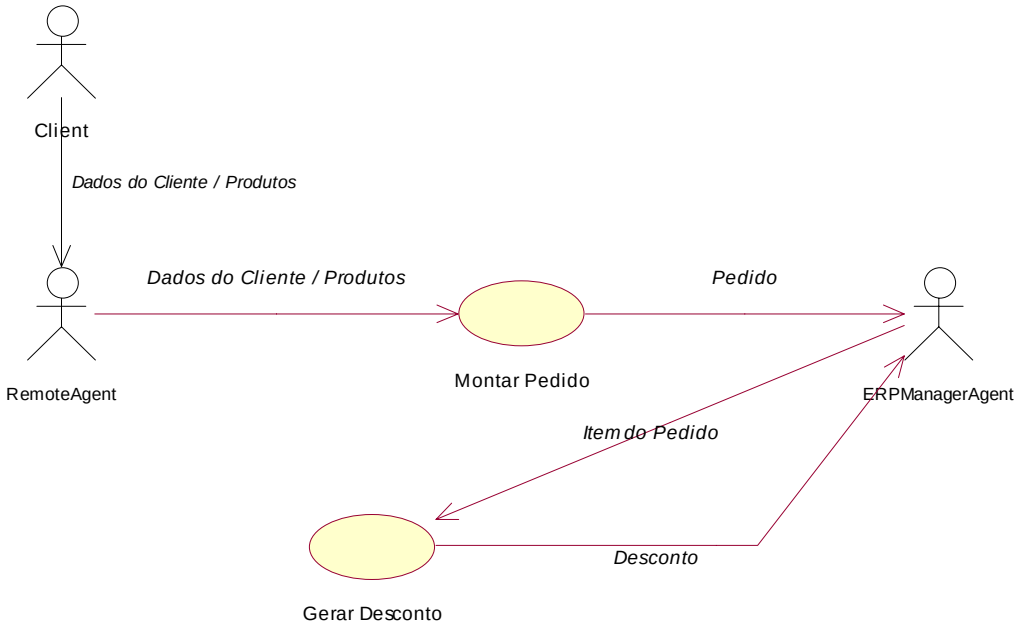

Figura B.1 – Caso de Uso Principal

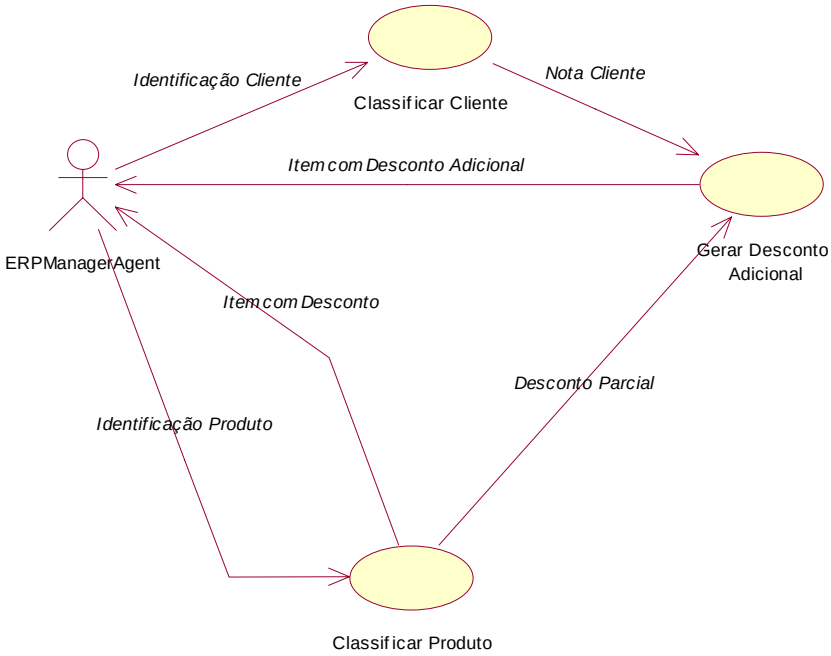

Figura B.2 – Caso de Uso Expandido – Gerar Desconto

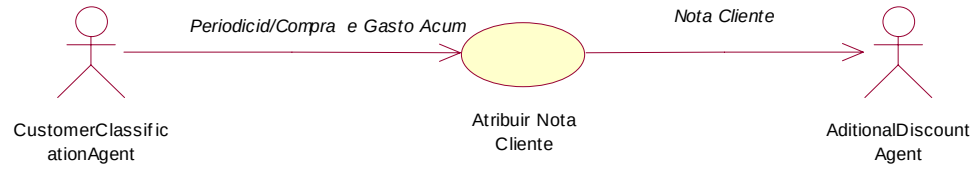

Figura B.3 – Caso de Uso Expandido – Classificar Cliente

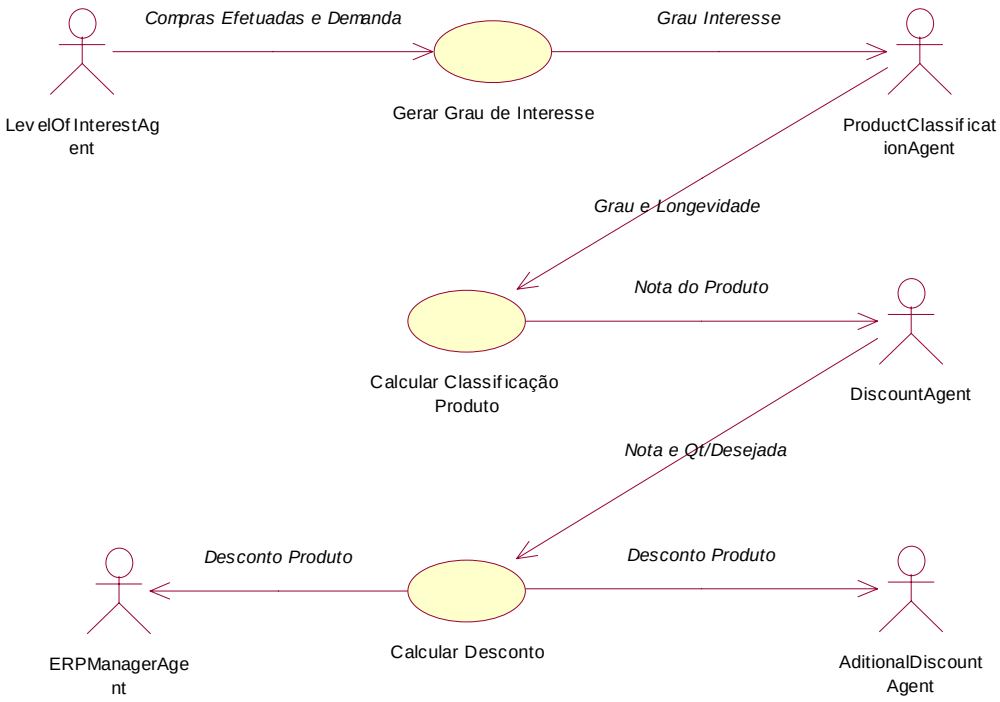

Figura B.4 – Caso de Uso Expandido – Classificar Produto

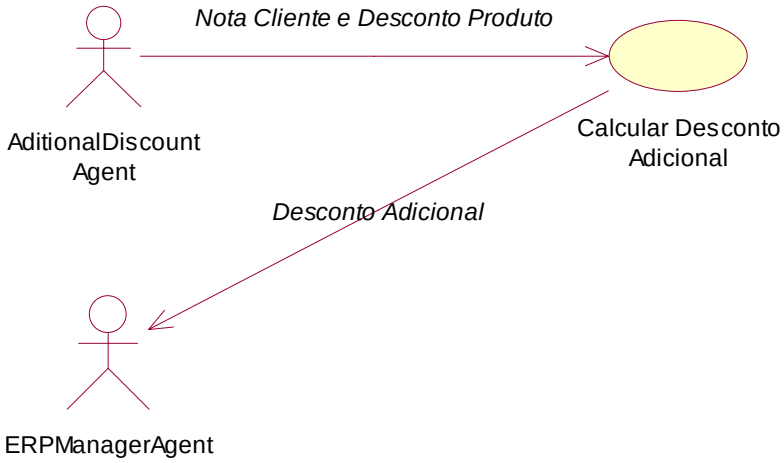

Figura B.5 – Caso de Uso Expandido – Gerar Desconto Adicional

Anexo C – Diagrama de Classes

Figura C.1 – Diagrama de Classes dos Agentes Modificados para Negociação

Anexo D – Código Fonte da Classe *Fuzzy*

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import com.ibm.aglet.*;
import com.ibm.aglet.event.*;
import java.util.Enumeration;

public class Fuzzy extends Aglet
{
	public AgletID aid;
	public AgletProxy proxy;

	//atributos da compra
	double e1_minimo,e1_maximo,e1_intervalos;
	double e1_a1,e1_a2,e1_b1,e1_b2,e1_b3,e1_c1,e1_c2,e1_c3,e1_d1,e1_d2,
		e1_d3,e1_e1,e1_e2;
	//atributos da procura
	double e2_minimo,e2_maximo,e2_intervalos;
	double e2_a1,e2_a2,e2_b1,e2_b2,e2_b3,e2_c1,e2_c2,e2_c3,e2_d1,e2_d2,
		e2_d3,e2_e1,e2_e2;
	//atributos do interesse
	double s_minimo,s_maximo,s_intervalos;
	double s_a1,s_a2,s_b1,s_b2,s_b3,s_c1,s_c2,s_c3,s_d1,s_d2,s_d3,s_e1,s_e2;

	double universoDiscursoE1, universoDiscursoE2, intervaloDifusorE1,
		intervaloDifusorE2, difMinimo, difMaximo, centroUnivE1, centroUnivE2;

	final static int NUMEROREGLAS = 25;
	final static int NUMEROVARENT = 02;
	final static int NUMEROVARSAI = 01;
	final static int NUMTOTCONJENT = 10;

	Variable variablesEntrada[];
	Variable variablesSalida[];
	Conjunto entradas[];
	Difusor difusores[];

	public int reglas[][];
	double modificadores[][];
	public double centrosAltura[][];

	public double entra[];
	public double sale[];

	double centroAltura1;
```

```
        double centroAltura2;
        double centroAltura3;
        double centroAltura4;
        double centroAltura5;

        public void onCreation(Object init)
        {
          entra = new double[2];
          sale = new double[1];

          reglas = new
int[NUMEROREGLAS][NUMEROVARENT+NUMEROVARSAI];
          modificadores = new double[NUMEROREGLAS][NUMEROVARENT];
          centrosAltura = new double[NUMEROREGLAS][NUMEROVARSAI];

          variablesEntrada = new Variable[2];
          variablesSalida =  new Variable[1];

          entradas = new Conjunto[10];
          difusores = new Difusor[2];

        modificadores[0][0]=1.000000;modificadores[0][1]=1.000000;
        modificadores[1][0]=1.000000;modificadores[1][1]=1.000000;
        modificadores[2][0]=1.000000;modificadores[2][1]=1.000000;
        modificadores[3][0]=1.000000;modificadores[3][1]=1.000000;
        modificadores[4][0]=1.000000;modificadores[4][1]=1.000000;
        modificadores[5][0]=1.000000;modificadores[5][1]=1.000000;
        modificadores[6][0]=1.000000;modificadores[6][1]=1.000000;
        modificadores[7][0]=1.000000;modificadores[7][1]=1.000000;
        modificadores[8][0]=1.000000;modificadores[8][1]=1.000000;
        modificadores[9][0]=1.000000;modificadores[9][1]=1.000000;
        modificadores[10][0]=1.000000;modificadores[10][1]=1.000000;
        modificadores[11][0]=1.000000;modificadores[11][1]=1.000000;
        modificadores[12][0]=1.000000;modificadores[12][1]=1.000000;
        modificadores[13][0]=1.000000;modificadores[13][1]=1.000000;
        modificadores[14][0]=1.000000;modificadores[14][1]=1.000000;
        modificadores[15][0]=1.000000;modificadores[15][1]=1.000000;
        modificadores[16][0]=1.000000;modificadores[16][1]=1.000000;
        modificadores[17][0]=1.000000;modificadores[17][1]=1.000000;
        modificadores[18][0]=1.000000;modificadores[18][1]=1.000000;
        modificadores[19][0]=1.000000;modificadores[19][1]=1.000000;
        modificadores[20][0]=1.000000;modificadores[20][1]=1.000000;
        modificadores[21][0]=1.000000;modificadores[21][1]=1.000000;
        modificadores[22][0]=1.000000;modificadores[22][1]=1.000000;
        modificadores[23][0]=1.000000;modificadores[23][1]=1.000000;
        modificadores[24][0]=1.000000;modificadores[24][1]=1.000000;
        }
```

```
public void configurarAgente(int codigo,int e1, int e2, int s)
{
  String sqlE1 = "SELECT CONJUNTO.*, minimo, maximo, intervalos " +
        "FROM VARIAVEL_PRODUTO INNER JOIN CONJUNTO ON
(VARIAVEL_PRODUTO.codVariavel = CONJUNTO.codVariavel) AND
(VARIAVEL_PRODUTO.cod_Produto = CONJUNTO.cod_Produto) AND
(VARIAVEL_PRODUTO.cod_Empresa = CONJUNTO.cod_Empresa) " +
        "WHERE (((CONJUNTO.cod_Empresa)=1) AND
((CONJUNTO.cod_Produto)=" + codigo +") AND ((CONJUNTO.codVariavel)=" + e1
+ "))";
  String sqlE2 = "SELECT CONJUNTO.*, minimo, maximo, intervalos " +
        "FROM VARIAVEL_PRODUTO INNER JOIN CONJUNTO ON
(VARIAVEL_PRODUTO.codVariavel = CONJUNTO.codVariavel) AND
(VARIAVEL_PRODUTO.cod_Produto = CONJUNTO.cod_Produto) AND
(VARIAVEL_PRODUTO.cod_Empresa = CONJUNTO.cod_Empresa) " +
        "WHERE (((CONJUNTO.cod_Empresa)=1) AND
((CONJUNTO.cod_Produto)=" + codigo +") AND ((CONJUNTO.codVariavel)=" + e2
+ "))";
  String sqlS  = "SELECT CONJUNTO.*, minimo, maximo, intervalos " +
        "FROM VARIAVEL_PRODUTO INNER JOIN CONJUNTO ON
(VARIAVEL_PRODUTO.codVariavel = CONJUNTO.codVariavel) AND
(VARIAVEL_PRODUTO.cod_Produto = CONJUNTO.cod_Produto) AND
(VARIAVEL_PRODUTO.cod_Empresa = CONJUNTO.cod_Empresa) " +
        "WHERE (((CONJUNTO.cod_Empresa)=1) AND
((CONJUNTO.cod_Produto)=" + codigo +") AND ((CONJUNTO.codVariavel)=" + s
+ "))";
    try
    {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      String fonteURL = new String("jdbc:odbc:CUBE");
      Connection dbConect = DriverManager.getConnection(fonteURL);
      Statement comando = dbConect.createStatement();
      ResultSet recSet = comando.executeQuery(sqlE1);
      if (RecordNumber(recSet) > 0)
      {
        recSet = comando.executeQuery(sqlE1);
        recSet.next();
        e1_minimo    = recSet.getDouble("minimo");
        e1_maximo    = recSet.getDouble("maximo");
        e1_intervalos  = recSet.getDouble("intervalos");
        e1_a1          = recSet.getDouble("ponto1");
        e1_a2        = recSet.getDouble("ponto2");
        recSet.next();
        e1_b1        = recSet.getDouble("ponto1");
        e1_b2        = recSet.getDouble("ponto2");
        e1_b3        = recSet.getDouble("ponto3");
        recSet.next();
        e1_c1        = recSet.getDouble("ponto1");
```

```
                e1_c2           = recSet.getDouble("ponto2");
                e1_c3           = recSet.getDouble("ponto3");
                recSet.next();
                e1_d1           = recSet.getDouble("ponto1");
                e1_d2           = recSet.getDouble("ponto2");
                e1_d3           = recSet.getDouble("ponto3");
                recSet.next();
                e1_e1           = recSet.getDouble("ponto1");
                e1_e2           = recSet.getDouble("ponto2");
        variablesEntrada[0] = new
Variable("Entrada1",e1_minimo,e1_maximo,e1_intervalos,(e1_maximo-
e1_minimo)/e1_intervalos,5);
                entradas[0] = new Conjunto(e1_minimo,e1_a2);
                entradas[1] = new Conjunto(e1_b1,e1_b3);
                entradas[2] = new Conjunto(e1_c1,e1_c3);
                entradas[3] = new Conjunto(e1_d1,e1_d3);
                entradas[4] = new Conjunto(e1_e1,e1_maximo);
             }
             recSet = comando.executeQuery(sqlE2);
             if (RecordNumber(recSet) > 0)
             {
                recSet = comando.executeQuery(sqlE2);
                recSet.next();
                e2_minimo       = recSet.getLong("minimo");
                e2_maximo       = recSet.getLong("maximo");
                e2_intervalos   = recSet.getLong("intervalos");
                e2_a1               = recSet.getDouble("ponto1");
                e2_a2           = recSet.getDouble("ponto2");
                recSet.next();
                e2_b1           = recSet.getDouble("ponto1");
                e2_b2           = recSet.getDouble("ponto2");
                e2_b3           = recSet.getDouble("ponto3");
                recSet.next();
                e2_c1           = recSet.getDouble("ponto1");
                e2_c2           = recSet.getDouble("ponto2");
                e2_c3           = recSet.getDouble("ponto3");
                recSet.next();
                e2_d1           = recSet.getDouble("ponto1");
                e2_d2           = recSet.getDouble("ponto2");
                e2_d3           = recSet.getDouble("ponto3");
                recSet.next();
                e2_e1           = recSet.getDouble("ponto1");
                e2_e2           = recSet.getDouble("ponto2");
                variablesEntrada[1] = new
Variable("Entrada2",e2_minimo,e2_maximo,e2_intervalos,(e2_maximo-
e2_minimo)/e2_intervalos,5);
                entradas[5] = new Conjunto(e2_minimo,e2_a2);
                entradas[6] = new Conjunto(e2_b1,e2_b3);
```

```
                entradas[7] = new Conjunto(e2_c1,e2_c3);
                entradas[8] = new Conjunto(e2_d1,e2_d3);
                entradas[9] = new Conjunto(e2_e1,e2_maximo);
            }
            recSet = comando.executeQuery(sqlS);
            if (RecordNumber(recSet) > 0)
            {
                recSet = comando.executeQuery(sqlS);
                recSet.next();
                s_minimo        = recSet.getLong("minimo");
                s_maximo        = recSet.getLong("maximo");
                s_intervalos    = recSet.getLong("intervalos");
                s_a1              = recSet.getDouble("ponto1");
                s_a2          = recSet.getDouble("ponto2");
                recSet.next();
                s_b1          = recSet.getDouble("ponto1");
                s_b2          = recSet.getDouble("ponto2");
                s_b3          = recSet.getDouble("ponto3");
                recSet.next();
                s_c1          = recSet.getDouble("ponto1");
                s_c2          = recSet.getDouble("ponto2");
                s_c3          = recSet.getDouble("ponto3");
                recSet.next();
                s_d1          = recSet.getDouble("ponto1");
                s_d2          = recSet.getDouble("ponto2");
                s_d3          = recSet.getDouble("ponto3");
                recSet.next();
                s_e1          = recSet.getDouble("ponto1");
                s_e2          = recSet.getDouble("ponto2");
                variablesSalida[0] =  new
Variable("Saída",s_minimo,s_maximo,s_intervalos,(s_maximo-
s_minimo)/s_intervalos,5);
              }
        dbConect.close();
          }
          catch(ClassNotFoundException except)
          {
       System.out.println(except);        // ClassNotFoundException
        }
        catch(SQLException sqlexcept)
        {
       System.out.println(sqlexcept);        // SQLException
        }
          //Cálculo do Difusor
          universoDiscursoE1 = e1_maximo - e1_minimo;
      if (universoDiscursoE1 <0)
      {
                universoDiscursoE1 = universoDiscursoE1 * -1;
```

```
        }
        universoDiscursoE2 = e2_maximo - e2_minimo;
      if (universoDiscursoE2 <0)
      {
              universoDiscursoE2 = universoDiscursoE2 * -1;
        }

       intervaloDifusorE1 = universoDiscursoE1 * 0.000025;
       intervaloDifusorE2 = universoDiscursoE2 * 0.000025;
       centroUnivE1 = e1_minimo + universoDiscursoE1/2;
       centroUnivE2 = e2_minimo + universoDiscursoE2/2;
       difMinimo = centroUnivE1 - intervaloDifusorE1;
       difMaximo = centroUnivE1 + intervaloDifusorE1;
       difusores[0] = new
Difusor(difMinimo,difMaximo,centroUnivE1,1,intervaloDifusorE1,difMinimo,difMaxi
mo,0,0);
        difMinimo = centroUnivE2 - intervaloDifusorE2;
       difMaximo = centroUnivE2 + intervaloDifusorE2;
        difusores[1] = new
Difusor(difMinimo,difMaximo,centroUnivE2,1,intervaloDifusorE2,difMinimo,difMaxi
mo,0,0);

        //Cálculo do Centro Altura
        centroAltura1 = s_minimo + ((s_a1 - s_minimo)/2);
        centroAltura2 = (s_b2);
        centroAltura3 = (s_c2);
        centroAltura4 = (s_d2);
        centroAltura5 = s_e2 + ((s_maximo - s_e2)/2);
       }

      public void calcular()
      {
        for(int i=0;i<NUMEROVARSAI;i++)
         {
           sale[i]=salidaConcreta(i);
      }
      }

      double salidaConcreta(int numeroSalida)
      {
         double respuesta;
         switch(numeroSalida)
         {
           case 0 : respuesta=salidaConcreta0();break;
           default:respuesta=0;break;
         }
      return respuesta;
      }
```

```
        double salidaConcreta0()
        {
           int numeroSalida=0;
           double con=0.000000;
           double y;
           double y1=0;
           double y2=0;
           double ymax;
           int i;
           int j;
           ymax=variablesSalida[numeroSalida].minimo;
           actualizarEntradas();
           for(i=0;i<(variablesSalida[numeroSalida].intervalos+1);i++)
           {
              con=0.000000;

y=variablesSalida[numeroSalida].minimo+i*variablesSalida[numeroSalida].intervalo;
              for(j=0;j<NUMEROREGLAS;j++)
              {
                 double temp;
                 temp=pertenenciaComposicion(numeroSalida,j,y);
                 con=Conjuncion(con,temp);
              }
              y1=y1+y*con;
              y2=y2+con;
           }
           if(Math.abs(y2)<0.000001)
              y2=100000.0;
           if(Math.abs(y1)<0.000001)
              y1=0.0;
           ymax=y1/y2;
           return ymax;
        }

        double pertenenciaComposicion(int numVar, int numRegla, double sal)
        {
           double ux;
           double uxa;
           double uxab;
           double comp=0;
           double x[] = new double[NUMEROVARENT];
           int inter[] = new int[NUMEROVARENT];
           if(!activarRegla(numRegla))
           {
              comp=Implicacion(0,0);
           }
           else
           {
```

```
            int casos=1;
            int i;
            for(i=0;i<NUMEROVARENT;i++)
            {
               casos=casos*difusores[i].puntos;
               inter[i]=1;
            }
            for(i=0;i<NUMEROVARENT;i++)
            {
               int k;
               for(k=0;k<NUMEROVARENT;k++)
               {
                  x[k]=difusores[k].minimo+difusores[k].intervalo*inter[k];
                  inter[k]=inter[k]+1;
                  if(inter[k]>=difusores[k].puntos)
                  {
                     inter[k]=1;
                  }
               }
               uxab=pertenenciaImplicacion(numVar,numRegla,x,sal);
               uxa=pertenenciaDifusores(x);
               ux=Composicion(uxa,uxab);
               if(ux>comp)
               {
                  comp=ux;
               }
            }
         }
         return comp;
      }

      double pertenenciaDifusores(double entra[])
      {
         double uxd;
         int j=0;
         uxd=pertenenciaDifusor(j,entra[j]);
         for(j=0;j<NUMEROVARENT;j++)
         {
                uxd=And(uxd,pertenenciaDifusor(j,entra[j]));
         }
         return uxd;
      }

      double pertenenciaImplicacion(int numSal, int numRegla, double entra[], double
sal)
      {
         double uxa,uxb;
         uxa=pertenenciaAntecedente(numRegla);
```

```
   uxb=pertenenciaConsecuente(numSal,numRegla,sal);
   return Implicacion(uxa,uxb);
}

double pertenenciaConsecuente(int numSal, int numRegla, double sal)
{
   double uxc;
   int conj;
   conj=reglas[numRegla][NUMEROVARENT+numSal];
   uxc=pertenenciaVariableSalida(numSal,conj,sal);
   return uxc;
}

double pertenenciaAntecedente(int numRegla)
{
   double ux;
   double uxa;
   int conj;
   int j=0;
   conj=reglas[numRegla][0];
   ux=pertenenciaVariableEntrada(j,conj,entra[j]);
   if(modificadores[numRegla][0]>0.0)
   {
      uxa=Math.pow(ux,modificadores[numRegla][0]);
   }
   else
   {
      uxa=1;
   }
   for(j=1;j<NUMEROVARENT;j++)
   {
      conj=reglas[numRegla][j];
      ux=pertenenciaVariableEntrada(j,conj,entra[j]);
      if(modificadores[numRegla][j]>0.0)
      {
         ux=Math.pow(ux,modificadores[numRegla][j]);
      }
      else
      {
         ux=1;
      }
      uxa=And(uxa,ux);
   }
   return uxa;
}

boolean activarRegla(int numRegla)
{
```

```
    int i;
    for(i=0;i<NUMEROVARENT;i++)
    {
            double bmn,bmx,cmn,cmx;
        int numCon;
        int indice;
        numCon=reglas[numRegla][i];
        indice=indiceEntradas(i,numCon);
        bmn=entradas[indice].minimo;
        bmx=entradas[indice].maximo;
        cmn=difusores[i].minimo;
        cmx=difusores[i].maximo;
        if(bmn>cmx||bmx<cmn)
            return false; //0
    }
    return true; //1
}

double pertenenciaDifusor(int numVar, double x)
{
    double ux=0;
    switch(numVar)
    {
        case 0:
            if(x<difusores[0].var1)
              ux=0;
            if(x<difusores[0].var2 && x>=difusores[0].var1)
              ux=1;
            if(x>=difusores[0].var2)
              ux=0;
            break;
        case 1:
            if(x<difusores[1].var1)
              ux=0;
            if(x<difusores[1].var2 && x>=difusores[1].var1)
              ux=1;
            if(x>=difusores[1].var2)
              ux=0;
            break;
        default:break;
    }
    return ux;
}

double pertenenciaVariableEntrada(int numVar,int numConj, double x)
{
    double ux=0;
    switch(numVar)
```

```
{
   case 0:
      switch(numConj)
      {
      case 0:
               if(x<(e1_minimo))
                  ux=1;
               if(x<(e1_a1)&&x>=(e1_minimo))
                  ux=1;
               if(x<(e1_a2)&&x>=(e1_a1))
                  ux=((e1_a2)-x)/((e1_a2)-(e1_a1));
               if(x>=(e1_a2))
                  ux=0;
               if(ux<0.0001)
                  ux=0;
         break;
      case 1:
               if(x<(e1_b1))
                  ux=0;
               if(x<(e1_b2)&&x>=(e1_b1))
                  ux=(x-(e1_b1))/((e1_b2)-(e1_b1));
               if(x<(e1_b3)&&x>=(e1_b2))
                  ux=((e1_b3)-x)/((e1_b3)-(e1_b2));
               if(x>=(e1_b3))
                  ux=0;
               if(ux<0.0001)
                  ux=0;
         break;
      case 2:
               if(x<(e1_c1))
                  ux=0;
               if(x<(e1_c2)&&x>=(e1_c1))
                  ux=(x-(e1_c1))/((e1_c2)-(e1_c1));
               if(x<(e1_c3)&&x>=(e1_c2))
                  ux=((e1_c3)-x)/((e1_c3)-(e1_c2));
               if(x>=(e1_c3))
                  ux=0;
               if(ux<0.0001)
                  ux=0;
         break;
      case 3:
               if(x<(e1_d1))
                  ux=0;
               if(x<(e1_d2)&&x>=(e1_d1))
                  ux=(x-(e1_d1))/((e1_d2)-(e1_d1));
               if(x<(e1_d3)&&x>=(e1_d2))
                  ux=((e1_d3)-x)/((e1_d3)-(e1_d2));
               if(x>=(e1_d3))
```

```
                ux=0;
            if(ux<0.0001)
                ux=0;
        break;
    case 4:
            if(x<(e1_e1))
                ux=0;
            if(x<(e1_e2)&&x>=(e1_e1))
                ux=(x-(e1_e1))/((e1_e2)-(e1_e1));
            if(x>=(e1_e2))
                ux=1;
            if(ux<0.0001)
                ux=0;
        break;
    default:break;
    }break;
case 1:
    switch(numConj)
    {
    case 0:
            if(x<(e2_minimo))
                ux=1;
            if(x<(e2_a1)&&x>=(e2_minimo))
                ux=1;
            if(x<(e2_a2)&&x>=(e2_a1))
                ux=((e2_a2)-x)/((e2_a2)-(e2_a1));
            if(x>=(e2_a2))
                ux=0;
            if(ux<0.0001)
                ux=0;
        break;
    case 1:
            if(x<(e2_b1))
                ux=0;
            if(x<(e2_b2)&&x>=(e2_b1))
                ux=(x-(e2_b1))/((e2_b2)-(e2_b1));
            if(x<(e2_b3)&&x>=(e2_b2))
                ux=((e2_b3)-x)/((e2_b3)-(e2_b2));
            if(x>=(e2_b3))
                ux=0;
            if(ux<0.0001)
                ux=0;
        break;
    case 2:
            if(x<(e2_c1))
                ux=0;
            if(x<(e2_c2)&&x>=(e2_c1))
                ux=(x-(e2_c1))/((e2_c2)-(e2_c1));
```

```
                if(x<(e2_c3)&&x>=(e2_c2))
                   ux=((e2_c3)-x)/((e2_c3)-(e2_c2));
                if(x>=(e2_c3))
                   ux=0;
                if(ux<0.0001)
                   ux=0;
             break;
          case 3:
                if(x<(e2_d1))
                   ux=0;
                if(x<(e2_d2)&&x>=(e2_d1))
                   ux=(x-(e2_d1))/((e2_d2)-(e2_d1));
                if(x<(e2_d3)&&x>=(e2_d2))
                   ux=((e2_d3)-x)/((e2_d3)-(e2_d2));
                if(x>=(e2_d3))
                   ux=0;
                if(ux<0.0001)
                   ux=0;
             break;
          case 4:
                if(x<(e2_e1))
                   ux=0;
                if(x<(e2_e2)&&x>=(e2_e1))
                   ux=(x-(e2_e1))/((e2_e2)-(e2_e1));
                if(x>=(e2_e2))
                   ux=1;
                if(ux<0.0001)
                   ux=0;
             break;
          default:break;
          }break;
       default:break;
   }
   return ux;
}

double pertenenciaVariableSalida(int numVar,int numConj, double x)
{
   double ux=0;
   switch(numVar)
   {
      case 0:
         switch(numConj)
         {
         case 0:
                if(x<(s_minimo))
                   ux=1;
                if(x<(s_a1)&&x>=(s_minimo))
```

```
            ux=1;
        if(x<(s_a2)&&x>=(s_a1))
            ux=((s_a2)-x)/((s_a2)-(s_a1));
        if(x>=(s_a2))
            ux=0;
        if(ux<0.0001)
            ux=0;
    break;
case 1:
        if(x<(s_b1))
            ux=0;
        if(x<(s_b2)&&x>=(s_b1))
            ux=(x-(s_b1))/((s_b2)-(s_b1));
        if(x<(s_b3)&&x>=(s_b2))
            ux=((s_b3)-x)/((s_b3)-(s_b2));
        if(x>=(s_b3))
            ux=0;
        if(ux<0.0001)
            ux=0;
    break;
case 2:
        if(x<(s_c1))
            ux=0;
        if(x<(s_c2)&&x>=(s_c1))
            ux=(x-(s_c1))/((s_c2)-(s_c1));
        if(x<(s_c3)&&x>=(s_c2))
            ux=((s_c3)-x)/((s_c3)-(s_c2));
        if(x>=(s_c3))
            ux=0;
        if(ux<0.0001)
            ux=0;
    break;
case 3:
        if(x<(s_d1))
            ux=0;
        if(x<(s_d2)&&x>=(s_d1))
            ux=(x-(s_d1))/((s_d2)-(s_d1));
        if(x<(s_d3)&&x>=(s_d2))
            ux=((s_d3)-x)/((s_d3)-(s_d2));
        if(x>=(s_d3))
            ux=0;
        if(ux<0.0001)
            ux=0;
    break;
case 4:
        if(x<(s_e1))
            ux=0;
        if(x<(s_e2)&&x>=(s_e1))
```

```
                    ux=(x-(s_e1))/((s_e2)-(s_e1));
                if(x>=(s_e2))
                    ux=1;
                if(ux<0.0001)
                    ux=0;
            break;
        default:break;
        }break;
    default:break;
  }
  return ux;
}

int indiceEntradas(int numVar,int numConj)
{
  int contador=0;
  int i;
  for(i=0;i<numVar;i++)
  {
    contador=contador+variablesEntrada[i].numeroConjuntos;
  }
  contador=contador+numConj;
  return contador;
}

void actualizarEntradas()
{
  double dx;
  int i;
  for(i=0;i<NUMEROVARENT;i++)
  {
    dx=entra[i]-difusores[i].centro;
    difusores[i].centro=difusores[i].centro+dx;
    difusores[i].minimo=difusores[i].minimo+dx;
    difusores[i].maximo=difusores[i].maximo+dx;
    difusores[i].var1=difusores[i].var1+dx;
    difusores[i].var2=difusores[i].var2+dx;
    difusores[i].var3=difusores[i].var3+dx;
    difusores[i].var4=difusores[i].var4+dx;
  }
}

double Implicacion(double x,double y)
{
  double rel;
  if(x<y)
  {
        rel=x;
```

```
    }
    else
    {
            rel=y;
    }
    return rel;
}

double Composicion(double x,double y)
{
    double z;
    if(x<y)
    {
            z=x;
    }
    else
    {
            z=y;
    }
    return z;
}

double Conjuncion(double x,double y)
{
    double z;
    if(x>y)
    {
            z=x;
    }
    else
    {
            z=y;
    }
    return z;
}

double And(double x,double y)
{
    double z;
    if(x<y)
    {
            z=x;
    }
    else
    {
            z=y;
    }
    return z;
```

```
        }

    //-------------------------------------------------------------------------
    //  MÉTODO: Print(String,String)
    //  Objetivo: Mostrar um texto no objeto TextArea da janela do contexto.
    //  Parâmetros: anAction - o rótulo da ação.
    //              aText - o texto a ser exibido.
    //
    public void Print(String anAction, String aText)
    {
      try
      {
        setText("[ "+getProxy().getAgletClassName()+
              " ]  "+anAction+" : "+aText);
      }
      catch(InvalidAgletException except){}
    }

    //-------------------------------------------------------------------------
    //  MÉTODO: RecordNumber(ResultSet)
    //  Objetivo: Retornar o número de registros de uma consulta.
    //  Parâmetros: umResultSet - a consulta com o número de registros a ser
    //                            verificado.
    //
    private int RecordNumber(ResultSet umResultSet)
    {
      int numeroRegistros = 0;

      try{
        while (umResultSet.next())
          numeroRegistros++;
      }
      catch(SQLException sqlexcept){
        numeroRegistros = 0;
      }
      return numeroRegistros;
    }
}
```