

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

Clayton Augusto Valdo

**CUBe: Um Sistema para Integrar Comércio Eletrônico
e ERP através de Agentes Móveis**

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

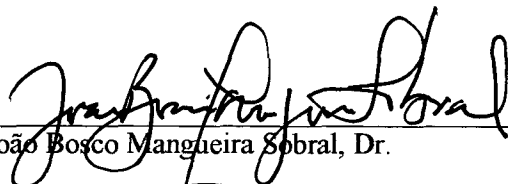
João Bosco Manguiera Sobral

Florianópolis, Setembro de 2000.

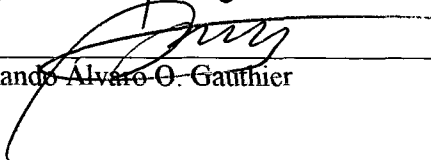
CUBe: Um Sistema para Integrar Comércio Eletrônico e ERP através de Agentes Móveis

Clayton Augusto Valdo

Esta Dissertação foi julgada adequada para a obtenção do título de **MESTRE EM CIÊNCIA DA COMPUTAÇÃO** na área de **SISTEMAS DE COMPUTAÇÃO** e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

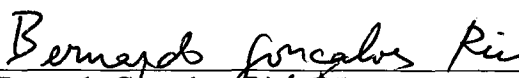


João Bosco Manguiera Sobral, Dr.

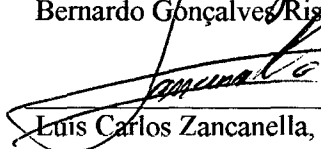


Fernando Alvaro O. Gauthier

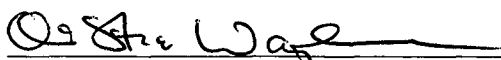
Banca Examinadora



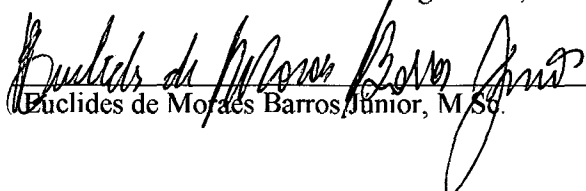
Bernardo Gonçalves Riso, Dr.



Luis Carlos Zancanella, Dr.



Christiane Annaliese G. von Wangenheim, Dra.



Euclides de Moraes Barros Junior, M.Sc.

A todas as pessoas que fazem da busca
pelo conhecimento parte preponderante de
suas vidas.

Este trabalho é dedicado à minha mãe
Maria, meu pai Ademar, minha irmã
Graciela, e em especial à minha noiva Val;
pessoas que sempre me incentivaram nas
horas mais difíceis e tiveram um papel
fundamental para a sua conclusão.

Sumário

| | |
|--|------|
| Lista de Figuras..... | viii |
| Lista de Tabelas | x |
| Lista de Anexos | xi |
| Lista de Abreviaturas e Siglas | xii |
| Resumo..... | xiii |
| Abstract | xiv |
| 1. Introdução..... | 15 |
| 1.1 Trabalhos Estudados..... | 16 |
| 1.2 Trabalhos Publicados..... | 16 |
| 1.3 Estrutura do Trabalho | 17 |
| 2. Comércio Eletrônico | 18 |
| 2.1 Visão Geral | 18 |
| 2.2 Transações: Modelo Tradicional X Comércio Eletrônico..... | 20 |
| 2.3 Mais do que a soma de suas partes..... | 21 |
| 2.3.1 <i>Distribuição da informação</i> | 22 |
| 2.3.2 <i>Ordem</i> | 23 |
| 2.3.3 <i>Pagamento</i> | 24 |
| 2.3.4 <i>Execução</i> | 24 |
| 2.3.5 <i>Suporte e serviços</i> | 24 |
| 2.4 Benefícios do Comércio Eletrônico | 25 |
| 2.5 Diferenças entre <i>e-business</i> e <i>e-commerce</i> | 25 |
| 2.5.1 <i>Business-to-business</i> | 26 |
| 2.5.2 <i>Business-to-consumer</i> | 27 |
| 3. Sistemas ERP | 28 |
| 3.1 Funcionalidades dos Sistemas ERP..... | 29 |
| 3.2 Termos relacionados aos Sistemas ERP | 29 |
| 4. Agentes..... | 32 |
| 4.1 Agentes de Software..... | 33 |
| 4.2 Características importantes dos agentes móveis | 34 |
| 4.3 Paradigmas da Computação em Rede | 35 |

| | | |
|-------|--|----|
| 4.4 | Aplicações de Agentes Móveis | 38 |
| 4.5 | Estado-da-Arte em Sistemas de Agentes Móveis | 38 |
| 4.5.1 | <i>Aglets</i> | 38 |
| 4.5.2 | <i>Voyager</i> | 39 |
| 4.6 | Elementos de um Sistema de Agentes Móveis | 39 |
| 4.6.1 | <i>Agente</i> | 39 |
| 4.6.2 | <i>Local</i> | 40 |
| 4.7 | Agentes Móveis com Java | 41 |
| 5. | O Modelo <i>Aglet</i> | 43 |
| 5.1 | Elementos Básicos..... | 43 |
| 5.2 | O Modelo de Eventos dos <i>Aglets</i> | 45 |
| 5.3 | O Modelo de Comunicação dos <i>Aglets</i> | 45 |
| 5.4 | O Pacote <i>Aglet</i> | 46 |
| 5.4.1 | Classe com <i>ibm.aglet.Aglet</i> | 46 |
| 5.4.2 | Interface com <i>ibm.aglet.AgletProxy</i> | 47 |
| 5.4.3 | Interface com <i>ibm.aglet.AgletContext</i> | 47 |
| 5.4.4 | Classe com <i>ibm.aglet.Message</i> | 48 |
| 5.4.5 | Interface com <i>ibm.aglet.FutureReply</i> | 48 |
| 6. | Mediação de Agentes em Comércio Eletrônico | 50 |
| 6.1 | O Modelo CBB..... | 50 |
| 6.2 | Negociação entre Agentes..... | 51 |
| 6.3 | AuctionBot..... | 54 |
| 6.4 | V-Market..... | 55 |
| 7. | Sistema CUBe – Análise e Projeto | 56 |
| 7.1 | Análise do Sistema CUBe..... | 56 |
| 7.1.1 | <i>Clientes</i> | 57 |
| 7.1.2 | <i>Empresa</i> | 58 |
| 7.1.3 | <i>Fornecedores</i> | 59 |
| 7.2 | Projeto do Sistema CUBe | 59 |
| 7.3 | Arquitetura do Sistema CUBe..... | 63 |
| 8. | CUBe – Concepção do Sistema | 65 |
| 8.1 | Sistema de Comércio Eletrônico | 65 |

| | | |
|-------|---|-----|
| 8.1.1 | <i>O contexto ClientWebServer</i> | 71 |
| 8.2 | Sistema ERP..... | 73 |
| 8.2.1 | <i>Contexto ERP Manager</i> | 74 |
| 8.2.2 | <i>Contexto ERP Seller</i> | 79 |
| 8.2.3 | <i>Contexto ERP Manufacturer</i> | 80 |
| 8.2.4 | <i>Contexto ERP Emitter</i> | 82 |
| 8.3 | Sistemas dos Fornecedores | 83 |
| 9. | Trabalhos Futuros | 85 |
| 10. | Conclusões..... | 87 |
| 11. | Anexos..... | 89 |
| 12. | Glossário..... | 103 |
| 13. | Referências Bibliográficas..... | 106 |

Lista de Figuras

| | |
|--|----|
| Figura 2.1 - O Ciclo do Comércio Eletrônico | 20 |
| Figura 2.2 - Comércio Eletrônico e Processos Comerciais | 22 |
| Figura 2.3 - Obtenção de Informações de seus Clientes | 23 |
| Figura 3.1 - Funcionalidades de um Sistema ERP..... | 30 |
| Figura 4.1 - Redução do Tráfego da Rede com Agentes Móveis..... | 34 |
| Figura 4.2 - Execução Assíncrona e Autônoma dos Agentes Móveis..... | 36 |
| Figura 4.3 - Paradigmas da Computação em Rede..... | 37 |
| Figura 4.4 - Agente e Local..... | 39 |
| Figura 5.1 - Principais Operações de um <i>Aglet</i> | 45 |
| Figura 5.2 - Principais Classes e Interfaces Definidas na API Aglet..... | 46 |
| Figura 6.1 - <i>Site</i> de Leilões do eBay's..... | 52 |
| Figura 6.2 - <i>Site</i> de Leilões do iBazar | 52 |
| Figura 6.3 - Etapas do Modelo CBB em Sistemas que utilizam a Mediação de Agentes | 53 |
| Figura 6.4 - Janela de Criação de Leilões do AuctionBot..... | 54 |
| Figura 6.5 - Janela para Exibição de Agentes Criados no VMarket..... | 55 |
| Figura 7.1 - CUBE e as 3 Partes Integrantes de seu Sistema..... | 57 |
| Figura 7.2 - A Especificação do Produto e sua Composição no Sistema CUBE..... | 60 |
| Figura 7.3 - A Especificação do Pedido e da Ordem dentro do Sistema ERP | 61 |
| Figura 7.4 - Especificação dos Agentes do Sistema CUBE..... | 62 |
| Figura 7.5 - Etapas do Modelo CBB e a sua Automação através dos Agentes do CUBE | 62 |
| Figura 7.6 - Estratégia e Comportamento do Agente dentro do Sistema CUBE | 63 |
| Figura 7.7 - Arquitetura do Sistema CUBE | 64 |
| Figura 8.1 - Estrutura Hierárquica das Páginas ASP do Sistema de Comércio Eletrônico dentro do Sistema CUBE | 65 |
| Figura 8.2 - Página <i>default.asp</i> do Sistema CUBE..... | 66 |
| Figura 8.3 - Página <i>clientes.asp</i> do Sistema CUBE | 67 |
| Figura 8.4 - Página <i>login.asp</i> do Sistema CUBE | 67 |
| Figura 8.5 - Página <i>compras_a.asp</i> do Sistema CUBE | 68 |

| | |
|---|----|
| Figura 8.6 - Página <i>agente.asp</i> do Sistema CUBe | 69 |
| Figura 8.7 - Página <i>pedido.asp</i> do Sistema CUBe..... | 70 |
| Figura 8.8 - Passagem de Valores entre <i>Web Servers</i> | 71 |
| Figura 8.9 - Contexto <i>Client Web Server</i> do Sistema CUBe..... | 72 |
| Figura 8.10 - Página <i>index.html</i> Gerado pelo <i>WebServerAgent</i> | 73 |
| Figura 8.11 - Contexto <i>ERP Manager</i> | 74 |
| Figura 8.12 - <i>ERPManagerAgent</i> Processando Requisição do <i>RemoteAgent</i> | 75 |
| Figura 8.13 - <i>ERPManagerAgent</i> Processando Requisições do <i>SellerAgent</i> | 76 |
| Figura 8.14 - <i>ERPManagerAgent</i> Processando Requisições do <i>ManufacturerAgent</i> | 76 |
| Figura 8.15 - <i>ERPManagerAgent</i> Processando Requisições do <i>EmitterAgent</i> | 77 |
| Figura 8.16 - <i>LocatorAgent</i> | 77 |
| Figura 8.17 - <i>RepositoryAgent</i> | 79 |
| Figura 8.18 - <i>SellerAgent</i> Processando Requisição do <i>ERPManagerAgent</i> | 80 |
| Figura 8.19 - <i>ManufacturerAgent</i> Processando Requisições | 82 |
| Figura 8.20 - <i>EmitterAgent</i> Processando Requisições | 82 |
| Figura 8.21 - Contextos dos Fornecedores..... | 83 |
| Figura 8.22 - Fornecedores Processando Requisições dos Agentes Remotos do Sistema ERP..... | 84 |

Lista de Tabelas

| | |
|--|----|
| Tabela 2.1 - Compra de um Item: Comércio Tradicional X Comércio Eletrônico | 21 |
| Tabela 5.1 - Atributos e Métodos da Classe <i>Aglet</i> | 46 |
| Tabela 5.2 - Métodos da Interface <i>AgletProxy</i> | 47 |
| Tabela 5.3 - Métodos da Interface <i>AgletContext</i> | 48 |
| Tabela 5.4 - Atributos e Métodos da Classe <i>Message</i> | 48 |
| Tabela 5.5 - Métodos da Interface <i>FutureReply</i> | 49 |
| Tabela 8.1 - Parâmetros dos Agentes no Sistema CUBe | 69 |
| Tabela 8.2 - Estados do Estoque de Itens no Sistema CUBe | 81 |

Lista de Anexos

| | |
|---|-----|
| Anexo 1 - Diagrama Use Case: Principal..... | 90 |
| Anexo 2 - Diagrama Use Case: ERP - Compra..... | 91 |
| Anexo 3 - Diagrama Use Case: ERP - Produção..... | 92 |
| Anexo 4 - Diagrama Use Case: ERP - Emissão..... | 93 |
| Anexo 5 - Diagrama Use Case: ERP - Externo..... | 94 |
| Anexo 6 - Diagrama Use Case: ERP - Venda..... | 95 |
| Anexo 7 - Diagrama Use Case: Provedor - Banco..... | 96 |
| Anexo 8 - Diagrama Use Case: Provedor - Fornecedor..... | 97 |
| Anexo 9 - Diagrama Use Case: Provedor - Transportadora..... | 98 |
| Anexo 10 - Logical View - Arquitetura do Sistema..... | 99 |
| Anexo 11 - Diagrama de Classes - agents..... | 100 |
| Anexo 12 - Diagrama de Classes - cube..... | 101 |
| Anexo 13 - Diagrama de Classes - dialogs..... | 102 |

Lista de Abreviaturas e Siglas

| | |
|------|---------------------------------|
| API | Application Program Interface |
| ASP | Active Server Pages |
| CBB | Consumer-Buying Behavior |
| EDI | Electronic Data Interchange |
| EFT | Electronic Funds Transfer |
| ERP | Enterprise Resource Planning |
| FAQ | Frequently Asked Question |
| FIFO | First In First Out |
| HTML | HyperText Meta-Language |
| HTTP | HyperText Transfer Protocol |
| IBM | International Business Machine |
| IIS | Internet Information Server |
| IP | Internet Protocol |
| JDBC | Java Database Connectivity |
| JVM | Java Virtual Machine |
| KQML | Knowledge Query Meta-Language |
| MRP | Manufacturing Resource Planning |
| OC | Ordem de Compra |
| ODBC | Object Database Connectivity |
| OE | Ordem de Emissão |
| Oen | Ordem de Entrega |
| OP | Ordem de Produção |
| ORB | Object Request Broker |
| OV | Ordem de Venda |
| RMI | Remote Method Invocation |
| ROI | Return Of Investment |
| RPC | Remote Procedure Call |
| UML | Unified Modeling Language |
| WAN | Wide Area Network |

Resumo

O objetivo deste trabalho é mostrar uma abordagem em Sistemas Distribuídos para a integração de aplicações em Comércio Eletrônico e ERP (*Enterprise Resource Planing*) em um sistema chamado CUBe, através da mediação dos Agentes Móveis; como tema de dissertação do curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

Como solução ao *Business to Consumer (b2c)*, presente no Comércio Eletrônico, utilizou-se a tecnologia de páginas ASP integradas a um Agente Remoto. O *Business to Business (b2b)* oferecido pelo sistema ERP e pelos sistemas dos fornecedores, foi desenvolvido em cima do paradigma dos Agentes Móveis. E para a automação das diversas tarefas envolvidas no comportamento de compra e venda deste sistema, utilizou-se o modelo do CBB (*Consumer Buying Behaviour*).

Abstract

The purpose of this work is to show an approach in Distributed Computing for the integration of applications in Electronic Commerce and ERP (Enterprise Resource Planing) to a system called CUBe, through mediation of the Mobile Agents; as subject of dissertation of the Graduate program in Computer Sciences from the “Universidade Federal de Santa Catarina”.

As a solution to Business to Consumer (b2c), present in the Electronic Commerce, the technology of ASP pages integrated to a Remote Agent, was used. The Business to Business (b2b), offered by the ERP System and by the suppliers’ systems, was developed on the paradigm of the Mobile Agents. And for the automation on the various tasks involved in the buying and selling behavior of this system, the CBB (Consumer Buying Behavior) was used.

1. Introdução

Este trabalho versa sobre uma aplicação de Comércio Eletrônico no contexto de *e-Commerce*, utilizando-se o paradigma dos Agentes Móveis e considerando a integração do ERP (*Enterprise Resource Planning*), no sentido de permitir maior interoperabilidade de recursos da empresa.

Com a globalização da economia, as empresas viram seus mercados cativos, restritos a regiões conhecidas espalharem-se de forma considerável por todo o globo, abrangendo regiões antes jamais imaginadas. Além disso, o cliente que antes estava amarrado a uma determinada empresa, ou por motivos geográficos ou por falta de melhores opções, agora ficou livre para escolher produtos de companhias localizadas em qualquer lugar do globo (DRUCKER, 2000). Nesta onda de globalização duas tecnologias cresceram substancialmente: os sistemas de informação e o comércio eletrônico.

Nos últimos três anos, os Sistemas de Planejamento de Recursos Empresariais, também conhecidos como ERP (*Enterprise Resource Planning*), consolidaram-se como uma tecnologia fundamental e determinante para a sobrevivência e a competitividade das empresas em nível de sistema de informação (LUZ NETO, 2000).

O comércio eletrônico também foi impulsionado pela difusão da *Internet* nos últimos anos. Por sua vez, com a digitalização das informações comerciais, as companhias conseguiram montar verdadeiras empresas virtuais tanto para transações com seus clientes (*business-to-customer* ou *b2c*) quanto para transações entre empresas (*business-to-business* ou *b2b*).

O grande problema das companhias atualmente é encontrar sistemas que integrem estas três grandes aplicações – *e-Commerce*, *e-Business* e *ERP* – em uma única solução. Muitas soluções adotadas, ou não são compatíveis entre si, ou utilizam tecnologias completamente heterogêneas, dificultando a troca de informações e o seu futuro crescimento ou atualização. Para satisfazer a estes requisitos de interoperabilidade e reusabilidade, a linguagem aliada à tecnologia emergente dos Agentes Móveis, foi uma solução encontrada para a implementação neste trabalho de um sistema integrador no domínio das aplicações acima mencionadas.

1.1 Trabalhos Estudados

Da literatura pesquisada, dois trabalhos sobre Comércio Eletrônico com Agentes Móveis foram tomados como base para esta pesquisa: AuctionBot (WURMAN, 1998) e VMarket (RIPPER, 2000).

AuctionBot é um sistema de Comércio Eletrônico implementado com agentes, e que implementa um leilão genérico da Universidade de Michigan. Vmarket é um *framework* para padronização de mercados virtuais e definição de categorias de transações sob demanda, também implementado com agentes, da PUC do Rio de Janeiro.

Tais trabalhos serviram para nortear o projeto e a implementação de um sistema, chamado CUBe, que buscou solucionar problemas acima discutidos, adotar algumas alternativas implementadas nos sistemas acima citados e outras soluções inovadoras, dentre elas:

- suporte a múltiplos produtos, ampliando a solução adotada pelos sistemas acima, visto que estes trabalham com uma linha definida de produtos, em geral livros e CDs;
- automação de várias etapas do modelo CBB (*Consumer Buying Behaviour*), característica presente nos sistemas acima citados, porém como são sistemas baseados em leilões possuem um trabalho centrado na negociação; no sistema CUBe, buscou-se além da etapa de negociação, a automação de mais etapas (1, 4, 5 e 6) do modelo CBB, discutidas em detalhes no capítulo 6;
- múltiplos processos de estratégia e comportamento, utilizado pelos sistemas estudados para facilitar o processo de escolha do usuário de estratégias e comportamentos dos agentes no ato da negociação e compra, previamente parametrizados, e que foi utilizado também neste trabalho.

1.2 Trabalhos Publicados

Durante a pesquisa e implementação do sistema, foi possível a publicação de um artigo, referente a este projeto, no XX Congresso do SBC/Semish em Julho de 2000, realizado na cidade de Curitiba, Paraná (VALDO, 2000).

1.3 Estrutura do Trabalho

Os capítulos deste trabalho estão divididos da seguinte forma: na seção 2 é mostrado o conceito de Comércio Eletrônico, implicações e características; na seção 3 é apresentada a definição de Sistemas ERP; na seção 4 é apresentado o conceito de Agentes Móveis, foco principal deste trabalho; na seção 5 é apresentada a ferramenta escolhida para a implementação do sistema CUBe; na seção 6 é explicado o modelo CBB de simulação de comportamentos de compra em software e também trabalhos relacionados a este projeto; na seção 7 é mostrado a análise, o projeto e a arquitetura do sistema CUBe proposto; no capítulo 8 é mostrada a concepção do sistema e soluções implementadas; no capítulo 9 são apresentadas as propostas para futuros trabalhos; e, no capítulo 10, as conclusões.

2. Comércio Eletrônico

Há algum tempo, grandes companhias comerciais têm usado o comércio eletrônico para conduzir suas transações empresariais. O intercâmbio de dados eletrônicos (*Electronic Data Interchange* ou *EDI*) em redes privadas começou em 1960, e os bancos estão usando redes dedicadas para transferência eletrônica de fundos (*Electronic Funds Transfer* ou *EFT*) há algum tempo. Recentemente, entretanto, com o aumento da qualidade e popularidade da *Internet*, o comércio eletrônico tem conquistado consumidores individuais assim como empresas de todos os portes.

A *Internet* já está mudando o modo pelo qual muitas companhias conduzem seus negócios. Com o aumento desta influência, e mais companhias utilizando a *Internet*, as possibilidades de se conduzir o comércio empresarial na *Internet* irá expandir-se bastante, e se tornar mais parte da rotina comercial do que é hoje.

2.1 Visão Geral

Para muitos, comércio eletrônico é definido como a compra e venda de produtos e serviços na *Internet*, mas há muitos mais aspectos. O comércio eletrônico tem incluído a negociação de transações de compra e transferência de fundos sobre redes de computadores. Está em crescimento agora a inclusão da compra e venda de novos artigos (*commodities*) tais como informação eletrônica. E as oportunidades das empresas tirarem vantagem das capacidades do comércio eletrônico são enormes, ao contrário da visão comercial adotada atualmente, ou seja, realizar estas mesmas transações sobre redes eletrônicas.

A *Internet* deu um impulso ao comércio eletrônico empresarial – em alguns casos, pequenas companhias estão descobrindo que podem conduzir comércio *on-line*, como suas irmãs maiores. E empresas de todos os portes estão descobrindo que eles podem tirar vantagens da *Internet* para baratear o custo do comércio eletrônico – substituindo outras redes, ou usando a *Internet* como um meio de comunicação alternativo, convertendo seus dados comerciais para a forma digital, e incorporando a isso suas práticas comerciais.

O movimento para a digitalização da informação não é novo – está sendo realizado há mais de uma década, e continua a crescer com os computadores pessoais tornando-se um padrão para um número maior de corporações. O que está fazendo a diferença para o comércio em geral, é que uma significativa força tem impulsionado as empresas a usarem a informação digital, processos comerciais computadorizados e a *Internet*; esta força é chamada de comércio eletrônico.

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 por 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.

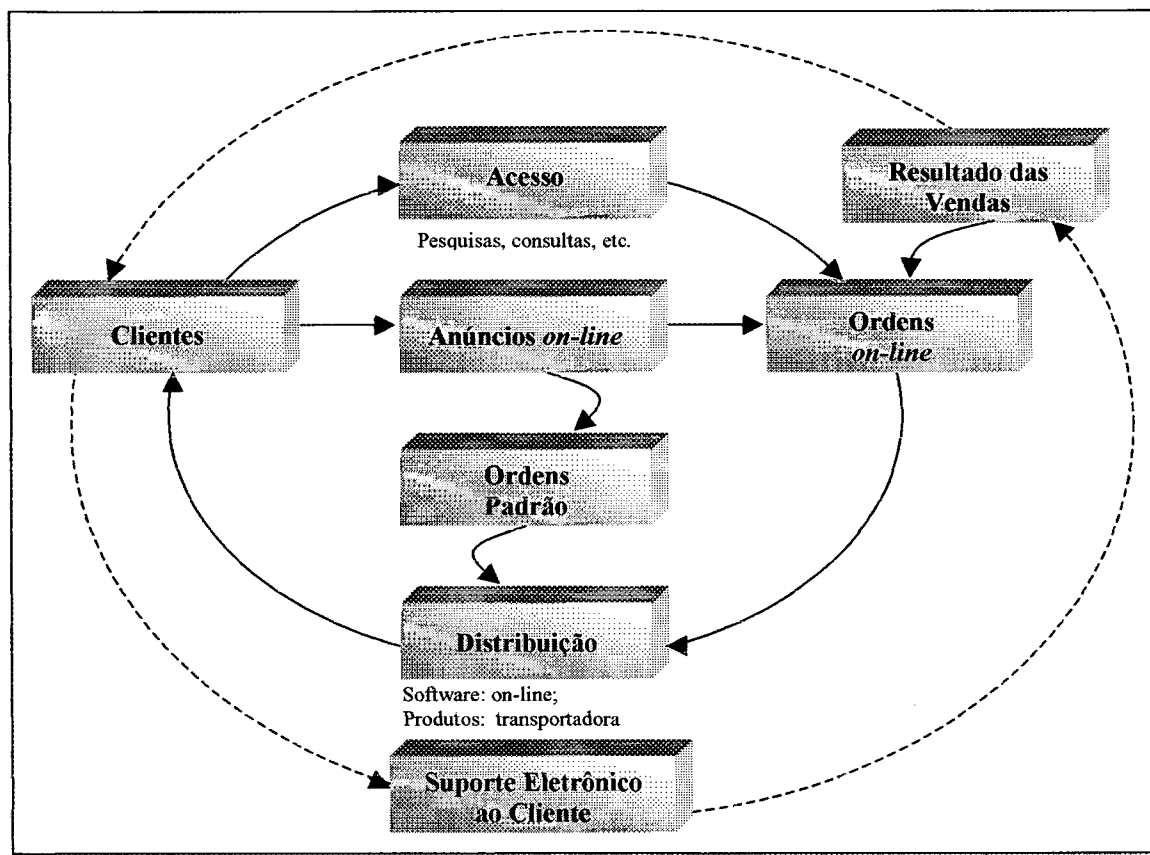
O comércio eletrônico é um sistema que inclui não somente aquelas transações tradicionais de compra e venda de bens e serviços para diretamente gerar renda, mas também aquelas transações de suporte que geram renda indiretamente, tais como: criar demanda para bens e serviços, oferecer venda de suporte e serviços a clientes, ou facilitar as comunicações entre parceiros comerciais (fig. 2.1).

O comércio eletrônico construído sobre a estrutura e as vantagens do comércio tradicional adiciona diversas flexibilidades oferecidas pela rede eletrônica; facilitando, com isso, as interações de departamentos, aumentando as relações com os clientes e eliminando os limites de tempo e local. Por exemplo, sistemas computacionais na *Internet* podem ser configurados para fornecer suporte a clientes 24 horas por dia, 7 dias por semana. Ordens de compra de seus produtos e serviços podem também ser aceitos a qualquer hora e provenientes de qualquer lugar.

Além disso tudo, o comércio eletrônico permite novas formas de comércio, assim como novas formas de se fazer comércio. Uma destas formas de comércio diferenciado 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.

Figura 2.1 - O Ciclo do Comércio Eletrônico



2.2 Transações: Modelo Tradicional X Comércio Eletrônico

Considere as tarefas que uma companhia tem que realizar quando seu empregado quer adquirir um item. Primeiro, deve gerar 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.

A tabela 2.1 compara as duas formas de se realizar uma transação comercial.

Tabela 2.1 - Compra de um Item: Comércio Tradicional X Comércio Eletrônico

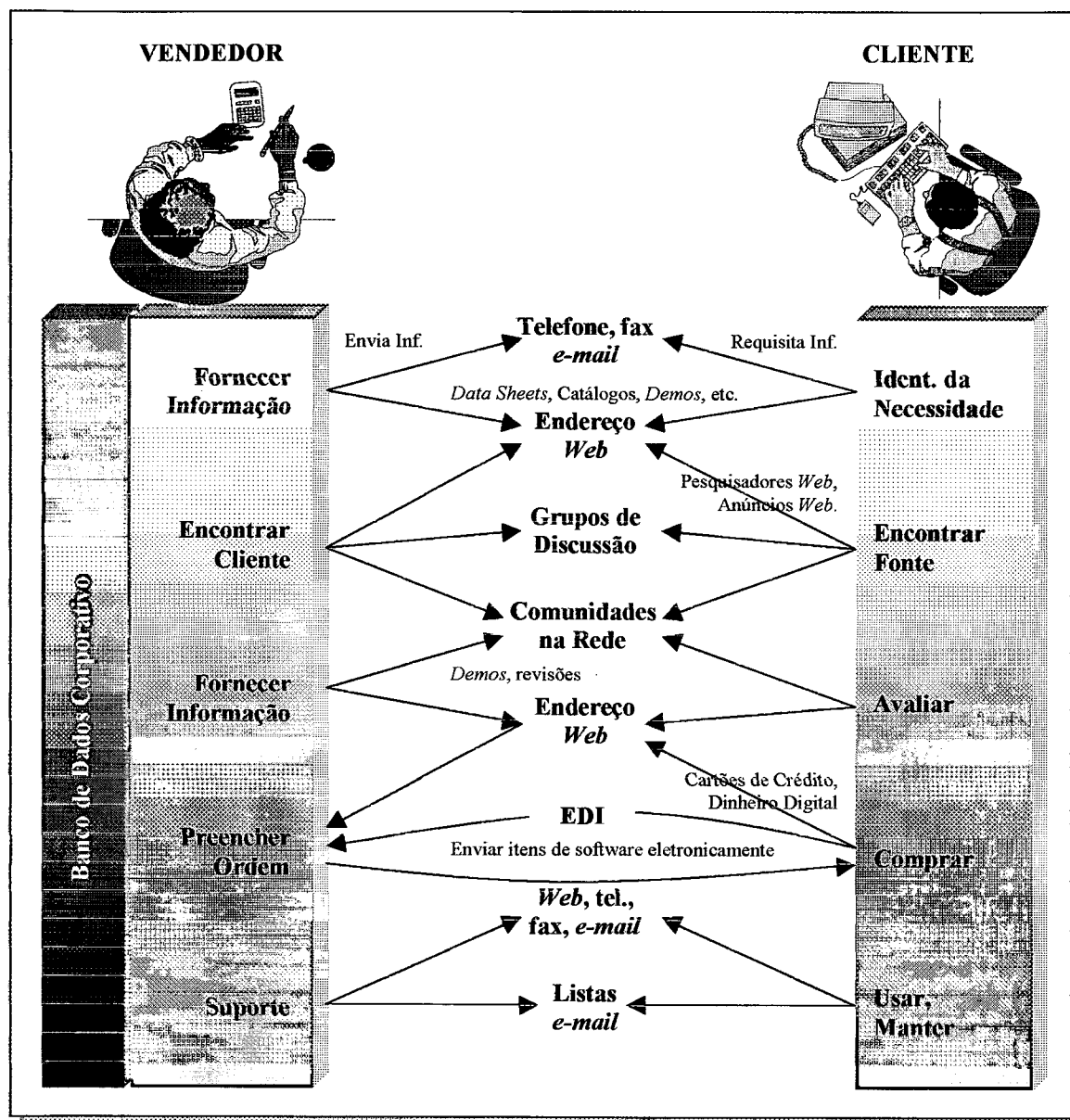
| Etapas do Ciclo de Venda | Comércio Tradicional (múltiplos meios envolvidos) | Comércio Eletrônico (único meio envolvido) |
|--|--|---|
| Adquirir informações sobre o produto | revistas, <i>folders</i> , catálogos | páginas da <i>Web</i> |
| Requisitar um item | formulários impressos, cartas | <i>e-mail</i> |
| Pegar aprovação da ordem de compra | | |
| Analisar catálogos, preços | catálogos | catálogos <i>on-line</i> |
| Analisar disponibilidade de itens e conformidade de preço | telefone, fax | |
| Enviar ordem (comprador) Receber ordem (fornecedor) | fax, correio | <i>e-mail</i> , <i>EDI</i> |
| Priorizar ordem | * | banco de dados <i>on-line</i> |
| Analisar inventário no estoque | formulário impresso, telefone, fax | banco de dados <i>on-line</i> , páginas da <i>Web</i> |
| Agendar entrega | formulário impresso | banco de dados <i>on-line</i> |
| Emitir fatura | formulário impresso | banco de dados <i>on-line</i> |
| Receber produto | meio de entrega qualquer | |
| Confirmar recebimento | formulário impresso | <i>e-mail</i> |
| Enviar fatura (fornecedor) Receber fatura (comprador) | correio | <i>e-mail</i> <i>EDI</i> |
| Agendar pagamento | formulário impresso | <i>EDI</i> banco de dados <i>on-line</i> |
| Enviar pagamento (comprador) Receber pagamento (fornecedor) | correio | <i>EDI</i> <i>EFT</i> |

2.3 Mais do que a soma de suas partes

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. Estes dados podem ser armazenados em um banco de dados e apresentados eletronicamente através de páginas na *Web*.

Vamos analisar agora os 5 processos: distribuição da informação, ordem, pagamento, execução, serviço e suporte. Todas estas etapas fazem parte do novo ciclo de vida do comércio eletrônico (fig. 2.2).

Figura 2.2 - Comércio Eletrônico e Processos Comerciais

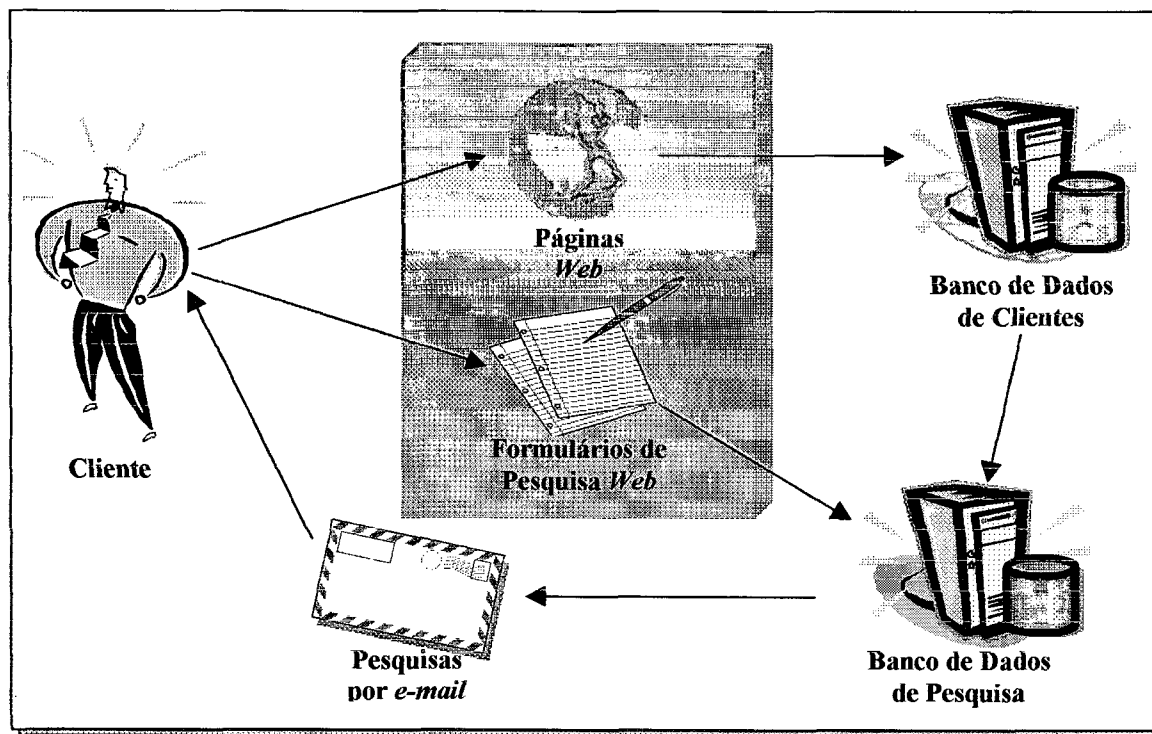


2.3.1 Distribuição da informação

Antes de se fazer uma venda, é necessário que os clientes alcancem seus produtos e serviços; isto é, fornecer dados para o processo de pesquisa de seus clientes.

A *Web* fornece um meio efetivo de comunicação com os clientes; pode-se projetar endereços na *Web* para incluir catálogos de produtos que podem ser encontrados eletronicamente e que fornecem novos tipos de informação sobre produtos. Manter um catálogo *on-line* usando a *Web*, pode gerar dados sobre requisições de produtos e quando estas requisições estão sendo realizadas. Pode-se também incluir um formulário para aceitar questões de clientes e enviar os relatórios diretamente ao grupo de suporte. Pode-se também compilar questões que crescem repetidamente – conhecidas como *FAQs* (*Frequently Asked Questions* ou Perguntas Frequentemente Realizadas) e distribuir estas informações através do *e-mail* e pela *Web* (fig. 2.3).

Figura 2.3 - Obtenção de Informações de seus Clientes



2.3.2 Ordem

Realizar uma ordem eletrônica de itens e serviços não deve ser problemático para os clientes. Formulários eletrônicos que espelham os tradicionais em papel são um bom modo de se manipular as requisições. Várias aplicações cliente-servidor existentes no mercado foram projetadas para este fim, mas devido a muitos sistemas na *Web* suportarem formulários eletrônicos, muitas companhias estão começando a migrar seus sistemas para a *Web*.

2.3.3 Pagamento

Este é o coração do processo de vendas atual – receber o dinheiro pelos itens ou serviços. Com uma grande variedade de mecanismos de pagamento, os consumidores podem escolher entre: cartões de crédito, cheques eletrônicos, dinheiro digital, etc.

As empresas têm respondido pela popularidade da *Web*, colocando suas folhas de dados de produtos e catálogos para compra em servidores *Web*. Também estão começando a usar *EDI* 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.

2.3.4 Execução

Independentemente de estarmos na Era Eletrônica ou na Era da Informação, nossa economia depende da transferência diária de quantidades maciças de informação. Muitas empresas ganham dinheiro gerando, transferindo ou analisando estas informações. A *Internet* pode ser usada para transferência de informação de produtos para clientes. Além dos formulários padronizados, tais como: *newsletters*, periódicos, relatórios de análise, e preços de produtos, pode-se incluir nesta lista de dados eletrônicos o *software*. Documentações, aperfeiçoamentos de programas e atualizações estão também bem integrados aos processos de distribuição sobre a *Internet*.

Mesmo para produtos físicos, 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* das entregas.

Uma vez fabricado o produto, é necessário distribuí-lo. 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; o *e-mail* e um *Web site* podem disponibilizar informações sobre o lançamento deste produto de forma clara, objetiva e com fácil acesso.

2.3.5 Suporte e serviços

O relacionamento de uma companhia com um cliente raramente termina com a venda. Na verdade, a venda pode ser somente o começo de um longo e frutífero

relacionamento com um cliente. 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 ela virá a oferecer no futuro.

Itens tais como notas técnicas sobre características e usos de seus produtos, *FAQs* que fornecem a seus clientes perguntas freqüentemente realizadas sobre algum produto, atualizações de *software*, e correções de problemas são apenas algumas das informações que podem estar disponíveis aos clientes na *Internet*. Sistemas inteligentes podem disponibilizar estas informações através de vários canais como: fax, *e-mail*, e a *Web*, tudo ao mesmo tempo.

Fornecer um formulário de perguntas em um *Web site*, ou simplesmente aceitando perguntas por *e-mail* (e não somente para o seu pessoal do suporte técnico), pode tornar-se um modo de assegurar que você está dando a informação certa para a pessoa certa.

2.4 Benefícios do Comércio Eletrônico

O comércio eletrônico pode oferecer à empresa benefícios a curto e a longo prazo. Não somente abrir novos mercados, permitindo alcançar novos clientes, mas pode também facilitar e tornar mais rápido o comércio para a base de clientes existentes. Mudando as práticas comerciais, como: ordem, fatura, e suporte a clientes, para um sistema baseado em rede pode também reduzir o volume de papel envolvido em uma transação empresarial.

Quanto maior o grau de informação digital disponível em uma transação comercial, melhor para a identificação das necessidades do cliente. A informação digital também possibilita a apresentação de soluções padronizadas; o que vai facilitar a obtenção da satisfação dos clientes e buscar seu futuro retorno, tornando o comércio eletrônico, acima de tudo, lucrativo.

2.5 Diferenças entre *e-business* e *e-commerce*

Os conceitos: *e-business* e *e-commerce* têm estado em evidência na área de comércio eletrônico. Muita discussão foi gerada em torno da definição destes dois temas; grandes companhias como a IBM têm investido milhões de dólares para que suas idéias se consolidem como verdadeiras. As definições aqui adotadas vão de encontro ao

exposto por Jason Smith (SMITH, 1999), 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.

Estes dois termos têm sido freqüentemente usados de forma idêntica, mas há uma sutil diferença: *e-commerce* é somente uma parte do *e-business*. O trabalho aqui proposto visa a integração destas duas áreas, já que afeta tanto a transação como a troca de informações de um modo geral.

2.5.1 *Business-to-business*

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 (*supply-chain*). A *Internet* permite transações comerciais valiosas, pode abrir canais com fornecedores, para coleta de relatórios; este tipo de *b2b* pode ter um ROI (*Return of Investment* ou Retorno de Investimento) quase que imediato.

Com a digitalização de relatórios e processos comerciais pode-se economizar milhões de dólares como resultado direto do aumento de produtividade, redução do custo de papel e eficiência nas aprovações. O sucesso destas iniciativas é que faz uma companhia incorporar o que é chamado de negócio eletrônico.

A *Web* não é somente um canal de vendas. Está aberta a desafios para que novos negócios sejam realizados em termos de estruturas existentes e relacionamentos com clientes, fornecedores e vendedores.

2.5.2 *Business-to-consumer*

O termo *business-to-consumer (b2c)* está relacionado às transações entre empresas e clientes, um nicho de mercado que está em plena ascensão.

Pessoas que compram na *Web* não estão comprando por causa dos melhores preços; estão procurando por produtos que não encontraram em nenhum lugar ou por eficiência em seus processos de pedidos (como um produto feito sob medida para suas necessidades).

Preço é sempre um tópico considerado, mas raramente é o motivador para uma compra *on-line*. A *Internet* fornece um modo mais barato e potencialmente mais satisfatório para certos tipos de transações comerciais. Isto aplica-se diretamente para compras repetitivas. Mas deve-se assegurar que a ordem é de fácil uso e incorpora várias características possíveis *off-line*.

Além do processo de pagamento, a aquisição de dados (*data mining*) é possível. Mas o *b2c* na *Internet* está longe de causar impactos significativos nas empresas. Deve-se ponderar apenas que é um processo irreversível para todos, em franca expansão e que mais cedo ou mais tarde revelará seu lado lucrativo, tanto para a empresa quanto para o consumidor.

3. Sistemas ERP

As pressões competitivas, presentes desde o final da década de 80, têm obrigado as empresas a recuperar sua competitividade através da redução de custos e diferenciação de seus produtos ou serviços. Uma série de ferramentas e filosofias gerenciais têm sido aplicadas nesta recuperação. Muitas dessas ferramentas e filosofias reconhecem a necessidade de se 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* ou Planejamento de Recursos Empresariais) surgiram para explorar essa necessidade de rápido desenvolvimento de sistemas integrados, ao mesmo tempo em que as empresas são pressionadas para terceirizar todas as atividades que não pertençam ao seu foco principal de negócios. Um ERP é um sistema informatizado integrado, desenvolvido por empresas especializadas, que abrange a maioria ou a totalidade dos processos empresariais. Estes sistemas eram inicialmente conhecidos como sistemas integrados de gestão empresarial, ou simplesmente “pacotes integrados”.

Os sistemas ERP podem ser definidos como sistemas de informação integrados na forma de pacotes de *software* comercial com a finalidade de dar suporte à maioria das operações de uma empresa. São geralmente divididos em módulos que se comunicam e atualizam uma mesma base de dados central. As informações alimentadas em um módulo são instantaneamente disponibilizadas para os demais módulos que delas dependem. Os sistemas ERP permitem, ainda, 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 (recursos financeiros, recursos humanos indiretos, vendas, distribuição, etc). 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 mercados 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 dos Sistemas ERP

Os sistemas ERP abrangem uma grande gama de funcionalidades e processos empresariais. Logicamente, de acordo com o fornecedor do *software* ERP, existe variação em amplitude (número de atividades e processos contemplados pelo sistema) e em profundidade (grau de especificidade e flexibilidade com que trata um determinado processo). 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).

Davenport (DAVENPORT, 1998), 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 (fig. 3.1).

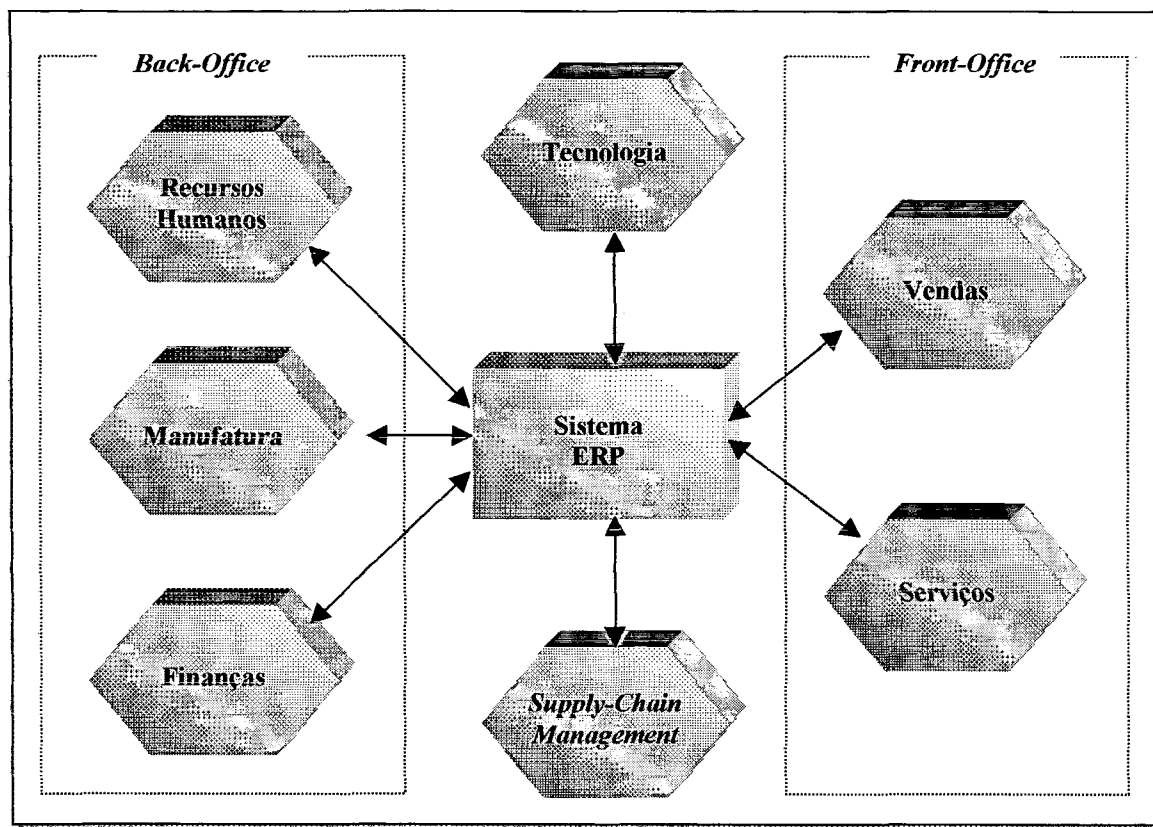
3.2 Termos relacionados aos Sistemas ERP

Segundo (SOUZA, 1999a), existem alguns termos relacionados aos sistemas ERP que, embora não os definam, são importantes para a compreensão dos aspectos envolvidos em sua utilização. Esses termos serão descritos a seguir resumidamente.

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.

Figura 3.1 - Funcionalidades de um Sistema ERP



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, taxas, leis, procedimentos).

4. Agentes

Muitas definições têm sido abordadas sobre agentes, agentes móveis, agentes inteligentes e agentes autônomos; cada qual abordando aspectos importantes e relevantes desta nova tecnologia. Este tema é tão vasto e complexo, que mesmo alguns estudiosos da área usam várias definições para este assunto. Vamos listar algumas destas definições de importantes pesquisadores da área de agentes.

“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).

“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).

“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, 1996).

“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).

De acordo com os estudos realizados, não existe uma definição única e correta; todas focam peculiaridades importantes e inerentes ao conceito de agentes. Alguns dos aspectos dos agentes citados nestas definições devem-se à área de estudo em que se concentram seus pesquisadores, por isso algumas destas definições focam características mais importantes para uma determinada área do que para outra; por exemplo: inteligência e autonomia são conceitos importantes para a área de Inteligência Artificial, enquanto que o conceito de mobilidade está mais ligado a Sistemas Distribuídos.

Os conceitos de agentes discutidos neste trabalho vão de encontro aos estudos e definições de (LANGE, 1998).

4.1 Agentes de Software

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. Para o sistema, um agente é um objeto de *software* que está situado em um ambiente de execução e possui obrigatoriamente as seguintes propriedades:

- *Reativo*: sente alterações do ambiente em que está e toma determinadas ações de acordo com estas mudanças.
- *Autônomo*: tem o controle sobre suas próprias ações.
- *Orientado por objetivos*: age de forma pró-ativa para conseguir chegar ao objetivo que lhe foi previamente delegado.
- *Execução contínua*: está sempre em execução a não ser que tenha chegado ao seu objetivo ou tenha recebido uma ordem em contrário.

E pode ainda possuir algumas propriedades complementares listadas abaixo:

- *Mobilidade*: um agente pode trafegar em uma rede, mudando de estação de acordo com um itinerário previamente designado.
- *Comunicativo*: um agente pode possuir a capacidade de se comunicar com outros agentes.
- *Aprendizagem*: um agente pode adaptar-se de acordo com experiências passadas.
- *Confiabilidade*: um agente pode ser construído de forma a ser confiável para o usuário final.

Como definido acima, a mobilidade é uma propriedade complementar dos agentes, ou seja, um agente pode ficar localizado em uma estação e comunicar-se com outros agentes ou mesmo com o sistema de diferentes formas. Os agentes podem ser divididos em:

- *Agentes Estacionários*: são executados somente no sistema onde foram iniciados. Caso necessitem comunicar-se com outros agentes, ou necessitem de informações que não estão naquele sistema, utilizam um mecanismo de mensagens como o RPC (*Remote Procedure Calling*).

- *Agentes Móveis*: não estão limitados ao sistema onde foram executados inicialmente. Possuem a habilidade de transportarem-se de um sistema em uma rede para outro. Esta capacidade de transporte permite a um agente móvel viajar para um determinado sistema que contenha um objeto com o qual deseja interagir, privilegiando-se por estar na mesma estação onde o objeto está localizado.

4.2 Características importantes dos agentes móveis

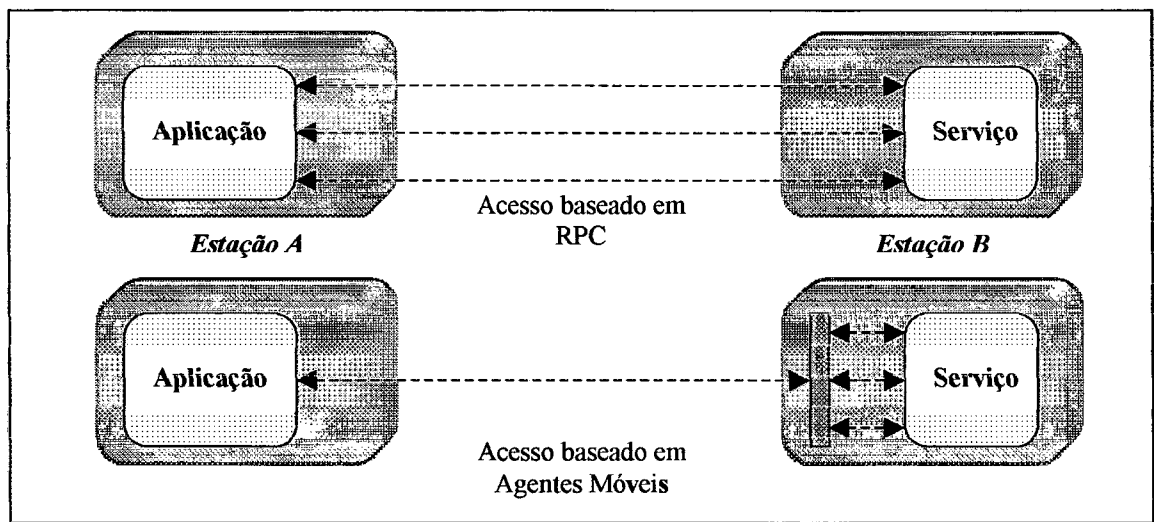
a) Redução do tráfego da rede

Sistemas Distribuídos frequentemente trabalham com protocolos que envolvem múltiplas interações para cumprir uma determinada tarefa. Com os agentes, isto não ocorre, pois estes são despachados para a estação de destino e executam lá (localmente) suas tarefas (fig. 4.1). Com isso movemos o processamento para os dados e não os dados para o processamento.

b) 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 então ser despachados de um computador central e executar localmente um controle sobre este sistema, ocasionando um tempo de atraso praticamente nulo.

Figura 4.1 - Redução do Tráfego da Rede com Agentes Móveis



c) 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.

d) 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 (fig. 4.2).

e) 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.

f) Naturalmente heterogêneos

Uma rede de computadores é fundamentalmente heterogênea (tanto de *hardware* quanto de *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.

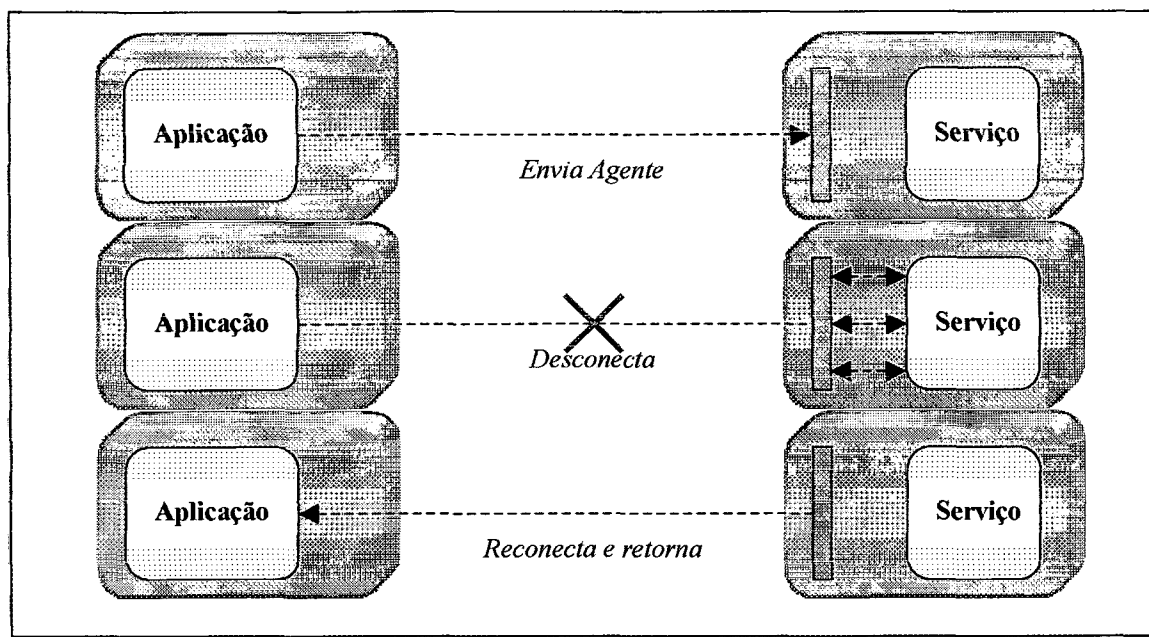
g) Robustez e Tolerância a Falhas

A habilidade dos agentes móveis em reagir dinamicamente a situações e eventos desfavoráveis o torna favorável à construção de sistemas distribuídos robustos e tolerantes a faltas.

4.3 Paradigmas da Computação em Rede

a) Paradigma Cliente-Servidor

Figura 4.2 - Execução Assíncrona e Autônoma dos Agentes Móveis



Neste paradigma, um servidor avisa a um conjunto de serviços que necessita de acesso a alguns recursos (tais como banco de dados). O código que implementa estes serviços estão armazenados localmente no servidor. Dizemos que o servidor armazena o *know-how*. Finalmente, o próprio servidor executa o serviço e tem então a capacidade de processamento (fig. 4.3(a)). Caso o cliente necessite usar um ou mais dos serviços fornecidos pelo servidor, deve utilizar os serviços fornecidos por ele. O servidor possui tudo: o *know-how*, recursos, e processamento. Uma grande quantidade de tecnologias utilizam este paradigma, como: *Remote Procedure Call (RPC)*, *Object Request Broker (ORB)*, e *Remote Method Invocation (RMI)*.

b) Paradigma Código sob Demanda

De acordo com este paradigma, você obtém o *know-how* apenas quando necessita. O cliente neste caso está habilitado a executar suas tarefas de acordo com um conjunto de código (*know-how*). Uma estação da rede fornece este código necessário. Uma vez recebido o código pelo cliente, a computação é descarregada no cliente. O cliente armazena então, a capacidade de processamento assim como os recursos locais (fig. 4.3(b)). Ao contrário do clássico paradigma cliente-servidor, o cliente não necessita de código pré-instalado porque todo o código necessário será carregado do servidor. Dizemos que o cliente tem os recursos e o processamento e o servidor possui o *know-how*. Os exemplos mais comuns desta arquitetura são os *Applets Java* (códigos

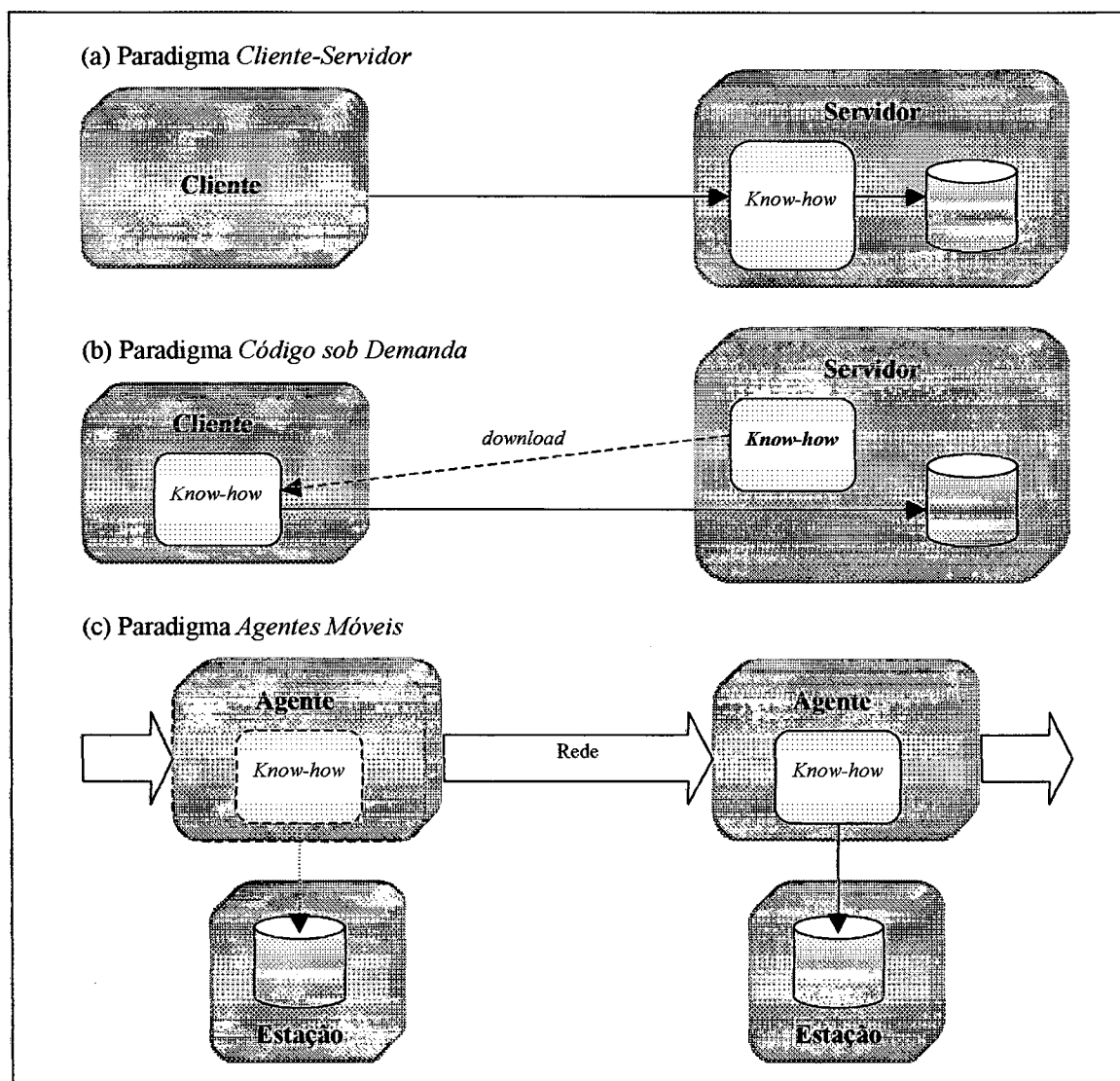
carregados em um *Web Browser* para execução local) e *Servlets Java* (códigos enviados para um *Web Server* para execução remota).

c) Paradigma Agentes Móveis

Uma característica chave do paradigma dos agentes móveis é que qualquer estação da rede possui um alto grau de flexibilidade, pois possui uma mistura de *know-how*, recursos e processamento. Esta capacidade de processamento pode ser combinada com recursos locais. O *know-how* associado aos agentes móveis não está atribuída a uma estação específica, mas disponível através da rede (fig. 4.3(c)).

Comparando-se os três paradigmas, vemos que a escala cronológica evolui para uma maior flexibilidade, ou seja, o cliente e o servidor fundindo-se em uma única estação.

Figura 4.3 - Paradigmas da Computação em Rede



4.4 Aplicações de Agentes Móveis

a) Comércio Eletrônico

Os agentes móveis são perfeitamente adaptáveis ao comércio eletrônico. Uma transação comercial necessita de acesso em tempo-real a recursos remotos como quotas em estoque e até mesmo situações mais complexas como uma negociação entre agentes. Diferentes agentes teriam diferentes metas e poderiam implementar e exercer diferentes estratégias para alcançar estes objetivos. Com isso poderíamos ter os agentes buscando os objetivos de seus criadores, e negociando em seu lugar.

b) Serviços de Redes de Telecomunicações

Suporte e gerenciamento de serviços de telecomunicações avançada são caracterizados por reconfiguração de redes dinamicamente e padronização de usuários. O tamanho físico destas redes e as necessidades restritas sobre os quais eles operam faz com que a tecnologia de agentes móveis formem a base para que tais sistemas ainda sejam efetivos.

c) Aplicações de *Workflow* e *Groupware*

Está presente nas aplicações de *workflow* o suporte a fluxos de informações entre os colaboradores. Os agentes móveis são particularmente úteis aqui, graças à sua mobilidade ele fornece um grau de autonomia para o *workflow*. Além disso, os agentes móveis (representando *workflows* individuais) podem possuir a informação e característica necessárias para se movimentarem dentro da organização, independente de alguma aplicação em particular.

4.5 Estado-da-Arte em Sistemas de Agentes Móveis

4.5.1 Aglets

O *Aglet* da IBM nada mais é que um agente Java móvel que suporta o conceito de autonomia de execução e roteamento dinâmico de itinerário. Pode-se pensar no *aglet* como uma generalização e extensão dos *applets* e *servlets* Java. *Aglets* são armazenados por um servidor *Aglet* da mesma forma que os *applets* são armazenados em um *Web browser*. O servidor *Aglet* fornece um ambiente para os *aglets* executarem, e 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*).

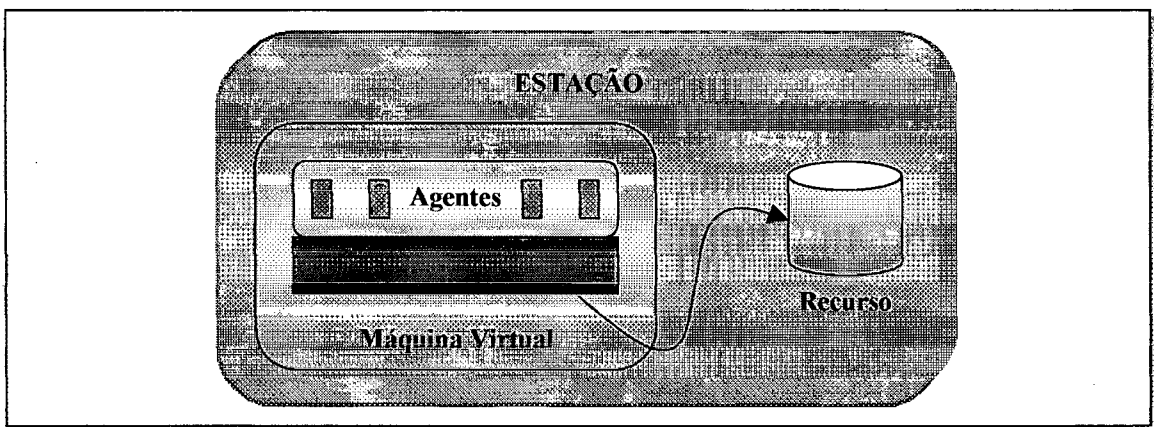
4.5.2 Voyager

Voyager da ObjectSpace é uma plataforma para computação distribuída acrescida de agentes em Java. *Voyager* fornece um extenso conjunto de objetos com capacidade de mensagem, também permite que os objetos movam-se como agentes em uma rede. *Voyager* combina as propriedades de um *ORB* baseado em Java com o sistema de agentes móveis. Desta forma permite que os programadores em Java criem aplicações em rede usando ambas as técnicas: a tradicional e a programação distribuída acrescida de agentes.

4.6 Elementos de um Sistema de Agentes Móveis

Os dois conceitos fundamentais no modelo de agentes móveis são: os agentes e seus ambientes de execução ou locais (fig. 4.4).

Figura 4.4 - Agente e Local



4.6.1 Agente

Um agente móvel é uma entidade que tem 5 atributos: estado, implementação, interface, identificador e mandante. Quando o agente move-se pela rede, ele leva estes atributos consigo. Vamos agora descrever em detalhes cada uma destes atributos:

- *Estado*: necessário para o agente retomar a execução no computador destino após a sua viagem. Em muitas linguagens pode-se dividir o estado de um agente

em: estado de execução, que é o seu estado em tempo de execução e, estado do objeto, que é o valor das variáveis de instância no objeto.

- *Implementação*: necessário para execução do agente independente de sua localização. Quando um agente trafega pela rede duas situações podem ser implementadas: levar todo o código necessário consigo ou, viajar até a estação destino e analisar o código existente, recuperando somente o código necessário.
- *Interface*: necessária para a comunicação do agente. Esta interface pode ser algo como um conjunto de assinaturas de métodos que permitem a outros agentes e aplicações o acesso a seus métodos, para a criação de uma interface de mensagens e comunicação entre os agentes.
- *Identificador*: necessário para reconhecer e localizar agentes em trânsito. Cada agente possui um identificador que é único durante toda sua vida, facilitando sua identificação e localização através de serviços de diretórios, por exemplo.
- *Mandante*: necessário para determinar responsabilidades morais e legais. Um mandante é uma entidade cuja identidade pode ser autenticada por algum sistema cujo agente esteja tentando acessar. Uma identidade neste caso poderia ser possivelmente um nome seguido de outros atributos necessários (código, senha, etc).

4.6.2 Local

Um local é o ambiente no qual os agentes operam, ou seja, é o contexto no qual um agente pode executar. Este local é visto como um ponto de entrada para que um agente visitante possa executar seu código. Além disso, o local pode ser caracterizado como o sistema operacional dos agentes. Vamos detalhar quatro conceitos que exercem uma importante regra nestes ambientes.

- *Máquina*: são as máquinas virtuais de um ou mais locais. Um local não pode executar um agente sozinho. As máquinas servem como ambientes virtuais para locais e seus agentes; um exemplo destas máquinas é a JVM (*Java Virtual Machine* – Máquina Virtual Java), existente nos sistemas de agentes móveis baseados em Java.

- *Recursos*: são os serviços e recursos que as máquinas e os locais fornecem, tais como: redes, bancos de dados, processadores e memória, discos e outros *hardwares*, além de serviços de *software*.
- *Localização*: é a combinação do nome do local no qual o agente está executando e do endereço da rede da máquina no qual o local existe. Esta localização é geralmente descrita como um endereço IP, e uma porta da máquina com um atributo de nome do local.
- *Mandantes*: um local tem dois mandantes: o seu construtor, que é o autor (fornecedor) do local de implementação, e o seu administrador, que é o mandante que tem a responsabilidade pela operação do ambiente de execução dos agentes (local).

4.7 Agentes Móveis com Java

Devido ao fato da linguagem Java ser orientada-a-objetos e baseada em rede faz com que ela se torne uma linguagem ideal para programação de agentes móveis. A seguir é listado algumas vantagens.

a) Independência de Plataforma

A linguagem Java é projetada para operar em ambientes heterogêneos. Para ativar uma aplicação Java para executar em qualquer lugar na rede, o compilador gera *bytecodes* neutros de arquitetura ao contrário de códigos nativos não portáteis. Para este código ser executado em um determinado computador, o sistema *runtime* do Java deve estar presente. Além disso, a linguagem Java não tem aspectos dependentes de plataforma; seus tipos de dados primitivos são rigorosamente especificados e não são dependentes de um determinado processador ou sistema operacional e; mesmo as bibliotecas que fazem parte do sistema são independentes de plataforma.

b) Segurança de Execução

Java é projetado para uso na *Internet* e *intranets*, e a demanda por segurança influenciou seu projeto de diversas maneiras:

- O modelo de ponteiros do Java elimina a possibilidade de sobrescrita de memória e corrupção de dados.
- Não permite *casting* de tipos ilegais ou de algum ponteiro aritmético.

- Os programas não conseguem forjar acesso a dados privados em objetos que eles não tenham permissão, uma estrutura que previne certos tipos de vírus.
- Caso um *bytecode* seja alterado, o sistema *runtime* do Java garante que o código seja desativado, para evitar a violação da semântica básica da linguagem Java.
- Possui um gerente de segurança para checar todas as operações potencialmente inseguras como: acesso a arquivos, conexões de redes, etc.

c) Leitura Dinâmica de Classes

Este mecanismo permite à máquina virtual carregar e definir classes em tempo de execução. Fornece um espaço de nome protegido para cada agente, permitindo a estes executarem de forma segura e independente uns dos outros. O mecanismo de leitura de classes é extensível e possibilita que as classes sejam carregadas através da rede.

d) Programação *Multithread*

Agentes são por definição autônomos; um agente executa independentemente de outro agente que resida no mesmo espaço. A possibilidade de execução em seu próprio processo, chamado *thread* de execução, é uma forma de se habilitar agentes a tornarem-se autônomos. Além da programação *multithread*, a linguagem Java também suporta um conjunto de primitivas de sincronização incluídas na linguagem e que permitem a interação entre os agentes.

e) Serialização de Objetos

Uma chave dos agentes móveis é que eles podem ser serializados e deserializados. A linguagem Java convenientemente fornece um mecanismo de serialização embutida que pode representar o estado de um objeto em uma forma serializada suficientemente detalhada para que o objeto seja reconstruído mais tarde.

f) Reflexão

O código Java pode descobrir informações sobre campos, métodos e construtores carregados e que podem ser usados para operar objetos de parceiros subordinados, tudo dentro das restrições de segurança. A reflexão implementa a necessidade dos agentes serem inteligentes sobre si mesmos e outros agentes.

5. O Modelo *Aglet*

Aglets são objetos Java que podem mover-se de uma estação em uma rede para outra. Este objeto móvel tem sua própria *thread* de execução, é dirigido por eventos e comunica-se por passagem de mensagens.

5.1 Elementos Básicos

Vamos agora analisar o modelo dos *aglets*. Este modelo define um conjunto de abstrações, e o comportamento necessário para integrar a tecnologia de agentes móveis em redes *WAN*, como a Internet por exemplo. 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*

Um *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*; isto é, esconde a real posição do *aglet* na rede.

c) Contexto

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

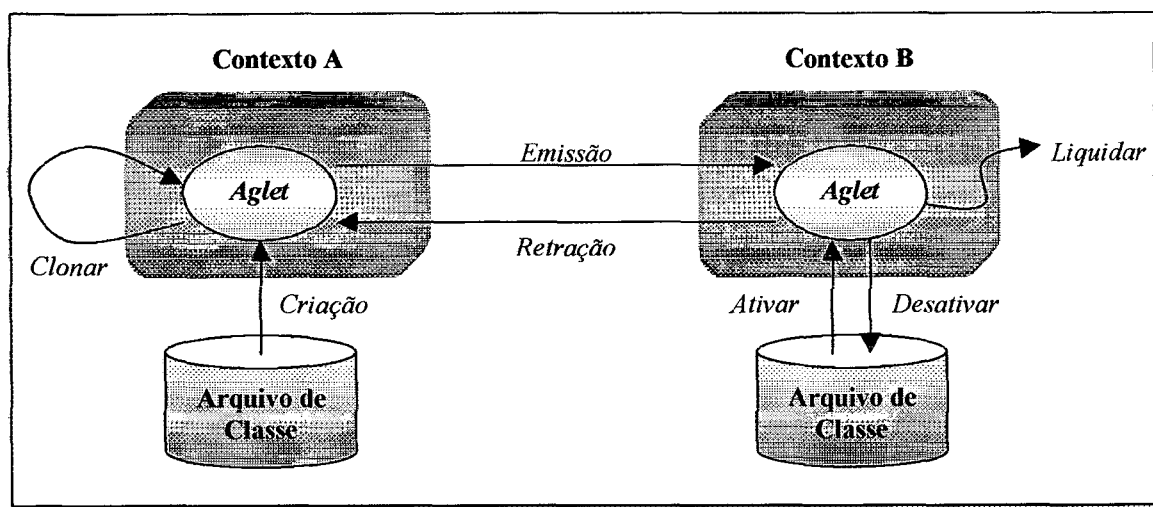
Com as abstrações acima descritas, pode-se controlar um objeto *aglet* da sua criação até a sua destruição; vamos descrever as principais operações de um *aglet* (fig. 5.1).

- *Criação (creation)*: sempre realizada em um contexto. Ao novo *aglet* é atribuído um identificador, inserido em um contexto e inicializado. O *aglet* iniciar 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 deve-se 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 de 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.

5.2 O Modelo de Eventos dos Aglets

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 modelo *aglet*:

Figura 5.1 - Principais Operações de um 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.

5.3 O Modelo de Comunicação dos Aglets

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

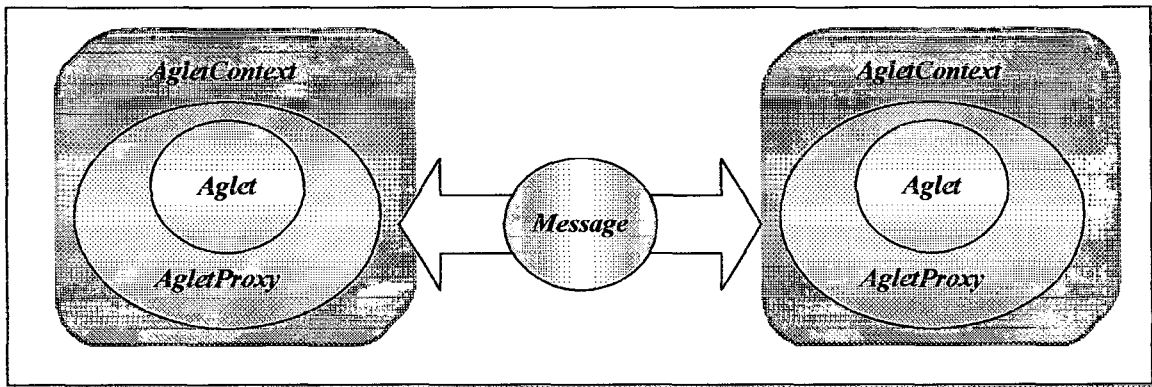
- *Message*: uma mensagem é um objeto trocado entre os *aglets*. Este modelo permite passagem de mensagens síncronas e assíncronas entre os *aglets*.

- *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 futuras e é usado para buscar resultados assim que eles estejam disponíveis. É possível também receber uma resposta e ignorar as subsequentes.

5.4 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 (fig. 5.2). A seguir será descrito as principais classes e interfaces desta API.

Figura 5.2 - Principais Classes e Interfaces Definidas na API *Aglet*



5.4.1 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*. A tabela 5.1 lista os principais atributos e métodos da classe *Aglet*.

Tabela 5.1 - Atributos e Métodos da Classe *Aglet*

| Atributos | |
|---------------------------|---|
| ACTIVE | Contém o estado do Aglet. |
| Métodos | |
| clone() | Clona o aglet e o proxy que o possui a sua localização. |
| deactivate(long duration) | Desativa o aglet durante um período de tempo |
| dispatch(URL destination) | Emite o aglet para o host especificado no argumento. |
| dispose() | Destrói e remove o aglet do seu contexto atual. |

| | |
|--------------------------------------|--|
| <code>getAgletContext()</code> | Obtém o contexto no qual o aglet está executando. |
| <code>getAgletID()</code> | Obtém informações do identificador do aglet. |
| <code>getAgletInfo()</code> | Obtém informações do objeto relacionado aquele aglet. |
| <code>getProxy()</code> | Obtém o proxy que mapeia um determinado aglet. |
| <code>onCreation(Object init)</code> | Este método é executado toda vez que o aglet é criado. |
| <code>onDisposing()</code> | Executado sempre que um Aglet está sendo liquidado. |
| <code>run()</code> | É o ponto de entrada para a execução do Aglet. Chamado sempre que um dos métodos: <code>creation</code> , <code>dispatch</code> , <code>retraction</code> ou <code>activation</code> é bem sucedido. |

5.4.2 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 importante regra da interface do *AgletProxy* é fornecer ao *aglet* transparência de localização. A tabela 5.2 é mostra seus métodos principais.

Tabela 5.2 - Métodos da Interface *AgletProxy*

| Métodos | |
|---|--|
| <code>activate()</code> | Força um aglet a se ativar. |
| <code>clone()</code> | Clona o aglet e seu proxy. |
| <code>deactivate(long duration)</code> | Desativa o aglet por um determinado período. |
| <code>dispatch(URL address)</code> | Emite o aglet para um determinado endereço. |
| <code>dispose()</code> | Destroi o aglet. |
| <code>getAgletID()</code> | Obtém o identificador do aglet. |
| <code>GetAgletInfo()</code> | Obtém informações do objeto do aglet. |
| <code>isActivate()</code> | Verifica se o aglet está ativado ou desativado. |
| <code>isRemote()</code> | Checa se o proxy está referenciando um aglet remoto. |
| <code>SendAsyncMessage (Message msg)</code> | Envia uma mensagem assíncrona ao aglet. |
| <code>SendMessage (Message msg)</code> | Envia uma mensagem síncrona ao aglet. |

5.4.3 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. A tabela 5.3 mostra os principais métodos da interface *AgletContext*.

Tabela 5.3 - Métodos da Interface *AgletContext*

| Métodos | |
|--|--|
| <code>CreateAglet(URL codebase, String code, Object init)</code> | Cria uma instância da classe <i>aglet</i> especificada. O arquivo de código da classe do <i>aglet</i> pode estar localizado no sistema de arquivos local, assim como no servidor remoto. |
| <code>GetAgletProxies(int type)</code> | Obtém um array com os proxies dos <i>aglets</i> no contexto corrente. |
| <code>GetAgletProxy(AgletID id)</code> | Obtém o proxy do <i>aglet</i> no contexto corrente. |
| <code>GetHostigURL()</code> | Obtém o URL do servidor do contexto. |
| <code>getName()</code> | Obtém o nome do contexto. |
| <code>RetractAglet(URL url, AgletID aid)</code> | Recupera um <i>aglet</i> especificado pela URL da estação e pelo seu identificador. |
| <code>shutdown()</code> | Desliga o contexto. |
| <code>start()</code> | Inicia o contexto. |

5.4.4 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. Na tabela 5.4 é listado seus principais métodos.

Tabela 5.4 - Atributos e Métodos da Classe *Message*

| Atributos | |
|-----------------------------|---|
| <code>SYNCHRONOUS</code> | Tipos de mensagem indicando como a mensagem será enviada. |
| Métodos | |
| <code>getTimeStamp()</code> | Obtém o tempo em milisegundos de quando a mensagem foi enviada. |
| <code>sendReply(...)</code> | Envia uma resposta com um valor específico (null, float, boolean, int, char, double, long, Object). |

5.4.5 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. A tabela 5.5 lista seus principais métodos.

Tabela 5.5 - Métodos da Interface *FutureReply*

| Métodos | |
|---|--|
| <code>getBooleanReply()</code> | Obtém a resposta como um primitivo boolean. |
| <code>getCharReply()</code> | Obtém a resposta como um primitivo char. |
| <code>getDoubleReply()</code> | Obtém a resposta como um primitivo double. |
| <code>getFloatReply()</code> | Obtém a resposta como um primitivo float. |
| <code>getIntReply()</code> | Obtém a resposta como um primitivo int. |
| <code>getLongReply()</code> | Obtém a resposta como um primitivo long. |
| <code>isAvailable()</code> | Verifica se a resposta está disponível ou não. |
| <code>waitForReply()</code> | Espera pela resposta até que a mesma esteja disponível |
| <code>waitForReply(long timeout)</code> | Espera pela resposta por um período de tempo limitado. |

6. 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 (CHAVES, 1996, RIPPER, 2000).

6.1 O Modelo CBB

O modelo de CBB (*Consumer-Buying Behavior* ou Comportamento de Compra de Clientes) (GUTTMAN, 1998a) 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. Muitos 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.

6.2 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 isso que este 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, muitos destes impedimentos desaparecem. Por exemplo, EggHead (<http://www.egghead.com>), eBay's (<http://www.ebay.com>), Arremate.com (<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.

A seguir é mostrado dois dos *sites* mais populares (fig. 6.1, fig. 6.2), um a nível mundial e o outro aqui no Brasil, respectivamente.

Figura 6.1 - Site de Leilões do eBay's

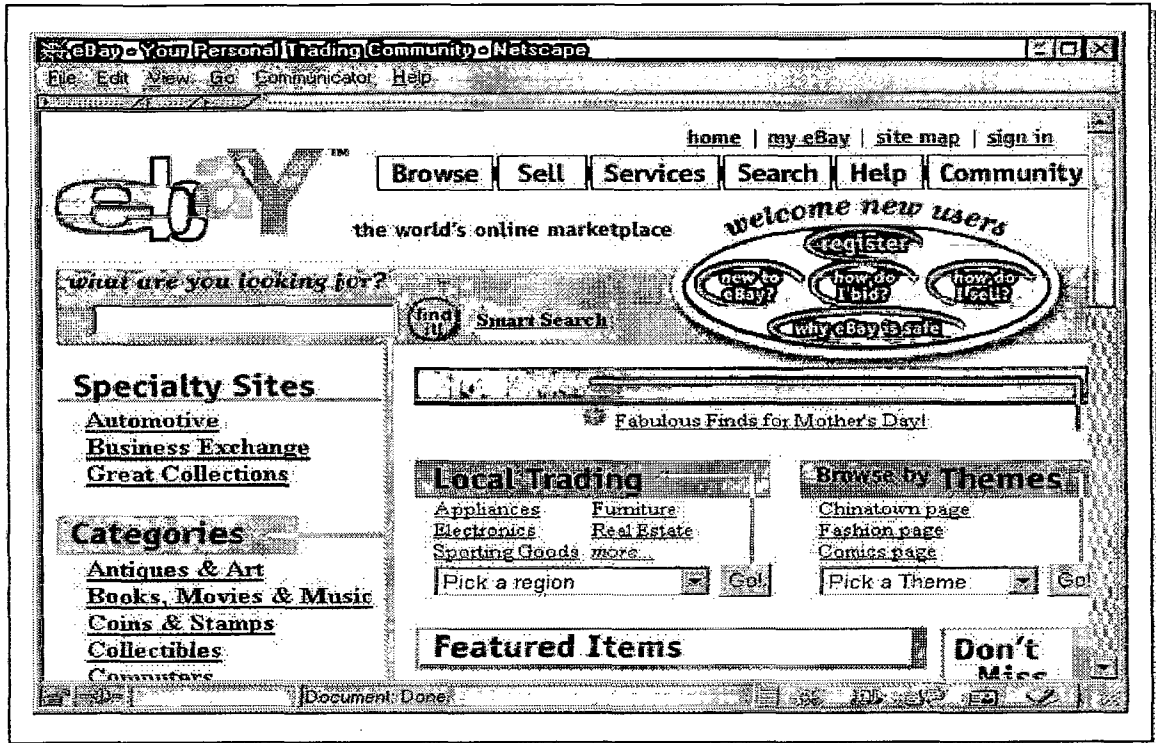
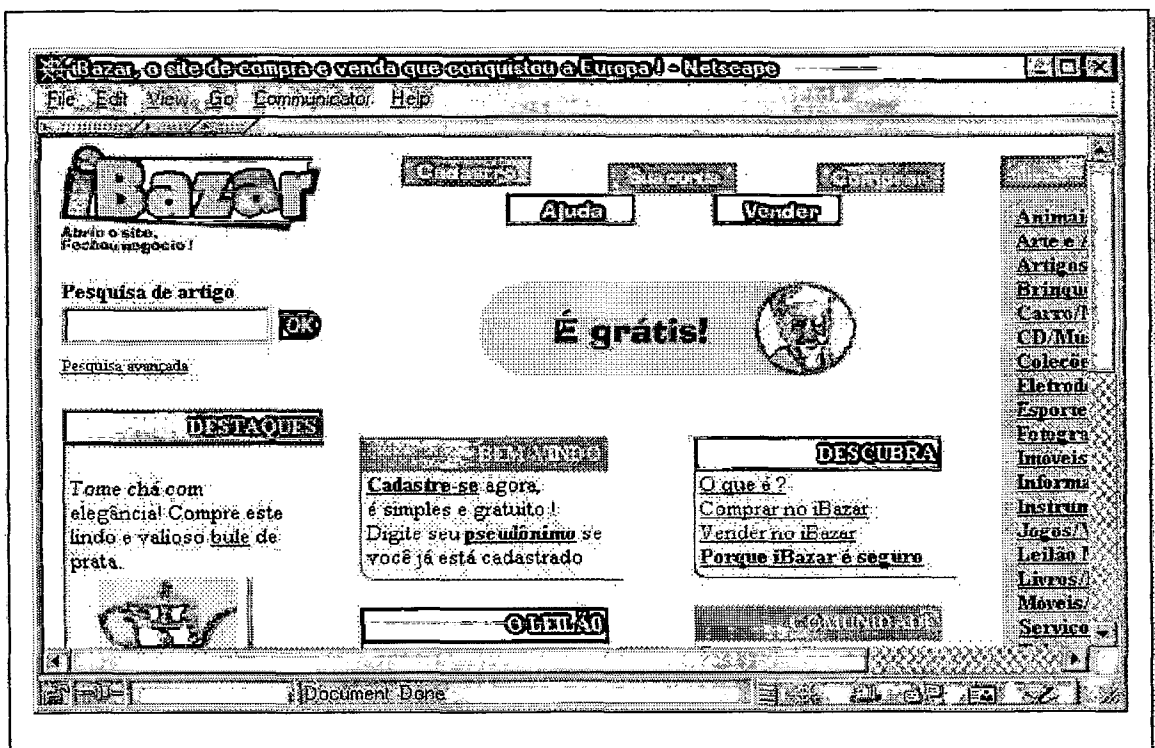


Figura 6.2 - Site de Leilões do iBazar



Para automatizar ainda mais os leilões e o processo de negociação existente neles, estudos vêm sendo realizados para a mediação de agentes neste 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-a-Tete/>), Jango (*Excite* - <http://www.jango.com>), AuctionBot (<http://auction.eecs.umich.edu>), V-Market (<http://harper.les.inf.puc-rio.br/vbookmark>), dentre outros. A figura abaixo compara os diferentes sistemas pesquisados e quais etapas do modelo CBB eles implementam através dos agentes (fig. 6.3).

Figura 6.3 - Etapas do Modelo CBB em Sistemas que utilizam a Mediação de Agentes

| | AuctionBot | Bargain Finder | Jango | Kasbah | T@T | V-Market |
|--|------------|----------------|-------|--------|-----|----------|
| 1. Identificação da necessidade | | | | | | |
| 2. Pesquisa de produto | | | X | | X | |
| 3. Pesquisa de mercado | X | X | X | X | X | X |
| 4. Negociação | X | | | X | X | X |
| 5. Pagamento e entrega | | | | | | |
| 6. Suporte e avaliação | | | | | | |

Vamos agora analisar dois sistemas – AuctionBot e V-Market – que utilizam a Mediação de Agentes e que serviram como base para a análise, projeto e concepção do sistema CUBE.

6.3 AuctionBot

AuctionBot (<http://auction.eecs.umich.edu>) é um servidor de leilões genéricos da Universidade de Michigan (WURMAN, 1998). Neste sistema usuários criam novos leilões para comprar ou vender produtos através da escolha de tipos de leilões e da especificação de seus parâmetros: tempo de liquidação, métodos para resolução de múltiplas propostas, número de vendedores permitido, dentre outros (fig. 6.4).

Compradores e vendedores podem então fazer propostas de acordo com os protocolos de negociação distribuído multi-lateral. Em um cenário típico, um vendedor poderia propor um preço reservado depois de criar um leilão e deixar o AuctionBot administrar e forçar o comprador a fazer propostas de acordo com os protocolos e parâmetros do leilão.

Figura 6.4 - Janela de Criação de Leilões do AuctionBot

The screenshot shows a Netscape browser window titled "Create Auction Application - Netscape". The address bar contains the URL "http://auction.eecs.umich.edu/...". The page content includes a header for "Michigan Internet AuctionBot" with a navigation menu. Below the header, the main heading is "Create Auction Application (continued)". The form contains two input fields: one for "What is the name of the good to be auctioned?" and another for "Provide a concise description of the good:". At the bottom, there is a partially visible field for "Enter a URL for partic...".

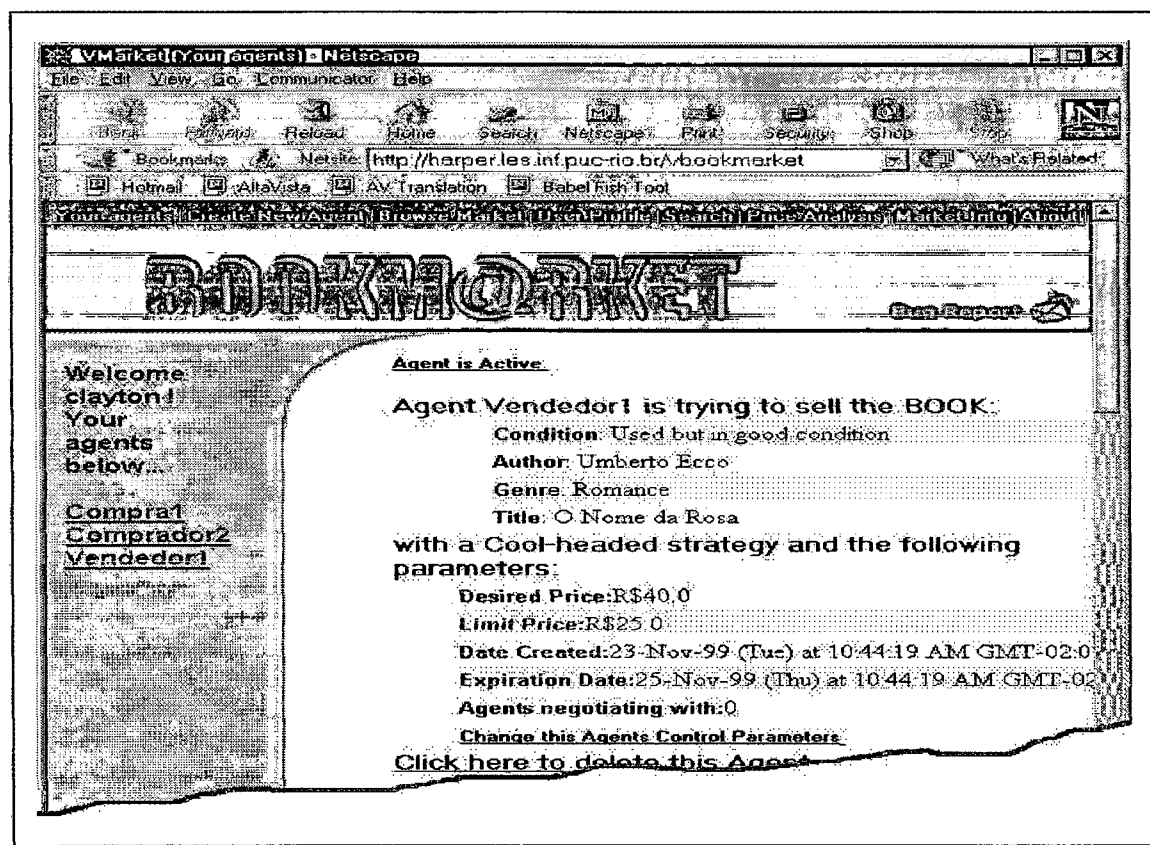
6.4 V-Market

VBookMarket é a primeira instância de um *framework* chamado V-Market desenvolvido no Laboratório de Engenharia da PUC-Rio (RIPPER, 2000).

O principal objetivo do projeto V-Market é dar aos consumidores e empresas a capacidade de padronização de mercados virtuais e definição de categorias de transações sob demanda. Este projeto expande a visão inicial de outro *framework* chamado Kasbah's (desenvolvido no MIT Lab). Os agentes do V-Market avaliam e negociam ativamente com compradores e vendedores interessados, representados por seus respectivos agentes, conforme mostrado na fig. 6.5 (<http://harper.les.inf.puc-rio.br/vbookmarket>).

Os agentes de *software* podem ser criados com qualquer conjunto de comportamentos desejados, desta forma, é permitido ao consumidor ter uma presença virtual no espaço de mercado para lutar por seus interesses, liberando-o de sua constante monitoração.

Figura 6.5 - Janela para Exibição de Agentes Criados no VMarket



7. Sistema CUBe – Análise e Projeto

Baseado em pesquisas anteriores e estudos da implementação de diversos sistemas de *e-Commerce* com a mediação de Agentes Móveis, especialmente aqueles descritos anteriormente em (RIPPER, 2000, WURMAN, 1998), foi projetado o CUBe – um sistema orientado a objetos para integração de diversos sistemas: de *e-Commerce*, *e-Business* e ERP, através da Mediação de Agentes móveis. Este sistema foi desenvolvido com a ferramenta de agentes apresentada anteriormente e visa dentre outras coisas:

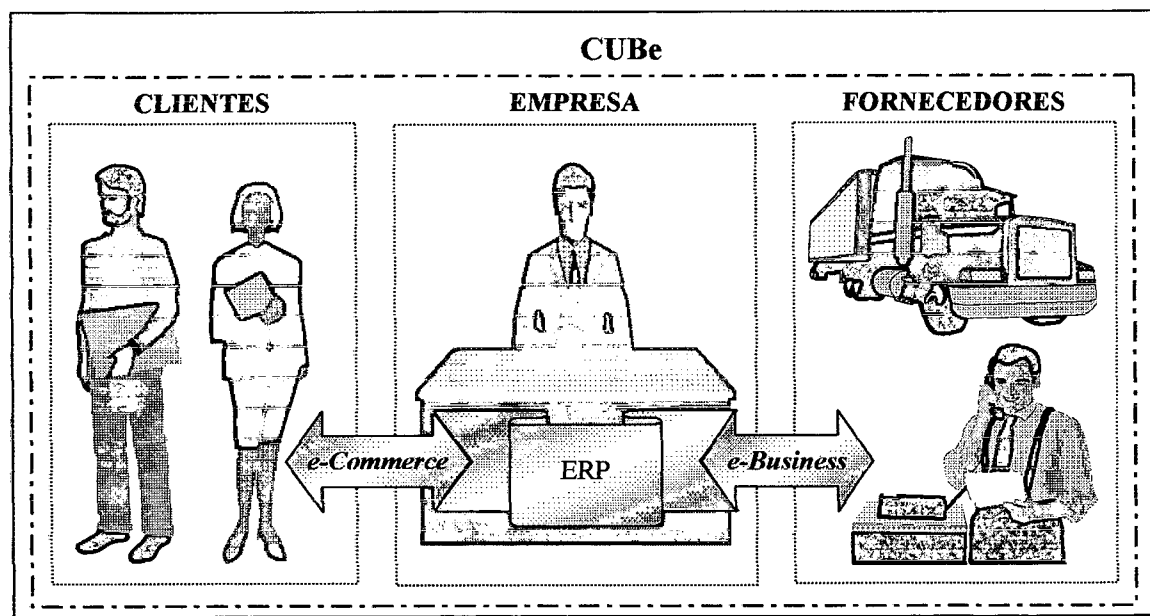
- *Interoperabilidade*: através da adoção de uma ferramenta baseada na linguagem Java – os Aglets da IBM.
- *Facilidade de uso*: através da automação de várias tarefas definidas no CBB (GUTTMAN, 1998a), fazendo com que algumas tarefas repetitivas presentes nos sistemas acima, sejam mediadas pelos agentes móveis; evitando que o usuário necessite constantemente monitorar o sistema, ou mesmo, realizar tarefas enfadonhas, como o processo de negociação, por exemplo.

7.1 Análise do Sistema CUBe

A análise do sistema CUBe foi realizada de forma a se obter um sistema interoperável, mas que ao mesmo tempo não abandonasse um de seus principais objetivos que é a facilidade de uso. A análise foi dividida em 3 partes distintas, conforme mostrado na fig. 7.1.

Para a integração de cada parte, agentes móveis atuam como mediadores das transações do sistema, além disso, cada módulo terá tecnologias apropriadas para a operacionalização dos diferentes meios de acesso, todas integradas através da Máquina Virtual Java (JVM); já o ambiente dos agentes e os agentes trabalharão, conforme descrito no capítulo 5. Para a fase de análise, utilizou-se a UML (*Unified Modelling Language* ou Linguagem de Modelagem Unificada) através da representação de *Use Cases* (Casos de Uso). Vamos detalhar agora cada uma das 3 partes.

Figura 7.1 - CUBe e as 3 Partes Integrantes de seu Sistema



7.1.1 Clientes

O acesso dos clientes ao sistema CUBe é feito pela *Internet* – através de páginas ASP fornecidas por um servidor *Web*, rodando IIS (*Internet Information Server*). Nas páginas existem itens a serem selecionados através de um formulário eletrônico padrão, montando-se com isso um pedido eletrônico padronizado.

Após o pedido, o usuário definirá o tipo de agente remoto que atuará no sistema ERP, sendo este parametrizado com características de comportamento previamente definidas, dentre os tipos de agentes 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.

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. Após 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 (ANEXO 1).

7.1.2 Empresa

Dentro do Sistema ERP, agentes integrantes deste sistema, trabalharão automatizando várias etapas descritas no modelo CBB, antes controladas manualmente por funcionários da empresa. Apenas tarefas exclusivamente de consulta, produção ou configuração ainda serão realizadas por funcionários; alguns exemplos destas tarefas são: emissão de relatórios e tarefas de configuração de informações do sistema, lançamento da ordem de produção para uma ordem de venda, reposição de itens em estoque, etc.

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.

O funcionamento do Sistema ERP começa com a chegada do Agente Remoto que entrará em negociação com o Agente Administrador do Sistema ERP e caso haja sucesso nesta etapa, irá firmar um pedido de compra, este então irá emitir este pedido ao Agente Vendedor que irá emitir uma Ordem de Venda (OV) com os dados relativos ao pedido; a tarefa de emissão de boletos de cobrança, também ficará subordinada a este agente (ANEXO 6).

Emitida a OV, esta é entregue ao Agente Produtor que irá recebê-la, analisar a sua viabilidade. Caso seja possível a produção, será emitida uma Ordem de Produção (OP), caso contrário, será gerada uma Ordem de Compra (OC) para compra do material em falta relativo à OV. Depois da OP ter sido manufaturada, os itens serão estocados e será emitida uma Ordem de Emissão (OE) para entrega dos itens solicitados (ANEXO 3).

A OE chega então ao Agente Emissor que receberá as informações relativas ao item (quantidade, destino, cliente, dentre outros), e este agendará uma entrega com uma transportadora (fornecedor do sistema) e emitirá uma Ordem de Entrega (OEn), fechando, então, o ciclo principal do sistema ERP; também será emitida uma Nota Fiscal do produto, para ser levada pela transportadora, junto com o pedido, para o cliente (ANEXO 4).

Existem ainda no sistema ERP outras informações relativas à entrega de boletos (ANEXO 5), compra de produtos e agendamento de transporte (ANEXO 2) que serão consideradas na construção do sistema CUBe.

Outros fatores como Gerenciamento de Recursos Humanos, *Supply-Chain Management*, Serviços, dentre outros, não foram considerados neste primeiro projeto, evitando-se uma complexidade excessiva, presente nos sistemas ERP, e todos seus módulos complementares.

7.1.3 Fornecedores

Para o sistema CUBe, os 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 de um pedido, realizando várias tarefas como: venda de peças em estoque, agendamento de entrega de produtos, pesquisa de preços, dentre outros.

Para este trabalho analisou-se a possibilidade de comunicação com 3 tipos distintos de fornecedores de serviços e produtos:

- *Bancos* – fornecedor de serviços de cobrança;
- *Transportadoras* – fornecedor de serviços de entrega de produtos e;
- *Fornecedores* – fornecedor de produtos necessários para a produção de itens manufaturáveis pela empresa.

Devido à complexidade e às particularidades de cada fornecedor, acima mencionados, definiu-se apenas que os sistemas relativos a esta terceira etapa seriam apenas receptores dos pedidos gerados por outras empresas, conforme mostra o ANEXO 7, ANEXO 8 e ANEXO 9, possibilitando que futuros trabalhos venham a acrescentar-se a este projeto inicial, conforme discutido mais adiante.

7.2 Projeto do Sistema CUBe

Na fase de projeto do sistema CUBe, buscou-se especificar o sistema em um diagrama de classes utilizando-se a linguagem UML, com todas as características definidas na fase de análise. Alguns pontos críticos do sistema, descritos anteriormente, e do projeto do sistema CUBe foram observados e as soluções encontradas são

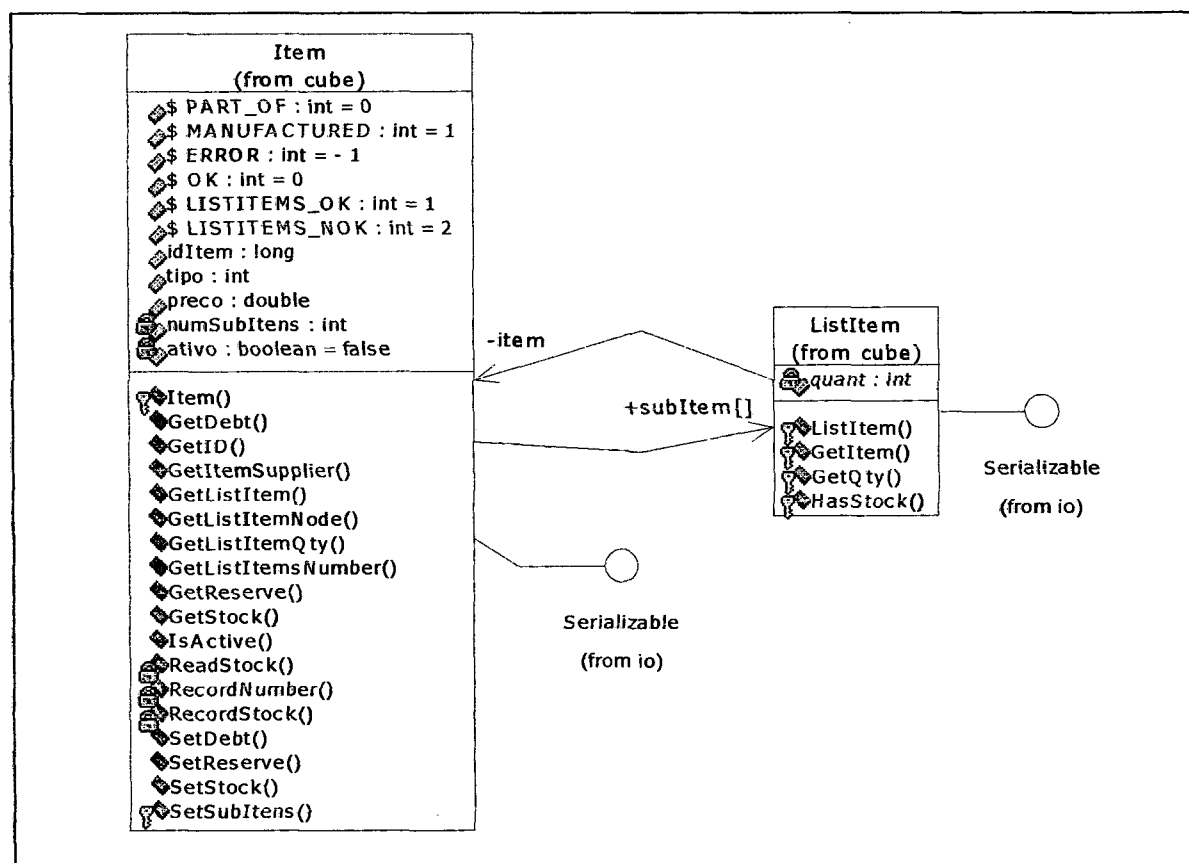
mostradas resumidamente a seguir. Para a visualização do diagrama de classes completo, pesquisar ANEXO 10, ANEXO 11, ANEXO 12 e ANEXO 13.

a) Suporte a múltiplos produtos

Uma das principais características, tanto nos sistemas de *e-Commerce* quanto nos sistemas de *e-Business* é a comercialização (compra/venda) de vários produtos; ao contrário de alguns sistemas mediadores vistos anteriormente, que geralmente trabalham com uma lista definida de itens (em geral livros e CDs), no sistema CUBe buscou-se uma generalização maior aproximando-o de um sistema real, onde a inclusão de um item novo é feita na base de dados da empresa e a partir daí ela está pronta para comercialização.

No sistema CUBe um produto poderá ser de qualquer tipo e, além disso, poderá ser criado a partir de uma lista de itens (composição do item produto). A fig. 7.2 ilustra este tópico do sistema.

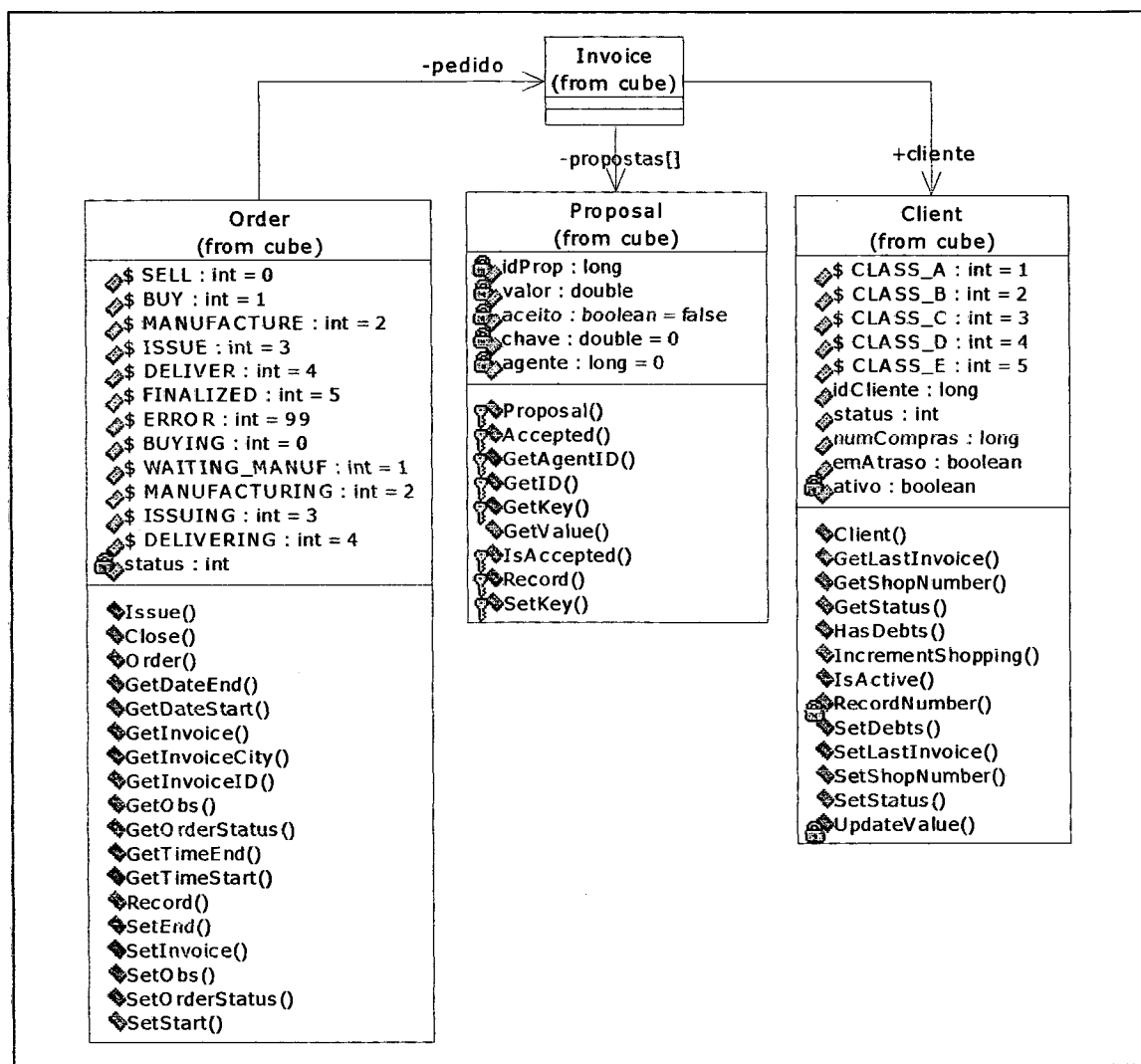
Figura 7.2 - A Especificação do Produto e sua Composição no Sistema CUBe



b) Automação de várias etapas do modelo CBB

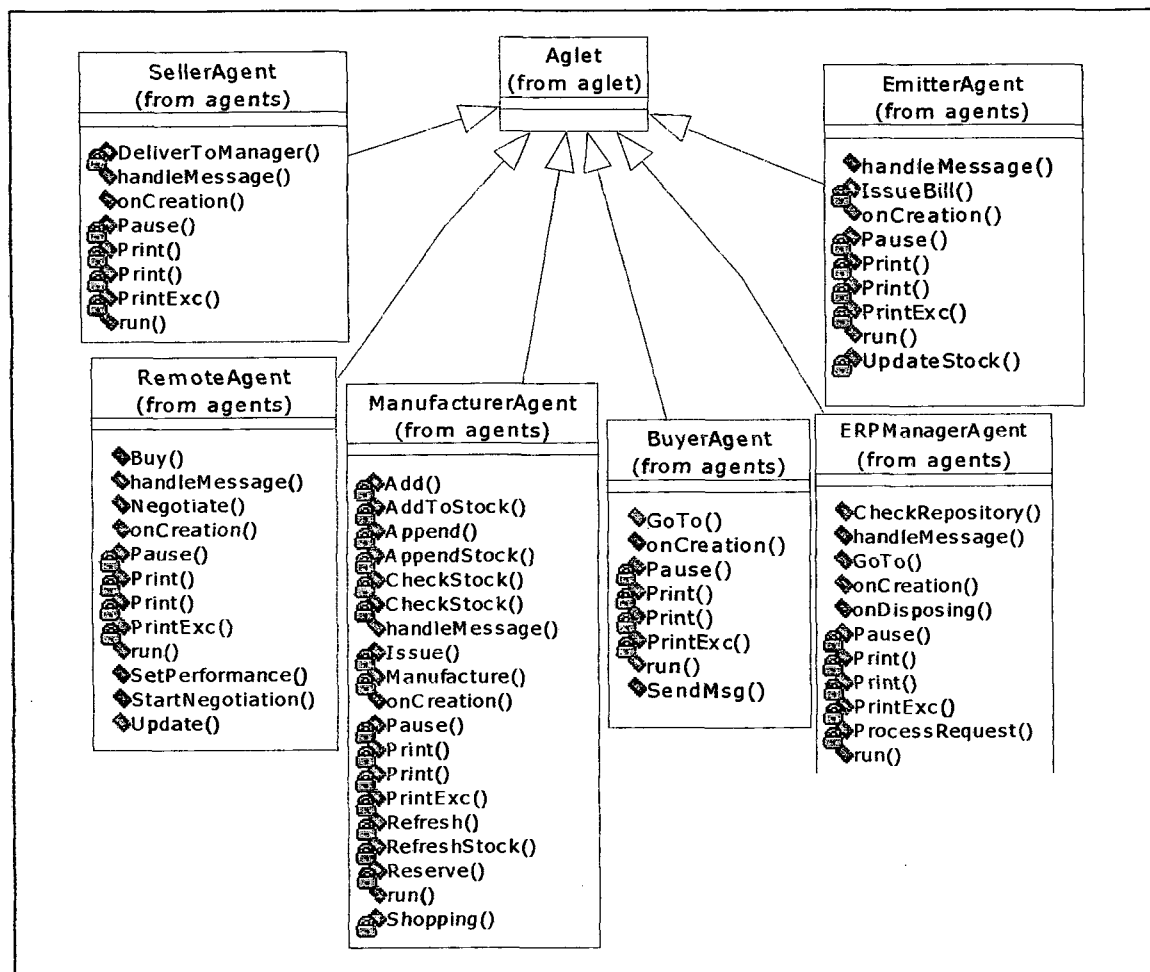
Graças aos agentes móveis e ao projeto integrador do sistema CUBE (fig. 7.3), é possível automatizar as etapas 1, 4, 5 e 6 do modelo CBB; as etapas 2 e 3 não serão automatizadas neste sistema, discutido mais adiante em trabalhos futuros; já que este projeto inicial comercializa produtos de única empresa, não fazendo sentido para tal, a implementação de mecanismos de pesquisa de produtos e de mercado, objetivos das etapas 2 e 3 do modelo CBB.

Figura 7.3 - A Especificação do Pedido e da Ordem dentro do Sistema ERP



No diagrama de classes definido na fig. 7.4, cada agente atuará dentro do sistema automatizando uma ou mais tarefas especificadas no modelo CBB; a fig. 7.5 mostra o modelo CBB e os agentes que trabalham em cada uma de suas etapas dentro do sistema CUBE.

Figura 7.4 - Especificação dos Agentes do Sistema CUBe



O objetivo deste modelo é buscar um nível de automação maior e mais integrado, complementando a implementação básica que os *frameworks* estudados anteriormente fazem (automação das etapas 3, 4 e 5); pretende-se com este modelo abranger as etapas 1, 4, 5 e 6 do modelo CBB.

c) Múltiplos processos de negociação e comportamentos

Para a inclusão de um comportamento aliado a uma estratégia de negociação, utilizou-se a classe *Strategy* que implementa características da interface *Behaviour*, conforme mostrado na fig. 7.6. Com esta estrutura, agentes com comportamentos variados, ao se encontrarem terão uma determinada estratégia de negociação; ao se mudar o comportamento do agente, muda-se também a sua estratégia no ato da negociação.

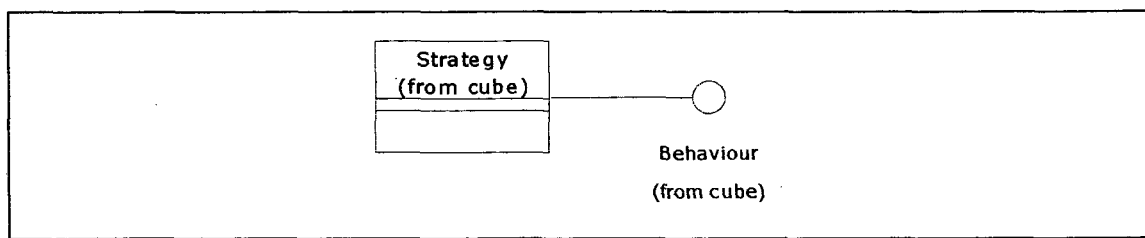
O objetivo do sistema CUBe, no que tange a este tópico, é deixar um determinado número de estratégias previamente definidas. Com isso ganha-se em flexibilidade, pois

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.

Figura 7.5 - Etapas do Modelo CBB e a sua Automação através dos Agentes do CUBe

| | Remoto | Administrador | Comprador | Vendedor | Produtor | Emissor |
|---------------------------------|--------|---------------|-----------|----------|----------|---------|
| 1. Identificação da necessidade | | | | | X | X |
| 2. Pesquisa de produto | | | | | | |
| 3. Pesquisa de mercado | | | | | | |
| 4. Negociação | X | X | | | | |
| 5. Pagamento e entrega | | | | | | X |
| 6. Suporte e avaliação | X | X | X | X | X | X |

Figura 7.6 - Estratégia e Comportamento do Agente dentro do Sistema CUBe

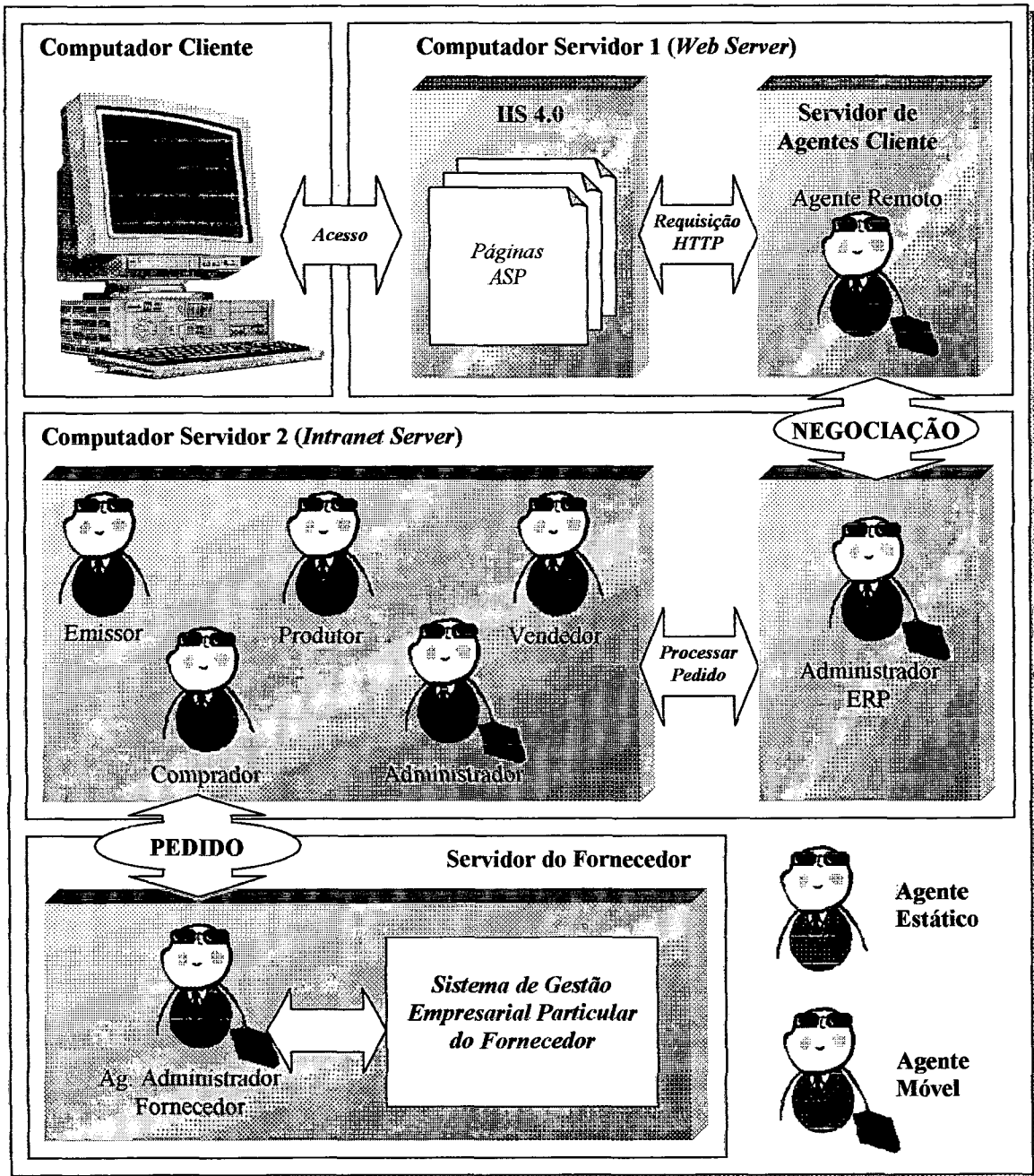


7.3 Arquitetura do Sistema CUBe

Na fig. 7.7 temos a arquitetura geral do sistema CUBe, e algumas tecnologias empregadas na sua construção.

Detalhes da arquitetura, da implementação do Sistema CUBe, dos objetos, da interface entre os mesmos, das funções, dos resultados e de alguns tópicos relevantes serão descritos no próximo capítulo.

Figura 7.7 - Arquitetura do Sistema CUBe



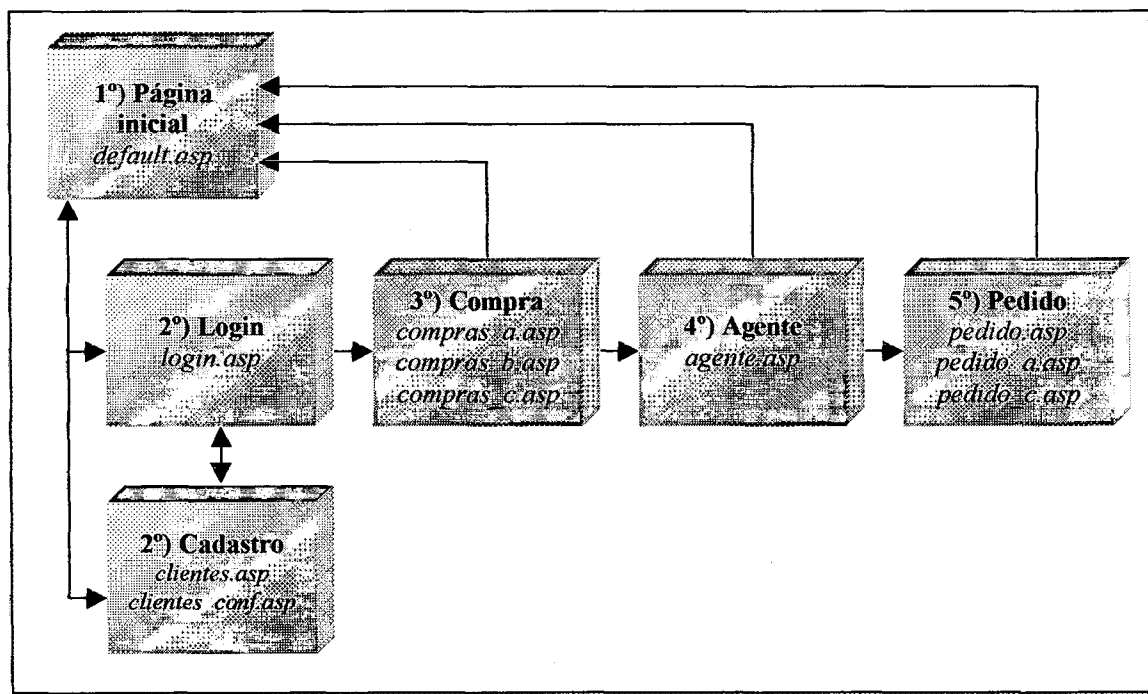
8. CUBe – Concepção do Sistema

Depois da fase de análise e projeto do sistema, buscou-se a sua implementação usando para tal as ferramentas descritas anteriormente. A seguir serão abordados todos os tópicos relevantes e características fundamentais do projeto.

8.1 Sistema de Comércio Eletrônico

A construção das páginas que serão acessadas pelo cliente baseou-se no padrão ASP (*Active Server Pages*), conseguindo-se uma integração maior e mais simples à uma base de dados através do ODBC (*Object Database Connectivity*). A estrutura hierárquica das páginas presentes no projeto do sistema CUBe, segue a estrutura a seguir (fig. 8.1).

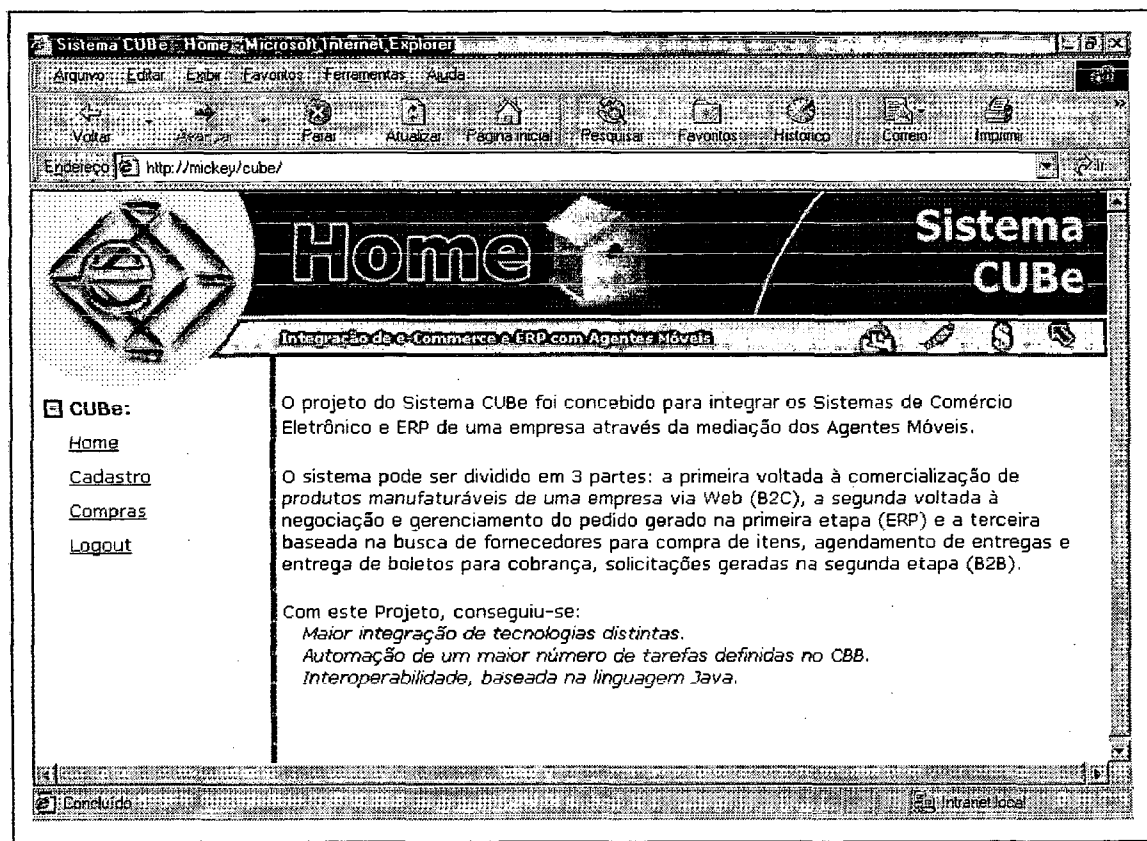
Figura 8.1 - Estrutura Hierárquica das Páginas ASP do Sistema de Comércio Eletrônico dentro do Sistema CUBe



Cada bloco acima representa uma etapa dentro do Sistema de Comércio Eletrônico do Sistema CUBe, e cada cor, representa um diferente nível hierárquico. As páginas simulam uma compra padrão, encontrada hoje em dia na *Internet*; a diferença encontra-se no nível dos Agentes, onde o usuário irá definir qual agente atuará como

negociador do pedido efetuado pelo usuário, e no nível do Pedido, onde o usuário irá parametrizar os valores da negociação para o agente. A página de abertura do sistema é mostrada na fig. 8.2.

Figura 8.2 - Página *default.asp* do Sistema CUBe



O usuário nesta página pode cadastrar-se, caso ele ainda não seja usuário do sistema, ou escolher a opção de compras. Caso o usuário clique na opção *Cadastro* será então remetido ao segundo nível: *clientes.asp* – mostrada na fig. 8.3.

Para a opção *Compras*, será aberta para ele a página do 2º nível: *login.asp* – nesta página existem dois campos (*eMail* e *Senha*) que permitem uma segurança básica para este sistema, conforme mostrado na fig. 8.4.

Depois do usuário ser validado, é então remetido ao 3º nível: *compras_a.asp*, *compras_b.asp*, *compras_c.asp* – nesta etapa o usuário irá montar uma cesta de produtos para compra; os produtos que são listados devem estar previamente cadastrados para serem disponibilizados para um pedido. Uma das janelas da etapa de compras é mostrada na fig. 8.5.

Figura 8.3 - Página *clientes.asp* do Sistema CUBe

Sistema CUBe - Cadastro de Clientes - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Voltar Avançar Para Atualizar Página inicial Pesquisar Favoritos Histórico Correio Imprimir

Endereço http://mickey/cube/clientes.asp

Cadastro Sistema CUBe

Preencha todos os campos para efetivação do cadastro!

Dados cadastrais:

Nome: Clayton A. Valdo

Endereço: R. Maria Luiza Rodrigues, 6

Cidade: Florianópolis Estado: SC

CEP: 88004-032

Telefone: (48) 346-4339 Fax:

eMail: cvaldo@terra.com.br

Senha: Confirm:

Cadastrar Limpar

Concluído Intranet local

Figura 8.4 - Página *login.asp* do Sistema CUBe

Sistema CUBe - Login - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Voltar Avançar Para Atualizar Página inicial Pesquisar Favoritos Histórico Correio Imprimir

Endereço http://mickey/cube/login.asp

Login Sistema CUBe

Entre com seu Login!

Digite seu email/senha abaixo:

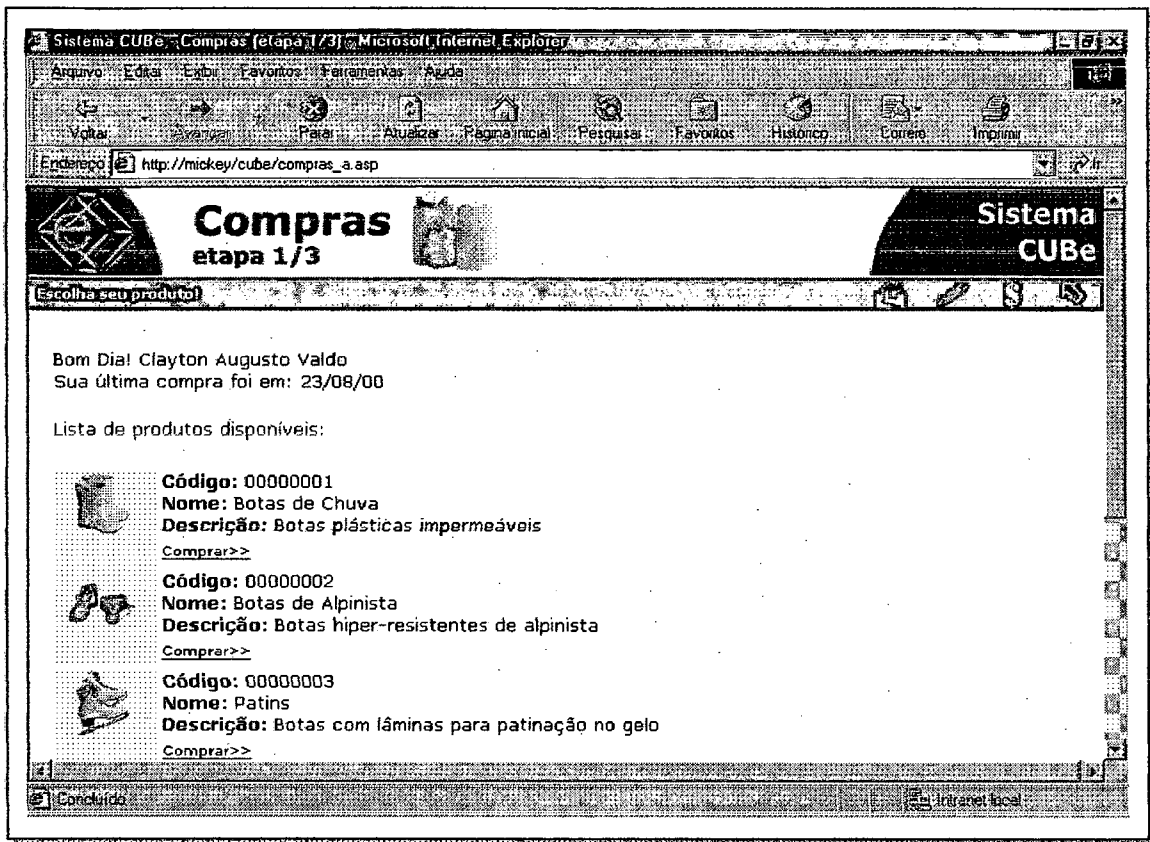
eMail:

Senha:

Login Limpar

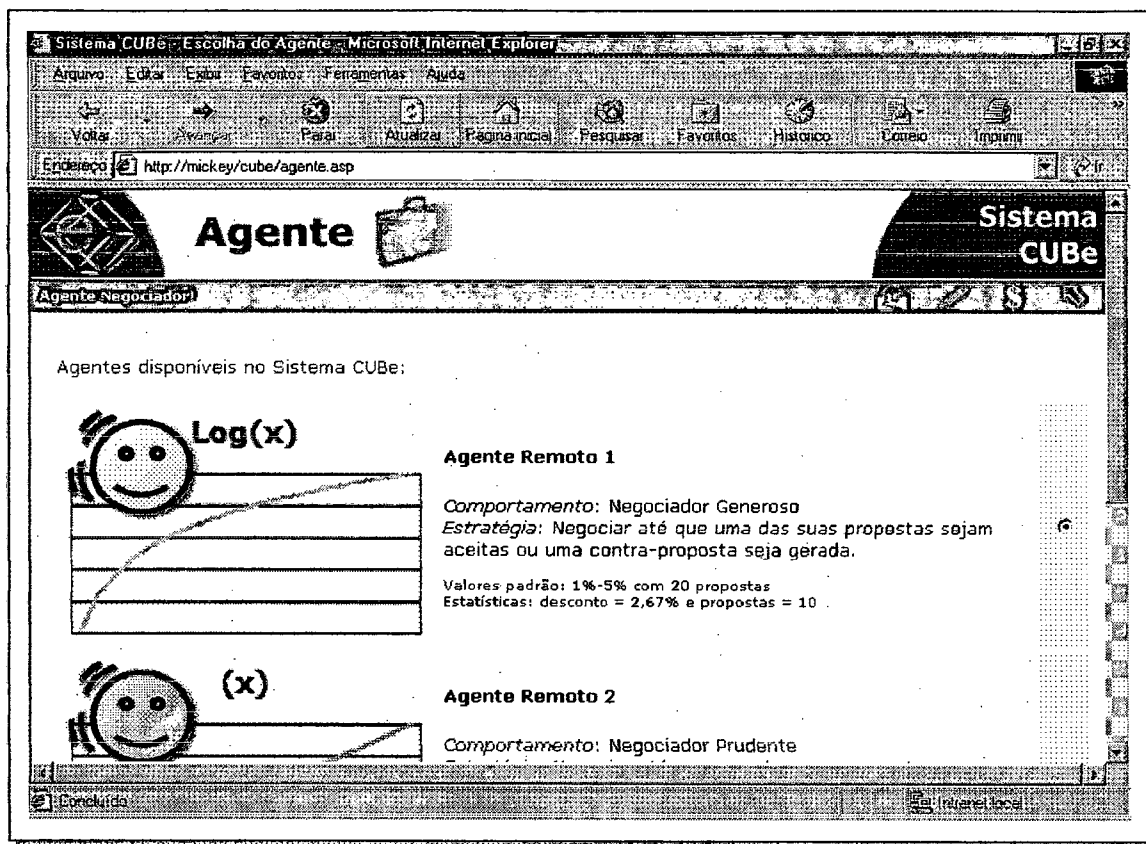
Caso você ainda não tenha feito seu cadastro, clique [aqui](#).

Intranet local

Figura 8.5 - Página *compras_a.asp* do Sistema CUBe

Após as 3 etapas do processo de compra dos produtos, o usuário terá então uma cesta com itens e suas respectivas quantidades, totalizando com isso um pedido; nos sistemas de comércio eletrônico tradicionais, o processo seguinte seria o de preenchimento dos dados de entrega dos itens para recebimento posterior; no Sistema CUBe, a próxima etapa permite ao usuário definir um agente que atuará como mediador, no processo de negociação do seu pedido com o agente administrador da empresa responsável pelos produtos selecionados na página de compras.

Assim, é possível de acordo com o modelo CBB, descrito anteriormente, uma aproximação mais realística de um processo de compra tradicional. A página mostrada na fig. 8.6 representa o 4º nível: *agente.asp* – nesta página temos 3 agentes distintos com comportamentos previamente definidos no Sistema CUBe; são eles: Generoso, Prudente e Rígido, sendo que cada um destes comportamentos identifica o agente dentro do sistema, o algoritmo de negociação adotado e a função de variação de cálculo durante a etapa de negociação do pedido.

Figura 8.6 - Página *agente.asp* do Sistema CUBe

Os padrões definidos são identificados e parametrizados na Tabela 8.1. Cabe salientar neste ponto que as funções de variação de negociação foram escolhidas ao acaso, porém o procedimento de parametrização, previamente definido, também foi adotado em outros trabalhos, em especial (CHAVES, 1996, RIPPER, 2000), visando trazer para o usuário uma maior facilidade na operação do sistema.

Tabela 8.1 - Parâmetros dos Agentes no Sistema CUBe

| Comportamento do Agente. | Função utilizada 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 afirmativa |

Depois de escolhido o tipo de agente que irá representar o pedido do usuário no sistema, temos o 5º nível: *pedido.asp* – nesta etapa tem-se as informações sobre: a entrega do pedido, os itens pertencentes ao pedido e as diretrizes para o agente, que são os valores de desconto mínimo e máximo que o agente deverá buscar durante a negociação, e o número de propostas máxima permitida, conforme mostrado na fig. 8.7.

Figura 8.7 - Página *pedido.asp* do Sistema CUBe

Sistema CUBe - Pedido (definição dos parâmetros) - Microsoft Internet Explorer

Arquivo: Editar: Exibir: Favoritos: Ferramentas: Ajuda

Voltar: Parar: Atualizar: Página inicial: Pesquisar: Favoritos: Histórico: Excluir: Imprimir

Endereço: http://mickey/cube/pedido.asp

Destinatário: Clayton

Endereço: Rua

Cidade: Florianópolis Estado: SC

CEP: 88000-000

Telefone: 233-0916

2- Informações do Pedido:

| Quantidade | Código/Nome | Preço unit. | Total Parcial |
|------------------|------------------|-------------|---------------|
| 1 | 1/Botas de Chuva | R\$ 10,00 | R\$ 10,00 |
| Total do Pedido: | | | R\$ 10,00 |

3- Diretrizes do Agente:

Nome: Agente Remoto 2
Vendedor Prudente
Estratégia: Negocia até que sua proposta seja aceita, caso nenhuma seja aceita, aceita a última contra-proposta gerada!

Barganhar descontos de 1 até 10 % no valor total do pedido.

Nº de propostas permitido: 100

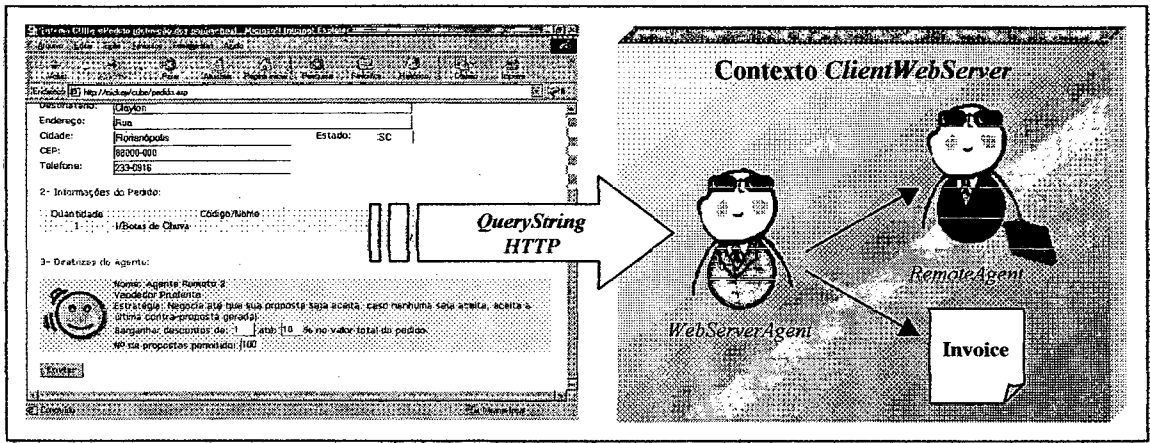
Enviar

Concluído

Quando o usuário proceder o envio do pedido, os dados são então repassados ao contexto *ClientWebServer*, através de uma *Query String* padrão *HTTP*; o contexto de agentes irá então capturar estas informações e instanciar o pedido (classe *Invoice*) de acordo com os parâmetros definidos nos níveis 3 e 5, e o Agente Remoto, responsável por este pedido, de acordo com o comportamento e diretrizes definidas nos níveis 4 e 5.

A fig. 8.8, representa a passagem de valores que ocorre entre o *Web Server IIS* e o contexto de agentes *ClientWebServer*.

Figura 8.8 - Passagem de Valores entre *Web Servers*



8.1.1 O contexto *ClientWebServer*

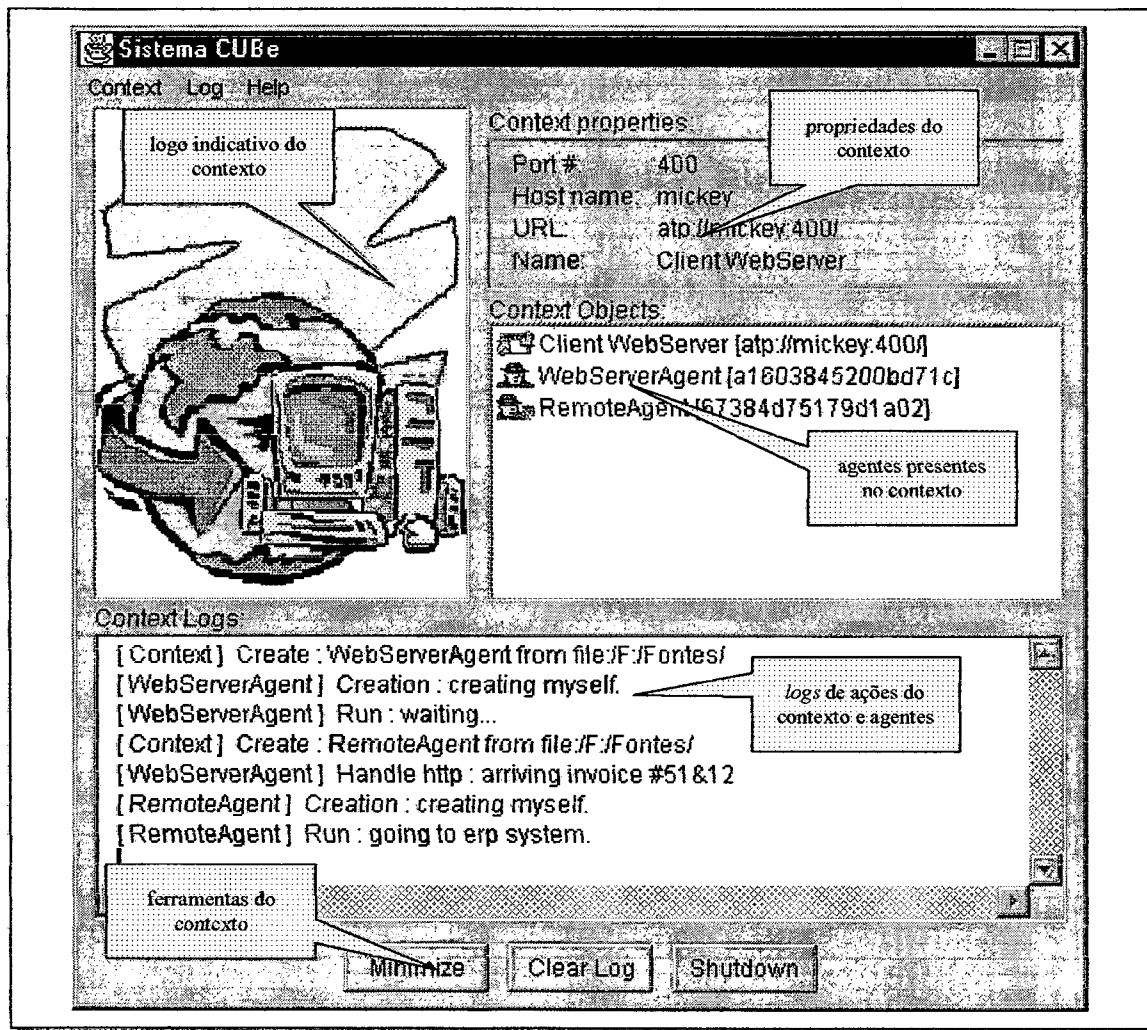
Todo agente é instanciado e realiza suas tarefas em cima de um contexto, conforme descrito anteriormente, todos os contextos do Sistema CUBe, tem o formato do contexto *ClientWebServer* (fig. 8.9), este contexto trabalha do lado cliente recebendo requisições HTTP, instanciando agentes remotos e seus respectivos pedidos e fornecendo como resposta uma página padrão HTML, com informações relativas ao pedido e ao agente, configurados nos níveis 3, 4 e 5, do Sistema de Comércio Eletrônico anteriormente definido.

Neste contexto, toda requisição HTTP é manipulada pelo agente estático *WebServerAgent*, que implementa um *listener* para este evento. Ao receber uma requisição, ele então retira as informações necessárias e instancia o pedido, representado pela classe *Invoice*, e um agente móvel, representado pela classe *RemoteAgent*, com um determinado comportamento, também definido e passado ao *WebServerAgent* na requisição HTTP.

Neste ponto, o agente remoto viaja até o contexto do Servidor ERP levando o pedido, e submete este pedido para verificação da sua aceitação ou não pelo Administrador daquele contexto, caso o pedido seja aceito, o agente entra então em um processo de negociação, baseado nos parâmetros definidos pelo usuário no Sistema de Comércio Eletrônico (desconto mínimo, desconto máximo e número máximo de propostas), retornando após o término desta negociação ao seu contexto de origem e destruindo-se logo em seguida.

Todas as ações realizadas pelos agentes (da criação à sua destruição) são gravadas pelos contextos, que possuem *listeners* específicos para cada ação realizada pelos agentes.

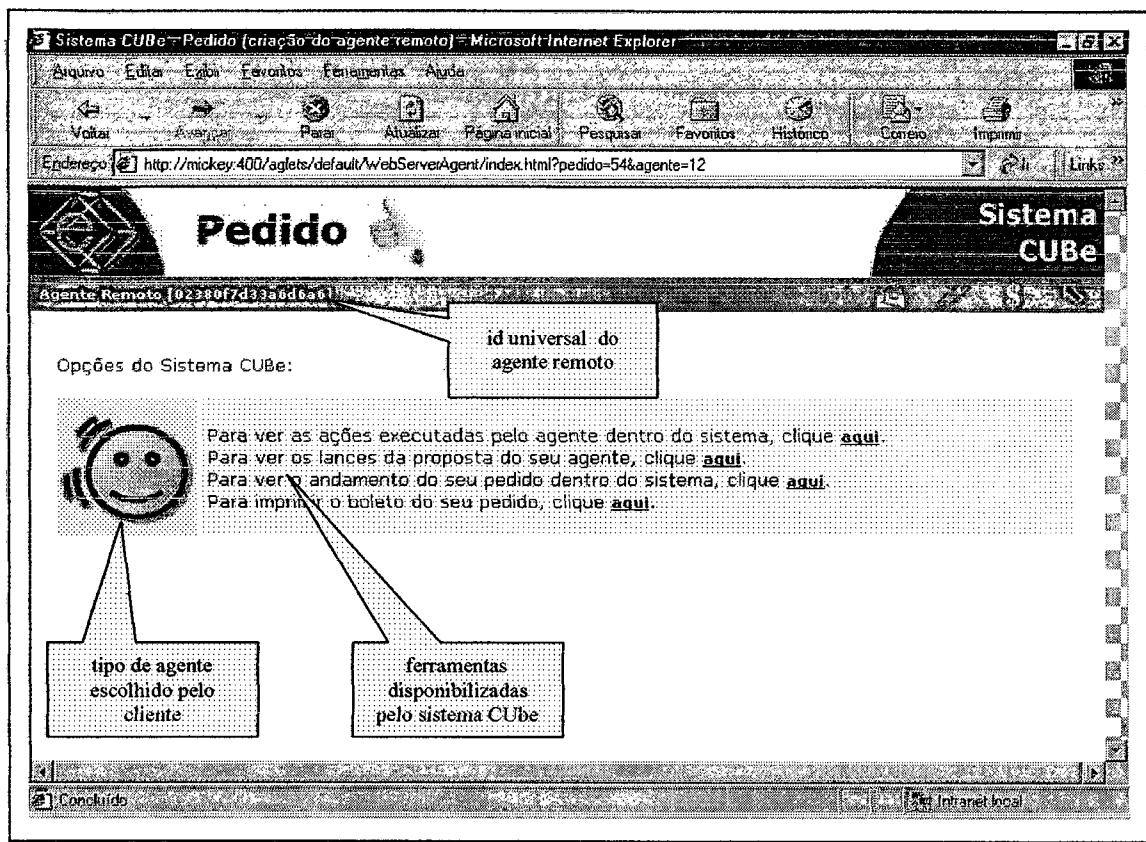
Figura 8.9 - Contexto *Client Web Server* do Sistema CUBe



Quando o *WebServerAgent* recebe uma requisição HTTP, ele instancia o Agente Remoto e cria uma página com valores do pedido e do agente definidos pelo usuário, retornando uma página HTTP, para o *browser* do solicitante, no caso, o cliente. Todas as informações geradas pelo agente: pedido, contexto, e outros dados relevantes, são gravados no banco de dados do sistema através do JDBC (*Java Database Connectivity*).

A seguir temos a fig. 8.10 que mostra a tela de resposta após a criação do pedido juntamente com o agente remoto.

Figura 8.10 - Página *index.html* Gerado pelo *WebServerAgent*



As ferramentas presentes neste página identificam quatro situações que podem ser de interesse para o usuário na consulta do seu pedido ou do agente remoto, responsável pelo pedido dentro do sistema CUBE.

8.2 Sistema ERP

No sistema ERP, existem 4 contextos: *ERPServer*, *Seller*, *Manufacturer* e *Emitter* que trabalham de forma conjunta simulando um sistema de manufatura da empresa que disponibiliza os itens no Sistema de Comércio Eletrônico.

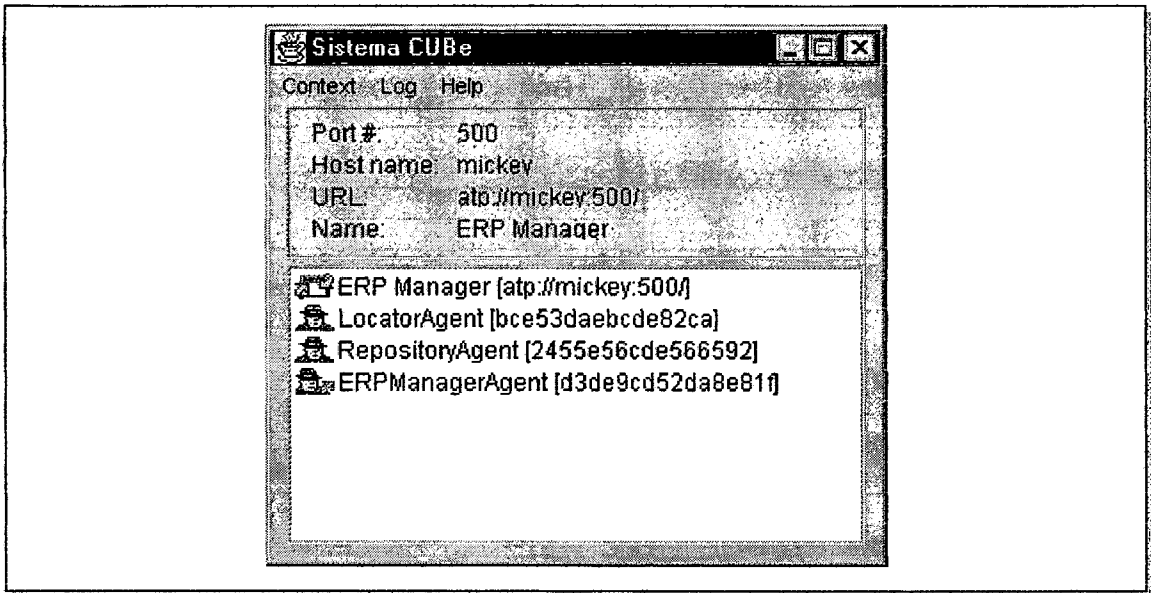
Cada contexto possui atribuições específicas, sendo que cada um deste executa tarefas anteriormente abordadas na fase de análise e projeto; além disso, geram requisições externas ao seu sistema, para complementar o pedido; dentre estas requisições pode-se citar: entrega de boletos para cobrança, requisição de entrega de pedidos ou requisição de compra de itens.

A seguir é analisado cada contexto em particular, citando-se as soluções implementadas e o fluxo de informações dentro de cada contexto.

8.2.1 Contexto *ERP Manager*

Quando o contexto *ERP Manager* é inicializado, são instanciados os agentes estáticos *LocatorAgent* e *RepositoryAgent* e o agente móvel *ERPManagerAgent*, conforme mostrado de forma minimizada na fig. 8.11.

Figura 8.11 - Contexto *ERP Manager*

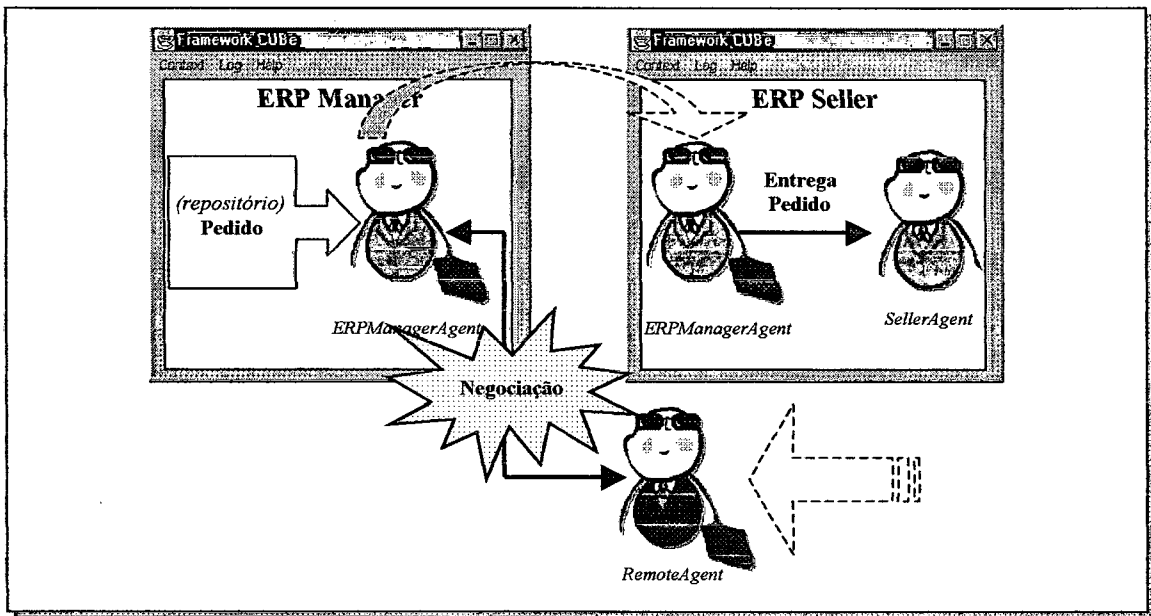


Cada agente dentro do contexto possui funcionalidades próprias, mas que operam de forma conjunta para a manutenção das tarefa previamente definidas; a seguir são descritos cada agente pertencente ao contexto *ERP Manager* em detalhes.

1. *ERPManagerAgent*: é o agente responsável em movimentar-se dentro do sistema ERP entregando pedidos, ordens, realizando negociações e outras tarefas específicas para a resolução de um pedido efetuado por um cliente. Cabe salientar aqui, que todas as requisições passadas a este agente são previamente armazenadas em um repositório, administrado pelo agente *RepositoryAgent*, explicado em detalhes mais adiante.
 - a) Como primeira tarefa pode-se citar a criação (*creation*) dos agentes que compõem o Sistema ERP e o envio (*dispatch*) deles para seus respectivos contextos.
 - b) Quando receber uma solicitação de um pedido, o agente *ERPManagerAgent*, caso esteja livre, entrará em estado de ocupado e realizará uma negociação com o agente *RemoteAgent*, que gerou a requisição; assim

como definido pelo cliente, o *ERPManagerAgent* parametrizará a negociação, utilizando-se para isso os mesmos critérios de comportamento definidos na interface *Behaviour* do Sistema CUBe, porém como estratégia, utilizará como parâmetros o número de compras já realizado pelo cliente e seu *status* (valor que identifica quantas vezes o cliente já atrasou algum pagamento) e, de acordo com estes valores adota um comportamento no ato da negociação. Depois de efetuada a negociação e o pedido tiver sido aceito, ele vai até o contexto *ERP Seller* entregar o pedido já aprovado e retorna ao seu contexto de origem (fig. 8.12).

Figura 8.12 - *ERPManagerAgent* Processando Requisição do *RemoteAgent*



- c) Após esta etapa, duas solicitações são geradas pelo agente *SellerAgent*: primeiro uma ordem de entrega de boleto de cobrança e a segunda uma ordem de venda (OV), ambas armazenadas previamente no repositório. Para a requisição de entrega de boleto, o agente *ERPManagerAgent* cria um agente *OfficeBoyAgent* com a ordem de entregar o boleto no contexto *Bank Manager*, para a solicitação de OV, ele viaja até o contexto *ERP Manufacturer* para entregar a OV, retornando em seguida ao seu contexto de origem (fig. 8.13).
- d) Duas solicitações podem ser geradas pelo agente *ManufacturerAgent*: primeiro uma ordem de compra de itens (OC) e segundo uma ordem de emissão (OE), também armazenadas previamente no repositório; para a OC, o

ERPManagerAgent cria o agente *BuyerAgent* para realizar a compra no contexto *Vendor Manager*; para a OE, o agente viaja até o contexto *ERP Emitter* e entrega esta ordem ao agente *EmitterAgent* (fig. 8.14).

Figura 8.13 - *ERPManagerAgent* Processando Requisições do *SellerAgent*

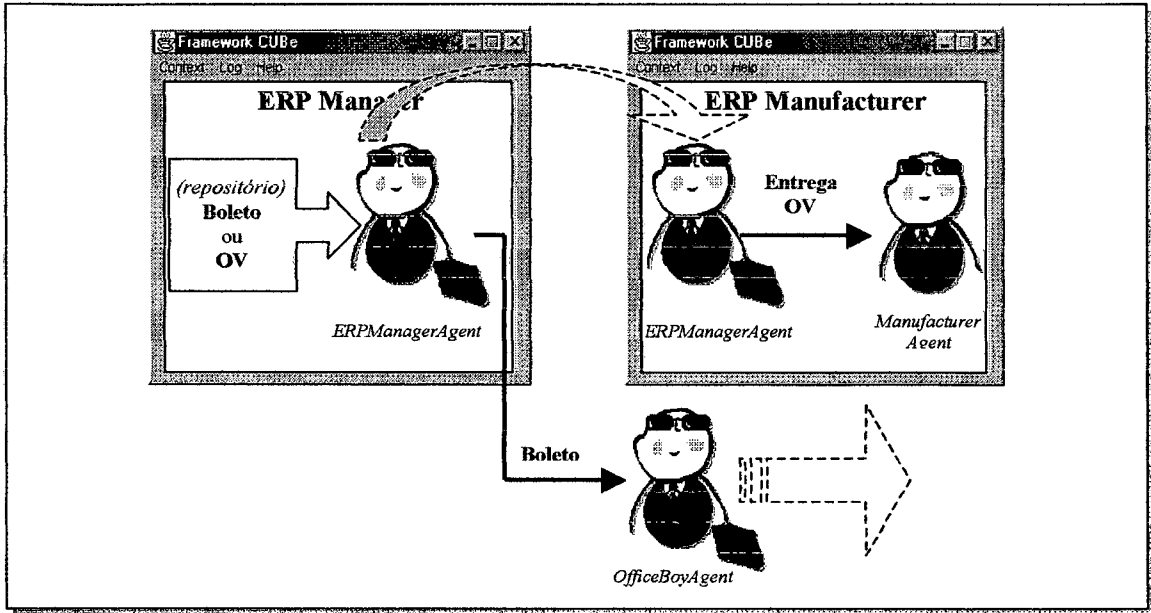
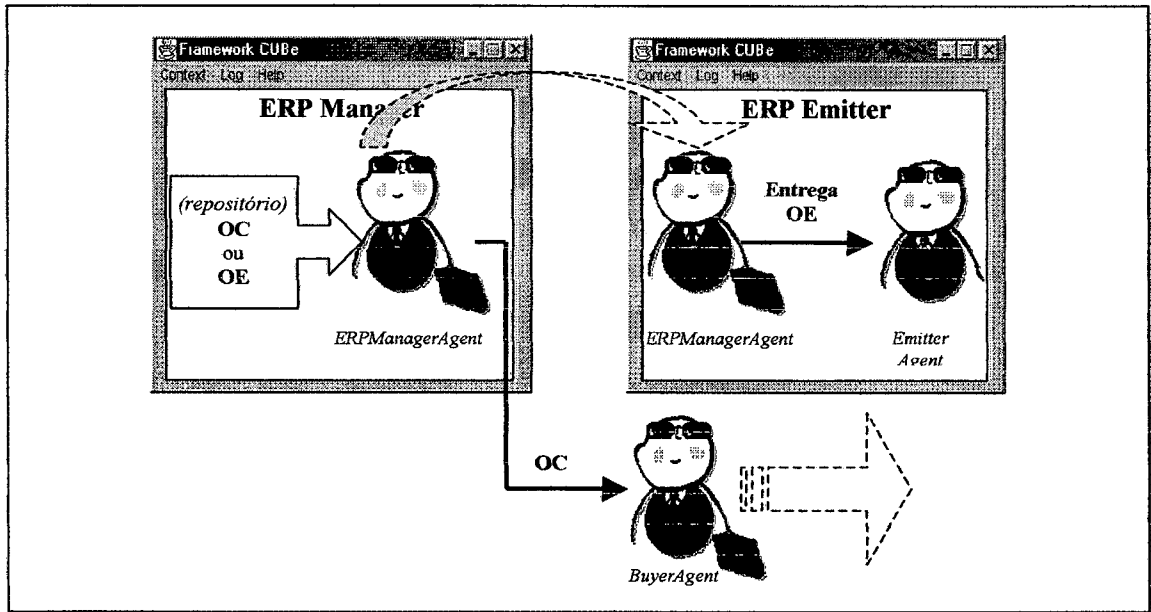


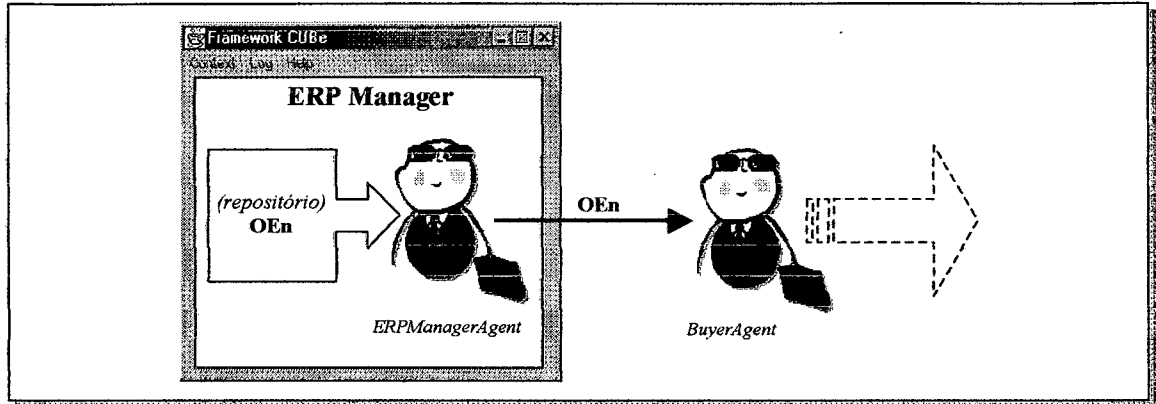
Figura 8.14 - *ERPManagerAgent* Processando Requisições do *ManufacturerAgent*



e) Do contexto *ERP Emitter* pode chegar a requisição de ordem de entrega (OEn), também armazenada previamente no repositório; para a OEn, o

ERPManagerAgent cria o agente *BuyerAgent* para realizar o agendamento da entrega no contexto *Deliverer Agent* (fig. 8.15).

Figura 8.15 - ERPManagerAgent Processando Requisições do EmitterAgent

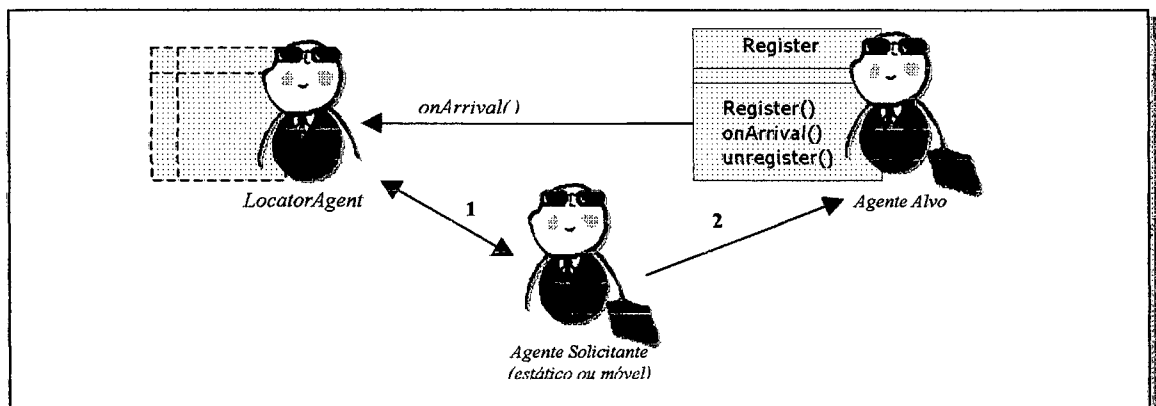


2. *LocatorAgent*: é o agente responsável em localizar um determinado agente dentro dos contextos, já que o processo de localização de agentes dentro dos contextos não é suportado pela API dos *aglets*. O agente *LocatorAgent* suporta 3 tipos de mensagens:

- a) *Where aglet*: retorna o *proxy* do *aglet* especificado.
- b) *Register aglet & proxy*: registra o *aglet* com seu endereço numa tabela.
- c) *Unregister aglet*: retira o *aglet* da tabela de posições.

E para fazer o elo de ligação entre o agente móvel e o agente *LocatorAgent*, foi definida a classe *Register* que implementa a interface *MobilityListener*; esta classe registra o novo endereço do agente (*proxy*) assim que ele chega a seu destino; mantendo com isso uma tabela atualizada; caso o agente seja destruído é retirado desta tabela (fig. 8.16).

Figura 8.16 - LocatorAgent



Desta forma um agente solicitante que necessite comunicar-se com um agente requisitado, enviará uma requisição (1) *Where agente* ao *LocatorAgent* e receberá como resposta o *proxy* do agente requisitado, caso este esteja previamente registrado, podendo desta forma comunicar-se (2) com o agente alvo.

A comunicação com o *LocatorAgent* é simplificada, visto que ao ser criado ele registra seu *proxy* como de domínio público dentro do contexto, através da função *setProperty()* da interface *AgletContext*.

Cabe salientar que este tipo de comunicação pode falhar, visto que no instante que o *LocatorAgent* está respondendo à pergunta (1) do agente solicitante, o agente requisitado pode mudar de contexto, gerando com isto uma exceção na comunicação (2), entre o agente solicitante e o agente requisitado; exceção esta que deverá ser tratada pelo agente fonte da consulta, de modo que não haja falhas de comunicação e conseqüentemente inconsistências no sistema.

3. *RepositoryAgent*: é o agente responsável em armazenar todas as mensagens enviadas para o *ERPManagerAgent*, devido ao fato deste agente ser móvel e com isto podendo ocasionar falhas de comunicação, conforme descrito anteriormente no agente *LocatorAgent*.

Com o agente *RepositoryAgent* todas as mensagens (*Message*) são enviadas de forma assíncrona a ele (*sendOnewayMessage()*) que irá armazenar estas mensagens em uma fila FIFO (*First In First Out*); após esta etapa, através do agente *LocatorAgent* enviará uma outra mensagem informando ao agente *ERPManagerAgent* que uma solicitação acabou de chegar.

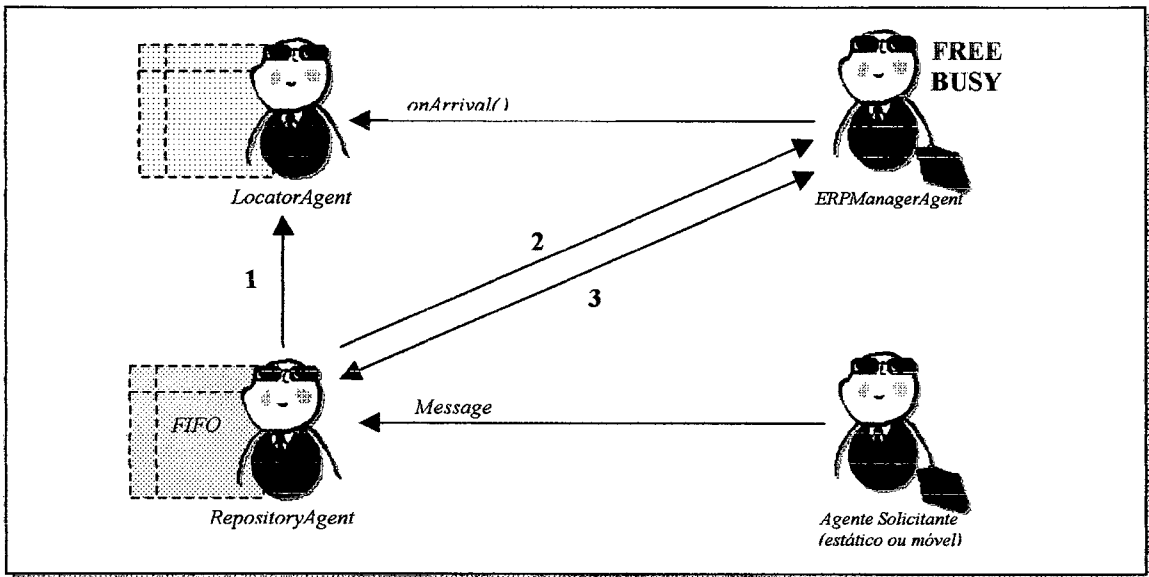
Caso o agente *ERPManagerAgent* esteja livre, ele então solicitará uma mensagem para retirar a solicitação da fila de espera, entrando em estado de ocupado e resolvendo a mesma. Assim que o agente *ERPManagerAgent* terminar alguma tarefa, perguntará ao agente *RepositoryAgent* se existe solicitações pendentes, e este responderá negativamente ou enviando a solicitação como resposta, repetindo-se o processo até que todas as solicitações estejam resolvidas.

Caso haja inconsistência na comunicação, conforme explicado no item 2, o agente *RepositoryAgent* simplesmente ignorará a exceção, tomando-se como

dedução que o agente *ERPManagerAgent* está ocupado trafegando pelos contextos do sistema ERP. Cabe salientar aqui, que um processo de tratamento de exceções mais apurado será discutido mais adiante nos trabalhos futuros.

A fig. 8.17, representa este processo de armazenamento de mensagens e o devido tratamento dado pelo agente *RepositoryAgent*.

Figura 8.17 - *RepositoryAgent*



Na figura acima, vemos o Agente Solicitante enviar uma mensagem, que será armazenada pelo agente *RepositoryAgent*; este último agente, logo em seguida, solicitará (1) ao agente *LocatorAgent* a posição atual (*proxy*) do agente *ERPManagerAgent*, enviando logo em seguida a informação de solicitações em espera (2); caso o agente esteja livre, é então estabelecido um contato por parte do agente *ERPManagerAgent* requisitando a solicitação (3) a ser tratada (respeitando a *FIFO*), caso o agente *ERPManagerAgent* esteja ocupado, nada é realizado.

Assim como o *LocatorAgent*, o agente *RepositoryAgent* registra seu *proxy* como de domínio público no contexto onde foi inicializado.

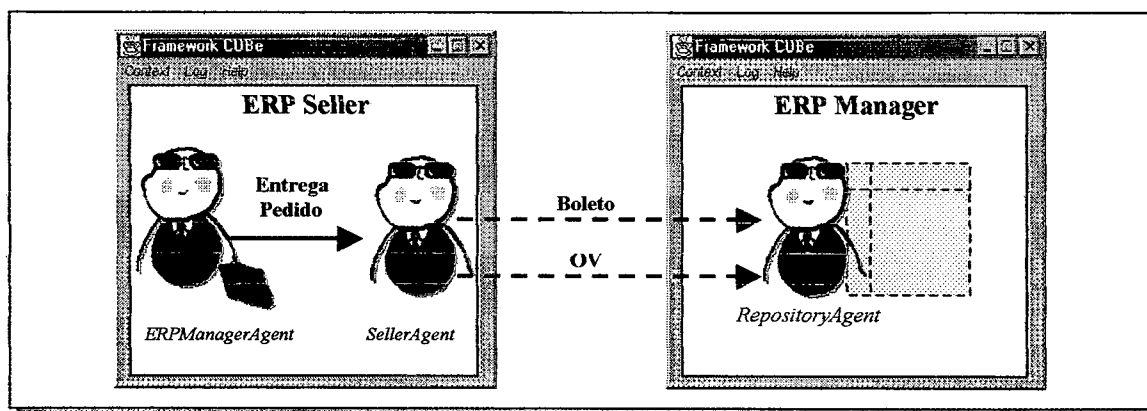
8.2.2 Contexto *ERP Seller*

O contexto *ERP Seller* recebe o agente estático *SellerAgent* que é despachado pelo agente *ERPManagerAgent* e recebe também este último quando da entrega de um pedido negociado e aprovado previamente.

O agente *SellerAgent* é criado no contexto *ERP Manager*, momento no qual seta alguns parâmetros iniciais, dentre eles o *proxy* do agente *RepositoryAgent*, necessário para a comunicação com o *ERPManagerAgent* para envio de requisições e, é então, despachado para o seu devido contexto.

Quando o agente *ERPManagerAgent* entra no seu contexto, recebe deste, através de uma mensagem um objeto *Invoice* (pedido do cliente), abre então uma ordem interna (*IOrder*) e cria um boleto (*Receipt*), retornando uma mensagem ao agente *RepositoryAgent* para envio deste boleto ao banco e outra mensagem para criação de uma ordem de venda (fig.8.18).

Figura 8.18 - SellerAgent Processando Requisição do ERPManagerAgent



8.2.3 Contexto *ERP Manufacturer*

O contexto *ERP Seller* recebe o agente estático *ManufacturerAgent* que é despachado pelo agente *ERPManagerAgent* e recebe também este último quando da entrega de uma ordem de venda (OV). Existem 3 mensagens que podem ser enviadas ao agente *ManufacturerAgent*, descritas a seguir.

1. Quando a solicitação chega, o agente *ManufacturerAgent* irá então, verificar no estoque a quantidade dos itens solicitados; o estoque baseia-se em três estados distintos, conforme detalhado na tabela 8.2.
 - a) Caso a quantidade de produtos prontos solicitados na OV seja suficiente, estes valores são colocados em reserva, o agente *ManufacturerAgent* emite então uma ordem de emissão (OE) deste pedido, entregando-a ao *RepositoryAgent*.

Tabela 8.2 - Estados do Estoque de Itens no Sistema CUBe

| Estado | Descrição |
|-------------------|--|
| <i>Quantidade</i> | Valor referente à quantidade disponível para comercialização, no caso de produtos prontos ou para produção, no caso de itens manufaturáveis. |
| <i>Reservado</i> | Valor referente à quantidade disponível para entrega (à uma transportadora) no caso de produtos prontos ou disponíveis em estoque para produção, no caso itens manufaturáveis. |
| <i>Débito</i> | Valor referente à quantidade faltante, tanto de produtos prontos, quanto de itens manufaturáveis. |

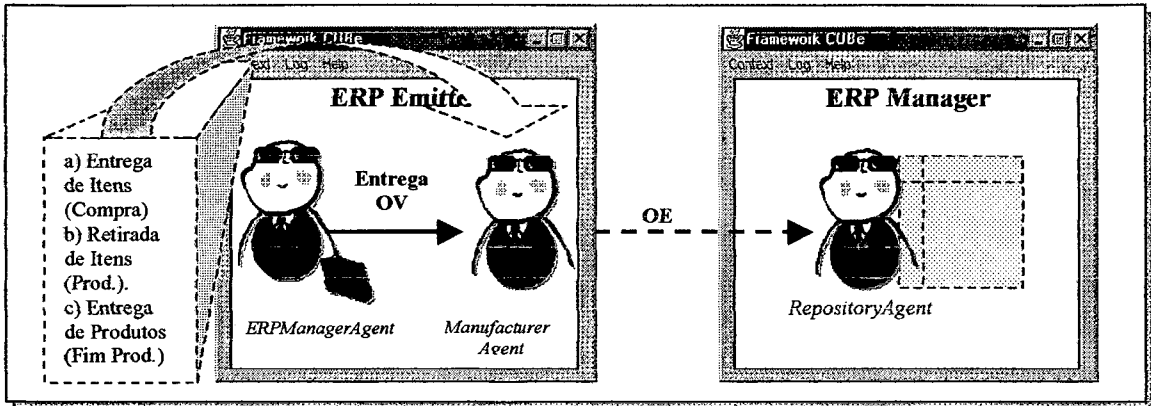
b) Por outro lado, caso algum dos produtos solicitados não seja suficiente, porém as peças que compõem este produto estejam disponíveis em estoque, o agente *ManufacturerAgent* coloca a quantidade de produtos e peças em reserva e emite uma ordem de produção (OP) para ser atendida por algum funcionário do setor.

c) E finalmente, caso os produtos nem as peças sejam suficientes para atender à OV, o agente *ManufacturerAgent* emite então uma ordem de compra (OC – classe *EOrder*) solicitando a quantidade de itens em falta e colocando os devidos valores em reserva e em débito no estoque, enviando então esta solicitação ao *RepositoryAgent*.

2. Quando uma ordem de compra (OC) entra no sistema (manipulada por algum funcionário responsável pela recepção e conferência dos itens da compra), o contexto envia uma mensagem usando a função (*sendMulticastMessage*), e o agente *ManufacturerAgent* que está registrado para receber esta mensagem, processa então o recebimento da OC e estoca as peças, liberando uma ordem de produção (OP), conforme descrito no item (b) anteriormente.
3. Quando uma ordem de produção (OP) é finalizada (também manipulada por algum funcionário do setor), o agente *ManufacturerAgent* irá receber do contexto uma mensagem e irá proceder conforme descrito no item (c) anteriormente explicado.

Em relação à comunicação com o agente *RepositoryAgent* para envio de suas requisições, o procedimento é similar ao do agente *SellerAgent*, explicado anteriormente. A fig. 8.19 a seguir, exemplifica todas as requisições manipuladas pelo agente *ManufacturerAgent*, citadas acima.

Figura 8.19 - *ManufacturerAgent* Processando Requisições



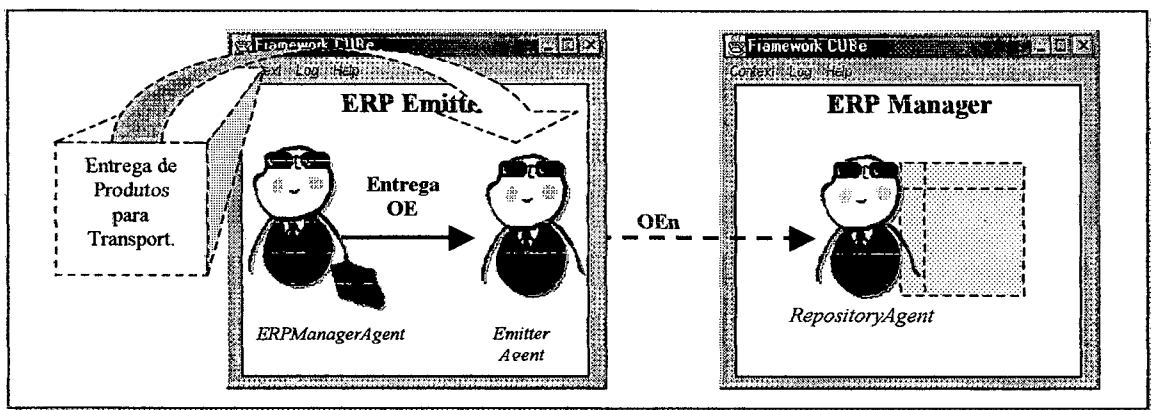
8.2.4 Contexto *ERP Emitter*

O contexto *ERP Emitter* recebe o agente estático *EmitterAgent* que é despachado pelo agente *ERPManagerAgent* e recebe também este último quando da entrega de uma ordem de emissão OE.

Ao receber a requisição de OE, o agente *EmitterAgent*, gera uma nota fiscal (classe *Bill*) e despacha uma requisição de ordem de entrega ao agente *RepositoryAgent*.

Também pode receber uma mensagem do contexto indicando que algum funcionário do setor está entregando o pedido, momento no qual o agente *EmitterAgent* retira os produtos do estoque (produtos em reserva), finalizando o pedido e suas ordens internas/externas dentro do Sistema CUBE (fig. 8.20).

Figura 8.20 - *EmitterAgent* Processando Requisições



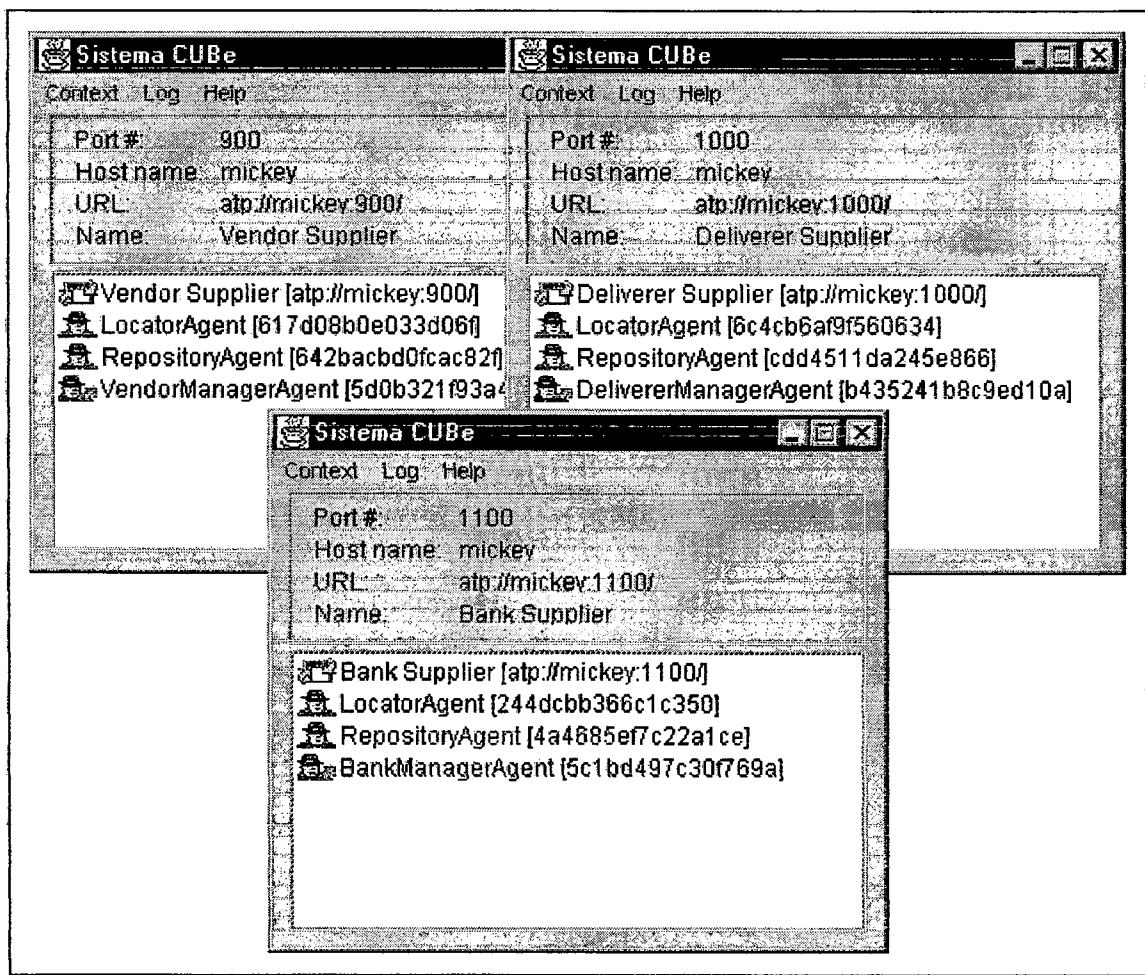
Em todas as etapas descritas anteriormente, da entrada à saída, o *status* do pedido é atualizado na base de dados do sistema, o que possibilita ao usuário fazer um

mapeamento do mesmo, obtendo informações *on-line* sobre a sua situação dentro do Sistema ERP.

8.3 Sistemas dos Fornecedores

No sistema ERP, existem 3 contextos que representam respectivamente o Fornecedor Banco (*Bank Supplier*), o Fornecedor Fabricante (*Vendor Supplier*) e o Fornecedor Transportadora (*Deliverer Supplier*), cada um possuindo seu sistema particular e atuando em conjunto com o Sistema ERP para a resolução de cobrança de pedidos, compra de itens em falta e agendamento de entrega (fig. 8.21), respectivamente.

Figura 8.21 - Contextos dos Fornecedores

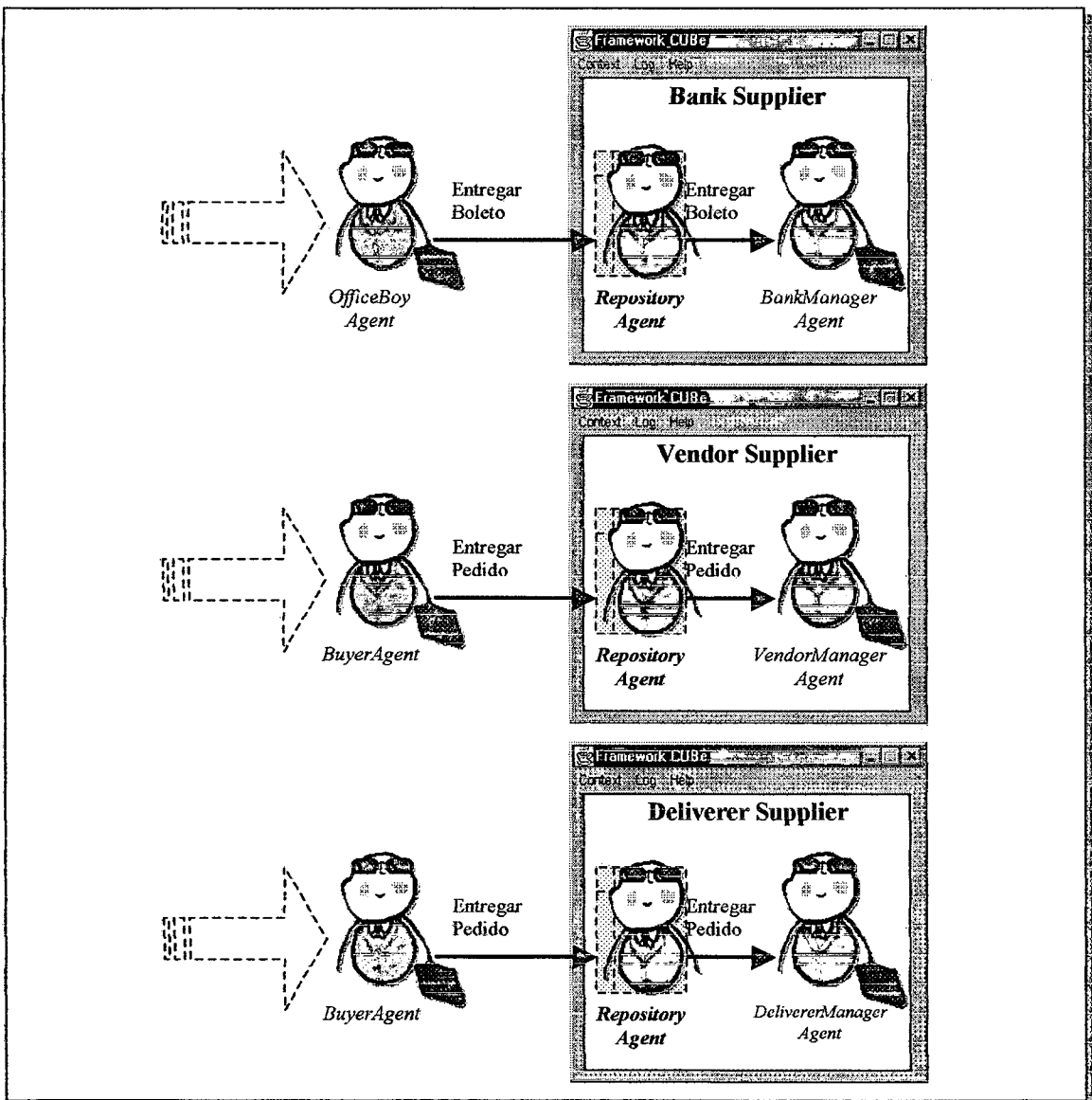


O objetivo inicial destes contextos é de forma simples e resumida serem apenas receptores dos pedidos enviados aos seus repositórios, não havendo nenhum tratamento

para estas requisições por parte de seus agentes administradores; isto deve-se principalmente ao fato que cada fornecedor aqui apresentado possui características próprias e peculiaridades do sistema que não trazem nenhum acréscimo ao Sistema CUBe já implementado, ao contrário, iriam aumentar demasiadamente a complexidade do projeto proposto e a sua implementação.

Cada contexto fornecedor, possui uma estrutura similar à do contexto *ERP Manager*, ou seja, cada requisição gerada é enviado ao seu repositório, e não diretamente ao administrador (fig. 8.22).

Figura 8.22 - Fornecedores Processando Requisições dos Agentes Remotos do Sistema ERP



9. Trabalhos Futuros

Durante a fase de análise, projeto e implementação do Sistema CUBe, várias propostas surgiram e são descritas a seguir como possíveis trabalhos complementares a este projeto inicial.

Tolerância a Faltas: é um requisito hoje importante em qualquer sistema que necessite de robustez e principalmente em sistemas críticos. No projeto CUBe, apenas algumas questões sobre Tolerância a Faltas foram abordadas, principalmente na comunicação entre agentes, como proposto e implementado nos agentes *LocatorAgent*, *RepositoryAgent* e na classe *Register*, conforme já descrito. Um trabalho complementar que abordasse outros tipos de Tolerância a Faltas, principalmente no que tange ao agente e ao contexto seria de suma importância.

Segurança: também seria um outro aspecto a ser considerado, visto que este tópico não foi tratado no projeto CUBe; mecanismos de segurança implementados a nível de agente (comunicação, trânsito, dentre outros), a nível de contexto (validação, criptografia, dentre outros) e também a nível do Comércio Eletrônico seriam três pontos centrais deste possível trabalho.

Banco de Dados: a questão do banco de dados, utilizado amplamente pelo sistema também seria um ponto a ser focado; um banco de dados distribuído que liberasse o agente do seu contexto inicial, evitando os mecanismos de acesso utilizados atualmente como ODBC e JDBC, seria outro ponto interessante para distribuição dos agentes de um mesmo sistema sobre uma *extranet* ou mesmo na *Internet*.

Colaboração: a questão da colaboração também seria outro ponto importante, visto que neste primeiro trabalho um Agente Remoto trabalha exclusivamente com requisições para uma única empresa. Mecanismos de colaboração para divisão de tarefas, dentre elas: pesquisa de preço, pesquisa de mercado e negociação; e mecanismos gerenciadores destas questões, seriam um trabalho futuro que incrementaria ainda mais o projeto.

Inteligência Artificial: como última proposta, pode-se citar a questão da inclusão de Inteligência Artificial através de ferramentas como KQML; possibilitando aos agentes do Sistema CUBe, aprofundarem-se ainda mais em aspectos inerentes aos agentes como: aprendizagem, negociação, colaboração, tomada de decisões, dentre

outros; atualmente muitos destes aspectos estão parametrizados dentro do Sistema CUBe; em relação à negociação, ferramentas que incluam a possibilidade de escolha de comportamentos e estratégias na forma escrita ou através de comandos seria um acréscimo a este trabalho inicial.

10. Conclusões

Esta dissertação descreve a análise, o projeto e a implementação do sistema chamado CUBe, para integração de diversas tecnologias, ligadas principalmente ao Comércio Eletrônico e aos Sistemas ERP. As ferramentas, arquitetura e soluções encontradas na implementação deste sistema também são descritas.

A tecnologia de agentes móveis, através dos *Aglets* (LANGE, 1998), forneceu a necessária interoperabilidade ao sistema.

A integração do Comércio Eletrônico com o ERP foi conseguido com a criação do contexto *Client Web Server*, onde o agente estático (*WebServerAgent*) implementa um *listener* de requisições *HTTP*.

Também conseguiu-se no projeto do Sistema CUBe, uma extensão às soluções já apresentadas, em especial aquelas vistas e discutidas no servidor de leilões AuctionBot (<http://auction.eecs.umich.edu>) e no *framework* VMarket (<http://harper.les.inf.puc-rio.br/vbookmarket>), implementando outras características do modelo CBB (GUTTMAN, 1998a) e garantindo a integração de vários sistemas, discutidos no capítulo 6, através da mediação dos agentes móveis.

O suporte a múltiplos produtos, tanto no Sistema de Comércio Eletrônico quanto no Sistema ERP, foi garantido na fase de análise e projeto e implementado no sistema final; conseguiu-se, também, parametrizar várias estratégias e comportamentos, fazendo com que os agentes do sistema possuíssem múltiplos processos de negociação, como estudado em (WURMAN, 1998, RIPPER, 2000). A implementação de várias etapas do modelo CBB (GUTTMAN, 1998a) foi garantida pelos procedimentos de negociação do agente *RemoteAgent* com o *ERPManagerAgent*, além de decisões e suporte presentes nos outros agentes internos ao sistema ERP.

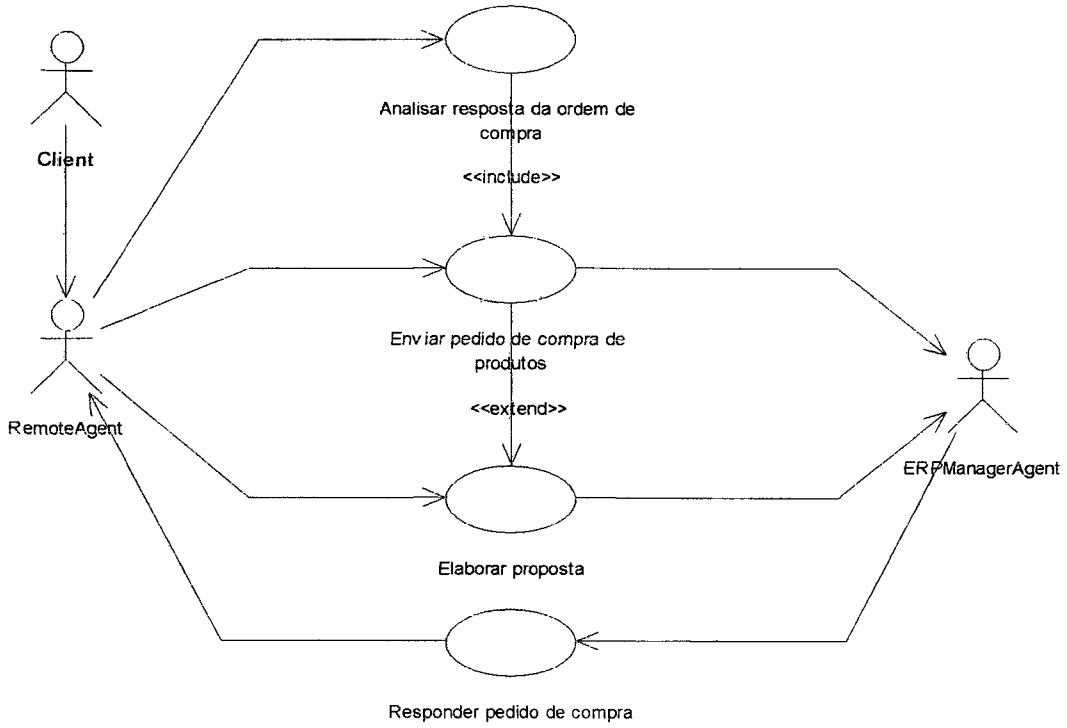
A adoção da UML (*Unified Modeling Language*) como linguagem de modelagem do sistema CUBe, teve como objetivos, dentre outras coisas, facilitar a continuidade deste projeto e possivelmente a sua expansão.

A crescente importância dos agentes móveis aliada a já difundida linguagem Java, está trazendo muitos benefícios aos Sistemas Distribuídos em geral. O Sistema CUBe é apenas uma solução inicial de um destes aspectos. Com a sua estrutura atual, e a

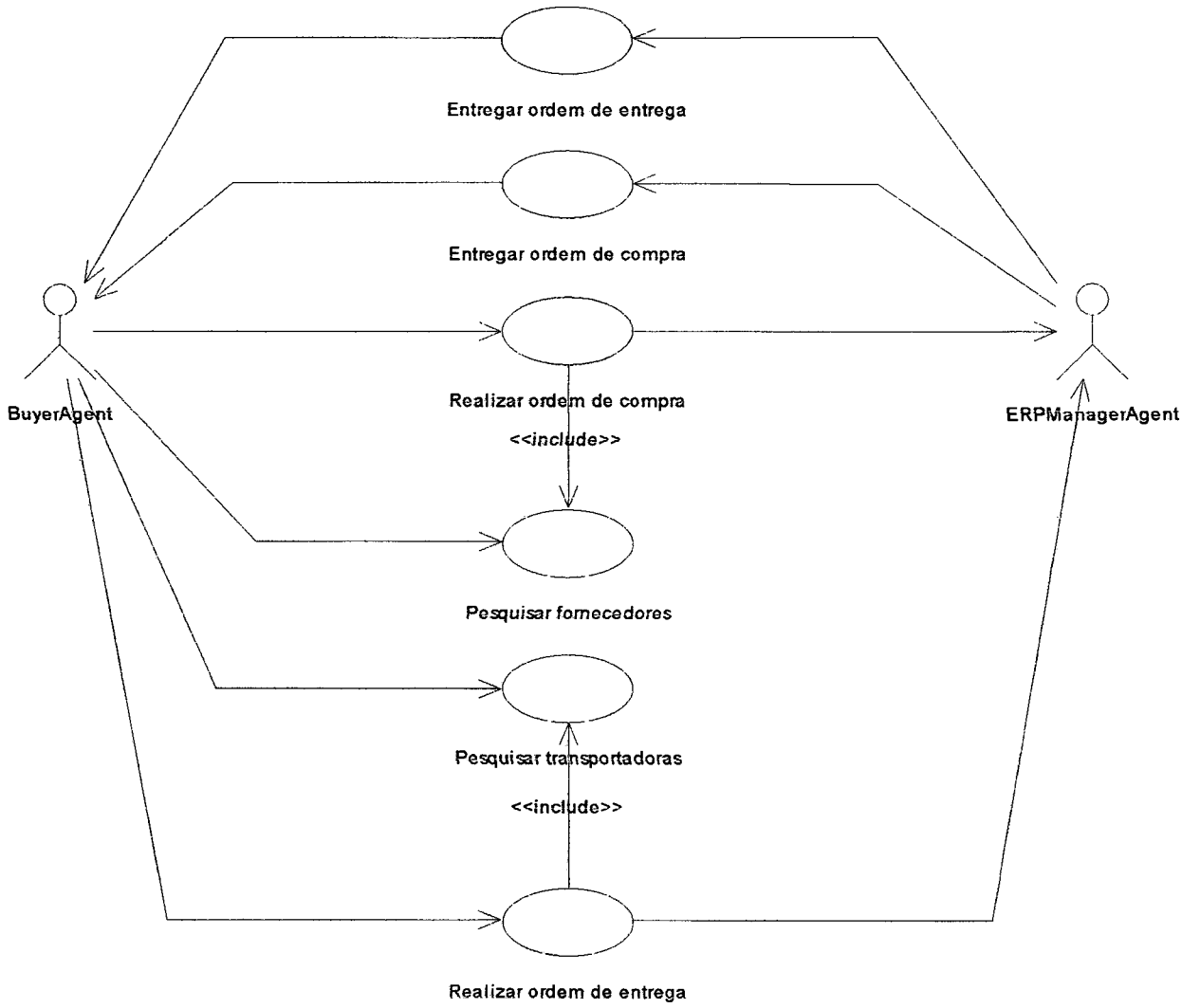
inclusão de alguns trabalhos discutidos no capítulo 9, este projeto inicial traz benefícios e soluções a muitos problemas encontrados hoje em dia.

11. Anexos

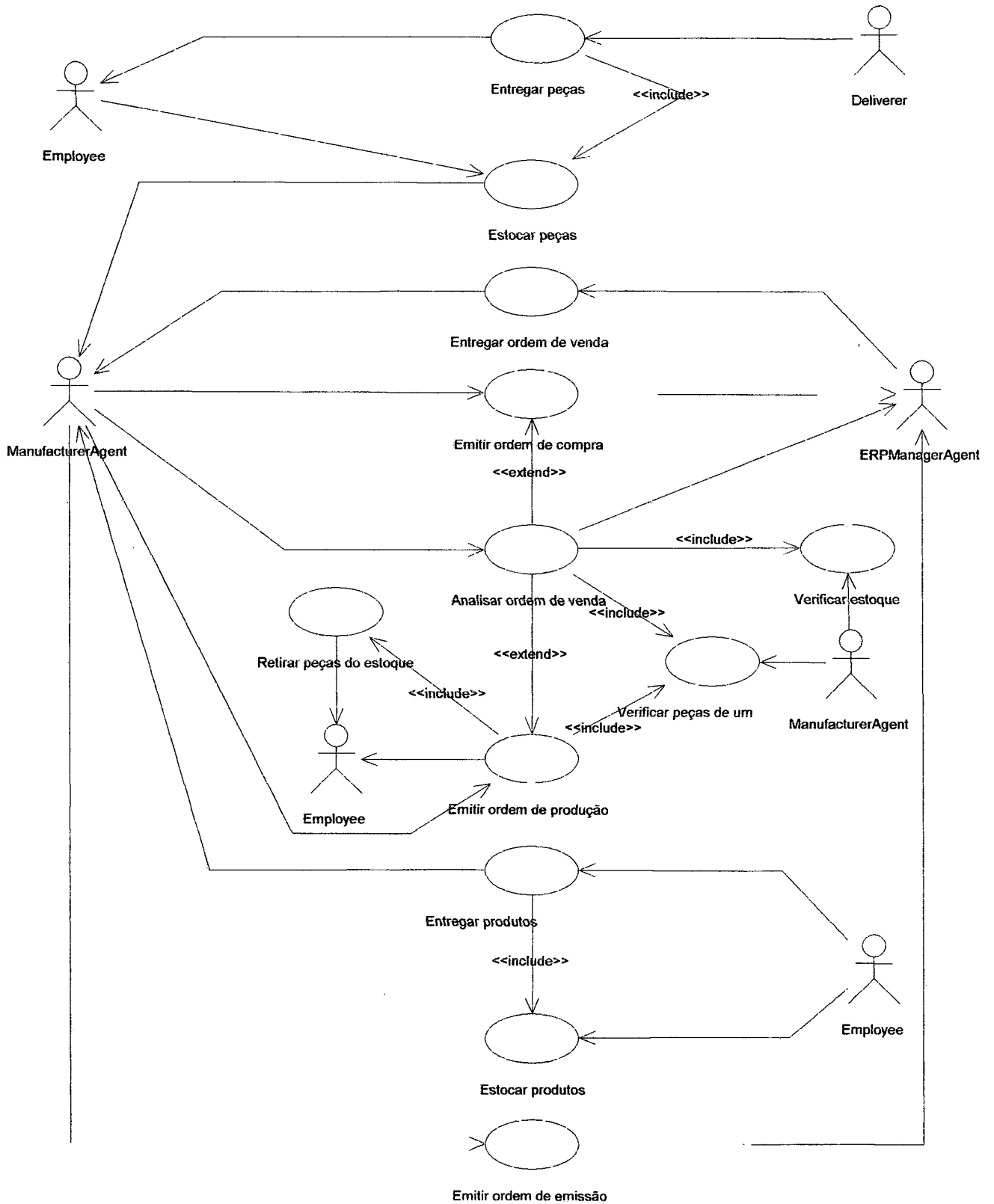
Anexo 1 - Diagrama Use Case: Principal



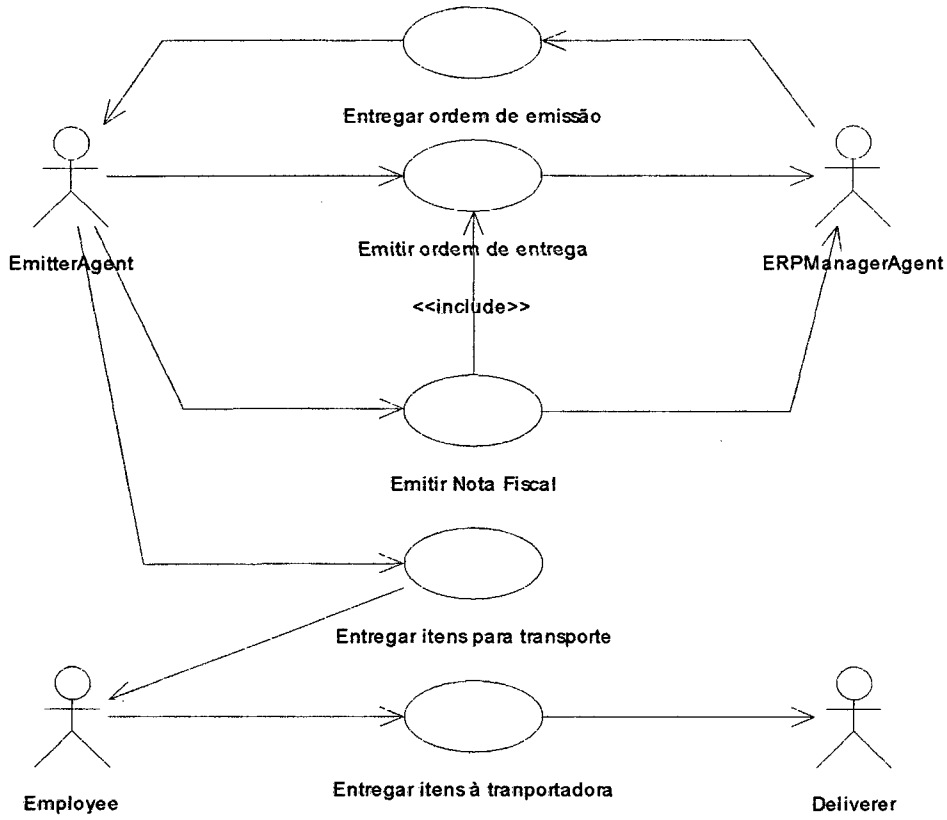
Anexo 2 - Diagrama Use Case: ERP - Compra



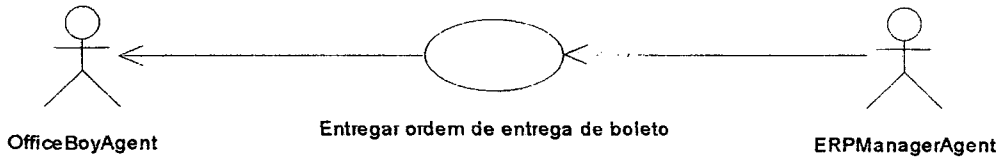
Anexo 3 - Diagrama Use Case: ERP - Produção



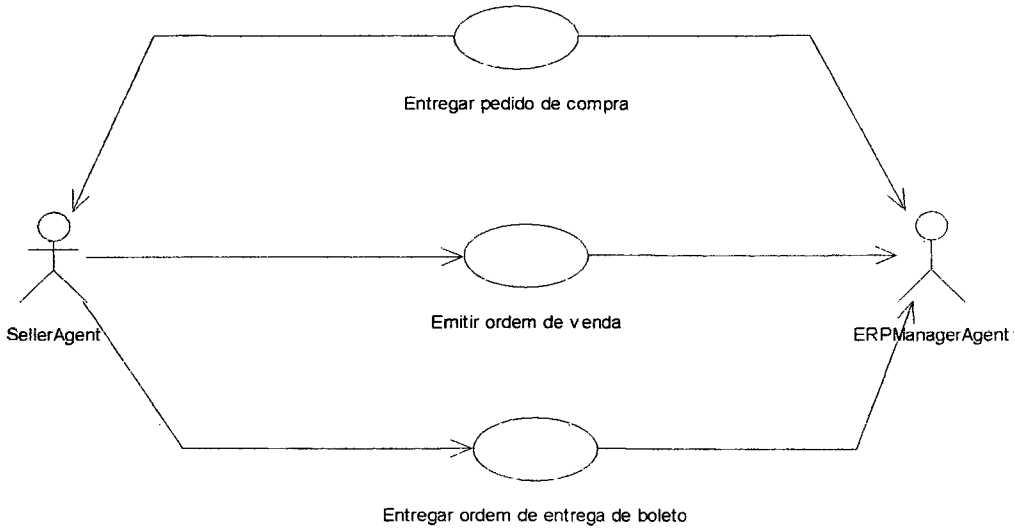
Anexo 4 - Diagrama Use Case: ERP - Emissão



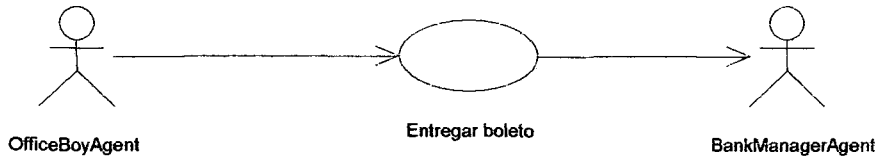
Anexo 5 - Diagrama Use Case: ERP - Externo



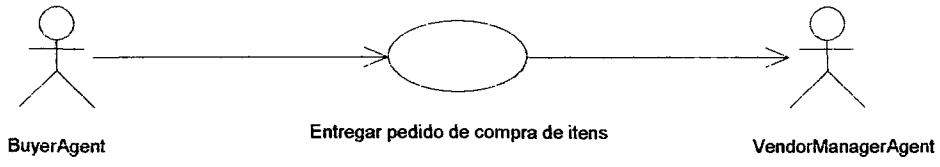
Anexo 6 - Diagrama Use Case: ERP - Venda



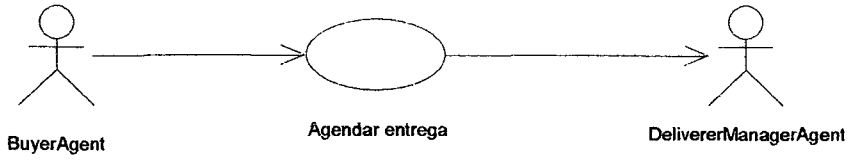
Anexo 7 - Diagrama Use Case: Proveedor - Banco



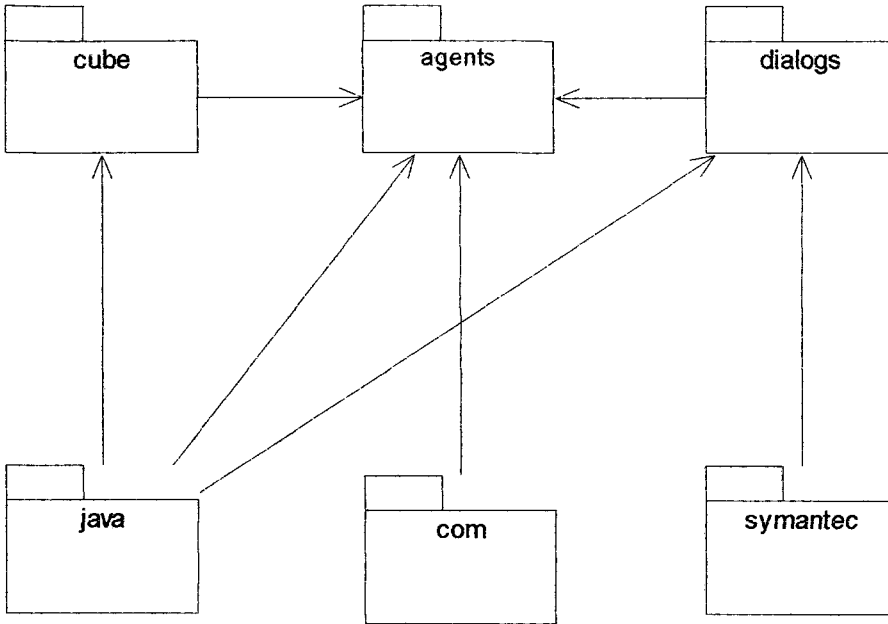
Anexo 8 - Diagrama Use Case: Proveedor - Fornecedor



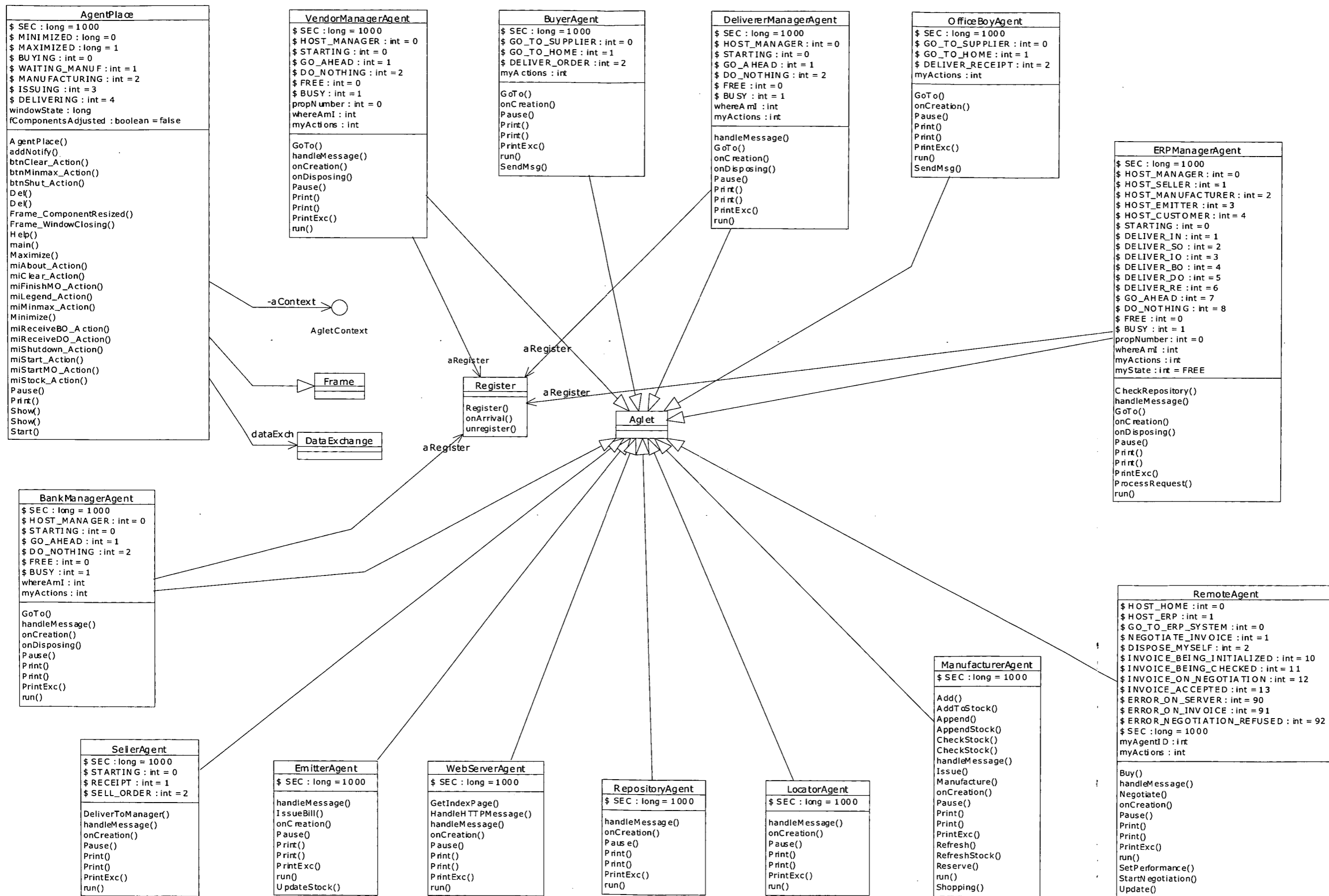
Anexo 9 - Diagrama Use Case: Proveedor - Transportadora



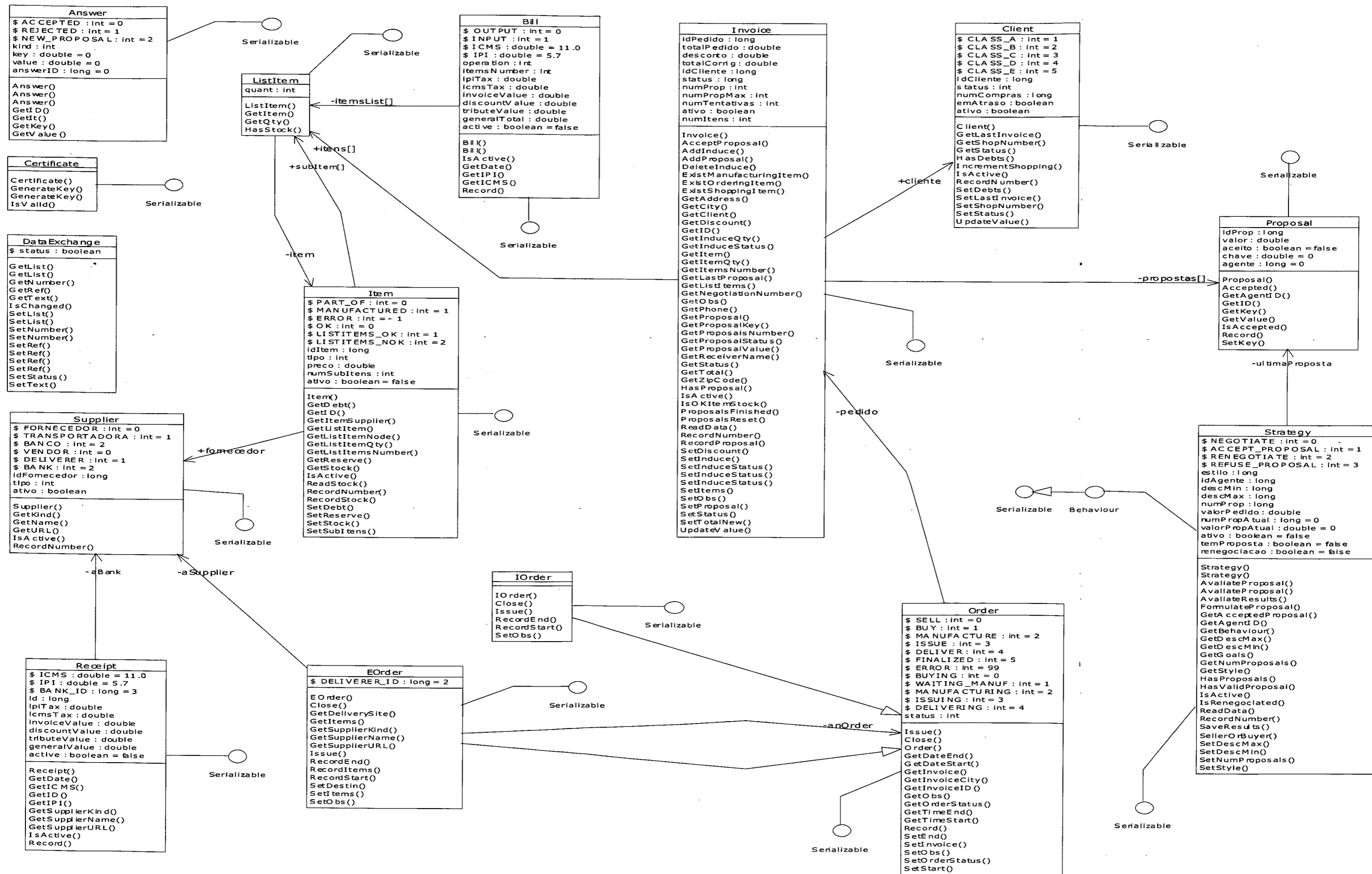
Anexo 10 - Logical View - Arquitetura do Sistema



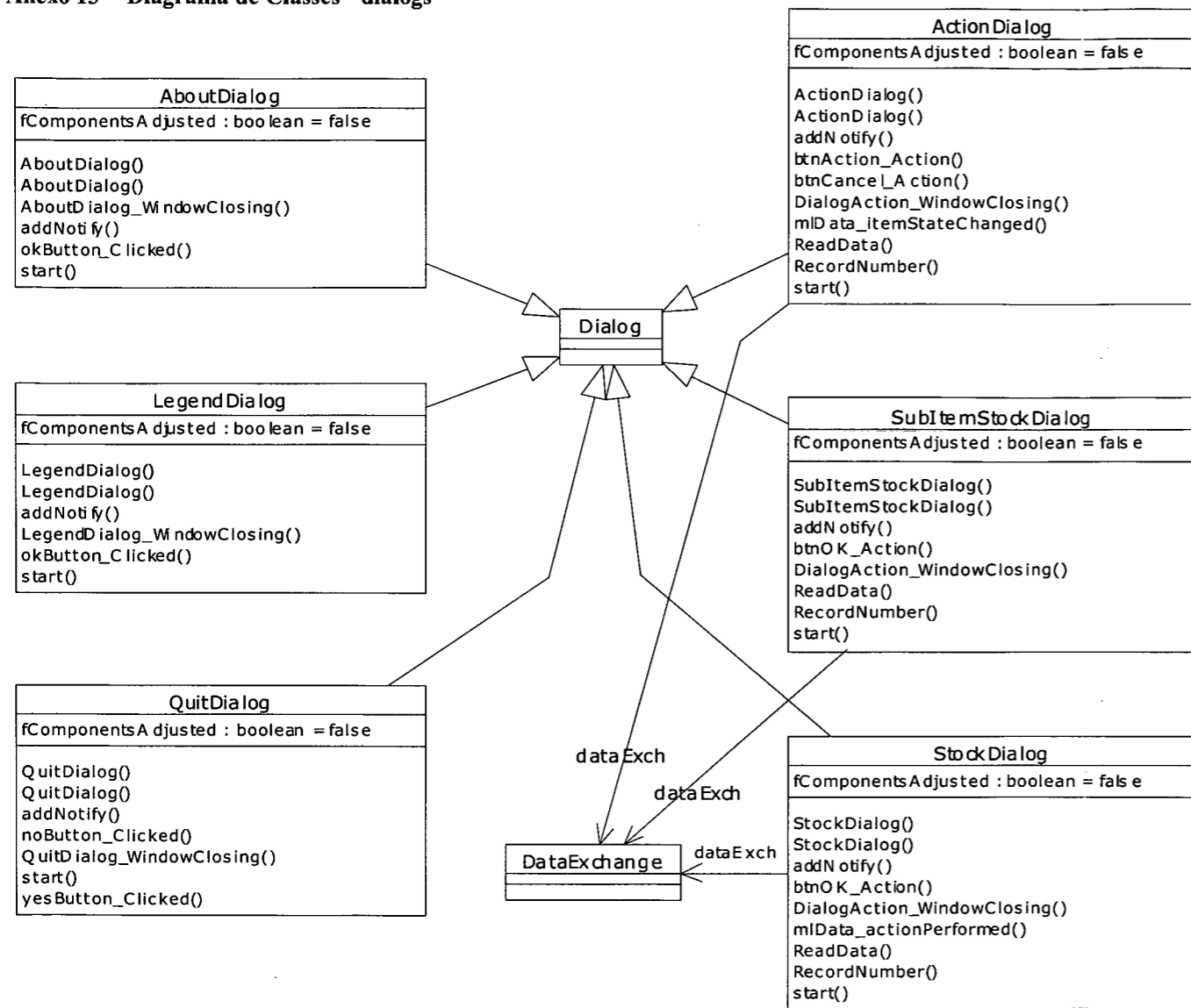
Anexo 11 - Diagrama de Classes - agents



Anexo 12 - Diagrama de Classes - cube



Anexo 13 - Diagrama de Classes - dialogs



12. Glossário

| | |
|-----------------------------|--|
| Activation | Ativação. |
| 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-customer | Transação comercial entre empresas e clientes. |
| Bytecodes | Arquivo pré-compilado gerado após a compilação de um programa em Java; portátil para vários sistemas operacionais e diferentes arquiteturas de computadores. |
| Casting | Mudança de tipo, podendo ocasionar perda de valor ou significado. |
| Cloning | Clonagem; duplicação exata de um original. |
| Commodities | Artigos ou objetos de utilidade; mercadorias. |
| Creation | Criação. |
| Data mining | Extração de dados; aquisição de dados. |
| Data-sheets | Pasta de dados. |
| Deactivation | Desativação. |
| Demos | Programas de demonstração; demonstração de algo. |
| 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. |
| Extranet | É a conexão de uma <i>Intranet</i> a outra utilizando-se para isso a <i>Internet</i> . |
| Folders | Folhetos; pastas com papéis. |
| 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. |

| | |
|--------------------|---|
| Groupware | Programas aplicativos cujo objetivo é auxiliar grupos de pessoas trabalhando cooperativamente. |
| Handle | Driver ou manipulador; parte do sistema operacional ou programa que controla um periférico. |
| Hardware | Unidades físicas, componentes, circuitos integrados, discos e mecanismos que compõem um computador ou seus periféricos. |
| Home-page | Página eletrônica com informações diversas e acessível a qualquer pessoa que esteja conectada à <i>Internet</i> . |
| 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 <i>Internet</i> , porém para troca de informações em um ambiente fechado e com um número pequeno de computadores. |
| Know-how | Conhecimento prático; habilidade. |
| Link | Conexão; meio de contato. |
| 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. |

| | |
|----------------------------|---|
| <i>QueryString</i> | Sequência de texto utilizada para consultas e/ou passagem de valores através de requisições HTTP. |
| <i>Reply set</i> | Conjunto de resposta. |
| <i>Retracting</i> | Retração. |
| <i>Runtime</i> | 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 um programa; (iii) (operação) executada apenas quando um programa está sendo executado |
| <i>Site</i> | Endereço onde pode-se localizar uma determinada <i>home-page</i> . |
| <i>Software</i> | 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. |
| <i>Status</i> | Estado; importância ou posição. |
| <i>String</i> | Cadeia ou sequência; quaisquer séries de caracteres alfanuméricos ou palavras consecutivas que são manipuladas e tratadas como uma unidade pelo computador. |
| <i>Supply-chain</i> | Cadeia de abastecimento. |
| <i>Thread</i> | Em cadeia; programa que consiste de várias seções menores independentes. |
| <i>Use cases</i> | Casos de uso; regras de negócio. |
| <i>Web</i> | Teia; também representa a <i>Internet</i> (WWW – <i>World Wide Web</i> ou Larga Teia Mundial). |
| <i>Web browser</i> | Programa navegador; utilizado na <i>Internet</i> para visualização das <i>Home-Pages</i> . |
| <i>Web server</i> | Servidor de <i>Internet</i> ; utilizado para prover acesso de clientes a páginas (<i>home-pages</i>) e outros serviços disponíveis na Internet. |
| <i>Workflow</i> | Fluxo de trabalho; programa que controla o fluxo de informações e trabalho de um grupo de usuários. |

13. Referências Bibliográficas

- BUENO, Francisco da Silveira. **Minidicionário da Língua Portuguesa**. Ed. rev. e atual. por Helena Bonito C. Pereira, Rena Signer. São Paulo: FTD: LISA, 1996.
- CHAVES, A.; MAES, P. **Kasbah: An Agent Marketplace for Buying and Selling Goods**. London, UK, Abril 1996.
- CHAVES, A. et al. **A Real Life Experiment in Creating an Agent Marketplace**. Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97). London, UK, Abril 1997.
- CORRÊA, Henrique L.; GIANESI, Irineu G. N. **Just in time, MRP II e OPT: Um enfoque estratégico**. São Paulo: Editora Atlas Ltda, 1993.
- DAVENPORT, Thomas H. **Putting the Enterprise into the Enterprise System**. EUA: Harvard Business Review, Julho/Agosto 1998, pág. 124.
- DRUCKER, Peter. **O Futuro já Chegou**. Exame, ano 34, edição 6, pág. 112-126, 22 Março 2000.
- FRANKLIN, Stan.; GRASSER, Art. **Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents**. Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Berlin, Springer-Verlag, 1996.
- GUTTMAN, R. H.; MAES, P. **Agent-mediated Integrative Negotiation for Retail Electronic Commerce**. Proceedings of the Workshop on Agent Mediated Electronic Trading (AMET'98), Minneapolis, Minnesota, Maio 1998a.
- GUTTMAN, R. H.; MAES, P. **Cooperative vs. Competitive Multi-Agent Negotiations in Retail Electronic Commerce**. Proceedings of the Second International Workshop on Cooperative Information Agents (CIA'98), Paris, France, Julho 3-8, 1998b.
- JENNINGS, N. R.; WOOLDRIDGE, M. J. **Software Agents**. IEEE Review, January, 1996, p. 17-20.
- KOSIUR, David. **Understanding Electronic Commerce**. USA: Microsoft Press, 1997, 288p.
- LANGE, Danny B.; OSHIMA, Mitsuru. **Programming and deploying Java mobile agents with aglets**. Massachusetts: Addison Wesley Longman, Inc, 1998, 225p.
- LUZ NETO, O. P. **Utilizando o ERP para Desbancar a Concorrência**. Developer's Magazine, ano 4, nº 41, p. 24-25, Janeiro 2000.

- MAES, Pattie. **Agents that Reduce Work and Information Overload.** Communications of the ACM, Vol. 37, No.7, pp. 31-40, 146, ACM Press, Julho 1994.
- 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, pág. 32-60.
- RIPPER, P. S. et al. **V-Market: A Framework for Agent Mediated E-Commerce Systems based on Virtual Marketplaces.** World Wide Web, Baltzer Science Publishers, 3(1), 2000.
- SMITH, Jason. **E-Commerce in the Future.** Special Issue on MarcommWise Knowledge Bank, 1999.
- SOUZA, Cesar A. e ZWICKER, Ronaldo. **Aspectos envolvidos na seleção e implementação de sistemas ERP.** A ser publicado nos Anais da XXXIV CLADEA, Outubro 1999.
- SOUZA, Cesar A. e ZWICKER, Ronaldo. **Um modelo de ciclo de vida de sistemas ERP: aspectos relacionados à sua seleção, implementação e utilização.** Publicado nos Anais da IV SemeAd, Outubro 1999.
- VALDO, Clayton A. e SOBRAL, João Bosco M. **Integração de e-Commerce e ERP através de Agentes Móveis.** Publicado nos Anais do XX SBC, Curitiba, Julho 2000.
- WURMAN, P.R.; WELLMAN, M. P.; WALSH, W. E. **The Michigan Internet AuctionBot: A configurable auction server for human and software agents.** Second International Conference on Autonomous Agents, Maio, pág. 301-308, 1998.