Rosane Maria Martins

# DATA AGENTS: Sistema de Compras na Internet usando Tecnologia de Agentes Móveis

Orientadores: Profª Luci Pirmez, Dra.
Prof. Luiz Fernando Rust da Costa Carmo, Dr. UPS

Núcleo de Computação Eletrônica - NCE
Instituto de Matemática - IM
Universidade do Estado do Rio de Janeiro - UFRJ

Rio de Janeiro, Junho de 2000.

DATA AGENTS:
SISTEMA DE COMPRAS NA INTERNET
USANDO TECNOLOGIA DE AGENTES MÓVEIS

Rosane Maria Martins

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE MATEMÁTICA/NÚCLEO DE COMPUTAÇÃO ELETRÔNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM INFORMÁTICA.

Aprovada por:

_____
Profª. Luci Pirmez, Dra.

_____
Prof. Luiz Fernando Rust da Costa Carmo, Dr. UPS

_____
Prof. Júlio Salek Aude, D.Sc.
.

_____
Prof. Vitório Bruno Mazzola, Dr. UPS

RIO DE JANEIRO, RJ - BRASIL

JUNHO DE 2000

**FICHA CATALOGRÁFICA**

MARTINS, ROSANE MARIA.

DATA AGENTS: Sistema de Compras na Internet usando Tecnologia de Agentes Móveis [Rio de Janeiro] 2000

IX, 84 p. 29,7 cm (IM/NCE/UFRJ, M.Sc., Informática, 2000)

Dissertação - Universidade Federal do Rio de Janeiro, IM/NCE

1. Agentes Móveis

I. IM/NCE/UFRJ II. Título ( série )

# Agradecimentos

A Deus, sempre presente em minha vida.

A Helio Ramos da Silva, por ter sabido suportar os longos momentos de ausência. Obrigada pelo amor, pela tolerância e pela paciência.

Aos meus amigos e familiares que estimularam e desejaram o meu completo sucesso no decorrer desta pesquisa.

À Família Chaves pela companhia e apoio constantes.

Aos meus orientadores pela disponibilidade de tempo, dicas e sugestões fornecidas no decorrer do presente trabalho. Obrigada pela incansável orientação e assistência, bem como pelos estímulos nos momentos de incertezas.

Meus sinceros agradecimentos ao Prof. Arnaldo Vieira pelo incentivo e apoio que me forneceu ao ingresso nas atividades acadêmicas como aluna de Mestrado.

Agradeço à Universidade do Estado do Rio de Janeiro pelas condições relacionadas ao tempo de dedicação para a realização deste trabalho.

Resumo da Tese apresentada ao IM/NCE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

DATA AGENTS: Sistema de Compras na Internet
usando Tecnologia de Agentes Móveis

Rosane Maria Martins

Junho/2000

Orientadores: Luci Pirmez
Luiz Fernando Rust da Costa Carmo

Programa: Informática

A abertura da Internet para uso comercial com sua interface gráfica de fácil uso, gerou um enorme crescimento na quantidade de usuários e variedade de serviços disponíveis nestes últimos anos. Atualmente, grande parte das instituições públicas e privadas fornecem algum tipo de informação ou serviço através da Internet. Tendo em vista este cenário, o presente trabalho apresenta uma aplicação móvel para o comércio eletrônico visando tornar mais fácil para o usuário final a manipulação destas informações. O presente trabalho descreve também a tecnologia dos agentes móveis e a plataforma de desenvolvimento utilizada na implementação da aplicação citada'. Por fim, discute a arquitetura do projeto e ressalta os resultados obtidos na experimentação realizada.

Abstract of Thesis presented to IM/NCE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

DATA AGENTS: Mobile Agents System for E-Commerce

Rosane Maria Martins

June/2000

Advisors: Luci Pirmez
Luiz Fernando Rust da Costa Carmo

Department: Informatics

The beginning of Internet as commercial usage with its easy graphic interface originated an enormous increase of users' quantity as well as a great variety of available services during the last years. Nowadays the majority of private and public institutions are prepared to provide any kind of information or service through the Internet. Considering this fact, the present work presents a mobile application for e-commerce in order to make easier the manipulation of this kind of information by the final user. This thesis also describes the mobile agent technology and the framework used to implement the application. Finally, it emphasizes the project architecture and the obtained results.

# S U M Á R I O


CAPÍTULO 1 - INTRODUÇÃO..... **01**
1.1 - CONSIDERAÇÕES INICIAIS..... **01**
1.2 - TRABALHOS RELACIONADOS..... **04**
1.3 - ESTRUTURA DO TRABALHO..... **06**

CAPÍTULO 2 - FUNDAMENTAÇÃO TEÓRICA..... **08**
2.1 - INTRODUÇÃO..... **08**
2.2 - ARQUITETURA CLIENTE/SERVIDOR..... **08**
2.3 - INTELIGÊNCIA ARTIFICIAL DISTRIBUÍDA..... **10**
2.4 - AGENTES NA INTERNET..... **12**
2.4.1 - Conceituação..... **13**
2.4.2 - Estrutura de um Sistema de Agentes Móveis e seu Ciclo de Vida..... **15**
2.4.3 - Infra-Estrutura..... **17**
2.4.4 - Plataformas de Desenvolvimento..... **19**
2.5 - ORIENTAÇÃO A OBJETOS E JAVA..... **21**
2.5.1 - Orientação a Objetos..... **22**
2.5.2 - Características da Tecnologia Orientada a Objetos..... **23**
2.5.3 - A Linguagem Java..... **25**
2.5.3.1 - Independência de Plataforma..... **26**
2.5.3.2 - Robustez..... **27**
2.5.3.3 - Segurança..... **29**
2.5.3.4 - Java e Banco de Dados..... **30**

2.6 - CONCLUSÃO..... **33**

CAPÍTULO 3 - ARQUITETURA E IMPLEMENTAÇÃO DO SISTEMA DATA AGENTS..... **36**
3.1 - INTRODUÇÃO..... **36**
3.2 - CENÁRIO ELETRÔNICO DE COMPRA..... **37**
3.3 - PROPRIEDADES DO MODELO..... **38**
3.4 - AMBIENTE DE DESENVOLVIMENTO..... **39**
3.4.1 - Aglets Software Development Kit: Principais Classes..... **40**
3.4.2 - Ciclo de Vida de um Aglet..... **42**
3.4.3 - Comunicação entre Aglets..... **44**
3.4.4 - Funcionamento Básico do Protocolo ATP (Agent Transfer Protocol)..... **46**
3.4.5 - Segurança em Aglets..... **46**
3.5 - ARQUITETURA PROPOSTA..... **47**
3.5.1 - Descrição Geral do Sistema..... **49**
3.5.2 - Funcionamento do Protótipo..... **51**
3.6 - Implementação..... **54**
3.6.1 - Descrição das Etapas e Módulos Desenvolvidos..... **54**
3.6.2 - O Ambiente de Simulação..... **57**
3.7 - CONCLUSÃO..... **59**

CAPÍTULO 4 - AVALIAÇÃO DA QUALIDADE E DO DESEMPENHO DO SISTEMA DATA AGENTS..... **61**
4.1 - INTRODUÇÃO..... **61**
4.2 - AVALIAÇÃO DA QUALIDADE..... **61**
4.2.1 - Avaliando Alinhamento Semântico..... **63**
4.2.2 - Avaliando a Agilidade do Sistema..... **66**
4.3 - AVALIAÇÃO DE DESEMPENHO..... **68**
4.4 - CONCLUSÃO..... **70**

CAPÍTULO 5 - CONCLUSÃO..... **72**
5.1 - CONTRIBUIÇÕES..... **72**
5.2 - PERSPECTIVAS PARA PESQUISAS FUTURAS..... **74**

REFERÊNCIAS BIBLIOGRÁFICAS..... **76**

## LISTA DE FIGURAS

FIGURA 1 - Infra-estrutura de sistemas de agentes móveis..... **18**
FIGURA 2 - Estrutura de um objeto..... **24**
FIGURA 3 - Projeto de banco de dados JDBC..... **31**
FIGURA 4 - Principais classes e interfaces..... **41**
FIGURA 5 - Ciclo de vida dos aglets..... **42**
FIGURA 6 - Métodos “request” do ATP..... **46**
FIGURA 7 - Arquitetura geral do sistema Data Agents..... **49**
FIGURA 8 - Módulo de Interface do Data Agents..... **50**
FIGURA 9 - Módulo de Controle - Estrutura Interna..... **53**
FIGURA 10 - Resultado da Pesquisa Efetuada..... **53**
FIGURA 11 - Contextualização dos processos ao longo da rede..... **58**
FIGURA 12 - Modelo de entidade e relacionamento do Data Agents..... **59**
FIGURA 13 - Tempo médio de execução dos agentes compradores..... **70**

## LISTA DE TABELAS

TABELA 1 - Tipos de dados..... **26**
TABELA 2 - Métricas geradas utilizando o método GQM..... **63**
TABELA 3 - Análise da Difusão Conceitual presente no código móvel..... **64**
TABELA 4 - Capacidade de Mudança..... **67**
TABELA 5 - Tempos médios de execução dos agentes compradores..... **69**

# CAPÍTULO 1:

## *Introdução*

## 1.1- Considerações Iniciais

O homem é diferenciado dos demais seres pelo grau de sua capacidade de apreensão do universo, ou seja, sua capacidade de conceber idéias; por ser capaz de expressá-las transmitindo-as a seus semelhantes; e por sua capacidade de raciocínio, manipulando juízos [43]. A forma como empreende estas atividades determina seu domínio sobre o meio, incluindo não apenas o ambiente físico em que habita mas também seu convívio social, diretamente condicionando sua existência. Assim, o tratamento das informações é fator extremamente relevante para as sociedades.

Efetivamente, em nosso século, as transformações oriundas da conjunção de duas tecnologias - comunicação e processamento de informações - vieram "revolucionar o mundo em que vivemos, abrindo as fronteiras com novas formas de comunicação, e permitindo maior eficácia dos sistemas computacionais". [85]

Se, durante a década de 1970, as mudanças no enfoque aos sistemas de computação centralizados levaram ao surgimento das redes locais, atualmente o objetivo não é apenas uma infinidade de redes locais existentes em organizações diversas, mas sim a possibilidade de que "um grande volume de usuários seja capaz de trocar informações, interligando-se as redes às quais eles estão conectados, formando assim uma ***inter-rede***[85]. Sendo assim, redes de computadores são uma realidade neste contexto.

Baseados nestas constatações, pode-se afirmar, em consonância com diversos pesquisadores [83], que a expectativa quanto aos ambientes de processamento de informações do futuro é de que sejam constituídos por redes heterogêneas, caracterizadas por concepção descentralizada, autonomia em termos de ações empreendidas por cada unidade, e filosofia de ambientes abertos, onde novas unidades

podem entrar e sair a qualquer momento. A Internet é apresentada como exemplo irrefutável desta tendência [83].

Paralelamente à evolução dos sistemas distribuídos e utilizando-se de seus resultados, pesquisadores de Inteligência Artificial (IA) iniciaram novos trabalhos que não buscavam mais o desenvolvimento de uma única entidade inteligente, mas sim de atividades distribuídas no espaço e no tempo, surgindo, assim, a Inteligência Artificial Distribuída (IAD) [40].

Originalmente, o enfoque adotado na área de IA foi o de concepção centralizada, efetuada por uma só pessoa ou um grupo de pessoas, para a resolução de um problema específico. Esse problema poderia ser melhor trabalhado se dividido em sub-problemas a serem resolvidos de forma coordenada originando a abordagem de Resolução Distribuída de Problemas. Desta última abordagem, evoluiu-se, no final da década de 1980, para a abordagem de "Inteligência Artificial Descentralizada" [25], ou Sistemas Multi-Agentes, como tornou-se conhecida posteriormente. Nesse enfoque, sistemas distribuídos são uma coleção de agentes autônomos em um mesmo ambiente que podem interagir, cada qual possuindo suas próprias capacidades e objetivos.

Devido à importância da manipulação de informações para o desenvolvimento das sociedades aliada às perspectivas atuais das redes de computadores, a tecnologia de sistemas multi-agentes vem obtendo um interesse crescente na Ciência da Computação. Aponta-se que, no próximo século, esta tecnologia terá papel central no desenvolvimento de sistemas distribuídos.

O termo agente ganha cada vez mais popularidade, sendo divulgado por publicações com público bem mais amplo do que a comunidade original de Inteligência Artificial [70],[72]. No entanto, não há um conceito único para a palavra "agente". É comum seu emprego designando entidades muito diversas, incluindo programas tradicionais extremamente simples [58],[82].

As definições do termo "agentes móveis" seguem um padrão, visto que ressaltam a possibilidade de migração de código e estado, bem como a autonomia de execução. O

recente avanço nesta área de agentes móveis tem sido impulsionado pela preocupação com a qualidade do software e pelas deficiências do modelo cliente/servidor em aplicações distribuídas. A popularidade da Internet como um mecanismo universal de acesso a informações criou a necessidade do desenvolvimento de aplicações com acesso a banco de dados que suportem esta tecnologia. Para o desenvolvimento deste tipo de aplicações, é necessário o uso de métodos, ferramentas e procedimentos próprios de forma a atender adequadamente às necessidades da sociedade presente. Desse modo, além das pesquisas existentes sobre diversos conceitos intrínsecos a esta nova área, também o estudo de métodos adequados para a modelagem e análise de aspectos relevantes destes sistemas apresenta-se bastante promissor.

A tecnologia Java tornou-se respeitada nos centros de computação pelo fato de ser portável e orientada para a Internet, o que a leva a ser a melhor linguagem para implementações Cliente/Servidor e computação móvel. Mais ainda, com a criação do JDBC (*Java Database Connectivity*) que permite o acesso a bancos de dados, tem despertado o interesse dos mais diversos fabricantes de Sistemas Gerenciadores de Banco de Dados (SGBD).

A união destas duas tecnologias, Java e *Web*, pode fornecer soluções para a recuperação de informações em bancos de dados. Atualmente, existem metodologias que possibilitam a implementação de *applet*s que acessem a bancos de dados através do JDBC, mas que possuem grandes limitações como, por exemplo, o fato de que o provedor de serviços *Web* precisa estar no mesmo servidor em que está o banco de dados, além do *browser* cliente precisar descarregar as classes do JDBC do servidor antes de inicializar seu contexto, ou seja, essas tecnologias convencionais são bastante limitadas principalmente em termos de desempenho.

Este trabalho tem por objetivo principal o uso de agentes móveis para facilitar a implementação de aplicações na *Web*. São apresentadas a arquitetura e a implementação de um sistema de recuperação de informações estruturadas que, a partir de um cliente, cria um agente que se deslocará até o servidor de banco de dados, fará as consultas necessárias e retornará os resultados ao usuário através de uma tela ou por e-mail.

Os objetivos secundários, voltados para a consecução desta meta maior, mas que também merecem destaque como contribuições à área, são:

1. a apresentação de uma visão panorâmica do paradigma de Agentes Móveis, de forma a sistematizar o conhecimento atual sobre o estado da arte desta disciplina;
2. o estudo do funcionamento da arquitetura cliente/servidor e suas ferramentas na Internet;
3. o estudo e a avaliação de ferramentas que auxiliam na construção de um sistema de agentes e que possibilitam o acesso a banco de dados relacionais distribuídos.

## 1.2- Trabalhos Relacionados

O conceito de agenciamento ou intermediação ("agency") vem se tornando cada vez mais popular. A idéia é ter processos semi-autônomos que apóiam o usuário na realização de tarefas, cumprindo um papel intermediador entre o usuário e os programas que efetivamente executam as tarefas a níveis mais baixos de abstração. Na maioria dos casos, entretanto, os agentes não prestam diretamente um serviço para o usuário, o usuário não está consciente do auxílio que recebe durante a realização das tarefas [6].

Um exemplo do uso desta técnica é o agente de compras ***Bargain Finder***[1], cuja funcionalidade está relacionada com a compra de CDs (*compact disc*) que tenham um preço melhor a partir de uma rede de fornecedores. Esse agente permite aos usuários compararem preços entre oito fornecedores de CDs distribuídos na Internet. Este agente de compras foi criado pela empresa *Andersen Consulting* como um protótipo que visa demonstrar o uso de agentes inteligentes no comércio eletrônico.

Um outro agente mais sofisticado que o BargainFinder é o ***Firefly***, que ajuda o usuário a procurar determinado tipo de música na rede, aprendendo as preferências musicais de cada usuário através da interação com os mesmos e até mesmo fazendo algumas sugestões[61].

[1] Disponível em: http://bf.cstar.ac.com/bf

O agente *Jango*[2] funciona como um catálogo onde é escolhida a categoria do produto que se pretende comprar, além da identificação do mesmo. A partir dessa escolha, o agente retorna uma lista de fornecedores onde a compra pode ser efetuada.

*Bargainbot*[3] é um agente que ajuda os usuários a comprarem livros na Internet. Sua arquitetura facilita a busca simultânea nas bases de dados de várias livrarias. Após apresentar o título do livro ou o autor, o agente sai a procura do mesmo nas livrarias onde existe o devido acesso de consulta ao banco de dados. Em seguida, o agente filtra as informações pesquisadas e apresenta ao usuário o título, autor, livraria e preço do livro.

Existem vários sistemas multi-agente funcionais, que permitem automatizar grande parte do processo de compra e venda de produtos. Para facilitar o estudo de mecanismos de negociação entre os agentes que compõem os sistemas multi-agentes, alguns destes baseiam-se em leilões. Estes sistemas funcionam como verdadeiros mercados eletrônicos. Os mais significativos são o AuctionBot [95], o FishMarket[76], o Kasbah [13] e o Tête-a-Tête [55]. Apesar da existência de inúmeras aplicações baseadas na tecnologia de agentes no contexto do comércio eletrônico, o presente levantamento bibliográfico revela a inexistência de trabalhos que explorem a propriedade de mobilidade de seus agentes.

Hoje é possível encontrar quase todo tipo de informação dentro da Internet. Entretanto, a *Web*, em especial, não foi planejada para suportar publicações organizadas e recuperação de informações de forma coordenada, levando até mesmo especialistas a gastarem uma grande quantidade de tempo para acessar e recuperar o recurso de informação desejado. Nesse contexto, surgiram as *Search Engines* [88],(*Altavista*, *Lycos*, *Excite*), que utilizam robôs (também conhecidos como *spiders* ou *crawlers*) [14], [45],[46] para automatizar, classificar e indexar a vasta quantidade de informação digital presente na *Web*. Tais ferramentas apresentam a vantagem de explorar diretamente de maneira rápida e a um custo baixo a Internet, trabalhando de forma

[2] Disponível em: http://jango.excite.com

[3] Disponível em: http://www.scu.edu.au/sponsored/ausweb/ausweb96/tech/aoun/paper.html

democrática, através do acesso uniforme a todas as informações disponíveis. Entretanto, essa possibilidade de acessar todos os documentos disponíveis também dificulta a recuperação de documentos relevantes.

Este trabalho contribui com a implementação de uma aplicação baseada em agentes móveis, a fim de demonstrar sua aplicabilidade no processo de busca de informações estruturadas de distribuídas no domínio do comércio eletrônico. Visando otimizar o processo de recuperação de informações que estão distribuídas em diversos bancos de dados, o presente trabalho apresenta um protótipo que realiza a seleção dos dados armazenados em cada servidor distinto, deslocando-se entre eles, para no final retornar todos os dados solicitados pelo usuário, sem a necessidade do cliente efetuar a chamada a cada um dos servidores separadamente. Em virtude da execução local da pesquisa, o acesso ao banco de dados não consumirá os recursos da rede, melhorando ainda mais a performance da recuperação da informação desejada.

## 1.3- Estrutura do Trabalho

Este trabalho foi organizado da seguinte forma: o capítulo seguinte apresenta os fundamentos teóricos necessários para a elaboração desta dissertação. É fornecida uma visão geral da arquitetura cliente/servidor, partindo de sua evolução histórica e suas diferentes abordagens. Os fundamentos sobre Inteligência Artificial Distribuída também são apresentados neste capítulo. Considerando os objetivos deste trabalho, a ênfase encontra-se nos aspectos relevantes de modelagem e análise voltados ao conceitos dos agentes móveis. Os principais conceitos e características da linguagem de programação Java, tais como independência de plataforma, orientação a objetos e integração com banco de dados, também são analisados.

No capítulo 3, são apresentados a arquitetura e o funcionamento do modelo proposto para a recuperação de informações estruturadas e distribuídas no domínio do comércio eletrônico.

O quarto capítulo aborda os aspectos relacionados à implementação e à avaliação de desempenho do protótipo.

O quinto e último capítulo tratam das conclusões obtidas neste trabalho, finalizando com a apresentação das perspectivas para desdobramentos futuros a esta pesquisa.

# CAPÍTULO 2:

## *Fundamentação Teórica*

## 2.1 - Introdução

Este capítulo apresenta uma visão geral da arquitetura cliente/servidor e da Inteligência Artificial Distribuída (IAD), partindo de sua evolução histórica, suas diferentes abordagens e seus principais temas. Além disso, apresenta também, uma visão sistematizada da tecnologia de agentes e as principais características da metodologia orientada a objetos e da linguagem de programação Java. Esse trabalho propõe-se a utilizar a IAD para auxiliar o processo de recuperação de informações estruturadas e distribuídas. Neste contexto, foi desenvolvido, em Java, um modelo baseado em agentes dentro do comércio eletrônico que pretende auxiliar as compras via Internet.

## 2.2 - Arquitetura Cliente/Servidor

A expressão cliente/servidor foi originalmente aplicada à arquitetura de software que descrevia o processamento entre dois programas - a aplicação e o serviço de suporte. Nesta época, os programas do cliente e do servidor não estavam, obrigatoriamente, separados fisicamente; podiam ser dois programas rodando na mesma máquina, por exemplo. Assim, as discussões sobre cliente/servidor eram limitadas à interação entre um cliente e um servidor. Com a evolução da ciência da computação, a concepção de programas capazes de gerenciar recursos ou prover serviços a vários outros programas tornou-se largamente aceita [68].

Para entender a importância do modelo de processamento cliente/servidor, deve-se, primeiro, compreender como surgiu o ambiente de processamento de dados corporativos. Inicialmente, a estrutura organizacional dos computadores era *top-down*.

Como o negócio era gerenciado de maneira centralizada , a gerência dos dados também era centralizada. Com o crescimento das organizações e o aumento de competitividade existente no mercado, a gerência centralizada evoluiu para um modelo mais flexível. As organizações fragmentaram-se em unidades menores, descentralizando a gerência do negócio. Como conseqüência, os dados e o processamento também foram descentralizados. Assim, cada unidade passou a possuir e controlar seus próprios sistemas de processamento de dados.

A necessidade de compartilhar informações entre as diversas unidades para a tomada de decisões corporativas surgiu com a descentralização das organizações. Logo, a necessidade de acesso centralizado a dados descentralizados ficou aparente. Em paralelo às mudanças das necessidades organizacionais, ocorreu a evolução do *hardware* e do *software*. O surgimento dos computadores pessoais (PCs) levou o poder de processamento a cada mesa de trabalho. Redes foram desenvolvidas para permitir o compartilhamento de dados entre as várias máquinas. Padrões de aplicações como o *MS Windows* e o *SQL* foram desenvolvidos. Essas tecnologias permitiram que usuários de diferentes sistemas compartilhassem informações entre si.

Além dos fatores técnicos existiram também os comerciais. PCs passaram a fornecer um grande poder de processamento a um baixo custo. O custo dos meios de armazenamento de dados nos PCs passaram a ser mais baixos que nos computadores de grande porte. As interfaces gráficas tornaram os computadores mais fáceis de serem usados.

Outro fator relevante foi a disponibilidade de ambientes de desenvolvimento de aplicativos, permitindo que aplicações completas fossem desenvolvidas muito mais rapidamente. Devido a todos esses fatores mencionados, o conceito cliente/servidor ganhou grande popularidade.

A tecnologia cliente/servidor é bastante utilizada em processamento. Os servidores oferecem serviços a processos usuários, ou seja, executam a tarefa solicitada e enviam uma resposta ao cliente que se traduz nos dados solicitados. Por sua vez, os

clientes solicitam um determinado serviço, através do envio de uma mensagem ao servidor. Enquanto o processo servidor está trabalhando a solicitação, o cliente está livre para realizar outras tarefas.

Nenhum tipo de conexão deve ser estabelecida antes do envio da solicitação e nem desfeita após a obtenção da resposta. A própria mensagem de resposta serve como confirmação do recebimento da solicitação. Em conseqüência de sua simplicidade, esse modelo é bastante eficiente. De modo geral, são necessários três níveis de protocolos. Os protocolos do nível físico e enlace de dados tratam da obtenção dos pacotes dos clientes, de sua entrega no servidor correspondente, e da correspondente entrega aos clientes dos pacotes gerados pelos servidores. O nível do protocolo de solicitação/resposta define um conjunto de solicitações legais e de respostas aceitas para tais solicitações.

Um aspecto importante nos sistemas cliente/servidor é a transparência, ou seja, para o usuário não deve existir diferença entre acessar um recurso local ou remoto, não deve ser motivo de preocupação para o usuário a localização do servidor e a natureza da comunicação.

## 2.3- Inteligência Artificial Distribuída

Nas abordagens clássicas de Inteligência Artificial (IA), a ênfase da inteligência é baseada em um comportamento humano individual e o foco de atenção volta-se à representação de conhecimento e métodos de inferência. Já a Inteligência Artificial Distribuída (IAD), é baseada em comportamento social e sua ênfase é para cooperações, interações e para o fluxo de conhecimento entre unidades distintas [31].

A IAD procura estudar o conhecimento e as técnicas de raciocínio que podem ser úteis para que o grupo de agentes computacionais integrem uma sociedade de agentes, desenvolvendo técnicas de Inteligência Artificial para tentar solucionar problemas de maneira distribuída. Essas técnicas passam a ser úteis na solução de

problemas amplos onde o domínio da questão é inerentemente distribuído no espaço, como por exemplo, pode-se citar os problemas de controle de tráfego aéreo, distribuição de energia elétrica, etc.

A existência das técnicas de Inteligência Artificial Distribuída pode ser melhor entendida e, até mesmo, justificada se comparada com a própria sociedade humana. Na sociedade humana, para um determinado problema ser solucionado, muitas vezes depende da interação de mais de uma agente (ser humano) e da comunicação destes para que a meta (solução do problema) seja atingida de maneira satisfatória.

Sob o aspecto da Resolução Distribuída de Problemas em Inteligência Artificial Distribuída (IAD), uma tarefa global é inicialmente definida e o problema principal é projetar as entidades distribuídas, para permitir a execução desta tarefa global. O objetivo é estudar a distribuição e a resolução colaborativa de uma determinada tarefa. Em IAD, entidades individuais que contribuem para a solução de um problema são chamadas de agentes. Agentes quando agrupados podem formar uma comunidade do tipo cooperativa com o objetivo de atingir uma meta comum (solucionar um problema global).

O mundo da Inteligência Artificial Distribuída pode ser dividido em dois enfoques principais [31],[90]: a Resolução Distribuída de Problemas (RDP) e Sistemas Multiagentes (SMA). Vários autores citam esta divisão no mundo da IAD, mas é importante observar que a linha que separa as duas áreas de pesquisa ainda é muito tênue.

Na Resolução Distribuída de Problemas, o enfoque principal é o *problema*. Seus objetivos principais são fazer uso da capacidade de processamento oferecida pela tecnologia de redes para atacar problemas naturalmente distribuídos, como a decomposição e a alocação de tarefas em uma rede de computadores. Os agentes envolvidos em RDP são programados para cooperar uns com os outros, dividindo e compartilhando conhecimento sobre o problema e sobre o processo de obter uma solução.

Existem uma série de aplicações para RDP, como, por exemplo:

- ***interpretação distribuída -*** redes de sensores, diagnóstico de falhas de redes de comunicação;
- ***planejamento e controle distribuídos -*** controle distribuído de processos em manufatura;
- ***sistemas especialistas cooperantes -*** controle de veículos autônomos, negociação entre especialistas, etc.

Nesta segunda linha de trabalhos da IAD, tradicionalmente chamada de *Sistemas Multi-Agentes (SMA)*, o enfoque principal não é um problema específico, como no caso anterior, mas a coordenação do comportamento inteligente entre um conjunto de agentes autônomos. Os agentes devem raciocinar a respeito das ações e sobre o processo de coordenação entre si. As suas arquiteturas são mais flexíveis e a organização do sistema está sujeita a mudanças visando a adaptação às variações no ambiente e/ou problema a ser resolvido. [31].

As aplicações atuais são inúmeras e sofisticadas. Domínios típicos são [19]: robótica, tomada de decisão administrativa, administração de balcões bancários, controle de tráfego aéreo, gerência de redes de telecomunicações, automação de escritórios e sistemas de atendimento ao público.

## 2.4 - Agentes na Internet

Com o extraordinário desenvolvimento da rede mundial de computadores, a Internet, o volume de informações disponível pelo mundo, o custo de movimentação de dados e a pequena experiência em computação de muitos usuários têm sido motivo de preocupação para a comunidade de redes e para os desenvolvedores de aplicativos. O tempo despendido para efetuar tarefas simples, como encontrar informações desejadas sobre produtos, serviços, clientes e/ou fornecedores, aumenta cada vez mais conforme a expansão do número de usuários.

Uma possível solução para estes problemas consiste na utilização de ***agentes***, que são programas que ajudam um usuário a realizar tarefas, agindo em seu interesse. Caso os agentes possam executar suas tarefas em diferentes computadores da rede, estes são denominados ***agentes móveis***. Este tipo de agentes pode mover-se para o lugar onde os dados estão armazenados e, utilizando inteligência, selecionar as informações de que o usuário necessita; economizando, assim, banda passante, tempo e dinheiro.

### 2.4.1 - Conceituação

Ainda não existe uma definição uniforme para o termo agente dentro da IAD. [73 ],[38]. Neste trabalho, agentes são considerados como sendo entidades de software que fazem um conjunto de operações em favor do usuário ou outro programa com algum grau de independência e autonomia. Agentes de software ou, simplesmente, agentes podem ser definidos num espaço de três dimensões: agência, inteligência e mobilidade [59].

Basicamente, o que diferencia uma definição da outra são as propriedades que caracterizam cada agente. As propriedades e sua intensidade caracterizam o comportamento do agente dentro de uma sociedade. Como qualidades consideradas desejáveis em um agente de software, pode-se citar:

- ***autonomia -*** um agente é capaz de tomar iniciativa e exercer um grau de controle sobre suas próprias ações;
- ***orientação a meta -*** os agentes não respondem somente ao ambiente, mas buscam, também, um objetivo[33];
- ***flexibilidade -*** as ações de um agente não seguem um roteiro, estes são capazes de realizar escolhas dinâmicas de quais ações realizar e em qual seqüência, em resposta a uma mudança do ambiente externo;
- ***inicialização própria -*** um agente pode perceber modificações em seu ambiente e decidir quando atuar, o que o difere dos programas padrões que são diretamente disparados pelos usuários;

- ***comunicação*** **-** um agente é capaz de comunicar-se com outros agentes, incluindo pessoas, com o intuito de obter informações ou conseguir ajuda para cumprir seus objetivos;
- ***inferência social*** **-** um agente deve ser capaz de inferir sobre as atividades de outros agentes;
- ***inteligência*** **-** um agente é capaz de raciocinar e aprender a partir das interações com outros agentes, com seus usuários e com o ambiente, podendo ser dotado de diferentes níveis de inteligência;
- ***mobilidade*** **-** um agente móvel é aquele que é capaz de se transportar de uma máquina para outra durante sua execução;
- ***continuidade temporal*** **-** um agente é um processo que está em execução continuamente, não como um instante computacional que mapea uma única entrada para uma única saída e termina.

É importante enfatizar que para os agentes executarem suas tarefas, eles podem ou não usar as capacidades supracitadas; e é em função dessas capacidades que eles podem receber outras denominações, tais como: agentes fixos, agentes móveis, agentes inteligentes, agentes móveis e inteligentes, entre outras.

Já as definições de agentes móveis seguem um padrão, visto que ressaltam a possibilidade de migração de código e estado, bem como a autonomia de execução.

Em [28], "agentes móveis são programas que têm a capacidade de migrar de uma máquina para outra de uma rede durante a sua execução, carregando consigo seu estado de execução: ***agente = comportamento, estado & localização"***

> Em [36],"um agente móvel é capaz de se mover em uma rede heterogênea, de um nó para outro, sobre seu próprio controle, interagindo com os recursos existentes ou com outros agentes, tipicamente retornando ao local de origem quando sua tarefa estiver concluída."

Os conceitos centrais envolvidos neste novo paradigma são ***agente*** e ***migração remota de procedimentos***. Um agente, como já citado anteriormente, é uma entidade que tem um objetivo, sendo capaz de realizar uma tarefa pré-definida. Para realizar essa tarefa, um agente possui especialidades, sendo ainda capaz de perceber e representar o

ambiente em que se encontra, e de se comunicar com outros agentes que compõem o sistema. O comportamento global do sistema não advém de cada tarefa realizada pelos agentes individualmente, mas do comportamento da sociedade de agentes como um todo. Em um ambiente de rede heterogênea de computadores, por exemplo, uma sociedade de agentes móveis realiza um trabalho cooperativo, a partir das especialidades individuais de cada agente.

### 2.4.2 - Estrutura de um Sistema de Agentes Móveis e seu Ciclo de Vida

Um sistema de agente móvel, segundo Markus Endler [28], possui os seguintes elementos: código, estado, atributos, lugar, região, deslocamento, encontro, , autoridade, permissões, servidor e hospedeiro.

O ***código*** de um agente móvel é o programa que define seu comportamento. Este programa pode ser escrito em qualquer linguagem que possa ser interpretada por plataformas e por máquinas diferentes, garantindo a plena mobilidade do agente. As linguagens indicadas para o programa do agente móvel são, portanto, aquelas que são diretamente interpretáveis, ou aquelas que são compiladas para uma linguagem intermediária baseada em interpretador e que possa ser facilmente executada e transportada. Alguns exemplos de tais linguagens são Tcl, Pearl e Java.

Um agente deve saber quais suas ações no passado para determinar suas ações futuras. Para tal, cada agente dispõe de um ***estado***. Desta forma, o agente pode retomar suas atividades após ter se transportado de um hospedeiro para outro, a partir do ponto onde havia parado. O estado mantém guardado, portanto, o ponto de execução e as variáveis necessárias para a realização das tarefas.

Os ***atributos*** são utilizados para descrever o agente para os seus hospedeiros. Exemplos: identificador único e usuário do agente. Os atributos também impõem limitações na mobilidade, estabelecendo limites de domínios nos quais ele deve trafegar e limites de tempo de ação sob a forma de um prazo de expiração. Outro fator

importante descrito nos atributos é a quantidade de recursos do hospedeiro que o agente irá utilizar.

O ***lugar*** (contexto) é o ambiente lógico de execução de agentes, e tem como atributos a identidade e autoridade. Além disso, um lugar disponibiliza um conjunto de serviços (recursos). Uma ***região*** (domínio) é o domínio administrativo associado a cada lugar (ex.: nce.ufrj.br).

Dá-se o nome de ***deslocamento*** à transferência de um agente de um lugar para outro, que só ocorre se o mesmo está autorizado a visitar o lugar destino. Um encontro é a interação direta entre dois ou mais agentes, geralmente posicionados em um mesmo lugar.

Uma ***autoridade*** é a identidade da pessoa ou empresa que o gerente ou o lugar representa. Autoridade e identificação servem como base para a autenticação e autorização.

As ***permissões*** determinam quais operações podem ser executadas por agentes e lugares e qual é a quantidade máxima de recursos que pode ser usada.

Um ***servidor*** é o programa que executa em cada máquina e cujas tarefas são: a criação, a ativação e a transferência de agentes, a execução dos mecanismos de autenticação e autorização e a gerência do uso dos recursos do hospedeiro. E um ***hospedeiro*** é uma máquina na qual um agente está executando.

Quanto ao ciclo de vida de um agente móvel, podem ocorrer as seguintes ações:

- ***criar -*** o agente "nasce", seu estado é inicializado;
- ***clonar -*** um clone de um agente é criado com o mesmo estado do original;
- ***disparar -*** um agente é remetido para outra máquina;
- ***restituir -*** um agente é chamado de volta ao seu lugar de origem;
- ***desativar -*** a execução do agente é congelada e seu estado é armazenado;
- ***ativar -*** a execução do agente é iniciada ou retomada;

- ***desalocar -*** o agente é destruído e os seus recursos liberados.

### 2.4.3 - Infra-Estrutura

Para que um agente possa realizar suas tarefas, ele deve ser capaz de interagir com os hospedeiros, comunicar-se com outros agentes e mover-se através de uma rede de computadores. Os hospedeiros, para isso, devem proporcionar uma ***infra-estrutura*** de agentes[77]. A entidade que é capaz de prover suporte para os agentes em um hospedeiro particular é denominado ***servidor de agentes***. Outra função de extrema importância destas infra-estruturas é garantir a segurança, tanto dos hospedeiros quanto dos agentes neles alocados. De fato, um dos maiores problemas enfrentados pelos agentes móveis é a questão de segurança. Como seria possível permitir que um agente externo executasse programas em um hospedeiro e ainda assim evitar que estes programas venham a ameaçar a integridade dos sistemas? A solução mais comumente adotada é a da utilização de uma máquina virtual no lugar de uma máquina física. Desta forma, os agentes não executam suas tarefas diretamente sobre o processador, a memória e o sistema operacional real, mas sim sobre uma máquina virtual, geralmente um interpretador e um sistema em tempo de execução que escondem os detalhes da arquitetura do hospedeiro e restringem as possibilidades de ação dos agentes a um ambiente restrito.

O modelo de infra-estrutura baseado no servidor de agentes é ilustrado na figura 1. O servidor provê serviços básicos para os agentes como, por exemplo, uma máquina virtual para a sua execução. O ambiente adequado para a execução do agente é denominado ***ambiente em tempo de execução***. Este ambiente deve proteger o hospedeiro de agentes de visitas "maliciosas", deve capturar o estado de um agente para prosseguir a execução no ponto correto e deve interagir com o servidor. A interação entre o usuário (dono do agente) e a infra-estrutura dá-se através de um cliente. O cliente não necessita ficar conectado permanentemente com o restante do sistema, permitindo o processamento assíncrono. Os ambientes, neste tipo de infra-estrutura, em tempo de execução devem oferecer ferramentas básicas para os agentes, permitindo, desta forma, a diminuição do tamanho destes e otimizando o tráfego na rede durante a

movimentação dos agentes. Se, por exemplo, um agente busca uma informação em um banco de dados, a infra estrutura é quem vai prover as ferramentas de busca. Este ambiente deve também garantir a mobilidade e a comunicabilidade dos agentes com os usuários. Para resolver problemas de segurança, os ambientes em tempo de execução devem limitar os recursos do sistema para que os agentes possam realizar apenas as tarefas oferecidas pelo servidor. Para procurar por algum dado, um agente não precisa formatar um disco. Algumas operações devem ser, portanto, desabilitadas ou rigidamente controladas.

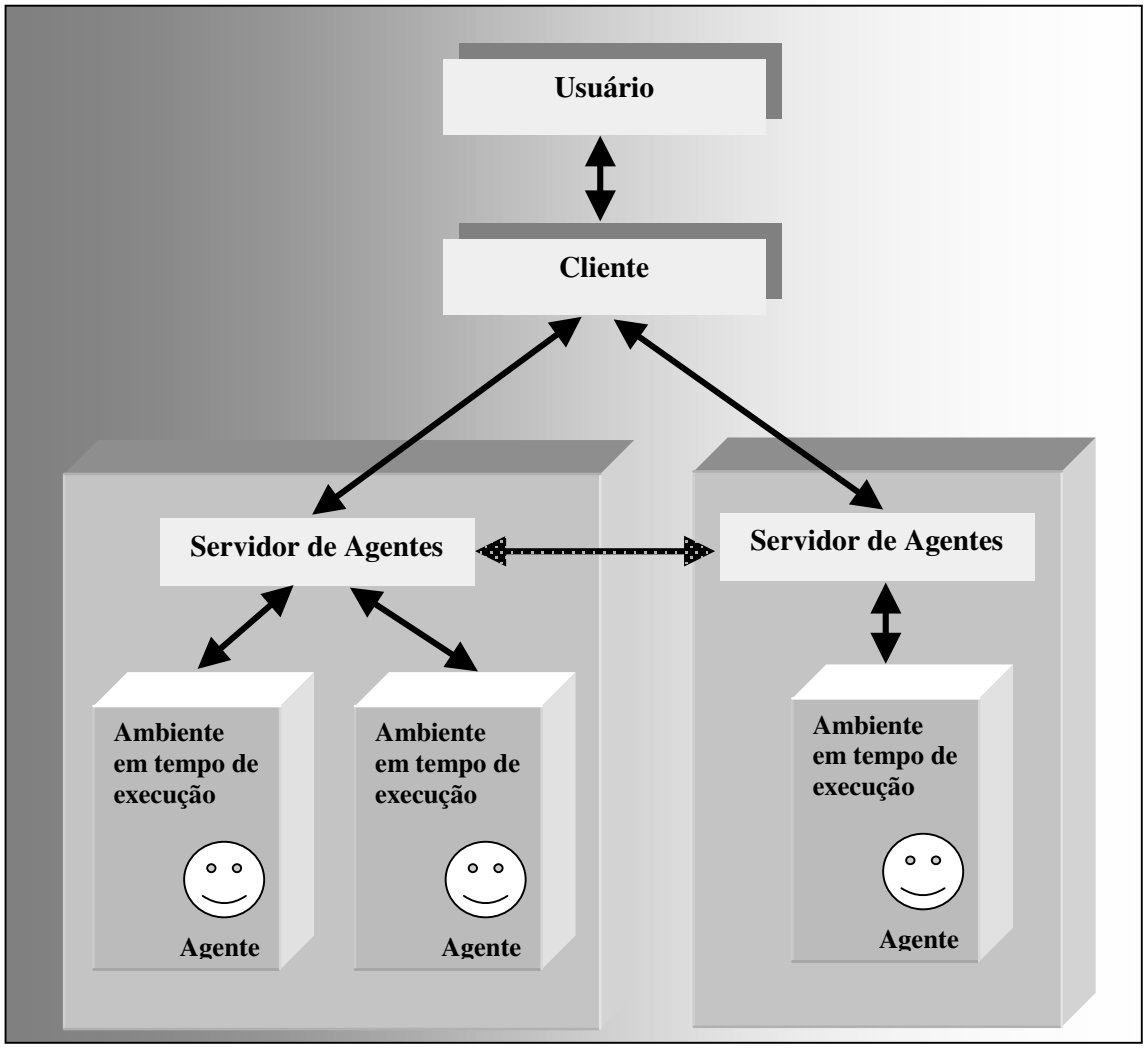


Figura 1 - Infra-estrutura de sistema de agentes móveis [77]

O servidor de agentes é responsável pela criação do ambiente virtual adequado para a execução dos servidores. Também é papel do servidor, o gerenciamento das comunicações entre os agentes e os usuários e entre os agentes. As migrações dos

agentes e as permissões de acesso assim como a correção de erros nos agentes e nas informações carregadas por eles são outras responsabilidades dos servidores de agentes.

### 2.4.4 - Plataformas de Desenvolvimento

No mercado, existem diversos sistemas para a construção de agentes que permitem ao usuário construir e gerenciar seus próprios agentes. Numerosos ambientes estão ainda em desenvolvimento e alguns deles disponíveis na *Web* para avaliação.

São descritos, nesta seção, alguns sistemas ou ambientes desenvolvidos para dar suporte à implementação de sociedades de agentes.

A General Magic Inc. criou o ***Telescript***[4], o primeiro sistema comercial de agentes móveis, e provavelmente, o mais conhecido ambiente que executa e gerencia aplicações baseadas em agente nos servidores. Entretanto, por ser baseado em uma linguagem proprietária teve uma "vida" curta. Em resposta à popularidade da Internet e ao sucesso da linguagem de programação Java, a General Magic decidiu reimplementar o conceito de agentes móveis no novo sistema ***Odyssey***[5]. Este sistema implementa toda a funcionalidade do Telescript utilizando as classes Java. O resultado é uma biblioteca de classes que habilita os desenvolvedores a criarem suas próprias aplicações de agentes móveis.

O ***Concordia***[6] é uma plataforma da Mitsubishi para o desenvolvimento e gerenciamento de aplicações de agentes móveis. Consiste de múltiplos componentes, todos escritos em Java, que, combinados, fornecem um ambiente completo para aplicações distribuídas. Este sistema é composto de uma máquina virtual Java padrão, um servidor e um conjunto de agentes; e provê mecanismos de segurança para execução segura de agentes, além de suportar mecanismos de *checkpoint* para tolerância a falhas.

[4] Disponível em: http://www.genmagic.com/telescript/index.html
[5] Disponível em: http://www.genmagic.com/technology/odyssey.html
[6] Disponível em: http:// http://www.meitca.com/HSL/Projects/Concordia/

O ***Voyager***[7] da Object Space é uma plataforma para a construção de sistemas distribuídos baseados em agentes. Implementa os mecanismos tradicionais de troca de mensagens, somados à capacidade de objetos moverem-se através da rede como agentes. Pode-se dizer que o Voyager combina as propriedades de um *object request broker* baseado em Java com as de um sistema de agente móvel. Desta forma, o Voyager permite que os programadores Java criem aplicações de rede utilizando ambas as técnicas: a tradicional e a programação distribuída baseada em agentes.

É importante ressaltar que os sistemas de agentes móveis baseados em Java são semelhantes em diversos pontos: além da linguagem, todos eles utilizam a versão padrão da máquina virtual Java, o mecanismo de serialização de objetos oferecido pela linguagem, e uma arquitetura baseada em servidor. Entretanto, no que se refere aos mecanismos de transporte de agentes e ao suporte à interação (troca de mensagens) esses ambientes diferenciam-se entre si.

Embora a maioria dos sistemas de agentes móveis sejam baseados no sistema da linguagem de programação Java, pode-se encontrar também outros sistemas baseados em outras linguagens. As mais importantes segundo Cockayne[16] são: Tcl, Scheme e Python.

O ***Agent Tcl***[8], desenvolvido pelo Dartmouth College é um sistema de agentes móveis que possui serviços de comunicação, mecanismos de segurança, ferramentas de *debugging* e *tracking*. Seu principal componente é um servidor que roda em cada máquina e que permite o estado de execução completo, incluindo variáveis locais e ponteiro de instrução para a migração. Quando um agente deseja migrar para uma outra máquina, o servidor da origem chama a função *agent_jump* que automaticamente captura o estado completo do agente e envia esta informação para o servidor de destino. O servidor de destino inicia uma execução Tcl, carrega a informação de estado do agente no ambiente de execução e reinicia a execução do mesmo a partir do ponto exato de onde ele interrompeu o cumprimento de sua tarefa.

[7] Disponível em: http://www.objectspace.com/voyager/index.html

[8] Disponível em: www.cs.darmouth.edu/~agent/

O ***Ara***[9], desenvolvido pela Universidade de Kaiserslautern, é uma plataforma baseada em Tcl para a execução segura de agentes móveis em redes heterogêneas. O projeto está mais relacionado ao suporte a sistemas de agentes móveis para uma execução segura e portável do que para as características a nível de aplicação, tais como: padrões de cooperação entre agentes, comportamento inteligente e modelagem de usuário.

O sistema ***TACOMA***[10] foi desenvolvido pela Universidade de Tromsϕ e Cornell. Implementado em C, este sistema é baseado em UNIX e no TCP, e suporta agentes escritos em C, Tcl/Tk, Perl, Python e Scheme. A plataforma oferece suporte ao desenvolvimento de sistemas de agentes móveis que podem ser utilizados na resolução de problemas tradicionalmente endereçados aos sistemas operacionais.

No capítulo 4, é fornecida uma descrição detalhada do sistema ***Aglets Software Development Kit (ASDK)***, ambiente usado na implementação do presente trabalho.

## 2.5 - Orientação a Objetos e Java

Esta seção apresenta as principais características da tecnologia de orientação a objetos e as vantagens e benefícios que esta tecnologia proporciona. Em seguida, é apresentada também a linguagem Java e as características da tecnologia de orientação a objetos suportada por esta linguagem, além de explicitar o motivo da utilização da mesma para o desenvolvimento deste trabalho.

### 2.5.1 - Orientação a Objetos

[9] Disponível em: http://www.uni-kl.de/AG-Nehmer/Projekte/Ara/index_e.html
[10] Disponível em: http://www.tacoma.cs.uit.no/

O termo "orientação a objetos" ou "baseado em objetos" significa que o software é organizado como uma coleção de objetos separados que incorporam tanto a estrutura quanto o comportamento dos dados. Esta concepção difere, em parte, da programação convencional, segundo a qual a estrutura e o comportamento dos dados têm pouca vinculação entre si. A abordagem baseada em objetos exige algumas características que incluem quatro aspectos fundamentais: identidade, classificação, polimorfismo e herança [78].

***Identidade*** significa que os dados são subdivididos em entidades discretas e distintas, denominadas objetos. Os objetos podem ser concretos, como um arquivo em um sistemas de arquivos, ou conceituais, como uma norma de escalonamento em um sistema operacional multiprocessado. No mundo real, um objeto limita-se a existir, mas, no que se refere a uma linguagem de programação, cada objeto dispõe de um único identificador pelo qual ele pode ser referenciado. O identificador pode ser implementado como um endereço, como um elemento de uma matriz ou como um valor exclusivo de um atributo. As referências dos objetos são uniformes e independentes do conteúdo.

***Classificação*** significa que os objetos com a mesma estrutura de dados (atributos) e o mesmo comportamento (operações ou métodos) são agrupados em uma classe. Uma classe descreve propriedades importantes para uma aplicação. A escolha de uma classe é arbitrária e depende da aplicação. Cada classe descreve um conjunto de objetos individuais. Cada objeto é dito ser uma instância de sua classe. Cada instância da classe tem seu próprio valor para cada atributo, mas compartilha os nomes de atributos e operações com outras instâncias da mesma classe.

***Polimorfismo*** significa que a mesma operação pode atuar de maneiras diferentes em classes diferentes. Uma operação é uma função de transformação que um objeto executa ou a que ele está sujeito. Uma implementação específica de uma operação por uma determinada classe é chamada de método. Como um operador baseado em objetos é polimórfico, pode haver mais de um método para sua implementação. Cada objeto sabe como executar suas próprias operações. A linguagem de programação orientada a

objetos seleciona automaticamente o método correto para implementar uma operação com base no nome da operação e lista de argumentos na classe do objeto que esteja sendo operado.

***Herança*** é o compartilhamento de atributos e operações entre classes com base em um relacionamento hierárquico. Uma classe pode ser definida de forma abrangente e depois refinada em sucessivas subclasses mais definidas. Cada subclasse incorpora (herda) todas as propriedades de sua superclasse e acrescenta suas próprias e exclusivas características. Esta é uma das principais vantagens da programação orientada a objetos, pois permite o reaproveitamento de código.

### 2.5.2 - Características da Tecnologia Orientada a Objetos

Existem diversas características fundamentais na tecnologia orientada a objetos, embora essas características não sejam exclusivas dos sistemas orientados a objetos [78].

A ***abstração*** consiste em buscar apenas os aspectos essenciais de uma entidade ignorando suas propriedades acidentais. Ou seja, concentrar-se no que o objeto é e faz, antes de decidir como ele deve ser implementado. O uso da abstração preserva a liberdade de se tomar decisões mais importantes, evitando, tanto quanto possível, preocupações com detalhes prematuros.

O ***encapsulamento*** consiste na separação dos aspectos externos de um objeto, acessíveis por outros objetos, dos detalhes internos da implementação daquele objeto, que ficam ocultos dos demais objetos. O encapsulamento impede que um programa se torne tão independente que uma pequena modificação possa causar grandes efeitos de propagação. A implementação de um objeto pode ser modificada sem que isso afete as aplicações que o utilizam. O encapsulamento não é exclusivo das linguagens orientadas a objetos, porém a capacidade de combinar estruturas de dados e seu comportamento em uma única entidade torna-a mais completa e mais poderosa do que as linguagens convencionais.

***Compartilhamento***. A herança da estrutura de dados e do seu comportamento permite que a estrutura comum seja compartilhada por diversas subclasses semelhantes sem redundâncias. O compartilhamento de código é uma das principais vantagens das linguagens orientadas a objetos. Mais importante que a redução do trabalho de codificação é a clareza conceitual proveniente de reconhecimento de que diferentes operações são, na realidade, a mesma. A possibilidade da reutilização de modelos e códigos em projetos futuros é enfatizada como uma justificativa para a tecnologia orientada a objetos. O desenvolvimento baseado em objetos fornece ferramentas como abstração, o encapsulamento e a herança que permitem "montar bibliotecas" de componentes reutilizáveis.

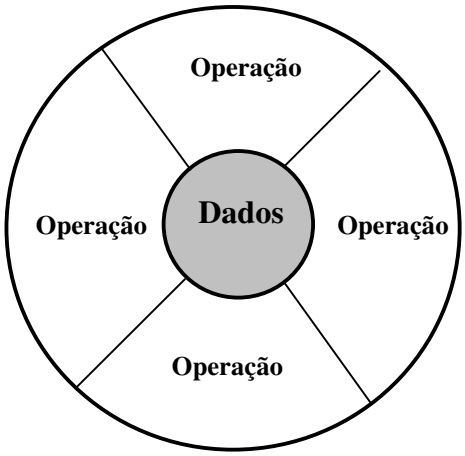


Figura 2 - Estrutura de um Objeto

***Ênfase na estrutura de objetos*** e não na estrutura de procedimentos significa preocupar-se em especificar o que um objeto é, e não como ele é utilizado. A figura 2 mostra a estrutura de um objeto. O uso de um objeto é altamente dependente dos detalhes da aplicação, os quais, freqüentemente, mudam durante o desenvolvimento. O desenvolvimento baseado em objetos coloca maior ênfase na estrutura dos dados e menor ênfase na estrutura de procedimentos, diferente do desenvolvimento tradicional baseado na decomposição funcional.

### 2.5.3 - A Linguagem Java

Java é uma linguagem de programação desenvolvida pela Sun Microsystems, cujo objetivo foi criar uma linguagem orientada a objetos dinâmica para uso nos mesmos tipos de aplicações desenvolvidas em C e C++, mas sem as dificuldades e os erros mais comuns destas linguagens. Conforme a experiência dos projetistas da linguagem, foram realizadas escolhas que adicionaram novas características ou suprimiram outras do C++, visando criar uma linguagem que facilitasse a geração de código robusto, confiável, facilmente utilizável em plataformas diversas.

Estas escolhas levaram alguns a chamarem Java de "C++ ++ --", onde o segundo ++ representaria as inovações (g*arbage collection, multithreading)* e o -- indicam as restrições (supressão da aritmética de ponteiros).

A linguagem Java é inteiramente orientada a objetos, de tal maneira que apenas os números não são objetos. Todas as demais representações de entidades são feitas por meio de objetos.

O paradigma de programação orientada a objetos já se consagrou como sendo adequado para aplicações dos mais variados portes, principalmente aplicações de maior complexidade como os encontrados em ambientes de rede/programação distribuída. A facilidade para o reaproveitamento de código também colabora para o aumento de produtividade.

A linguagem Java fornece inúmeras bibliotecas de classes, tornando possível o desenvolvimento de aplicações completas apenas utilizando estas classes básicas (que formam o chamado *Java Core*). Como exemplo destas bibliotecas de classes, têm-se: interface com redes; interface gráfica; interface com banco de dados (JDBC).

A Sun caracteriza Java como uma linguagem simples, orientada a objetos, distribuída, *multithread* e dinâmica. O objetivo desta seção é dar uma visão mais geral das características de orientação a objetos que a linguagem suporta. Maiores detalhes e exemplos mais aprofundados específicos de Java podem ser vistos em [37], [56],[94] e [96].

### 2.5.3.1 - Independência de Plataforma

Independência de Plataforma é, provavelmente, a característica mais marcante da linguagem Java. Desdobra-se em independência da arquitetura de *hardware* e independência do sistema operacional onde o programa deve ser executado.

A linguagem Java aborda a questão de independência de plataforma em diferentes níveis descritos a seguir.

- ***tipos de dados e semântica*** - as linguagens C e C++ apresentam em sua definição formal alguns aspectos que deixam margem a interpretação, permitindo que fabricantes diferentes utilizem interpretações diferentes, provocando uma incompatibilidade entre os *softwares* desenvolvidos em diferentes plataformas. Um exemplo é a precisão de números em diferentes arquiteturas. Uma simples pergunta como "quantos bytes tem uma variável do tipo int na linguagem de programação C" não pode ser respondida sem o conhecimento da plataforma na qual se pretende executar o código C. Na linguagem Java, qualquer que seja a plataforma, os tipos de dados tem um formato definido e único, como exemplificado na tabela abaixo:

| Tipo de Dado | Representação |
|---|---|
| Boolean | 1 bit |
| Byte | 8 bits |
| Char | 16 bits (*unsigned*) |
| Short | 16 bits |
| Int | 32 bits |
| Long | 64 bits |
| Float | 32 bits |

Tabela 1 - Tipos de Dados

Outro possível ponto de conflito é na interpretação da semântica da linguagem, como, por exemplo, a resolução de argumentos de uma função ou a seleção de

métodos sobrecarregados. Na definição da linguagem Java tais aspectos não deixam margem a interpretações conflitantes.

- ***Java virtual machine -*** principal meio pelo qual é atingida a independência de plataforma na linguagem Java. O alvo do compilador Java não é o *assembler* de um processador específico, pois isto limitaria o código gerado a uma determinada arquitetura. O compilador gera o código aberto (chamado *bytecodes*) para um processador hipotético, para a "*Java Virtual Machine*" (Máquina Virtual Java ou JVM). As implementações da JVM nas diversas combinações de plataformas de *hardware*/*software* possibilitam a execução dos *bytecodes*. As primeiras JVMs implementavam na prática um interpretador de *bytecodes*, acarretando conseqüentemente num pior desempenho na execução do código. Atualmente já estão disponíveis ambientes onde antes de serem executados, os *bytecodes* são compilados para o código objeto da plataforma de execução (são os chamados *Just in Time Compilers*). Desta maneira, as perdas na velocidade de processamento são consideravelmente reduzidas.

#### 2.5.3.2- Robustez

Outro requisito desejável numa linguagem de programação é a sua robustez. Em linhas gerais, robustez pode ser definida como a capacidade do software executar corretamente a função para a qual foi programada, e nada além disso.

Num ciclo tradicional de desenvolvimento de *software*, as metodologias adotadas pretendem garantir a robustez, além da obrigatória fase de testes. Mesmo com os mais rigorosos procedimentos, podem ocorrer casos em que algum erro acabe passando desapercebido. Estes pequenos erros podem ficar latentes no meio do código, sendo despertados apenas em condições excepcionais. Entretanto, ao serem ativados, podem levar ao funcionamento imprevisível do *software* e a conseqüências imprevisíveis.

A flexibilidade de C / C++ e sua rapidez baseiam-se em parte na confiança de que o programador tem absoluto controle sobre o que o *software* estará realizando em

tempo de execução. Apesar disso, boa parte dos programadores C / C++ já devem ter passado pela situação onde horas/dias são gastos para determinar que o comportamento "estranho" do *software* era devido a um ponteiro mal comportado que acabava invadindo indevidamente uma área de memória.

Inúmeras das diferenças entre C / C++ e Java resultam de decisões que visaram tornar a linguagem Java mais robusta. Entre elas, as principais são descritas a seguir.

- ***ausência de aritmética de ponteiros -*** ponteiros são uma das características da linguagem C / C++ que tem grande potencial "destrutivo" num eventual erro de programação. Tal fato ocorre em virtude dos ponteiros permitirem que se acesse, praticamente, qualquer posição da memória, principalmente, por engano ou ainda com "segundas" intenções. Para enfrentar este problema, a aritmética de ponteiros foi abandonada na linguagem Java. Todos os acessos às posições de memória (que, na prática, são as propriedades dos objetos) são realizados pelos métodos fornecidos pelos objetos, e não diretamente.
- ***verificações em tempo de compilação -*** as restrições da linguagem Java são verificadas na compilação do código. Entre outras pode-se destacar a checagem das classes de objetos manipulados, pois em Java não é realizada a conversão automática de tipos.
- ***verificações em tempo de execução -*** outra deficiência da linguagem C / C++ é quanto a verificações em tempo de execução. Nestas linguagens, fica a cargo do programador a responsabilidade de realizar tais verificações, possibilitando o funcionamento incorreto do sistema e até paralisações caso alguma situação errônea não tenha sido devidamente tratada. Um exemplo seria o acesso a um vetor além dos limites originalmente determinados. A linguagem Java realiza uma série de verificações em tempo de execução automaticamente, sendo que situações de erro causam exceções. O tratamento de tais exceções pode ser definido pelo programador, realizando as ações adequadas ou deixadas a cargo do ambiente Java. A contrapartida para este nível maior de controle é o sacrifício no desempenho (menor velocidade de processamento).
- ***garbage collection -*** outra parcela significativa de problemas em aplicações são decorrentes de erros na alocação e desalocação de memória para os objetos. É

comum liberar o espaço de um determinado objeto e posteriormente referenciar inadvertidamente tal objeto, quando isto já não deveria ser mais possível. Para contornar este problema, o ambiente Java encarrega-se de determinar quando o objeto alocado pode ser liberado (processo conhecido como *garbage collection*), liberando o programador de tais preocupações (e eventuais erros).

#### 2.5.3.3 - Segurança

A segurança é um aspecto fundamental para a linguagem Java. Na principal aplicação desta linguagem (distribuição de *applets* embutidas em páginas WWW para serem executadas nas máquinas dos usuários), o usuário estará recebendo um código através de uma rede de dimensões mundiais (a Internet). Neste momento, surge a questão de como ter certeza de que não se está trazendo um vírus ou um cavalo de tróia para dentro do computador?

A linguagem Java especifica um modelo de segurança para enfrentar esta questão. O modelo de segurança Java é uma característica chave da arquitetura da linguagem que a torna uma opção apropriada para ambientes de redes de computadores [92]. Segurança é importante porque estes ambientes permitem um potencial ataque a partir de qualquer computador que tenha acesso à rede. Estas preocupações tornam-se especialmente fortes quando programas (*software*) são trazidos através da rede e executados localmente, como ocorre com *applets* Java. É provável que navegando pela rede o usuário encontrará *applets* de origem não confiável. Para enfrentar este ambiente potencialmente hostil, os mecanismos de segurança da linguagem Java estabelecem uma distinção de tratamento dependendo da origem do código a ser executado.

O modelo de segurança Java propõe-se a proteger o usuário de programas hostis trazidos pela rede de fontes não confiáveis. Para tanto, o ambiente Java utiliza-se de uma *sandbox* dentro da qual o programa Java é executado. O programa pode realizar qualquer coisa dentro dos limites da *sandbox*, mas não consegue realizar nenhuma ação fora de seus limites. *Applets* Java trazidos a partir da rede não podem realizar inúmeras ações, entre elas:

- ler ou escrever no sistema de arquivos da máquina do usuário;
- estabelecer conexões de rede para qualquer servidor, a não ser com o servidor do qual o *applet* foi carregado;
- criar novos processos;
- carregar dinamicamente novas bibliotecas e chamar diretamente métodos nativos.

Ao evitar que o código trazido através da rede execute certas operações, o modelo de segurança Java pretende proteger o usuário de programas hostis, facilitando o tratamento de *software* originado de fontes não confiáveis. Ao invés de exigir que o usuário se preocupe em evitar que esse tipo de software chegue ao seu computador, o modelo de *sandbox* permite que um código de qualquer fonte seja executado; mas, durante a execução, a *sandbox* evita que o código de origem não confiável realize qualquer ação que possivelmente poderia comprometer a segurança do sistema.

Para assegurar o funcionamento adequado da *sandbox*, o modelo de segurança Java envolve diversos aspectos da arquitetura da linguagem. Se houver áreas na arquitetura Java na qual a segurança for fraca, um programador mal-intencionado potencialmente poderia explorar esta área para contornar as restrições da *sandbox*.

#### 2.5.3.4 - Java e Banco de Dados

Na era da informação, o banco de dados é uma ferramenta indispensável para coletar e manipular dados. No entanto, o problema do armazenamento e recuperação de informações é complexo devido à heterogeneidade natural dos computadores nas diversas companhias [39].

Para conseguir a conectividade com os diversos tipos de banco de dados, Java oferece diversos benefícios para desenvolvedores criarem aplicações *front-end* para servidores de banco de dados. Esta linguagem possui a capacidade de criar aplicativos robustos independentes de plataforma e *applets* básicos prontos para a *Web.* Os

desenvolvedores consideram a utilização de Java um passo importante para desenvolver soluções de conectividade *fron-end* para todas as plataformas [39].

Assim, para escrever aplicações que acessem banco de dados, foi criado um pacote contendo uma série de interfaces que podem ser utilizadas para a conexão com o banco de dados. Essas interfaces receberam o nome de API JDBC.

O *Java Database Connectivity* é uma API (*Application Programatic Interface*) que possibilita a adição de comandos SQL (*Structured Query Language*) em uma aplicação Java, ou seja, é um conjunto de classes para integração da linguagem com bancos de dados relacionais.

A API JDBC foi projetada para permitir aos desenvolvedores criarem *front-ends* para banco de dados sem ter que reescrever código. Apesar do padrão lançado pelo comitê ANSI, cada sistema de banco de dados comercializado tem um modo particular de conexão, e, em alguns casos, de comunicação com seus sistemas [39].

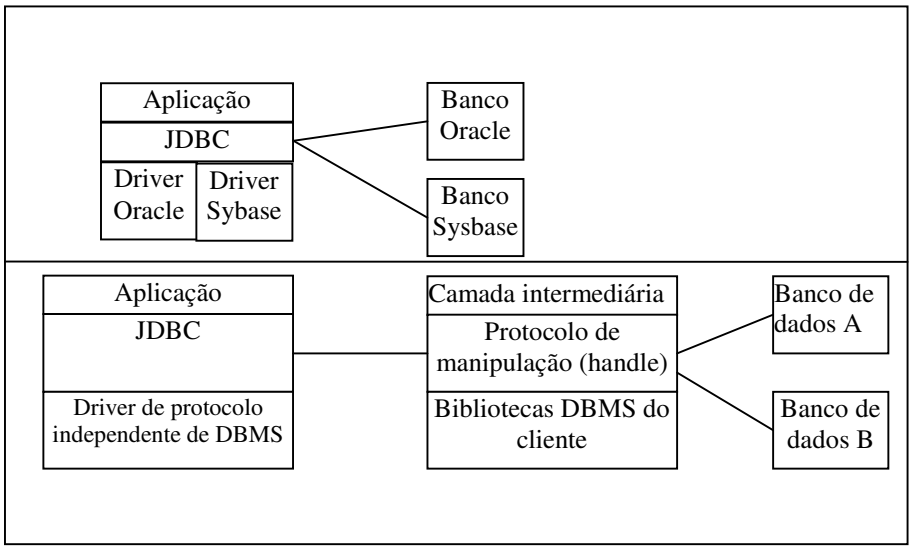


Figura 3: Projeto do banco de dados JDBC [69]

A API JDBC provê ao desenvolvimento de aplicações uma simples API que é uniforme e independente de banco de dados. A API provê um padrão para escrever, e um padrão que conversa com todos os diversos *designs* encontrados. O segredo é uma série de interfaces em Java que são implementadas em um *driver*. Este *driver* comunica-

se com o banco de dados, sendo responsável pela adequação dos comandos em JDBC em instruções correspondentes suportadas pelo banco de dados. Na figura 3, é demonstrada uma aplicação que foi escrita uma vez e utilizada com os diversos *drivers*. Os *drivers* são utilizados para desenvolver a camada intermediária em um projeto de banco de dados multi-camada [39].

Conforme demonstrado na figura 3, o driver JDBC faz a ligação entre a aplicação e o banco de dados. No primeiro caso, apresentado na parte superior da figura , tem-se o driver JDBC acessando diretamente o banco de dados Oracle e Sysbase. Já na parte inferior da figura, o driver JDBC faz solicitações ao banco de dados utilizando o protocolo da rede para acessar os bancos de dados.

Os drivers JDBC devem suportar ANSI SQL-2 Entry Level, mas o JDBC permite aos desenvolvedores passar uma expressão de seleção diretamente para o driver conectado podendo ou não ser ANSI SQL, ou SQL (*Structured Query Language*) [39].

A especificação do JDBC foi baseada no padrão X/Open SQL Call Level Interface (CLI), deixando-o muito parecido com um driver ODBC (*Open Database Connectivity*), que também foi baseado neste padrão. A principal diferença é que o ODBC é uma implementação em linguagem C nativa enquanto que o JDBC pode ser implementado independentemente de plataforma, seguindo os ideais da linguagem Java. Java provê, ainda, uma ponte entre JDBC-ODBC, que traduz de JDBC para ODBC. Esta implementação, executada como método próprio ("nativo"), é pequena e eficiente [39].

Em geral, existem dois níveis de interface na API JDBC: a camada de aplicação, onde o desenvolvedor usa a API para fazer chamadas para o banco de dados via SQL e recebe o resultado, e a camada de *driver*, que manipula toda a comunicação na implementação de um *driver* específico [39].

Todas as aplicações de JDBC (ou *applet*) devem ter no mínimo um *driver* JDBC, e este driver é específico para o tipo de DBMS (*Database Management Systems*) usado.

Este *driver*, porém, não precisa estar diretamente associado a um único banco de dados [39]. O *driver* da ponte JDBC-ODBC é único e funciona para mais tipos de bancos de dados, desde que suporte chamadas ODBC.

A estrutura do JDBC inclui dois conceitos chaves :

- ser uma interface independente de qualquer Sistema de Gerenciamento de Banco de dados (DBMS), permitindo um acesso genérico a banco de dados através de SQL, e ter uma interface uniforme para diferentes fontes de dados;
- o programador deve escrever apenas uma interface para interagir com um banco de dados relacional e, utilizando o JDBC, o programa poderá acessar qualquer fonte de dados sem ter uma programação adicional para realizar esta operação.

## 2.6 - Conclusão

Para que o desenvolvimento deste projeto fosse satisfatório, foi identificada a necessidade do conhecimento de conceitos pertinentes às tecnologias de Cliente/Servidor, Inteligência Artificial Distribuída, Orientação a Objetos e Java. Cada uma delas foi utilizada para permitir uma funcionalidade diferente no modelo virtual proposto.

O modelo de processamento cliente/servidor tornou-se rapidamente uma das abordagens mais utilizadas na indústria da computação. A evolução da informática possibilitou a várias pessoas compartilharem recursos distribuídos como informações, periféricos, serviços e aplicações. O compartilhamento de recursos quando executado corporativamente, através da interação entre clientes e servidores, pode será alcançado utilizando-se o modelo cliente/servidor.

Entretanto, conforme vão aumentando a quantidade de processos a serem executados, escalonados e gerenciados, da mesma maneira aumenta a complexidade da arquitetura e dos sistemas [68].

Sistemas baseados em cliente servidor têm basicamente duas grandes desvantagens descritas a seguir.

- ***complexidade -*** o desafio maior está em dispor todos os recursos a trabalhar em conjunto. Computadores pessoais, por exemplo, poderão ter uma rede com diversos *drivers* de ODBC (*Open Database Connectivity*) instalados, que farão pesquisas em diversos bancos de dados. Além disso, existem ainda muitos problemas de comunicação da aplicação, principalmente com versões de ODBC erradas e camadas de redes incompatíveis. Ao aperfeiçoar um sistema, o problema está em distribuir a nova versão do sistema a todos os usuários, principalmente quando esta mudança implica em mudanças no banco de dados. Assim, fatores como a máquina de cliente, o *software* de cliente, a camada intermediária do banco de dados, o servidor, os mecanismos de conexão do servidor, as aplicações do servidor e a rede são alguns do recursos que podem gerar conflitos;
- ***comunicação -*** em um ambiente de central/terminal, a linha de comunicação é usada para todo fluxo de dados. Não é usada, porém, para selecionar dados, atualizar tabelas, ou achar respostas. Todo este trabalho é executado no servidor e só a exibição resultante é enviada através da linha. No entanto, para qualquer solicitação, haverá uma série de mensagens trocadas. Aplicações pesadas, nas quais existem muitas consultas e exibições tendem a carregar a linha, tornado o aplicativo mais lento [68].

Foram apresentados também neste capítulo, os fundamentos da Inteligência Artificial Distribuída, iniciando pela exposição das características clássicas associadas à abordagem de resolução distribuída de problemas e à abordagem de sistemas multi-agentes. O assunto é bastante vasto. Para maiores detalhes sobre o tema, uma coletânea dos principais artigos clássicos é dada por [10]. Uma visão sistematizada sobre a literatura pode ser encontrada em [91]. Por fim, uma referência que recebeu uma boa acolhida da comunidade de IAD é [60].

Em seguida, procurou-se dar um maior enfoque à tecnologia de agentes móveis, ressaltando sua fundamentação teórica. Foram abordados os conceitos mais relevantes,

considerando os objetivos da presente dissertação. Uma abordagem mais detalhada pode ser encontrada nas principais referências indicadas pela literatura [16], [26].

Finalmente, foram abordados os conceitos mais relevantes relativos ao paradigma da orientação a objetos e à linguagem de programação Java. Uma abordagem mais detalhada pode ser encontrada nas principais referências indicadas pela literatura [78], [15], [18], [56].

Cabe ressaltar que a partir das características desta linguagem de programação, verificou-se a adequação de seu emprego na implementação deste trabalho. Java foi escolhida para a implementação deste trabalho por possuir algumas peculiaridades que a tornam propícia para o desenvolvimento de software na Internet. Outras características que também influenciaram na escolha foram a sua simplicidade (a alocação e o gerenciamento memória são problemas com os quais o programador Java não se preocupa) e sua portabilidade, característica considerada primordial no ambiente Internet. Deve-se também destacar a importância do *Java Database Connectivity* como um mecanismo importante para a linguagem estender suas aplicações para o mundo das bases de dados.

# CAPÍTULO 3:

## *Arquitetura e Implementação do Sistema Data Agents*

## 3.1 - Introdução

Nos últimos anos, a Internet tem se tornado a mídia preferida para transmissão e disseminação de informações, acesso a dados e comunicação pessoal. Inúmeras aplicações distribuídas e multi-plataformas como comércio eletrônico, bibliotecas digitais, educação à distância e sistemas hipermídia vêm ocupando espaço cada vez mais importante na Web.

No entanto, apesar destas facilidades, o crescimento exponencial da rede traz um grande problema - a sobrecarga de informações e de trabalho, dificultando a execução das atividades na mesma. Sendo assim, a popularidade da Web como um mecanismo de acesso universal à informação criou a necessidade de desenvolvimento de aplicações cliente/servidor baseadas na *Web*, pois à medida que a quantidade de informação em uma rede cresce, torna-se mais eficiente processar, consultar e atualizações no sistema computacional onde o dado está localizado. Este processamento geralmente é realizado através de uma interface cliente-servidor. Neste esquema, geralmente, o cliente fica limitado às consultas suportadas pelo *host* ou, então, exige que o servidor envie todos os dados para que o cliente possa realizar o processamento, aumentando muito o tráfego em uma rede.

Como alternativa para solucionar este problema, pode-se fazer uso da emergente tecnologia de agentes. Os agentes dotados de mobilidade, representando remotamente aplicações e usuários, propiciam vantagens para certas aplicações como a redução do tráfego na rede, maior flexibilidade, possibilidade de balanceamento de carga e processamento assíncrono, permitindo assim o desenvolvimento de soluções mais eficientes para problemas relacionados a comércio eletrônico, computação móvel, gerência remota de recursos, *data mining*, entre outros.

Agentes móveis é uma nova tecnologia para aplicações distribuídas, suportando ainda a distribuição de processos pelos elementos da computação móvel. São programas que se movem (ou são movidos) pela rede, migrando de servidores para servidores, interagindo com os recursos de cada elemento conectado à rede ou com outros agentes que estejam presentes nestes.

Este capítulo descreve a arquitetura e a implementação do Data Agents, um sistema de recuperação de informações estruturadas e distribuídas, estabelecendo as vantagens de se aliar as funcionalidades oferecidas pelos sistemas gerenciadores de banco de dados à capacidade de migração de objetos. Este sistema foi desenvolvido segundo o paradigma de orientação a objetos com uma linguagem de programação também orientada a objeto. Atualmente, existem diversas linguagens de programação orientadas a objeto, assim como, várias plataformas de desenvolvimento de agentes móveis. A linguagem utilizada para a implementação do sistema Data Agents foi Java, da Sun Microsystems Inc., e o ambiente de desenvolvimento de sistemas agentes móveis escolhido foi ***Aglets Software Development Kit (ASDK)*** versão 1.1b1, da IBM.

## 3.2 - Cenário Eletrônico de Compra

Este trabalho foi elaborado com base em um experimento realizado para investigar e testar a conveniência de se utilizar agentes móveis em um ambiente distribuído e multi-plataforma para produzir uma solução de pedido de compra em um mercado global. O cenário é baseado em um processo bastante convencional e freqüente na vida de muitas pessoas atualmente: busca de livros em lojas virtuais; pois na área do comércio eletrônico, a obtenção de informações rápidas e concisas é uma necessidade freqüente, a qual promove uma significativa redução de custos e tempo.

Os agentes considerados no contexto deste trabalho são agentes que ajudam, de alguma forma, o usuário a fazer compras através da Internet. Este tipo de agente, designado por agente de compras, pode desempenhar diversas tarefas, como por exemplo: ajudar o usuário a decidir que produto comprar; fazer sugestões, com base no

seu conhecimento sobre seu usuário; descobrir novidades, descontos e preços especiais; encontrar as lojas que vendem o produto ou serviço desejado, entre outras coisas.

Das possíveis funções descritas acima, o protótipo Data Agents pretende ajudar a encontrar as lojas que vendem o produto desejado e a listar os preços dos produtos encontrados. Sendo assim, o objetivo do trabalho proposto é tornar o processo de compras através da Internet mais fácil para o usuário, já que torna desnecessário a procura de vários fornecedores para o produto pretendido repetidas vezes. Além disso, uma vez que o processo de procura do produto passa a ser substancialmente mais rápido, o tempo disponível para fazer compras deixa de ser um fator limitativo e o usuário passa a encontrar mais vezes o produto almejado com o melhor preço.

Para alcançar este objetivo, o funcionamento do protótipo é o seguinte:

- o usuário escolhe o tipo de produto, especifica as características desejadas para esse produto (características essas que serão as restrições às condições de procura);
- o agente de compras procura, entre os produtos desse tipo, aqueles com as características desejadas;
- como resultado de pesquisa, o sistema Data Agents envia um e-mail ou mostra uma tela ao usuário com uma lista de produtos, seus respectivos preços e onde encontrá-los.

## 3.3 - Propriedades do Sistema Data Agents

As propriedades dos agentes apresentam-se em níveis menores ou maiores dentro de uma sociedade, dependendo da arquitetura montada. Estas propriedades, no entanto, podem ser ampliadas, de acordo com o problema a ser resolvido. As propriedades neste sistema são:

- ***cooperação -*** os agentes, dentro da sociedade, buscam cooperar, compartilhando as informações que cada um possui para atender o objetivo que a sociedade necessita alcançar em conjunto;
- ***autonomia -*** cada agente, dentro da sociedade, constrói sua própria agenda, e é capaz de persegui-la independente do usuário;
- ***veracidade -*** nesta sociedade, os agentes são verídicos, uma vez que não conseguem comunicar informações falsas porque sua arquitetura não permite;
- ***racionalidade -*** todos os agentes atuam de forma a atender os objetivos para os quais foram criados, e por sua dependência, eles não deixarão de ser alcançados;
- ***benevolência -*** nesta sociedade, cada agente possui o seu papel e há um agente que coordena as atividades, de forma que os objetivos não entrem em conflito e os agentes tentem realizar o que foi pedido.

## 3.4 - Ambiente de Desenvolvimento

O ***Aglets Software Development Kit***[11] ***(ASDK)***, desenvolvido pelo laboratório de pesquisa da IBM em Tóquio, é um ambiente para o desenvolvimento de sistemas de agentes móveis que estende o modelo de *applets* de Java: ( **aglet = ag**ent + appl**et**). A plataforma provê um conjunto de classes usadas para descrever agentes, mensagens, itinerários, identificadores de agentes, contextos, etc. Este ambiente baseia-se, principalmente, na característica multiplataforma de Java que suporta a migração e a execução de código entre computadores interligados através da Internet.

O ASDK foi escolhido para o desenvolvimento deste projeto por várias razões :

- utiliza a linguagem de programação JAVA;
- implementa a capacidade de mobilidade dos agentes entre os computadores ligados à Internet;
- apresenta facilidades de segurança: para um agente mover-se para outra máquina, exige-se que a máquina hospedeira rode um servidor de *aglet*, solucionando alguns dos problemas relacionados à segurança;

- suporta comunicação assíncrona e síncrona entre agentes locais e/ou remotos;
- implementa a noção de itinerários.

No ambiente ASDK, um *aglet* é um objeto em Java que pode mover-se de um *host* para outro pela Internet, ou seja, um *aglet* que roda em um *host* pode parar sua execução repentinamente, migrar para um *host* distante e retomar a execução ao chegar ao seu destino. Quando o *aglet* se move, leva seu código de programa bem como seu estado (dados) [62]. Conforme definição de Bret Sommers [86], um *aglet* é considerado um agente móvel pois é um objeto que possui comportamento, estado e localização.

*Aglet* também é definido como um agente de *software* autônomo baseado em Java. Os agentes são autônomos porque decidem para onde ir e o que fazer. Eles podem receber pedidos de fontes externas, como outros agentes, mas cada um decide se quer ou não obedecer a estes pedidos externos.

A seguir, será apresentada com mais detalhes a tecnologia de *aglets*: as principais classes e interfaces, o ciclo de vida e a forma de comunicação. Além disso, são descritos o protocolo utilizado para transferir os aglets e algumas questões relativas à segurança na utilização dessa tecnologia.

### 3.4.1 – Aglets Software Development Kit: Principais Classes

A principal vantagem do ASDK é que um *aglet* pode ser executado em toda a máquina que tenha instalado este pacote, não havendo necessidade de verificar qual é o *hardware* ou o sistema operacional existente, ou qual é a natureza da implementação particular do ASDK no *host* em que o *aglet* está executando [47].

[11] Disponível em: http://www.trl.ibm.co.jp/aglets

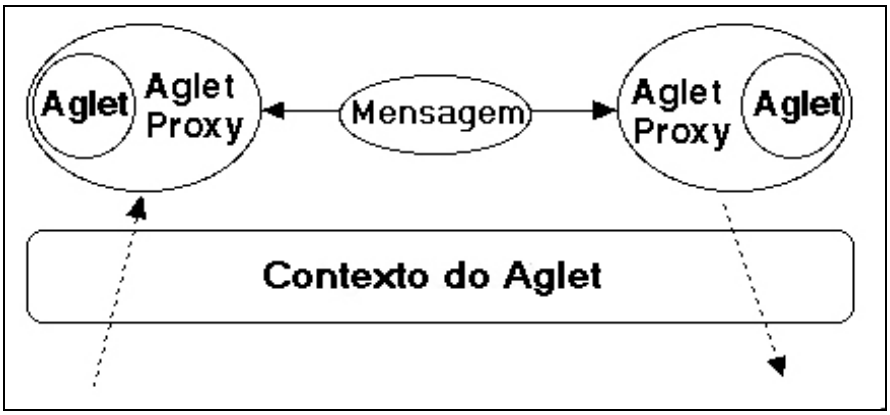


Figura 4 - Principais classes e interfaces

As principais interfaces e classes definidas neste pacote e a relação entre estas, ilustradas na figura 4, são descritas a seguir.

- ***aglet.Aglet*** - a classe abstrata *Aglet* define os métodos fundamentais para um agente móvel controlar a sua mobilidade e o seu ciclo de vida. Todos os *aglets* definidos devem ser instanciados a partir da classe abstrata *Aglet*, para que se possa ter acesso a todos os seus atributos.
- ***aglet.AgletID*** - cada instância de um *aglet* tem sua própria e única identidade que é imutável durante o seu ciclo de vida. Essa identidade consiste de alguns atributos como a identificação do usuário, o *host* de origem, entre outros.
- ***aglet.AgletProxy*** - a interface *AgletProxy* funciona como um manipulador de um *aglet* provendo um modo seguro de outros *aglets* acessá-lo. Esta interface proporciona, também, uma transparência de localização ao *aglet*. Por questões de segurança, quando um *aglet* deseja comunicar-se com um outro, este último deve obter, inicialmente, o objeto *proxy* do primeiro *aglet*, e então interagir utilizando esta interface. Dessa forma, o objeto *proxy* do *aglet* age como um objeto de proteção contra outros agentes maliciosos.
- ***aglet.AgletContext*** - um contexto pode ser considerado como um ambiente uniforme para o *aglet*, não importando se o *host* é um PC com Windows ou uma estação de trabalho Unix. Qualquer *aglet* pode obter uma referência para o objeto AgletContext. Essa referência pode ser usada para obter informações sobre o

local, como o endereço do contexto do *host* e a enumeração de AgletProxies, ou criar um *aglet* novo no contexto.

- ***aglet.Message*** - os objetos *aglets* comunicam-se trocando objetos da classe *Message*.
- ***aglet.FutureReply*** - o receptor, através dessa interface, pode determinar se uma resposta está disponível, ou esperar, por um intervalo de tempo especificado, pelo resultado. Se a resposta não for devolvida dentro do tempo determinado, o receptor pode continuar sua execução.

### 3.4.2 – Ciclo de Vida de um Aglet

A compreensão do conceito de um *aglet* é facilitada através de uma comparação a um ciclo de vida. Segundo Danny Lange [47], muitos eventos podem ocorrer durante o ciclo de vida de um *aglet*, e cada um deles é associado a um método da classe *Aglet* (vide figura 5). São eles: criação; clonagem; desalocação; disparo; restituição; desativação e ativação.

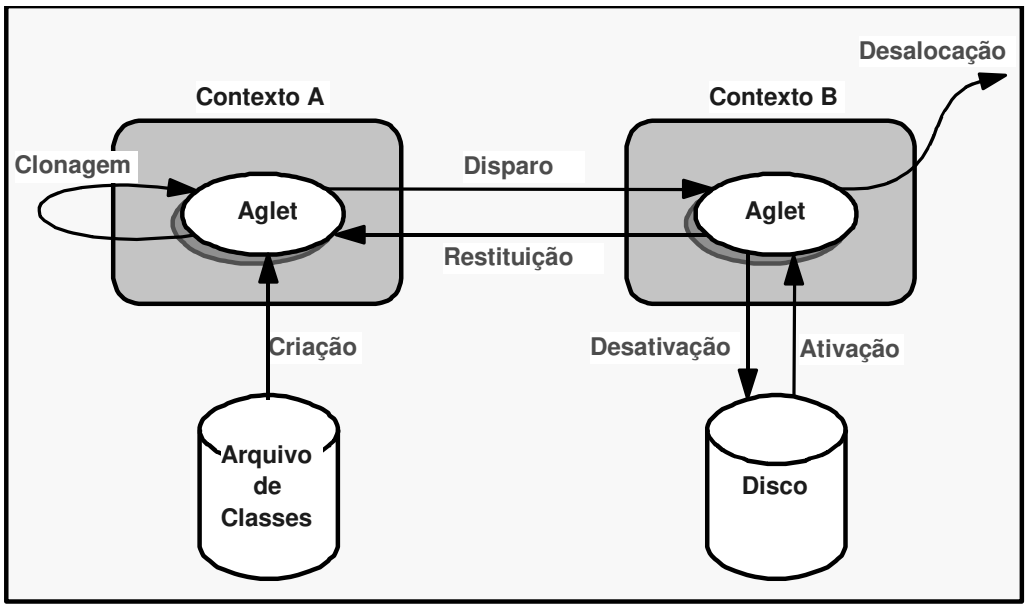


Figura 5: Ciclo de vida dos aglets

Um *aglet* pode ser criado de dois modos: instanciado a partir de uma classe de *aglet* ou clonado de um *aglet* existente. O *aglet* é criado em um contexto onde passa a maior parte de sua "vida". Quando um aglet se move de uma máquina para outra, na realidade, ele migra de um contexto para outro. Durante o processo de ***criação*** três métodos são chamados:

- o método construtor do *aglet* - só é chamado uma vez no ciclo de vida para criar um *aglet* não inicializado;
- o método *onCreation* () é chamado, também, apenas uma vez no ciclo de vida de um *aglet* para inicializar um *aglet* novo. Para personalizar a inicialização de um *aglet,* este método pode ser sobrescrito para executar determinadas instruções na criação do mesmo;
- o método *run()* descreve todos os procedimentos que um *aglet* deve repetir durante sua execução. Este método é invocado na criação, no disparo, na restituição, ou na ativação do *aglet*.

A ***clonagem*** é um modo alternativo de criar novos *aglets*. Uma chamada bem-sucedida do método *onClone()* em um determinado *aglet* cria uma cópia idêntica deste dentro do contexto corrente. Por razões de segurança, este método não devolve diretamente o clone, mas sim o AgletProxy, a fim de proteger os métodos públicos do novo aglet do acesso direto. [47].

Um *aglet* aloca vários recursos enquanto está em um contexto. Todos esses recursos são liberados após o término de suas tarefas. O sistema elimina todas as linhas que pertencem ao *aglet* que está sendo desalocado. Além do próprio contexto eliminar o *aglet*, o *garbage collector* do Java elimina a memória por ele alocada em virtude de todas as referências terem sido eliminadas, inclusive referências a outros *aglets*. A chamada do método *dispose* () conduz imediatamente à chamada do método *onDisposing()*, que pode ser usado para personalizar o processo de ***desalocação***. [47].

O método *dispatch()* permite a ***disparo*** de um *aglet* de um *host* para outro e possui como argumento o URL (*Uniform Resource Locator*) do *host* de destino. Além do *host* e do nome do domínio, a URL inclui informações a respeito do protocolo

particular usado para a comunicação com o servidor remoto. Nesse caso, é usado o protocolo ATP (*Agent Transfer Protocol*) para disparar os *aglets* [47].

O método *retract*(), responsável pela ***restituição*** de um aglet, solicita o retorno do *aglet* localizado no *host* de destino para o *host* de origem. [47]. Essa solicitação é feita de maneira assíncrona. Para executar a solicitação de retorno, é necessária a especificação da URL da máquina remota e a identificação do *aglet.* Todo *aglet* possui um identificador único, de modo que ele, estando em uma rede, possa ser acessado exclusivamente combinando seu identificador com o URL do *host* destino. A chamada para restituir um *aglet* distante conduz à execução do método *onReverting()*. Este último método pode ser sobrescrito para preparar o *aglet* para o retorno ou ser usado para prevenir o retorno [47]. A definição de um itinerário de "viagem" para o *aglet* é um outro modo de permitir a migração de agentes. O itinerário de destino pode ser simples, contendo apenas um endereço de um host, ou pode conter uma lista de *hosts* a serem visitados.

O *aglet* permite o seu armazenamento temporário em um meio secundário (***desativação***). Para tal, primeiramente, o contexto é armazenado e o *aglet* é informado sobre o intervalo de tempo que vai permanecer desativado. O período especificado é apenas uma sugestão para o sistema. A chamada do método *desative*() invoca, automaticamente, o método *onDeactivating()*. Este método é, usualmente, sobrescrito para permitir ao *aglet* terminar sua tarefa atual antes de ser desativado. A duração dessa desativação do *aglet* é indicada em milissegundos através do argumento de duração. Decorrido o intervalo de tempo especificado, ocorre a ***ativação*** do aglet pelo contexto. [47].

### 3.4.3 – Comunicação entre Aglets

Um *aglet* pode comunicar-se com outro através do envio e recebimento de objetos do tipo mensagem. Um objeto da classe mensagem tem como argumento, um atributo determinando o tipo da mensagem e um objeto arbitrário que é passado como

parâmetro. O *aglet* receptor pode tratar a mensagem através do método *Aglet.handleMessage(). Aglets* suportam os seguintes tipos de troca de mensagens:

- *now-type* – o tipo *now-type* é síncrono e bloqueia o *aglet* emissor até que o receptor complete o tratamento da mensagem; esse tipo de mensagem é implementado através do método *sendMessage(Message msg)* da classe *AgletProxy*;
- *future-type* - o tipo future-type é assíncrono e não bloqueia a execução do aglet emissor; o método do envio retorna um objeto do tipo FutureReply que pode ser usado para obter as respostas;
- *oneway-type* - esse tipo de mensagem também é assíncrono e não bloqueia a execução do aglet emissor; e difere do tipo future-type pelo fato do aglet receptor não precisar enviar uma resposta.

O *aglet* receptor deve definir o seu método *handleMessage(Message msg)* para tratar as mensagens que chegam. Neste método, o objeto mensagem é passado como argumento e pode ser usado para executar determinadas operações de acordo com o rótulo da mensagem. Se a mensagem for tratada, este método deve retornar um tipo *boolean* indicando se foi ou não bem executada. Em caso de retornar falso, o transmissor da mensagem recebe um aviso de erro de execução da mensagem (*NotHandledException).* No entanto, esse método não funciona para mensagens do tipo *Oneway.*

Todas as mensagens que chegam são basicamente armazenadas em um objeto de fila de mensagens, e então manipuladas uma a uma. As mensagens nos *aglets* seguem a lei de ordem de transmissão; isto é, as mensagens chegam na mesma ordem em que foram enviadas. Pode-se especificar ainda prioridades associadas com tipos de mensagens. As mensagens com alta prioridade são enfileiradas nas altas posições.

### 3.4.4 – Funcionamento Básico do Protocolo ATP (*Agent Transfer Protocol*)

O ATP, protocolo simples a nível de aplicação, é responsável por transmitir um agente de uma maneira independente do sistema. Ele define 4 (quatro) métodos padrão de solicitação para serviços de agentes, conforme ilustrado na figura 6:

- ***disparo ("dispatch")*** - uma solicitação de "*dispatch"* é enviada de A para B quando é necessário enviar um agente do serviço A para o serviço B;
- ***restituição ("retract")*** - uma solicitação de "*retract"* é enviada de A para B quando é necessário retornar um agente do serviço B para o serviço A;
- ***recuperação ("fetch")*** – uma solicitação de "*fetch"* é enviada de B para A quando o serviço B, para executar o agente, necessita recuperar o código executável da origem deste agente, ou seja, do serviço A; a resposta de A é uma mensagem cujo corpo contém o código de *status* e o código executável requerido.
- ***mensagem ("message")*** - uma solicitação de "*message"* é enviada de A para B quando um agente que está no serviço A deseja enviar uma mensagem para um agente que está no serviço B.

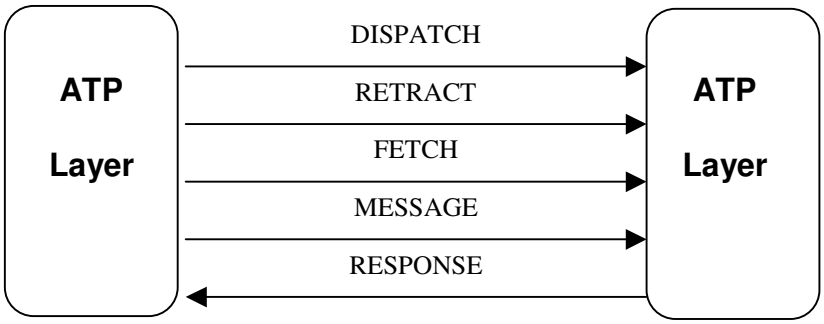


Figura 6 - Métodos "request" do ATP

### 3.4.5 - Segurança em Aglets

Dentro das configurações do servidor de *aglets*, é possível definir os níveis de acesso que os *aglets* criados em uma máquina possuem e os níveis de acesso dos *aglets* provenientes de outras máquinas. Nessa configuração, incluem-se níveis de acesso a arquivos do sistema, rede e outros dispositivos gerais como, por exemplo, habilitar

chamadas JDBC. Por padrão, o servidor de *aglets* disponibiliza apenas a possibilidade de mostrar janelas com avisos e a leitura das configurações do servidor de *aglets* instalado. Qualquer outra necessidade, é preciso que seja configurada. Caso não estejam configuradas corretamente, tanto as classes definidas no diretório AGLET_EXPORT_PATH que podem ser acessadas remotamente, bem como as classes principais de segurança do Java poderão ser substituídas remotamente.

## 3.5 - Arquitetura Proposta

No complexo cenário apresentado pelas diversas tecnologias de informação atuais, o campo emergente dos agentes móveis promete a construção de repositório de dados organizados, que auxiliam o usuário em suas diversas atividades de ordem "informacional".

A concepção de interfaces apropriadas e de uso natural torna-se tão importante quanto o desenvolvimento das próprias funcionalidades do sistema. O número de pesquisas na área atesta que, talvez, o maior desafio da concepção de ferramentas de busca seja a criação de novas interfaces que auxiliem o usuário a lidar com o volume, a complexidade e o dinamismo dos futuros repositórios de informação digital.

Tais interfaces devem permitir práticas radicalmente novas de trabalho individual ou em grupo. Inúmeras tecnologias existem ou se encontram atualmente em desenvolvimento que podem ser aplicadas na concepção de tais interfaces: a navegação hipertexto que facilita a exploração de grandes espaços de informação [32], [50]; técnicas de procura e recuperação de recursos de interesse específicos [32], [81], [87]; técnicas de visualização do contexto e de vários níveis de detalhes da informação desejada; além da integração do espaço e do tempo em ambientes de trabalho computacionais [49].

A arquitetura do Data Agents pretende introduzir o conceito de agentes móveis como apoio aos serviços oferecidos por livrarias digitais. A aplicação proposta pretende

ser uma arquitetura de *software* apta a prover agentes adaptados às necessidades de cada usuário em particular. Capaz de oferecer serviços de informação, onde os agentes trabalham sob a responsabilidade do usuário, liberando-o daquelas tarefas rotineiras de procura de informação em livrarias tradicionais, geralmente limitadas no espaço (quanto às coleções existentes) e no tempo (extremamente consumidoras de tempo).

O uso da tecnologia de agentes móveis dota a arquitetura do Data Agents de características que já são consenso no desenvolvimento de repositórios organizados de informação, ou seja: distribuição das bases de dados sobre vários servidores distantes; interoperabilidade entre servidores permitindo o compartilhamento de recursos através da rede; abertura para adaptação de outras tecnologias de serviço; escalabilidade para adaptação de novos serviço e novas representações da informação.

O desenvolvimento da arquitetura do Data Agents leva em consideração uma série de princípios que norteiam a sua especificação, concepção e implementação. Aqueles considerados mais importantes são:

- **reatividade espontânea dos agentes** - *os sistemas gerenciadores de base de dados atuais integram "triggers", regras e procedimentos que podem ser usados pelos agentes para oferecerem serviços ativos sob a responsabilidade do usuário;*
- **escalabilidade** - *o sistema provê mecanismos que facilitem a distribuição da carga de processamento de forma que a operação de novos agentes não afete, de forma drástica, o desempenho do servidor ou da rede;*
- **extensibilidade** - *o sistema permite a definição/expansão dos serviços disponíveis no servidor de acordo com as mudanças dos recursos da livraria digital;*
- **segurança** - *o sistema controla o acesso através de mecanismos de autenticação que permitem, ao mesmo tempo, livre acesso as operações dos agentes e privacidade para o usuário;*
- **busca simultânea -** *o sistema possibilita a busca simultânea em diversos recursos de informação, fornecendo ao usuário resultados mais variados e uma economia significativa de tempo.*

### 3.5.1 - Descrição Geral do Sistema

A arquitetura, conforme ilustrada na figura 7, é composta pelos seguintes elementos:

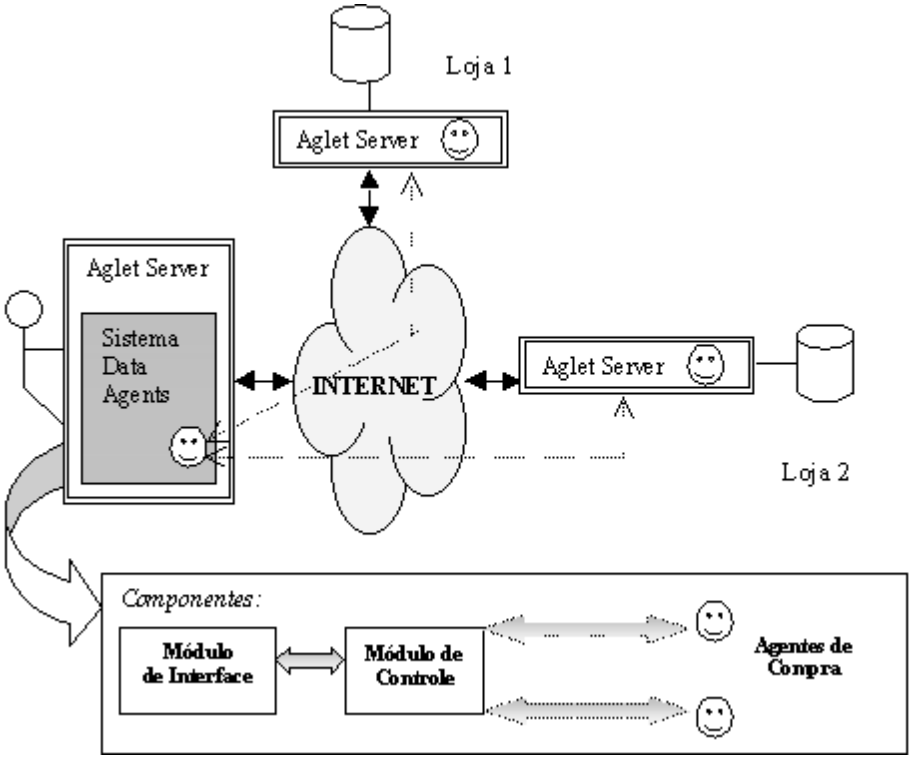


Figura 7 – Arquitetura geral do sistema Data Agents

- ***módulo de interface*** **-** é o módulo através do qual o usuário entra em contato com o sistema e realiza o seu pedido. É também responsável por apresentar ao usuário o resultado obtido pelo conjunto de agentes por meio de uma tela ou por e-mail.

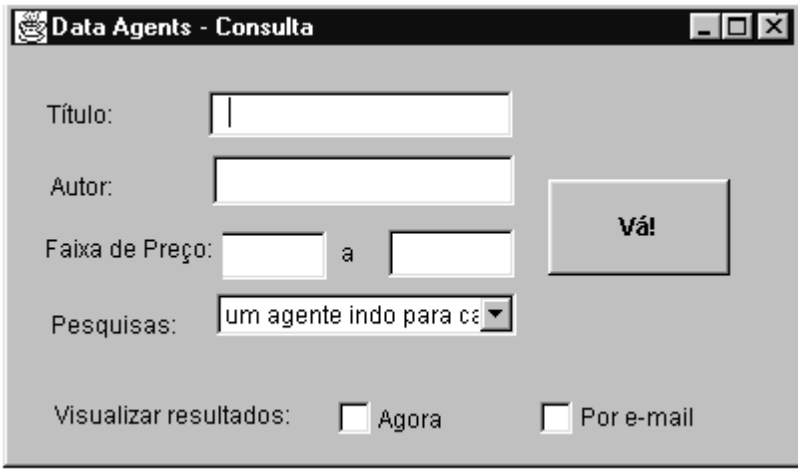


Figura 8 – Módulo de Interface do Data Agents

"Título", "autor", "faixa de preço" e "tipo de pesquisa" são informações que o usuário deve fornecer a este módulo para que seja possível a criação e o disparo dos agentes compradores de acordo com as restrições impostas pelo usuário.

Existem três possibilidades para efetuar a pesquisa a partir do *host* de origem:

- ***um agente indo para cada servidor*** **-** um agente é criado e enviado para cada um dos servidores cadastrados no sistema para efetuar a pesquisa; após ter realizado sua tarefa, cada agente envia o seu resultado para o *host* de origem como uma mensagem e, finalmente, é liberado;
- ***um agente percorrendo todos os servidores –*** é criado apenas um agente que possui no seu plano de viagens todos os servidores a serem visitados; o agente passa por todos os servidores, efetua a pesquisa sobre o produto solicitado e, no final de seu itinerário, envia o resultado final ao *host* de origem através de uma mensagem; o agente é liberado no último servidor que visita;
- ***um agente percorrendo todos os servidores até encontrar a primeira ocorrência –*** é criado apenas um agente que possui no seu plano de viagens todos os servidores a serem visitados; entretanto, o agente encerra a sua viagem ao encontrar a primeira ocorrência que satisfaça os requisitos estabelecidos pelo usuário.

- ***módulo de controle:*** é o componente responsável pela criação e envio dos agentes compradores aos servidores cadastrados para dar início à pesquisa solicitada pelo usuário. É também responsável pela agregação de todas as respostas obtidas pelos agentes compradores e pelo envio destas ao módulo de interface para que as mesmas sejam entregues ao usuário final.

- ***agentes compradores:*** procuram auxiliar os usuários da *Web* a encontrarem pontos de vendas de produtos que estejam disponíveis on-line, realizando a busca em paralelo. Esses agentes permitem que um usuário possa localizar com mais rapidez o produto em uma loja, sendo também um instrumento de compração de preços. Realizam o contato com a loja acessando o seu banco de

dados, efetuam o pedido e interpretam as respostas geradas, convertendo-as em um formato entendido pelo módulo de controle. Têm como meta a verificação do banco de dados contido no endereço destino, selecionando aquelas informações consideradas relevantes de acordo com as regras pré-estabelecidas pelo usuário. Estas informações vão formar a base de conhecimento a ser utilizada pelo agente comprador para tomar decisões apropriadas no processo de avaliação dos itens encontrados.

Com esta arquitetura, a extensão do protótipo para tratar de novos produtos e novas lojas é simples, apesar de ser necessário construir um módulo de controle para cada novo produto. Dessa forma, diversos recursos sobre o mesmo tipo de informação poderiam ser organizados em grupos distintos e atendidos por um membro específico do protótipo (livrarias, lojas de CDs, jornais) possibilitando um maior foco e fornecendo uma melhor visualização para que o usuário possa comparar preços de vários produtos presentes na Web.

### 3.5.2 – Funcionamento do Protótipo

Nesta seção, procurar-se-á descrever os componentes do protótipo, analisando o seu funcionamento. A arquitetura do modelo caracteriza-se pela presença e interação de três tipos de agentes:

- agente comprador – é o único componente móvel do modelo; viaja através da rede até os servidores onde efetua as buscas necessárias;
- *agente* de controle (módulo de controle) – cria a quantidade de *agentes* de pesquisa necessários de acordo com a especificação recebida do *agente* de interface;
- *agente* de interface (módulo de interface) – apresenta uma interface gráfica onde o usuário fornece os dados referentes a solicitação de busca, e cria o *agente* de controle passando os parâmetros de buscas através de mensagens.

A interação entre os agentes funciona através de uma série de mensagens pré-codificadas, as quais podem ou não conter algum argumento para o agente. Por

exemplo: a mensagem “caminho” contém como argumento o vetor indicando os locais que o agente comprador deve pesquisar (seu itinerário), enquanto a mensagem “iniciar” indica que o agente comprador deve iniciar a sua viagem para efetuar as pesquisas solicitadas.

Estando os agentes organizados dessa forma, circula na rede somente o *agente comprador* que vai até o servidor onde executa a busca localmente, enviando os resultados obtidos através da rede. O processo completo de pesquisa é iniciado com a criação do módulo interface que cria uma interface gráfica onde o usuário indica as suas preferências para a realização da busca. Como pode ser visto na figura 9, o módulo de controle ao receber os requisitos do usuário, cria um ou mais agente(s), converte-o(s) em um *array* de bytes, que, por sua vez, é passado à camada do ATP (*Agent Transfer Protocol*) para que este seja enviado ao seu destino. Este protocolo constrói, então um “*bit stream*” que contém informações gerais, tais como: nome do sistema, identificação do mesmo e o *array* de bytes proveniente da camada de *runtime*.

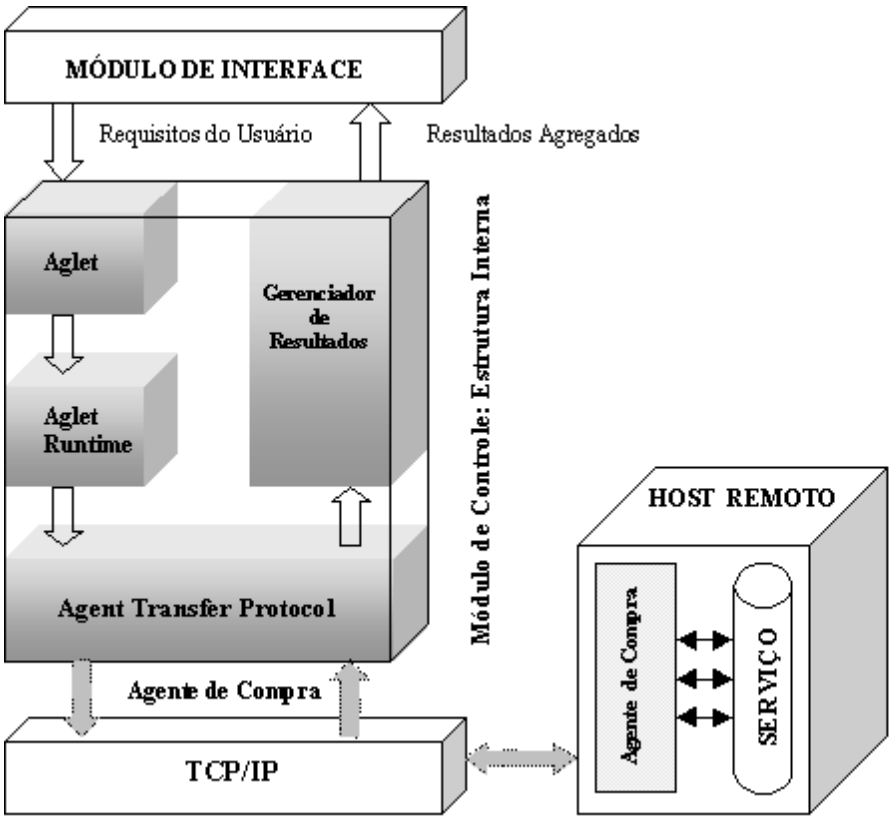


Figura 9 - Módulo de Controle - Estrutura Interna

Cada agente comprador ao retornar ao *host* de origem, envia todo o seu conteúdo ao componente do módulo de controle, Gerenciador de Resultados, para que este possa agregar todas as respostas obtidas e envia-las ao módulo de interface. Este último, por sua vez, envia o resultado para o usuário por e-mail ou por meio de uma tela (figura 10).

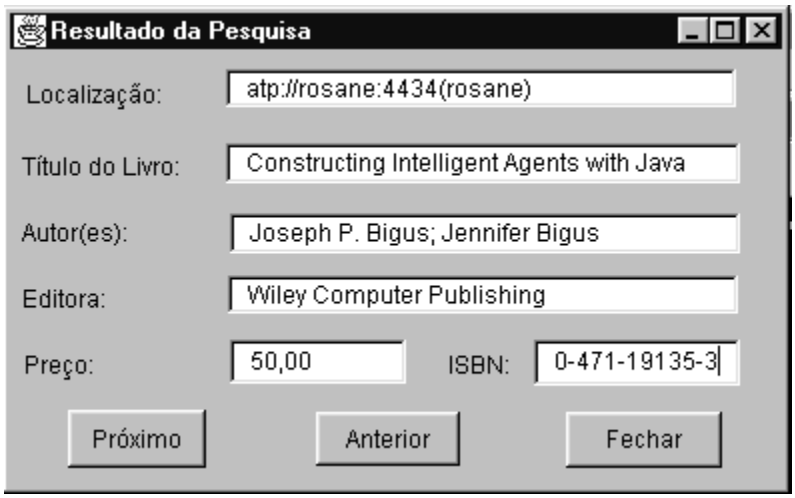


Figura 10 - Resultado da pesquisa efetuada

## 3.6 - Implementação

Muitos fatores influenciaram na implementação do sistema. O primeiro deles foi encontrar um ambiente de apoio a desenvolvimento que implementasse a mobilidade dos agentes, tal como apresentada na seção 3.4. Esta ferramenta possibilitou a inclusão de uma série de propriedades no protótipo, como, por exemplo a noção de itinerários.

O segundo foi a escolha de uma metodologia de desenvolvimento do sistema. Optou-se pela metodologia em espiral [9] pela facilidade de incorporar a análise de riscos e revisões do planejamento em qualquer fase do projeto, sem no entanto abdicar do uso de uma abordagem estruturada e sistemática.

As diversas voltas à espiral correspondem às etapas de execução do sistema. No final de cada fase, uma versão operacional do sistema Data Agents é criada. À medida em que o projeto progride, as versões passam a ter menos restrições, aproximando-se, assim, dos objetivos primordiais do sistema.

A análise de sistema e o planejamento geral constituem as primeiras etapas da volta em torno da espiral. Em seguida, são efetuados, em cada volta, o planejamento detalhado e a análise de requisitos para a fase correspondente. Define-se depois, um plano de testes. O passo seguinte é o detalhamento dos diversos aspectos referentes à arquitetura dessa fase. Cada volta à espiral termina com a concretização do Data Agents de acordo com as especificações dessa fase e a realização dos testes planejados.

### 3.6.1. - Descrição das Etapas e Módulos Desenvolvidos

Nesta seção, são descritas as fases que levaram, de uma forma incremental, a que o sistema Data Agents se aproximasse cada vez mais do objetivo de tornar o processo de compras através da Internet mais fácil para o usuário, tornando desnecessário a procura de vários fornecedores para o produto pretendido repetidas vezes.

- ***Fase um -*** a fase inicial de desenvolvimento do sistema Data Agents apresenta a configuração mais simples para o sistema. A solução apresentada é muito limitada, permitindo a existência de apenas um itinerário (um agente indo para um servidor) e uma loja associada ao sistema.
- ***Fase dois -*** a segunda fase consistiu na eliminação da limitação inicial de apenas uma loja estar associada ao sistema, podendo os agentes percorrem, a partir desta fase, três servidores.
- ***Fase três -*** a introdução do segundo tipo de itinerário, um agente percorrendo todos os servidores, foi a característica desta fase.
- ***Fase quatro -*** o terceiro e último tipo de itinerário foi introduzido no sistema nesta fase (um agente percorrendo todos os servidores até encontrar a primeira ocorrência).

O sistema Data Agents, como descrito na seção 3.5.1, é composto pelos módulos de interface, de controle e pelos agentes compradores. O módulo de interface e o de controle foram implementados como *aglets* estáticos, sendo nomeados na implementação como ***aglet de interface*** e ***aglet de controle***, respectivamente. Os agentes compradores, únicas entidades móveis do protótipo, foram implementados como ***aglets de pesquisa***. Cada um dos *aglets* possui características e funcionalidades próprias as quais são citadas a seguir.

- **Aglet de Interface**

O Aglet de Interface é um agente estático baseado em uma interface gráfica. É o primeiro agente a ser criado, e mostra uma caixa de diálogo para o usuário escolher os itens referentes à pesquisa a ser executada.

No caso de cancelamento da busca, a classe de interface envia uma mensagem para o *aglet* de interface indicando que o mesmo deve ser desativado. O *aglet* de interface recebe esta mensagem e inicia o processo de desalocação. Caso contrário, o *aglet* de interface chama um método que cria um *aglet* de controle para aquela busca, enviando-lhe três mensagens.

A primeira mensagem refere-se à origem da busca. Esta mensagem é exibida na tela dos diversos outros servidores indicando de onde partiu a pesquisa em questão. A segunda refere-se à mensagem de seleção que é executada pelos aglets de pesquisa. A terceira e última refere-se à quantidade de colunas que devem ser exibidas para o usuário.

- **Aglet de Controle**

O *aglet* de controle é criado pelo *aglet* de interface para controlar todos os agentes de pesquisa disparados conforme o tipo de procura solicitada. Logo após a sua criação, o *aglet* de controle cria uma estrutura de dados para cada *aglet* de pesquisa, contendo os drivers que são utilizados, a localização dos servidores de pesquisa (endereço atp), os usuários, as senhas e o caminho completo incluindo a porta de acesso ao banco de dados. Em seguida, é criada uma interface de espera indicando que o *aglet* de controle foi criado e que espera o retorno das pesquisas para exibir o resultado.

- **Aglet de Pesquisa**

Cada *aglet* de pesquisa, após ser criado, recebe através de mensagens os dados necessários para efetuar a pesquisa, incluindo os servidores a serem pesquisados, o AgletProxy do agente que o criou, local onde ele foi criado, a *string* de seleção e os dados referentes a localização dos servidores de pesquisa (endereço atp), os usuários, as senhas e o caminho completo para efetuar a busca no servidor. Após receber todos esses dados, o *aglet de pesquisa* inicia a viagem para o primeiro servidor da lista de endereços.

Ao chegar a cada um dos *hosts* definidos no plano de viagem, o *aglet* executa o método Lista(), que faz a conexão com o banco de dados localmente e realiza a pesquisa. Após receber o resultado, o *aglet* envia uma mensagem ao gerenciador de resultados, um componente do *aglet* de controle, contendo como argumento o vetor com todos os dados da tabela pesquisados, convertidos em *string*.

### 3.6.2 - O Ambiente de Simulação

A fim de se projetar, implementar e testar o sistema Data Agents, bem como estabelecer uma estrutura básica de programação em tempo real em um sistema distribuído, desenvolveu-se um ambiente de simulação com o uso de uma rede local, padrão Ethernet, com taxa de transmissão a 10 Mbps, na qual estavam colocadas três estações, consistindo de microcomputadores compatíveis IBM/PC: modelos Pentium 200 e 32 Mb de RAM. A escolha do tipo de rede foi devido à disponibilidade dos equipamentos no Laboratório de Informática da Área de Ensino e Pesquisa do Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro.

O ambiente operacional utilizado foi o Windows NT versão 4.0 e todas as linhas de código do sistema em questão foram escritas na linguagem Java e nas bibliotecas do plataforma ASDK para proporcionar o suporte necessário à aplicação.

Um diagrama do sistema, sob o ponto de vista de localidade dos processos e tratamento de eventos é mostrado na figura 11. O sistema proposto atua em um ambiente simulado de procura de livros na Internet. Este ambiente foi desenvolvido para a Web, motivado pelo esforço crescente na elaboração de ferramentas de busca de informações, as quais apresentam inúmeras vantagens para os usuários, tais como a independência de localização e redução de tempo no acesso à informação desejada.

Conforme ilustrado na figura 11, um ou mais agentes compradores são criados na máquina 1, e coletam informações de interesse do usuário nas máquinas 2, 3 e/ou 4. As informações coletadas e o *host* visitado são armazenados no corpo dos agentes compradores para posterior utilização pelos outros componentes do sistema.

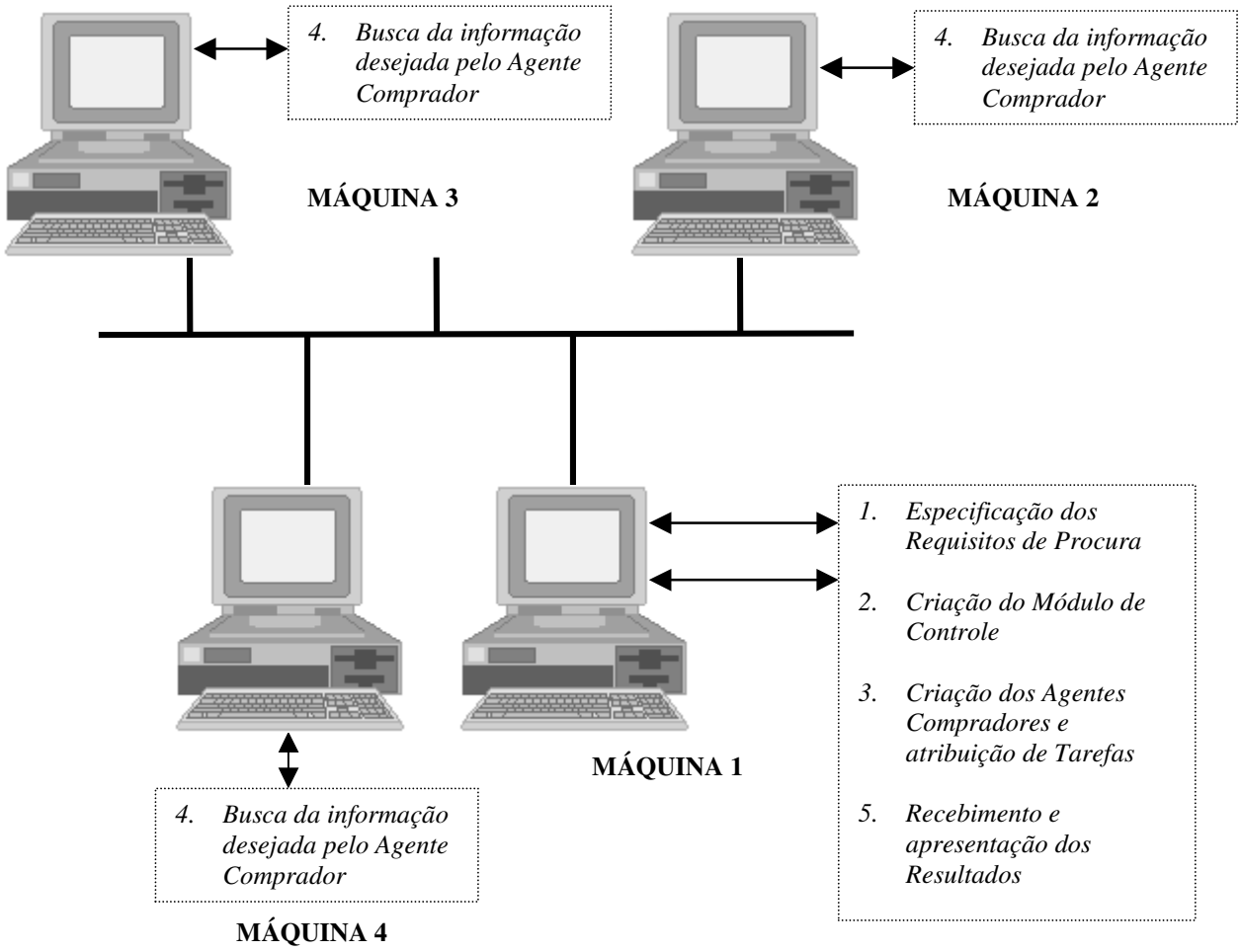


Figura 11 - Contextualização dos Processos ao longo da Rede

O modelo de entidade e relacionamento do banco de dados utilizado pelo sistema e disponível em cada máquina servidora é ilustrado na figura 12.

É um modelo simples, que possui 4 (quatro) tabelas: gêneros, autores, livro_autor e livros, sendo a busca de livros efetuada, basicamente, sobre as tabelas de livros, autores e livro_autor. Nas tabelas do dicionário de dados foram inseridos dados fictícios. Para comprovar o funcionamento de chamadas JDBC utilizou-se para o estudo de caso banco de dados Access.

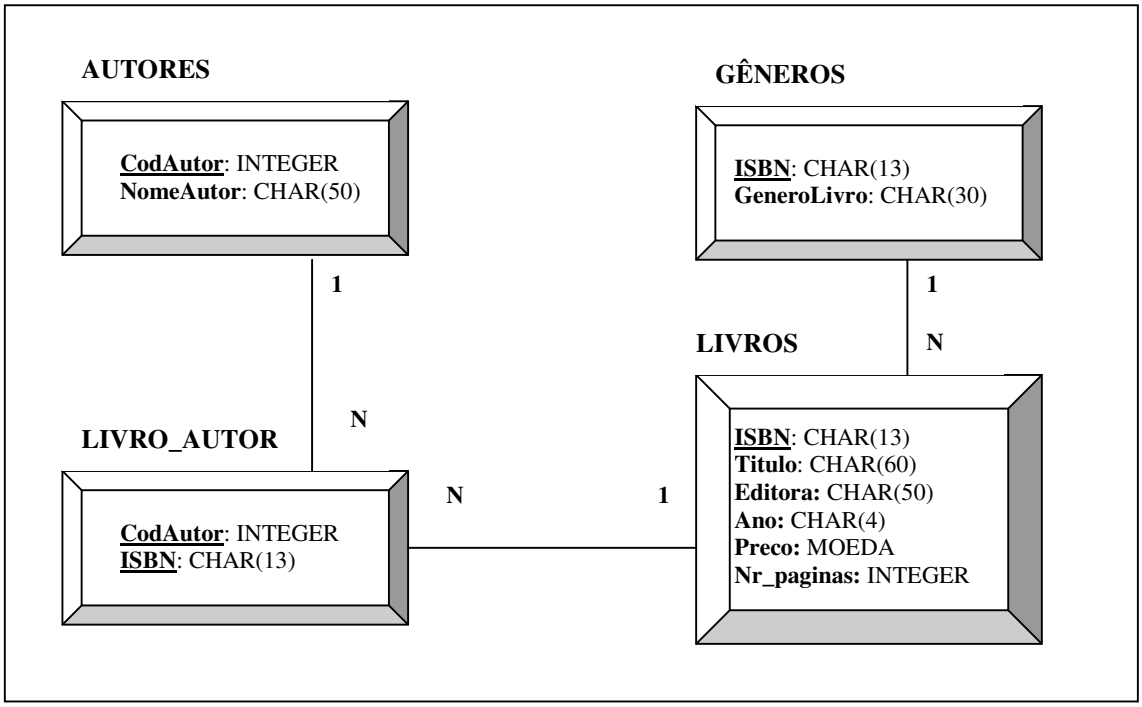


Figura 12 - Modelo de entidade e relacionamento do Data Agents

## 3.7 - Conclusão

O presente capítulo procura mostrar o ambiente de desenvolvimento, a arquitetura e o funcionamento de um sistema de recuperação de informações estruturadas que, a partir de um cliente, cria um ou mais agentes que se deslocam até um servidor de banco de dados, fazem as consultas necessárias e retornam os resultados ao usuário através de uma tela ou por e-mail.

Esta aplicação, que tem por objetivo tornar as compras através da Internet mais fáceis para o usuário, também investiga e testa a conveniência de se utilizar agentes móveis em um ambiente distribuído e multi-plataforma para produzir uma solução de pedido de compra em um mercado global.

Os aspectos referentes às experiências realizadas a fim de demonstrar sua funcionalidade são abordados no próximo capítulo.

# CAPÍTULO 4:

## *Avaliação da Qualidade e do Desempenho do Sistema Data Agents*

## 4.1 - Introdução

Este capítulo tem por objetivo avaliar o comportamento dos agentes do Sistema Data Agents em um ambiente distribuído para produzir uma solução de compra em um mercado global. Esta avaliação foi feita através da verificação de agilidade do sistema com a metodologia GQM (Goal - Question - Metric) de Basili [3]. Outra análise realizada foi sobre o tempo médio de execução dos agentes na realização da tarefa após visitar um número determinado de nós.

Este capítulo está organizado da seguinte forma: na seção 4.2 são descritos os parâmetros utilizados na avaliação da qualidade do software em questão e as experiências realizadas. Na seção 4.3 são relatados os resultados obtidos na avaliação de desempenho. Por último, a seção 4.4 apresenta as conclusões obtidas com as experimentações realizadas.

## 4.2 - Avaliação da Qualidade

A tecnologia de código móvel foi utilizada no sistema Data Agents, uma solução de pedido de compra em um mercado global. Diversos cenários gerados durante o estudo de caso são utilizados para avaliar a agilidade dos sistemas. A avaliação do sistema é conduzida através da metodologia GQM (*Goal - Question - Metric*) de Basilie [3], através da qual um número de métricas relevantes são identificadas.

O estudo de caso teve um importante papel na implementação deste sistema, revelando, principalmente, os requisitos primordiais do sistema Data Agents. Estes requisitos formam a base dos cenários para mudança utilizados para avaliar a implementação e a abstração do código móvel. Os requisitos são:

- poder manipular a rápida adição de novos agentes de compra;
- poder manipular a adição de novos endereços a serem inseridos no sistema;
- permitir mudanças na lógica do negócio.

Avaliar arquiteturas de software é uma tarefa notoriamente difícil. Há poucas técnicas e/ou medidas estabelecidas. E, apesar da Engenharia de Software ser uma ciência que se empenha para emular as ciências clássicas, ainda há um "longo caminho a ser percorrido". Ao invés de equações formais, têm-se metodologias para desenvolver métricas, tais como a abordagem "*Quality Function Deployment*" [44], Métricas de qualidade de Software [9] e GQM [3]. A metodologia de Basili - GQM - foi escolhida devido a sua popularidade e suporte dentro da comunidade de Engenharia de Software.

O objetivo da metodologia GQM é baseado na assunção de que para se ganhar uma medida prática, deve-se primeiro entender e especificar os objetivos do software que está sendo avaliado e os objetivos do processo de mensuração. Uma vez que estes princípios tenham sido estabelecidos, é possível direcionar a investigação e a medição para os dados que definem os objetivos operacionalmente. Um objetivo geral para a aplicação em questão pode ser estabelecido como: "Avaliar os agentes implementados a partir de uma perspectiva de recuperação de informações estruturadas e distribuídas para satisfazer a motivação de suportar a agilidade do sistema."

Tendo estabelecido o objetivo, o processo continua pela geração de um amplo conjunto de questões que devem fornecer alguma indicação dos assuntos individuais encapsulados pelo objetivo principal. Logo, procura-se gerar um número considerável de questões, inclusive questões redundantes e/ou inválidas.

Após diversas atividades de refinamento, um conjunto de métricas de software que podem ser utilizadas para avaliar o sistema de código móvel é mostrado na tabela 2:

| Métrica | Natureza da Métrica |
|---|---|
| (1) | Identificação das abstrações baseadas na informação do mundo real. Comparar com as abstrações baseadas na informação do software. |
| (2) | Identificação das abstrações baseadas em processo do mundo real. Comparar com os processos evidentes no software. |
| (3) | Identificação dos elementos móveis do mundo real. Comparar com os elementos móveis do software. |
| (4) | Identificação dos elementos estáticos do mundo real. Comparar com elementos estáticos do software. |
| (5) | Número de componentes. |
| (6) | Número de linhas de código |
| (7) | Número de comentários. Calcular a média: comentários por método. |
| (8) (9) (10) | Número de mudanças para cada requisito. |
| (11) | Número de arquivos que são mudados para cada requisito. |
| (12) | Número de invocação de métodos de objetos. |
| (13) | Número de métodos públicos. |

Tabela 2 - Métricas geradas utilizando o Método GQM

A maioria das métricas geradas é extremamente limitada em seu objetivo. No entanto, através de combinações, é possível se chegar a algumas medidas úteis de um software. Nas seções seguintes, é examinado como estas métricas foram utilizadas na avaliação do sistema implementado.

### 4.2.1 - Avaliando Alinhamento Semântico

Papaioannou e Edwards [67] afirmam que o código móvel pode aumentar o alinhamento semântico entre os softwares e os processos do mundo real que ele pretende suportar. Para comprovar esta afirmação, estes autores desenvolveram uma maneira de medir o quanto as abstrações do mundo real são personificadas no software, bem como quanto se parecem com o modelo do mundo real. Para isto, criaram o termo chamado ***Difusão Conceitual*** (*Conceptual Diffusion*). Difusão Conceitual é definida

como o grau de conceito semântico de cada componente do software no domínio da aplicação. Por isso, pode-se dizer que:

$$DC = \frac{A}{B} \qquad \textbf{(1)}$$

onde:
DC = difusão conceitual,
A = o número de conceitos inclusos nessa abstração, e
B = o número de componentes onde esta abstração está personificada.

A DC pode ser examinada em diferentes níveis de granularidade para se obter diferentes perspectivas em uma situação. Por exemplo, em um sistema que pretende suportar o processo de busca de informações sobre livros em diversas livrarias virtuais, o conceito da lógica de uma pesquisa deve estar presente. Na análise, foi observado que este está dividido entre três componentes separados do sistema. Assim, pode-se afirmar que o conceito de uma pesquisa no sistema Data Agents possui uma taxa de difusão conceitual de 3 (veja tabela 3).

| **Objetos** | **Abstração da Informação** | **Abstração do Processo** | | | **Lógica da Pesquisa** |
|---|---|---|---|---|---|
| | ***Pesquisa*** | *requisição* | *distribuição/ controle* | *pesquisa propriamente dita* | |
| Aglet de Interface | | ✓ | | | ✓ |
| Aglet de Controle | | | ✓ | | ✓ |
| Aglet de Pesquisa | ✓ | | | ✓ | ✓ |
| **DC** | **1** | **1** | **1** | **1** | **3** |

Tabela 3 - Análise da Difusão Conceitual presente no Código Móvel

A tabela 3 também mostra os resultados das métricas (1) e (2). Estas métricas são exemplos de difusão conceitual em um nível maior de granularidade. Por exemplo, a métrica (1) requer a identificação de todos os conceitos baseados na informação no domínio do mundo real, e uma comparação com seus correspondentes nos sistemas em

questão. Uma vez que "pesquisa" é uma abstração baseada na informação, ela pode ser incluída nos resultados da métrica (1). Deve-se utilizar a DC para se obter um *insight* em quão bem os conceitos ou abstrações são incorporadas no software.

O Alinhamento Semântico entre as abstrações do mundo real e os componentes de software tem sido considerado um fator de relativa importância na tentativa de construir sistemas mais ágeis [20]. Foi criado para se obter uma perspectiva geral do sistema através de uma combinação das métricas (1) até (4).

$$AS = \left\{ \frac{ls}{lr}, \frac{Ps}{\Pr}, \frac{Ms}{Mr}, \frac{Ss}{Sr} \right\} \qquad \textbf{(2)}$$

onde:

AS = alinhamento semântico, S = componentes estáticos,

l = abstrações baseadas na informação, s = no software, e

P = abstrações baseadas nos processos, r = no mundo real.

M = componentes móveis,

Assim, por exemplo, $\frac{Ps}{\Pr}$ é a razão das abstrações baseadas em processos no software para as abstrações baseadas em processos no mundo real.

Esta métrica pode ser utilizada para analisar um sistema e para avaliar quanto ele reflete as semânticas do domínio da aplicação. Uma comparação com o alinhamento semântico ideal de {1,1,1,1} pode ser utilizada como um parâmetro para avaliar a dificuldade em se entender um software, dado o contexto do domínio da aplicação.

Pela combinação dos resultados das quatro primeiras métricas, pode-se afirmar que o Alinhamento Semântico do Sistema Data Agents é { 1, 3/3, 1/3, 2/3}

Os resultados da análise da Difusão Conceitual e do Alinhamento Semântico mostram que o sistema em questão é fácil de entender, pois as abstrações no mundo real alinham-se razoavelmente aos componentes dos softwares. Ao considerar o alinhamento dos componentes estáticos e do componente móvel, um terço dos componentes no domínio é modelado como móvel na implementação, e dois terços como elementos estáticos. Se os sistemas distribuídos tradicionais forem examinados, poder-se-á constatar que eles não possuem facilidade para a construção de componentes móveis em um sistema. Por isso, são considerados incapazes de implementar quaisquer das abstrações móveis. Uma vez que sistemas de código móvel são igualmente adeptos à construção de componentes estáticos, pode-se concluir que os sistemas de código móvel, de fato, aumentam o alinhamento semântico entre o mundo real e os sistemas de software por eles suportados, para qualquer sistema que não é construído completamente a partir de componentes estáticos.

### 4.2.2 - Avaliando Agilidade do Sistema

Para avaliar a agilidade de um sistema é necessário efetuar mudanças no mesmo. O estudo de caso destacou diversos requisitos do mundo real para a agilidade que um sistema distribuído deve possuir (ver tabela 4). Usando estes requisitos como cenários para mudança, algumas modificações na implementação foram realizadas para avaliar a agilidade do sistema.

A metodologia GQM possibilitou a derivação de diversas métricas que podem ser utilizadas para medir certos aspectos de um sistema após a realização de alguma modificação. Estas medidas são especificadas pelas métricas (8), (9), (10), (11). Individualmente, elas permitem a medição de fatias limitadas da mudança para um sistema. No entanto, pela combinação delas, é possível produzir uma medida mais eficaz. Este conjunto tem sido chamado de ***capacidade de mudança*** e é descrito por:

$$CM = \left\{ \sum_{á}^{â} ão, \sum_{á}^{â} ãs, \sum_{á}^{â} ãi, \sum_{á}^{â} ã\, a^{o} \right\} \qquad \textbf{(3)}$$

$á \rightarrow â$

onde:

CM = capacidade de mudança, s = número de arquivos,

o = número de objetos, i = número de interações,

ã = alteração requisitada, a° = número de entidades conceituais entre estados ***á*** e ***â***.

Uma entidade conceitual é análoga à abstração ou conceito referidos nas seções anteriores. Por exemplo, poderia ser uma procura. Interações são aquelas trocas de informação entre objetos, usualmente via invocações de métodos, embora para os agentes isto também se aplica a qualquer diálogo (mensagens) que eles possam ter. Mudanças para aquelas interações normalmente implica em mudanças da assinatura do método.

A métrica capacidade de mudança pode ser usada para comparar sistemas ou para obter uma medida de agilidade do sistema relativo ao ideal CM = {0,0,0,0}. Para o sistema Data Agents, a capacidade de mudança para cada requisito está resumida na tabela 4.

Estes resultados mostram que o sistema é fácil de ser alterado. Adicionar novas facilidades de procura requer somente a instanciação de novos agentes compradores e a inclusão de novas livrarias no cadastro do sistema.

| **Requisitos** | **Data Agents** |
|---|---|
| Adição de novos agentes compradores | {0,1,1,0} |
| Inclusão de novas livrarias | {2,2,1,2} |
| Possibilidade de efetuar mudanças na lógica da procura | {1,1,0,1} |

Tabela 4 - Capacidade de Mudança

A métrica da capacidade de mudança pode ser utilizada por um projetista para avaliar o grau de sensibilidade a mudanças de um sistema. Por exemplo, considerando o

conjunto CM {5,20,20,1}, pode-se concluir que, para esta mudança, embora somente uma entidade conceitual (1) esteja sendo mudada, há 20 (vinte) mudanças para os arquivos fonte, 5 (cinco) mudanças para os objetos, e 20 (vinte) mudanças para as interações destes objetos. Mudar a assinatura de 20 (vinte) métodos em 5 (cinco) objetos para efetuar uma mudança em uma simples entidade pode causar sérios problemas e levar o projetista a ter que revisar o grau de difusão conceitual desta entidade.

## 4.3 - Avaliação de Desempenho

A análise de desempenho baseia-se nos tempos de execução dos agentes compradores de acordo com os três tipos de itinerários disponíveis no sistema. Foi avaliado o tempo que estes agentes levam para ir até a fonte dos dados, realizar a pesquisa e retornar os resultados para o módulo de controle. Os testes foram realizados em uma máquina Pentium 233 com 64 Mb de RAM, sobre o JDK 1.1.7A para ambiente Windows 95. Os resultados apresentados na tabela 5 estão em milissegundos. A simulação foi realizada em uma mesma máquina por se querer mensurar o tempo médio de execução dos agentes compradores, e não o tráfego da rede.

Para a compreensão da tabela 5, considera-se:

- ***itinerário 1 (um agente indo para cada servidor) -*** são criados vários agentes, sendo que cada um deles é enviado para um dos servidores, respectivamente, cadastrados no sistema para efetuar a pesquisa; após ter realizado sua tarefa, cada agente envia o seu resultado para o *host* de origem como uma mensagem e, finalmente, é liberado;
- ***itinerário 2 (um agente percorrendo todos os servidores) –*** é criado apenas um agente que possui no seu plano de viagens todos os servidores a serem visitados; o agente passa por todos os servidores, efetua a pesquisa sobre o produto solicitado e, no final de seu itinerário, envia o resultado final ao *host* de origem através de uma mensagem; o agente é liberado no último servidor que visita.
- ***itinerário 3 (um agente percorrendo todos os servidores até encontrar a primeira ocorrência) –*** é criado apenas um agente que possui no seu plano de

viagens todos os servidores a serem visitados; entretanto, o agente encerra a sua viagem ao encontrar a primeira ocorrência que satisfaça os requisitos estabelecidos pelo usuário.

| Número de Livrarias | Itinerário 1 | Itinerário2 | Itinerário3 |
|---|---|---|---|
| 5 | 988 | 1800 | 110 |
| 10 | 1044 | 1998 | 232 |
| 25 | 3558 | 5282 | 492 |

Tabela 5 - tempos médios de execução dos agentes compradores

A figura 13 mostra que o comportamento dos agentes muda, consideravelmente, com a mudança do tipo de itinerário (as curvas estão muito distantes umas das outras). Porém, a diferença de desempenho (ao nível de velocidade) é mais significativa entre os itinerários dois e três. Entretanto, em termos de conteúdo do resultado da pesquisa, deve-se considerar que os resultados obtidos pelos agentes através dos itinerários um e dois são mais completos. É claro que o tempo de processamento é diferente dependendo do itinerário empregado. Entretanto, pode-se concluir que, de uma maneira geral, tendo até 10 (dez) servidores a serem visitados , o sistema funciona bem.

Deve-se ressaltar, também, que, no itinerário dois, o tempo de resposta aumenta mais rapidamente quando o número de livrarias é grande, porque os agentes móveis passam a ter um tamanho maior que dificulta mais a migração. Já em relação ao itinerário três, é de se esperar que esse tipo de itinerário tenha o menor tempo de processamento, já que o agente retorna ao cliente ao encontrar a primeira ocorrência que satisfaça o usuário.

Por fim, analisando o tempo de execução e a qualidade das informações recebidas, pode-se observar que o itinerário um fornece uma melhor resposta; haja vista que são vários agentes compradores trabalhando em paralelo para atender os interesses

do usuário. Pode-se perceber, inclusive, que a curva torna-se mais tênue em relação ao itinerário dois.

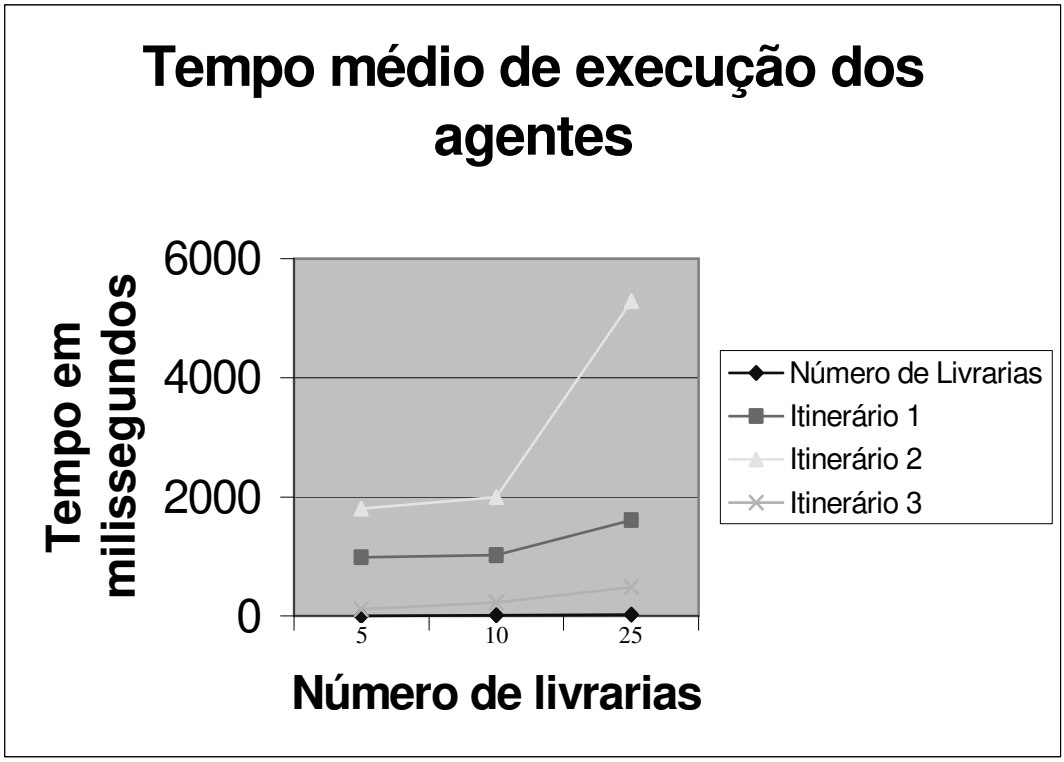


Figura 13 - Tempo médio de execução dos agentes compradores

## 4.4 - Conclusão

Avaliar *softwares* nunca foi e nunca será uma tarefa fácil. A motivação para este trabalho experimental descrito neste capítulo foi avaliar as abstrações dos agentes móveis. As avaliações quanto à qualidade foram realizadas segundo a metodologia GQM de Basili. Utilizando esta técnica, um conjunto de métricas tangíveis foram desenvolvidas. O trabalho mostrou que, pela redução de difusão conceitual em um sistema, as abstrações de código móvel são capazes de oferecer um alinhamento semântico mais próximo do sistema que o projeto pretende suportar. Quanto à capacidade de mudança, os resultados mostraram que o sistema é fácil de ser alterado.

Adicionar novas facilidades de procura, por exemplo, requer somente a instanciação de novos agentes compradores e a inclusão de novas livrarias no cadastro do sistema.

Quanto à avaliação de desempenho, foram realizados testes levando-se em consideração o tempo médio que os agentes compradores levam para ir até a fonte dos dados, realizar a pesquisa e retornar os resultados para o módulo de controle. Vários experimentos foram realizados de modo a avaliar a performance dos agentes de acordo com o itinerário seguido.

Em relação ao tempo de resposta, os resultados das simulações indicam que o sistema Data Agents possui um melhor desempenho quando há um só agente comprador percorrendo todos os servidores disponíveis, mas retornando à origem ao encontrar a primeira opção que satisfaz os interesses do usuário.

Observa-se que, se há um só agente percorrendo todos os servidores de modo a apresentar para o módulo de controle todos os resultados obtidos, o desempenho piora, pois o tamanho do agente é aumentado a cada nó visitado.

Os resultados, indicam, portanto, que o sistema apresenta um melhor desempenho (tempo X qualidade das informações coletadas) quando emprega o itinerário um (são criados vários agentes sendo que cada um deles é enviado para um dos servidores, respectivamente, cadastrados no sistema). Pode-se concluir, então, que este tipo de itinerário é melhor por possuir vários agentes compradores trabalhando em paralelo para atender os interesses do usuário.

# CAPÍTULO 5:

# *Conclusão*

## 5.1 - Contribuições

Com o extraordinário desenvolvimento da rede mundial de computadores, a Internet, o volume de informações disponível pelo mundo, o custo da movimentação de dados e a dificuldade de se encontrar informações relevantes têm sido motivo de preocupação para a comunidade de redes e para os desenvolvedores de aplicativos.

Uma possível solução para estes problemas consiste na utilização de agentes, que são programas que ajudam o usuário a realizar tarefas, agindo em seu interesse. Caso os agentes possam executar suas tarefas em diferentes computadores da rede, estes são denominados agentes móveis. Estes agentes podem mover-se para o lugar onde os dados estão armazenados e selecionar as informações das quais o usuário necessita, economizando, assim, banda passante, tempo e dinheiro.

O presente trabalho procurou destacar as vantagens advindas do paradigma de agentes móveis, através da implementação de um sistema voltado à área de comércio eletrônico. O sistema Data Agents implementa uma aplicação para investigar e testar a conveniência de se utilizar agentes móveis em um ambiente distribuído, produzindo uma solução de pedido de compra em um mercado global. Esta aplicação é capaz de oferecer serviços de informação, onde os agentes trabalham sob a responsabilidade do usuário, liberando-o de tarefas rotineiras de procura de informação em livrarias tradicionais, geralmente limitadas no espaço e no tempo. Visando otimizar o processo de recuperação de informações distribuídas em diversos bancos de dados, o sistema Data Agents realiza a seleção das informações armazenados em cada servidor distinto, deslocando-se entre eles, para, no final, retornar todos os dados solicitados pelo usuário, sem a necessidade do cliente efetuar a chamada a cada um dos servidores separadamente. Os agentes compradores podem ser disparados pelos usuários para efetuarem pesquisa no banco de dados de qualquer livraria registrada no sistema, evidenciando sua mobilidade. Já os agentes de interface e de controle permanecem fixos

na máquina do usuário que se propõe a efetuar uma compra na Internet. Em virtude da execução local da pesquisa, o acesso ao banco de dados não consome os recursos da rede, melhorando significativamente a performance da recuperação da informação desejada.

Para a implementação, foi feito uso da linguagem de programação Java e do ambiente *Aglets Software Development Kit (ASDK)*, que é um conjunto de classes destinadas a auxiliar na construção de sistemas de agentes móveis na Internet. O ambiente ASDK apresentou-se satisfatório durante a implementação dos agentes dentro do sistema proposto. A sua estrutura em camadas demonstrou que é uma ferramenta que tem um campo de aplicação muito abrangente para construção de aplicações distribuídas na rede, pois não impõe restrições quanto a arquitetura dos agentes implementados. Entretanto, cabem algumas considerações. Optou-se por esta plataforma em virtude de ser baseada na linguagem de programação Java, apesar da pouca documentação existente. Mesmo assim, o ASDK presta-se bem para a análise estrutural do problema modelado, apresentando maior simplicidade e documentação do que as outras plataformas pesquisadas.

De acordo com os testes realizados com o sistema de agentes desenvolvido, pode-se dizer que os resultados obtidos foram satisfatórios, tendo em vista que a partir da necessidade de um produto, o cliente obtém como resposta informações sobre o mesmo, de forma a decidir mais rapidamente qual a melhor opção para a satisfação de seu desejo de compra. Assim, considerando inúmeros fornecedores, o agente comprador informa a seu usuário quais as livrarias que podem melhor satisfazer suas necessidades.

As avaliações quanto à qualidade foram realizadas segundo a metodologia GQM de Basili. Utilizando esta técnica, um conjunto de métricas tangíveis foram desenvolvidas. O trabalho mostrou que, pela redução de difusão conceitual em um sistema, as abstrações de código móvel são capazes de oferecer um alinhamento semântico mais próximo do sistema que o projeto pretende suportar. Quanto à capacidade de mudança, os resultados mostraram que o sistema é fácil de ser alterado.

Adicionar novas facilidades de procura, por exemplo, requer somente a instanciação de novos agentes compradores e a inclusão de novas livrarias no cadastro do sistema.

Quanto à avaliação de desempenho, foram realizados testes levando-se em consideração o tempo médio que os agentes compradores levam para ir até a fonte dos dados, realizar a pesquisa e retornar os resultados para o módulo de controle. Vários experimentos foram realizados de modo a avaliar a performance dos agentes de acordo com o itinerário seguido. Os resultados, indicam, portanto, que o sistema apresenta um melhor desempenho (tempo X qualidade das informações coletadas) quando emprega o itinerário um (são criados vários agentes sendo que cada um deles é enviado para um dos servidores, respectivamente, cadastrados no sistema). Pode-se concluir, então, que este tipo de itinerário é melhor por possuir vários agentes compradores trabalhando em paralelo para atender os interesses do usuário.

## 5.2 - Perspectivas para Pesquisas Futuras

Um sistema de agentes móveis é um sistema onde os agentes podem migrar de uma máquina para outra. Neste tipo de sistema, como os agentes movem-se através da rede, eles consomem recursos de cada servidor que visitam. O valor agregado de um sistema de agente móvel depende da habilidade dos agentes em migrar de um servidor para outro na rede. Servidores que recebem a visita destes agentes expõem-se a uma eventual sobrecarga e riscos adicionais. Estudos sobre uma compensação de riscos e custos de acesso através do estabelecimento de um sistema de comércio eletrônico onde os agentes móveis "comprariam" recursos computacionais podem ser indicados como desdobramentos para este trabalho. Poder-se-ia implementar um sistema de pesquisa que estabelece mercados para recursos computacionais para os agentes móveis. O propósito destes mercados seria requerer que os agentes móveis "pagassem" por todos os recursos computacionais necessários para a conclusão de suas tarefas no *host*. Maiores detalhes podem ser conferidos em [11].

O processo de negociação de compra e venda não foi considerado neste trabalho por ser julgado exterior ao escopo dos objetivos da pesquisa. Entretanto, apesar de não ter sido explorado no presente estudo, é um excelente candidato para uma investigação e expansão em algum trabalho futuro. Poder-se-ia empregar, por exemplo, um processo de negociação considerando mais detidamente aspectos da Teoria de Jogos [98], [99], [100].

Outra sugestão seria a implementação de um módulo inteligente, permitindo que o sistema fizesse sugestões ao usuário, à medida que fosse aprendendo o

comportamento e as preferências deste no decorrer das experimentações. O desenvolvimento deste módulo inteligente, baseado em um mecanismo de inferências a partir de regras, e a possibilidade da existência de mais de um tipo de produto a ser pesquisado são tarefas que se apresentam como promissoras na evolução deste trabalho. A representação do conhecimento através de regras de produção é uma das técnicas mais populares utilizadas devido a algumas de suas características, dentre elas: modularidade (forma); facilidade de implantação (escrita e programação) e, também, pelo fato de existir inúmeros pacotes ("*shells*"), para o desenvolvimento de sistemas especialistas, que as usam. Segundo Waterman [93], as regras de produção são apropriadas para representar conhecimentos oriundos de recomendações, diretrizes e estratégias. Pretende-se, com isso, dotar a arquitetura do sistema Data Agents de características de comportamento adaptativo e permitir a definição/expansão dos serviços disponíveis na aplicação de acordo com as mudanças dos recursos dos diversos bancos de dados. Além disso, os próprios usuários finais poderão definir e/ou expandir as funcionalidades de seus agentes.

Embora se tenha atingido os objetivos propostos através do desenvolvimento de uma aplicação útil e confiável, há uma expectativa muito grande que, em um futuro não muito distante, a tecnologia empregada neste trabalho possa revolucionar a forma de aquisição de produtos e serviços[13].

## Referências Bibliográficas:


1. ACM. Special issue on intelligent agents. **Communications of the ACM**, v.37, n.7, July 1994.
2. AGHA, G**. Actors: A Model of Concurrent Computation in Distributed Systems.** Cambridge: The MIT Press, 1986.
3. BASILI, V.R.; CALDIERA, G.; ROMBACH, H.D. The Goal Question Metric Approach, **Encyclopedia of Software Engineering**. USA: Wiley and Sons, 1994. p. 528-532.
5. BARR, A.; FEIGENBAUM, E. A., eds. **The Handbook of Artificial Intelligence,** Vol.1. Los Altos: Morgan Kaufmann, 1981.
6. BAX, Marcello. **Agentes de Interface para Bibliotecas Digitais.** Disponível na INTERNET via www.url: http://cuba.eb.ufmg.br/SABiO/archive.htm. Arquivo consultado em 1999.
7. BAZAN, Ana L.C. **Coordenação entre Agentes Individualmente Motivados.** Disponível na INTERNET via www.url: http://www.inf.ufrgs.br/ ~flima/ArtigoMultiAgentes.html. Arquivo consultado em 2000.
8. BIRMINGHAM, W. **An Agent-Based Architecture for Digital Libraries.** Disponível na INTERNET via www.url: http://www.cnri.reston.va.us/ home/dlib/July95/07birmingham.html. Arquivo consultado em 1995.
9. BOEHM, W. et al. Quantitative Evaluation of Software Quality. In: PROC. 2nd INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 1976, pp. 592-605
10. BOND, A.H.; GASSER, L. An Analysis of Problems and Research in DAÍ. In: BOND, A.H.; GASSER, L. (eds). **Readings in Distributed Artificial Intelligence.** California: Morgan Kaufmann Publishers, 1988. p 3-35.
11. BREDIN, J. **Market-based mobile-agent planning: a thesis proposal.** Disponível na INTERNET via www.url: http://www.cs.darmouth.edu/~jonathan/ agents/ proposal/index.html 2000. Arquivo consultado em 2000.
12. CHANDRASEKARAN, B. Natural and social system metaphors for distributed problem solving: introduction to the issue. **IEEE Transactions on S.M.C.**, 11(1): 1-5 January 1981.

13. CHAVEZ, A.; MAES, P. **Kasbah: An Agent Marketplace for Buying and Selling Goods.** Disponível na INTERNET via www.url: http://agents.www.media.mit.edu/groups/agents/publications/kasbah/paam96.ps.gz. Arquivo consultado em 1998.
14. CHEONG, F. **Internet Agents - Spiders, Wanderers, Brokers and Bots.** USA*:* New Riders, 1996.
15. COAD, P.; YOURDON, E. **Análise Baseada em Objetos**. Rio de Janeiro: Campus, 1996.
16. COCKAYNE, W.; ZYDA, M. **Mobile Agents**. Greenwich: Manning, 1998.
17. COELHO, Alexandre Rodrigues**. Linguagens e Protocolos de Comunicação entre Agentes.** Orientador: Jomi Fred Hübner. Blumenau: FURB, 1998. 70p. (Relatório Final Projeto PIBIC/CNPq)
18. CORNELL, G.; HORSTMANN, C. S. **Core Java.** São Paulo: Makron Books, 1997.
19. CORRÊA FILHO, Milton. **A Arquitetura de Diálogos entre Agentes Cognitivos Distribuídos.** Rio de Janeiro: UFRJ/COPPE, 1994. Tese de Doutorado.
20. COUTTS, I.; EDWARDS, J. Support for Component Based Systems: Can Contemporary Technology Cope? **Intelligent Systems for manufacturing**, edited by L.M. Camarinha-Matos et al. Kluwer Academic Publishers, pp. 279-288 (1998).
21. DASGUPTA, P.; NARASIMHAN, N.; MOSER, L.E.; MELLIAR-SMITH, P.M**. A Supplier-Driven Electronic marketplace using Mobile Agents.** Disponível na INTERNET via www.url: http://alpha.ece.ucsb.edu/~pdg/research/papers/ICTEChtml/onecol.html. Arquivo consultado em 1999.
22. DAVIS, R.; SMITH, R.G. Negociation as a metaphor for distributed problem solving. In: BOND, A.H. & GASSER, L., **Readings in Distributed Artificial Intelligence.** California: Morgan Kaufmann Publishers, 1988, p. 333-356.
23. DECKER., K.S. Distributed Problem-Solving Techniques: A Survey. **IEEE Transactions on S.M.C.**, 17(5): 729-740, September/October 1987.
24. DECKER, K.S.; DURFEE, E.H.; LESSER, V.R. Evaluating Research in Cooperative Distributed Problem Solving. In: GASSER, L.; HUHNS, M.N. **Distributed Artificial Intelligence II.** California: Morgan Kaufmann Publishers, 1989. p 485-519.

25. DEMAZEAU, Y.; MÜLLER, J.P. **Descentralized Artificial Intelligence**. North-Holland: Elsevier Science Publishers, 1990. p. 3-16.
26. DIAS, M. M. *et al.* **Aplicação da Tecnologia de Agentes Móveis.** Disponível na INTERNET via www.url: http://www.inf.ufsc.br/~jbosco/adriana/Madalena/artmada.html. Arquivo consultado em 1999.
27. DURFEE, E.H. The Distributed Artificial Intelligence Melting Pot. **IEEE Transactions on S.M.C.** *,* 21(6): 1301-1306, November/December 1991.
28. ENDLER, Markus. **Novos Paradigmas de Interação usando Agentes Móveis**. Disponível na INTERNET via www.url: http://www.ime.usp.br/~endler/paperlinks/sbrcslides.ps. Arquivo consultado em 1998.
29. FARACO, Rafael Avila. **Uma Arquitetura de Agentes para Negociação dentro do Domínio do Comércio Eletrônico.** Orientador: Fernando Álvaro Ostuni Gauthier. Florianópolis: UFSC, 1998. Dissertação.
30. FERRADIN, Mauri. **Protótipo de um Sistema de Pré-Matrícula via Internet utilizando agentes com a banco de dados.** Orientador: Maurício Capobiaco Lopes. Blumenau: FURB, 1998. Monografia.
31. FLEISCHHAUER, Luciana Irene Amaral. **O uso da tecnologia de agentes na integração da Programação da Produção.** Orientador: Fernando Álvaro Ostuni Gauthier. Florianópolis: UFSC, 1996. Dissertação.
***32.*** FOX, E. et al. Digital Libraries. **Communications of the ACM *38***, 4, p.23-28, Abril, 1995.
33. FRANKLIN, Stan, GRAESSER, Art. **Is it an agent or just a program?: a taxonomy for autonomous agents.** Disponível na INTERNET via www.url: http://www.msci.memphis.edu/~franklin/Agentprog.html. Arquivo consultado em 1996.
34. GASSER, L. Social conceptions of knowledge and action: DAÍ foundations and open systems semantics. **Artificial Intelligence**, 47: 107-138, 1991.
35. GASSER, L.; HUHNS, M.N. **Distributed Artificial Intelligence II.** California: Morgan Kaufmann Publishers, 1989. 519p.
36. GRAY, R. *et al.* **Mobile Agents for Mobile Computing** – Technical Report PCS-TR96-285 – Department of Computer Science – Darmouth College – 1996.
37. HAROLD, Elliotte. **Java Networking Programming**. USA: O´Reilly, 1997.

38. HEILMANN, K. et al. **Intelligent Agents: a technology and business application analysis.** Disponível na INTERNET via www.url: http://haas.berkeley.edu/~heilmann/agents/index.html. Arquivo consultado em 1995.

39. HELLER, P.; ROBERTS, S. **Java 1.1 Developer´s Handbook.** USA: SYBEX Inc, 1997.

40. HEWITT, C. Open Information Systems Semantics for Distributed Artificial Intelligence. **Artificial Intelligence**, 47: 79-106, 1991.

41. HUHNS, M. H. **Distributed Artificial Intelligence.** California: Morgan Kaufmann, 1987, 390p.

42. JACKSON, Peter. **Introduction to Expert Systems.** USA: Addison-Wesley, 1990. 526p.

43. JOLIVET, R. **Curso de Filosofia.** Rio de Janeiro:Livraria Agir Editora, 1986. 445p.

44. KOGURE, M. et al. **Quality Function Deployment and CWQC in Japan**, Quality Progress, Outubro 1983. pp.25-29.

45. KOSTER, M. ***A* Standard for Robot Exclusion.** Disponível na INTERNET via www.url: http://informação.webcraler.com/mak/projects/robots/robots.html. Arquivo consultado em 1995.

46. KOSTER, M. **Guidelines for Robots Writers.** Disponível na INTERNET via www.url: http://informação.webcraler.com/mak/projects/robots/robots.html. Arquivo consultado em 1996.

47. LANGE, Danny B.; OSHIMA, Mitsuru. **Programming and Deploying Java Mobile Agents with Aglets.** Massachusetts: Addison-Wesley, 1998.

48. LEE, K. ; MANSFIELD, W.; SHETH, A. A Framework for controlling cooperative agents**. IEEE Computer**, 26, 7, p. 8-16. July 1993.

49. LEGGETT, J. et al. Principles of human-centered information systems. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON ARTIFICIAL REALITY AND TELE-EXISTENCE/CONFERENCE ON VIRTUAL REALITY SOFTWARE AND TECHNOLOGY (ICAT/VRST) ´95, 1995, Japão. p. 173-176.

50. LEGGETT, *et al.* Viewing Dexter with open eyes**. Communications of the ACM,** 37, 2, p.76-86, Fevereiro, 1994.

51. LEMAÎTRE, C.; SÁNCHEZ, V.G.; EXCELENTE, C.B.; ZAMORA. L. Cooperative Network for Existing Expert Systems. In**:** X SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 1993, Porto Alegre, p. 3-15.
52. LESSER, V.R. A Retrospective View of FA/C Distributed Problem Solving. **IEEE Transactions on S.M.C.**, 21(6): 1347-1362, November/December 1991.
53. LESSER, V.R.; CORKILL, D.D. Funcionality Accurate, Cooperative Distributed Systems. In: BOND, A.H.; GASSER, L. (eds) **Readings in Distributed Artificial Intelligence.** California: Morgan Kaufmann Publishers, 1988. p. 295-310.
54. MINSK, M. **The Society of Mind**. Nova York: Simon & Shuster, 1985.
55. MOUKAS, A, GUTTMAN, R., MAES, P. **Agent-mediated Electronic Commerce: An MIT Media Laboratory Perspective.** Disponível na INTERNET via www.url.http://ecommerce.media.mit.edu/papers/ker98.pdf. Arquivo consultado em 1999.
56. NEWMAN, Alexander. **Usando Java.** Rio de Janeiro: Campus, 1997.
57. NIGAY, L.; COUTAZ, J. PAC-Expert: **Towards an automatic generation of dialogue controllers.** Tech. Rep. RT 83, Laboratoire de Génie Informatique - IMAG, University of Grenoble, France, 1992.
58. NWANA, Hyacinth S. **Software Agents: An Overview. Knowledge Engineering Review.** Disponível na INTERNET via www.url: http://ww.cs.umb.edu/ agent/papers/ao.ps. Arquivo consultado em 1999.
59. O´CONNOR, D. *et al.* **Intelligent Agent Strategy: White Paper.** Disponível na INTERNET via www.url: http://activist.gpl.ibm.com:81/WhitePaper/ptc2.htm. Arquivo consultado em 1996.
60. O´HARE; JENNINGS. **Foundations of Distributed Artificial Inteligence.** USA: John Wiley & Sons, 1996.
61. ORLANTA, *et al.* **Intelligent Agents & The Internet: Effects on Electronic Commerce and Marketing.** Disponível na INTERNET via www.url.: http://asterix.ist.utl.pt/massdit/agcompras/referencias.html. Arquivo consultado em 1997.
62. OSHIMA, M.; KARJOTH, G. **Aglets Specification**. Disponível na INTERNET via www.url.: www.trl.ibm.co/aglets/spec_alpha.html. Arquivo consultado em 1997.

63. PARUNAK, H. V. D. Manufacturing Experience with the Contract Net. In: HUHNS, M.N. **Distributed Artificial Intelligence.** California: Morgan Kaufmann Publishers, 1987. p 285-310.

64. PARUNAK, H. V. D. Distributed AI and manufacturing Control: Some Issues and Insights. In: DEMAZEAU, Y.; MÜLLER, J.P. **Descentralized Artificial Intelligence.** North-Holland: Elsevier Science Publishers, 1990. pp 81-101.

***65.*** PASTORINO, Alexandre S.; LOPES, João Landislau. **A Proposta Cliente/Servidor e a Informatização da UFPel -** Disponível na INTERNET via www.url.: http://akira.ucpel.tche.br/bbvirt/pos/ufpel.ps. Arquivo consultado em 1998.

66. PAULA, G. E., RAMOS, F.S., RAMALHO, G.L. **Uma Arquitetura para Agentes Negociadores baseada em Teoria dos Jogos.** Disponível na INTERNET via correio eletrônico: gep@di.ufpe.br. Arquivo consultado em 1999.

***67.*** PAPAIOANNOU, Todd; EDWARDS, John. **Towards Understanding and Evaluating Mobile Code Systems.** Disponível na INTERNET via www.url: http://luckyspc.lut.ac.uk/Docs/. Arquivo consultado em 2000.

68. PIRES, Paulo. **Arquitetura Cliente/Servidor - A Chave para o Projeto de Sistemas Distribuídos.** Disponível na INTERNET via www.url: http:www.cos.ufrj.br/~pires/arqClienteServidor.ps.zip Arquivo consultado em 1998.

69. PIERITZ, Rogério. **Código Móvel na Internet com Acesso a Banco de Dados Relacional**. Orientador: Jomi Fred Hubner. Blumenau: Universidade Regional de Blumenau, 1997. 72p. Monografia.

70. REINHARDT, Andy. A rede inteligente. **Byte Brasil**, 3(10): 42-52, Outubro, 1994.

71. RICH, Elaine; KNIGHT, Kevin. **Inteligência Arificial.** Makron Books, 1994. 722p.

72. RIECKEN, Doug. Intelligent Agent. **Communications of the ACM.** 37(7): 18-21, July, 1994.

73. RIVER, Charles. **Agent technology.** Disponível na INTERNET via www.url: http://www.internet.ve/ramirez/MBA/. Arquivo consultado em 1996.

74. RODRÍGUEZ-AGUILAR, J. A; NORIEGA, P.; SIERRA, C.; PADGET, J. **A Java-based Electronic Auction House.** Disponível na INTERNET via

www.url.http://www.iiia.csic.es/~sierra/Publications.html. Arquivo consultado em 1999.

75. RODRÍGUEZ-AGUILAR, J. A; NORIEGA, P.; SIERRA, C.; MARTÍN, F.J, GARCIA, P. **Competitive Scenarios for Heterogeneous Trading Agents.** Disponível na INTERNET via www.url: http://www.iiia.csic.es/~sierra/ Publications.html. Arquivo consultado em 1999.
76. RODRÍGUEZ-AGUILAR, NORIEGA, P., J. A, Sierra, C, Martín, F.J. **Towards a Test-bed for Trading Agents in Electronic Auction Market.** Disponível na INTERNET via www.url.http://www.iiia.csic.es/~sierra/Publications.html. Arquivo consultado em 1999.
77. RUBINSTEIN, Marcelo. **Avaliação do Desempenho de agentes Móveis no gerenciamento de redes.** Disponível na INTERNET via www.url: http://www.gta.ufrj.br/ftp/gta/TechReports/Rubi99/Rubi99.ps.gz. Arquivo contado em 1999.
78. RUMBAUGH, et al. **Modelagem e Projetos Baseados em Objetos.** Rio de Janeiro: Campus, 1994.
79. RUSSEL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach.** USA: Prentice Hall, 1995. 932p.
80. SHAM, M.J.; WHINSTON, A.B. Learning and Adaptation in Distributed Artificial Intelligence Systems. In: GASSER, L.; HUHNS, M.N**. Distributed Artificial Intelligence Systems II.** California: Morgan Kaufmann Publishers, 1989. p. 413-429.
81. SHEN, S. *et al.* An Interoperable Architecture for Digital Information Repositories. In: PROCEEDINGS OF DIGITAL LIBRARIES ´94. Março, 1994.
82. SHOHAM, Y. Agent-oriented programming. **Artificial Intelligence.** 60:51-92, 1993.
83. SICHMAN, Jaime; DEMAZEAU, Yves. A Model for the Decision Phase of Autonomous Belief Revision in Open Multi-Agent Systems**. Journal of the Brazilian Computer Society**, 3(1): 40-50, July 1996.
84. SICHMAN, Jaime; DEMAZEAU, Yves; BOISSIER, O. When can knowledge-based systems be called agents? In: XII CONGRESSO DA SOCIEDADE

BRASILEIRA DE COMPUTAÇÃO, IX SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 1992. p.172-185.

85. SOARES, Luiz Fernando G., LEMOS, Guido & COLCHER, Sérgio. **Redes de Computadores: das LANS, MANS, e WANS às redes ATM.** Rio de Janeiro: Campus, 1995. 705 p.
86. SOMMERS,B. **Agents: not just for Bond anymore.** Disponível na INTERNET via www.url: http://www.javaworld.com/javaworld/jw-04-1997/jw-04-agents.html Arquivo consultado em 1998.
87. STREETER, L. et al. Interface styles: Direct manipulation versus social interaction. In: PROCEEDINGS OF THE CHI ´95. New York: ACM Press, 1995.
88. SULLIVAN, D. **A Webmaster's Guide to Search Engines.** Disponível na INTERNET via www.url: http://calafia.com/. Arquivo consultado em 1997.
89. SUNDSTED, Todd. **Agents on the Move.** Disponível na INTERNET via www.url: www.javaworld.com/javaworld/jw-07-1998/jw-07-howto.html. Arquivo consultado em 1998.
90. TOMINAGA, Paulo. **Processos de Negociação em Sistemas Multi-Agentes: Modelagem e Análise com Redes de Petri.** Orientador: Wagner Teixeira da Silva. Brasília: UnB, 1996. Dissertação.
91. TOMINAGA, Paulo; SILVA, Wagner T. **Inteligência Artificial Distribuída.** Disponível na INTERNET via www.url: ftp://ftp.cic.unb.br/pub/ publications/ report/tr.94-01.ps. Arquivo consultado em 1997.
92. VERNNES, B. **Java´s security architecture - an overview of the JVM´s security model and a look at this built-in safety features.** Disponível na INTERNET via www.url: http://www.javaworld.com/javaworld/jw-08-1997/jw-08-hood. html. Arquivo consultado em 1998.
93. WATERMAN, D. **A Guide to Expert systems**, USA: Addison-Wesley, 1986.
94. WATSON, Mark. **Intelligent Java Applications for the Internet and Intranets.** San Francisco: Morgan Kaufmann Publishers, 1997.
95. WURMAN, P. R., WELLMAN, M. P., WALSH, W.E. **The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents.** Disponível na INTERNET via www.url: http://auction.eecs.umich.edu/ papers.html. Arquivo consultado em 1999.

96. JEPSON, B. **Programando Banco de Dados em Java.** São Paulo: Makron Books, 1997.

97. ZEIGLER, B. **Object-Oriented Simulation with Hierarchical.** USA: Academic Press, 1992.

98. ZLOTKIN, G., ROSENSCHEIN, J. S. Negotiation and Task Sharing among Autonomous Agents in Cooperative Domains. In: PROCEEDING OF IJCAI´89, Detroit Michigan. pp. 912-917, 1989.

99. ZLOTKIN, G., ROSENSCHEIN, J. S. Negotiation and Conflict Resolution in Non-Cooperative Domains. In: THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI-90), Boston, Massachusetts, pp.100-105, 1990.

100. ZLOTKIN, G., ROSENSCHEIN, J. S. Mechanism for Automated Negotiation in Stated Oriented Domains. **Journal of Artificial Intelligence Research**, 5. pp. 163-238, 1996.