UNIVERSIDADE SALVADOR - UNIFACS
PROGRAMA DE PÓS-GRADUAÇÃO EM REDES DE COMPUTADORES
MESTRADO PROFISSIONAL EM REDES DE COMPUTADORES

JOACIR SIMÕES FERREIRA

AGENTES MÓVEIS PARA COLETA DE INFORMAÇÕES USANDO O DATAMINING

Salvador
2007

JOACIR SIMÕES FERREIRA

AGENTES MÓVEIS PARA COLETA DE INFORMAÇÕES USANDO O DATAMINING

Dissertação apresentada ao Curso de Mestrado Profissional em Redes de Computadores da Universidade Salvador, como requisito parcial para obtenção do título de Mestre.

Orientador: Prof. Dr. Joberto Sérgio Barbosa Martins.

Salvador
2007

FICHA CATALOGRÁFICA
(Elaborada pelo Sistema de Bibliotecas da Universidade Salvador - UNIFACS)

Ferreira, Joacir Simões
Agentes móveis para coleta de informações usando datamining/Joacir Simões Ferreira - 2009.
80 f.

Dissertação (mestrado) – Universidade Salvador – UNIFACS. Mestrado em Redes de Computadores, 2009.

Orientador: Prof. Dr. Joberto Sérgio Barbosa Martins

1. Agentes móveis. 2. Sistemas Distribuídos 3. do Conhecimento. 4. Mineração de dados I. Castelo Branco, Favônia Reis, orient. II. Universidade Salgado de Oliveira - Universo. III. Título.

CDD: 004.36

PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO
MESTRADO EM SISTEMAS E COMPUTAÇÃO

JOACIR SIMÕES FERREIRA

AGENTES MÓVEIS PARA COLETA DE INFORMAÇÕES USANDO O DATAMINING

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Redes de Computadores, Universidade Salvador - UNIFACS, do Curso de Mestrado Profissional em Redes de Computadores, pela seguinte banca examinadora:

Prof. Joberto Sérgio Barbosa Martins (Orientador)
Doutor em Informatique, Universite de Paris VI (Pierre et Marie Curie), U.P. VI, França.
Universidade Salvador - UNIFACS

Profa. Débora Abdalla dos Santos
Doutora em Ciência da Computação, Universidade Federal de Pernambuco, UFPE, Brasil.
Universidade Federal da Bahia - UFBA

Prof. Manoel Gomes de Mendonça Neto
Doutor em Ciência da Computação. Universidade de Maryland Em College Park, UMCP, Estados Unidos.
Universidade Salvador – UNIFACS

Salvador, 01 de agosto de 2007.

AGRADECIMENTOS

Agradeço a Deus em primeiro lugar, e reservo também um agradecimento especial para os responsáveis por minha existência neste mundo, aos

Agradeço com todo meu carinho a minha noiva, amada e companheira, pela paciência em meus longos períodos de ausência dedicados à elaboração deste trabalho e por sua ajuda preciosa nos momentos em que me confortava dando amor, atenção e suporte nos diagramas lógicos de objetos. Meus pais, JOSÉ FERREIRA FILHO e EUGÊNIA SIMÕES DE A. FERREIRA, que desde cedo incentivaram os meus estudos, e tornaram possível o meu desenvolvimento escolar, enxergando que somente a educação é capaz de elevar o ser humano em sua caminhada moral e social.

Agradecimentos especiais ao meu orientador, professor Dr. Joberto Sérgio Barbosa Martins e Professor Dr. Manoel Gomes de Mendonça Neto, pelos conhecimentos fornecidos e pela paciência e dedicação oferecidas ao longo de todo o caminho.

Finalmente, agradeço a todos os meus amigos que diretamente ou indiretamente contribuíram na realização deste trabalho.

“Se você acha que pode ou que não pode, nos dois casos você tem razão.”

Henry Ford.

RESUMO

A gestão do conhecimento tem tido papel de destaque na rotina dos profissionais e especialistas, principalmente, por aqueles que atuam em atividades com caráter essencialmente criativo que dependem diretamente do conhecimento das pessoas que as realizam. Localizá-los e recuperá-los adequadamente contribui para a melhoria da gestão do conhecimento. Observa-se, porém, que as informações localizam-se em ambientes distribuídos, sendo necessária uma aplicação específica para recuperar adequada e seletivamente as informações desejadas. Nos últimos anos, a mobilidade de código surgiu como uma nova abordagem para a construção de aplicações distribuídas. Os Agentes Móveis são uma técnica surgida nas áreas de Computação Móvel e Sistemas Distribuídos cujo objetivo é resolver o problema de integração e mobilidade da informação. Esse trabalho é motivado pela necessidade de conhecer e avaliar novas formas de coleta de dados em redes Coorporativas, de modo eficiente, com o menor esforço humano, baseando-se numa estratégia de agentes móveis, capazes de coletar dados dos dispositivos com posterior interpretação dos mesmos.

Palavras-chaves: Agentes móveis. Gestão do conhecimento. Mineração de dados

ABSTRACT

The management of the knowledge has had paper of prominence in the routine of the professionals and specialists, mainly, for whom they act in activities with essentially creative character that they depend directly on the knowledge of the people who carry through them. To locate them and to recoup adequately contribute them for the improvement of the management of the knowledge. It is observed however, that the information be situated in distributed environments, being necessary a specific application to selectively recoup adequately and the desired information. In recent years, the code mobility appeared as a new boarding for the construction of distributed applications. The Mobile Agents are one technique appeared in the areas of Mobile Computation and Distributed Systems whose objective is to decide the problem of integration and mobility of the information. This work is motivated by the necessity to know and to evaluate new forms of collection of data in Corporative nets, efficient way, with the lesser human effort, being based on a strategy of agents mobile, capable to collect given of the devices with posterior interpretation of the same ones.

Key-words: Mobile agents. Knowledge Management. Datamining.

LISTA DE FIGURAS

Figura 1 -	Servidor Tahiti	23
Figura 2 -	Instâncias Servidor e Cliente do Tahiti	24
Figura 3 -	Modelo Conceitual da Plataforma Aglets	25
Figura 4 -	Contexto Aglet	26
Figura 5 -	Modelo do Ciclo de vida de um agente	27
Figura 6 -	Padrão de comunicação mestre-escravo	31
Figura 7 -	Fases do processo de descoberta do conhecimento	34
Figura 8 -	Arquitetura da ferramenta ExperText	40
Figura 9 -	Abstract Factory utilizado no acesso a diferentes tipos de bancos de dados	42
Figura 10 -	Arquitetura do sistema SACI	44
Figura 11 -	Arquitetura dos Agentes – Agente Coordenador	45
Figura 12 -	Sistema SACI – Funcionamento do Protótipo. Processo de financiamento	48
Figura 13 -	Rede (física) de implementação do Protótipo	49
Figura 14 -	Modelo de Dados utilizado pelo Sistema SACI	50
Figura 15 -	Modelo de Dados uitilizado pela ferramenta ExperText	51
Figura 16 -	Fluxo de Recuperação e Busca – Sistema SACI	53
Figura 17 -	Fluxo de Autalização de Dados de Pesquisados – Sistema SACI	55
Figura 18 -	Funcionamento Geral do sistema	56
Figura 19 -	Fluxo de Atividades Básicas da Ferramenta	57
Figura 20 -	Operação do sistema com grande quantidade de arquivos	61
Figura 21 -	Performance da rede de comunicação de dados	62

LISTA DE TABELAS

Tabela 1 - Quantidade de projetos obtidas por área de atuação 59

SUMÁRIO

1	**INTRODUÇÂO**	12
1.1	OBJETIVO DA DISSERTAÇÃO	13
1.2	CONTEXTO DA PESQUISA	13
1.3	ABORDAGEM ADOTADA	14
1.4	ESTRUTURA DA DISSERTAÇÃO	15
2	**AGENTES MÓVEIS**	17
2.1	DEFINIÇÃO E PRINCIPAIS CARACTERÍSTICAS	17
2.2	PRINCIPAIS PLATAFORMAS DE DESENVOLVIMENTO	21
2.3	MODELO CONCEITUAL	25
2.4	AGENTES MÓVEIS – CONSIDERAÇÕES ADICIONAIS	29
3	**RECUPERAÇÃO DE DADOS E GESTÃO DO CONHECIMENTO**	32
3.1	O PROCESSO DA RECUPERAÇÃO DE DADOS (DATAMINING)	32
3.2	IMPORTÂNCIA DO CONHECIMENTO	35
3.3	A FERRAMENTA EXPERTEXT	38
4	**ARQUITETURA DO SISTEMA**	43
4.1	FUNCIONALIDADES DO SISTEMA SACI	46
5	**IMPLEMENTAÇÃO DO PROTÓTIPO**	48
5.1	CENÁRIO	48
5.2	OPERAÇÃO DOS AGENTES	52
5.3	OPÇÕES NO USO DA FERRAMENTA	55
5.4	COMO FUNCIONA O SISTEMA	56
5.5	TESTES E RESULTADOS	58
5.5.1	**Testes realizados com o uso da ferramenta**	59
5.5.1.1	Processo de escolha e análise dos projetos financeiros	59

5.5.1.2 Operacionalização do processo de cadastro e instalação do sistema 60
5.5.1.3 Compatibilidade da ferramenta com outros aplicativos 60
5.5.1.4 Operacionalização com grande quantidade de arquivos de rede 61
5.5.1.5 Performance da rede de comunicação de dados 61
5.6 ANÁLISE 62

6 CONSIDERAÇÕES FINAIS 64
6.1 CONTRIBUIÇÕES 65
6.2 PERSPECTIVAS 66

REFERÊNCIAS 68

APÊNDICE A - Instalação do Ambiente AGLET 73

1 INTRODUÇÃO

Nos últimos anos, a mobilidade de código surgiu como uma nova abordagem para a construção de aplicações distribuídas. Essa nova abordagem baseia-se na idéia de que um processo tenha condições, de forma dinâmica, de suspender sua execução corrente em um determinado local, mover-se ou ser movido para outro local e lá continuar sua execução. Nessa abordagem, o programador possui controle sobre o momento em que se faz necessária a movimentação de seu processo e sobre o local para onde tal processo deva ir. Além disso, o processo de movimentação em si é totalmente transparente ao programador, transparência essa, fornecida por plataformas de suporte à mobilidade, tais como as plataformas Voyager (GLASS, 1999) e Aglets da IBM (AGLETS..., 2002).

O aumento do número de redes instaladas, da mobilidade dos usuários envolvidos, bem como da sua complexidade, tanto em relação aos serviços como em relação à sua heterogeneidade nas tecnologias envolvidas, tem dificultado a atividade de gerência do sistema e de implementação de soluções mais eficientes em diversas áreas do conhecimento. Neste ambiente, há dificuldade pela diversidade de formas de controle, monitoração e coleta de dados, visto que, os produtos envolvidos na rede tornam-se gradativamente mais inteligentes e mais heterogêneos na forma como são conectados e utilizados.

A Computação Móvel teve uma evolução considerável nos últimos anos impulsionando o desenvolvimento das tecnologias dos sistemas portáteis (Laptop, PDA, Celulares, etc.) permitindo maior mobilidade dos indivíduos. Esta mobilidade vem a dificultar a coleta de dados, devido a falta de uma referência fixa com relação às entidades clientes envolvidas.

Os Agentes Móveis são uma técnica surgida nas áreas de Computação Móvel e Sistemas Distribuídos cujo objetivo é resolver o problema de integração e mobilidade da informação (OLIVEIRA, 2004).

Muitos estudos têm sido feitos, visando à compreensão e desenvolvimento de ferramentas capazes de analisar dados, transformando-os em informações úteis. Estudos como os realizados por Oliveira (2004), onde é apresentada a ferramenta EXPERTEXT, construída com o objetivo de classificar automaticamente documentos escritos na forma textual.

Esse trabalho é motivado pela necessidade de conhecer e avaliar novas formas de coleta de dados em redes Coorporativas, de modo eficiente, com o menor esforço humano, baseando-se numa estratégia de agentes móveis, capazes de coletar dados dos dispositivos com posterior interpretação dos mesmos. Como ferramenta de análise dos dados foi utilizado o EXPERTEXT já mencionado anteriormente.

Especificamente, o trabalho de dissertação propõe um estudo da integração dos conceitos de agentes móveis e de *datamining*, bem como da implementação de um modelo de validação para a coleta de informações.

1.1 OBJETIVO DA DISSERTAÇÃO

O objetivo desta dissertação é desenvolver um sistema que colete informações em uma rede corporativa usando agentes móveis, informações estas que através de técnicas de mineração de dados (*dataminig)* e do uso da ferramenta *Expertext* irão gerar relatórios que servirão de base para tomada de decisões gerenciais. Como ambiente de testes, será feito um estudo de caso em uma rede corporativa. O trabalho visa também, garantir na medida do possível, a portabilidade da solução proposta, atributo essencial para o desenvolvimento de aplicações que utilizem agentes móveis. A principal contribuição deste trabalho será oferecer um estudo que sirva como base para futuras aplicações que envolvam o uso de agentes móveis na coleta de informações em bases distribuídas do conhecimento.

Neste contexto, o trabalho pretende estudar as principais técnicas e plataformas de desenvolvimento utilizadas na implementação de agentes móveis, identificando a mais viável e estável para a realização da pesquisa, combinando a uma ferramenta de categorização automática de documentos, a *Expertext*.

1.2 CONTEXTO DA PESQUISA

Esta dissertação vem contribuir com os trabalhos desenvolvidos pelo Núcleo de Pesquisa em Redes e Computação (NUPERC) da Universidade Salvador (UNIFACS), onde foram desenvolvidas pesquisas nas áreas de gestão do conhecimento envolvendo técnicas de mineração de dados abrangendo a

elaboração de uma ferramenta para mapeamento do conhecimento e o trabalho de Oliveira (2004), abrangendo a implementação da ferramenta EXPERTEXT, construída com o objetivo de fornecer uma forma de acompanhar todo o processo de combinação de *expertise* de classificadores de textos, baseados no teorema de Bayes.

Como ambiente de testes, foi utilizado a rede corporativa da Desenbahia, onde foi desenvolvido e implementado um modelo de testes de recuperação e análise de dados distribuídos, combinando agentes móveis e a ferramenta Expertext.

O trabalho foi desenvolvido na Agência de Fomento do Estado da Bahia (Desenbahia), onde o autor desta dissertação atua como analista de tecnologia e segurança da informação, bem como faz parte do grupo de pesquisas institucionais para a gestão do conhecimento.

O sistema implementado apresentou resultados relevantes no ambiente de estudo a que foi submetido, a exemplo do conhecimento obtido da grande quantidade de projetos financeiros na área denominada microcrédito com pendências e irregularidades dos proponentes junto ao Serasa e a outras instituições de controle do crédito, demonstrando a carência de um maior investimento e atuação pró-ativa nas áreas de controle e cadastro de propostas.

1.3 ABORDAGEM ADOTADA

No desenvolvimento da solução para busca e tratamento de informações em Redes Coorporativas utilizando-se sistemas distribuídos (agentes móveis), duas abordagens podem ser utilizadas: (1) a construção de toda a infra-estrutura necessária à mobilidade e comunicação dos agentes, ou (2) a utilização de uma infra-estrutura já existente. Avaliando-se o investimento necessário à construção de uma plataforma de agentes, no que se refere ao escopo deste trabalho, chega-se à conclusão de que a segunda alternativa é aquela que se apresenta como mais adequada e eficiente com relação à solução do problema.

Para a utilização de uma plataforma já existente, fez-se então necessário um estudo aprofundado das técnicas, linguagens e bibliotecas relacionadas. No caso da plataforma adotada, os *Aglets*, um estudo sobre Java (LEMAY, 2001) - a linguagem

de programação utilizada para o desenvolvimento dos agentes nesta plataforma - e sobre as próprias classes relacionadas aos *Aglets* (GRIMSHAW, 2001; HORVAT, 2000; LANGE, 1998b) mostra-se suficiente e eficiente do ponto de vista da abordagem.

A plataforma *Aglets* foi escolhida por "funcionar de forma apropriada no ambiente da rede coorporativa (intranet) e, por ser reconhecidamente robusta e utilizada por um grande número de servidores." (HORVAT, 2000, p.8).

Como ferramenta de análise e apresentação dos dados, foi escolhida a Ferramenta *Expertext,* que apresentava razões tais como: ser uma ferramenta baseada em *java* apresentando com isso compatibilidade de integração com a plataforma *Aglet;* o aplicativo estar disponível, fácil acesso aos autores, entre outras.

1.4 ESTRUTURA DA DISSERTAÇÃO

Foi apresentada uma introdução, é contextualizando o tema e apresentado o problema a ser trabalhado. Foram apresentados os objetivos da pesquisa bem como a sua justificativa.

No capítulo 2, são abordados aspectos relacionados aos agentes móveis, tais como definição, características, modelo conceitual, principais aplicações e principais plataformas de desenvolvimento.

No Capítulo 3, é abordado o processo de recuperação de dados e gestão do conhecimento bem como, uma análise da ferramenta *Expertext*, ferramenta usada como apoio à análise dos dados coletados.

No Capítulo 4, é apresentada a arquitetura do sistema proposto.

No Capitulo 5, são apresentados aspectos sobre a implementação do protótipo bem como a descrição do ambiente de desenvolvimento utilizado.

No Capítulo 6, são apresentadas as considerações finais, onde são analisados os resultados do trabalho desenvolvido e apresentadas sugestões para trabalhos futuros.

O Apêndice A, desta dissertação contêm informação adicional, necessária à implementação do ambiente de desenvolvimento utilizado no trabalho desenvolvido.

2 AGENTES MÓVEIS

2.1 DEFINIÇÃO E PRINCIPAIS CARACTERÍSTICAS

Um agente móvel é um elemento de software auto-contido sendo muitas vezes definido como um *agente de software*. Um agente de software é uma entidade autônoma responsável pela execução de uma atividade, que possui a capacidade de *aprender* e *cooperar* com outros agentes de software. Além disso, um agente móvel é um *agente de software móvel*, capaz de mover-se baseado em algum padrão ou regra. Esta mobilidade é particularmente interessante porque permite que um agente possa atuar em diferentes localizações de uma rede (BERNADES, 1999).

Um sistema de agentes móveis é uma plataforma de software que define os mecanismos de mobilidade e de comunicação entre agentes de software (BERNADES, 1999). Os componentes básicos de um sistema de agentes móveis são:

a) agente - contêm identificador, estado, implementação e interface, além da indicação em nome de quem está agindo (responsável);
b) lugar ou agência - o ambiente de execução de um agente.

Um agente migra em um ambiente distribuído de uma agência a outra. Uma agência representa um lugar lógico no sistema distribuído. Quando um agente migra, sua execução é suspensa na agência original, o agente é transportado (i.e., código, dados e estado de execução) para uma outra agência no ambiente distribuído, onde a execução é retomada (SILVA, 1999).

Cada agente móvel possui um conjunto de modelos que definem seu comportamento. Semelhante a agentes usados em Inteligência Artificial, existe um modelo para o *ciclo de vida* do agente, um modelo *computacional*, um modelo de *segurança* e um modelo de comunicação (BITTENCOURT, 1998). O ciclo de vida de um agente define como o mesmo se comportará durante sua existência, em que circunstâncias haverá clonagem do agente, quando uma instância deve ser excluída, etc. O modelo de comunicação não define um protocolo de comunicação (por exemplo, *Network Management Protocol* (SNMP) onde é definido as regras específicas para a realização de uma comunicação entre os entes envolvidos, mas sim a forma como os agentes se comunicam entre si e com o ambiente. Além dos

modelos citados, o modelo mais importante de um agente móvel é o modelo de *locomoção* que define como um agente move-se na rede. De acordo com este modelo, um agente é capaz de decidir, por exemplo, qual o próximo nodo de uma rede a ser visitado e se isso já ocorreu, que outra ação de locomoção deve ser tomada.

Os modelos que definem o agente devem interagir entre si para formar o comportamento do agente móvel (GRANVILLE, 2000). Por exemplo, se todos os nodos de uma rede foram visitados, será que o agente deve ser excluído? Uma relação entre o modelo de locomoção e do ciclo de vida do agente responde esta questão. Qual o próximo nodo a ser visitado, tendo em vista que o agente deve procurar por dispositivos com novos dados? Uma relação entre o modelo de locomoção e o modelo computacional resolve esta outra questão.

As capacidades de aprender e cooperar dos agentes móveis, citadas anteriormente, tem impacto direto no comportamento dos mesmos. Isso significa que um agente é capaz de analisar os dados coletados e com isso tomar alguma decisão, e principalmente registrar essa ação internamente e poder assim agilizar futuras operações. Isto é, aprender (GRANVILLE, 2000). Além da interação com os dispositivos de rede um agente deve interagir com outros agentes, para que as tarefas possam ser executadas cooperativamente. Esta interação permite ainda que um agente possa aprender não apenas com as suas ações, mas também através do conhecimento das ações de outros agentes. Isto é, colaboração (GRANVILLE, 2000).

Os agentes móveis não possuem uma grande complexidade, apesar de realizarem tarefas complexas como quando são utilizados na detecção de problemas de performance em uma rede (BERNADES, 1999). Cada agente de um sistema pode ser muito simples, executando um conjunto de tarefas bem limitado. Por outro lado, o comportamento resultante da interação entre estes agentes simples é capaz de solucionar problemas complexos de uma rede. Isso facilita muito, a tarefa de criação e manutenção dos agentes por parte do usuário, sem que se perca com isso capacidade de resolução dos problemas.

De acordo com Lange (1998), a tecnologia de agentes móveis pode ser empregada em diferentes aplicações dentre elas: comércio eletrônico,

gerenciamento de grupo de trabalhos, gerenciamento de redes, serviços de telecomunicação, recuperação de informações distribuídas e redes ativas.

Alguns outros conceitos adotados para os agentes móveis são os seguintes:

Para Koch e Westphal (2001),

a) agentes devem ser **realmente autônomos**, capazes de controlar a si mesmo e a suas ações e trabalhar independente do que aconteça com outros agentes, tendo total liberdade para adaptar-se a possíveis mudanças no ambiente.
b) agentes podem ser **guiados por metas e baseados em regras**, sendo que um conjunto de regras constitui a base de conhecimento para qualquer sistema e as metas são os objetivos de trabalho dos agentes. Defini-se metas estáticas como parâmetros de configuração fixados na criação do agente, e metas dinâmicas como aquelas adicionadas ao comportamento inicial do agente e que foram derivadas de mudanças do ambiente ou de mensagens recebidas.
c) agentes devem ser capazes de **aprender novas habilidades**, e essas novas regras podem ser armazenadas dinamicamente nas bases locais de conhecimento dos agentes; outra possibilidade é inferir novas regras a partir do conhecimento atual do ambiente.
d) especificamente para Agentes Gerentes de Rede, **eles devem interagir com soluções comerciais usando protocolos padronizados**, ou seja, os agentes devem saber, por meio de regras em suas bases de conhecimento, como interagir com protocolos de domínio público como o *Simple Network Management Protocol* (SNMP), como resgatar informações dos arquivos de registro (log) e também como interagir com outros sistemas de gerência existentes. Tornar o seu sistema compatível significa facilidade de aceitação no mercado.

No mercado encontramos também o mecanismo de *Remote Procedure Call* (RPC), que permite que programas chamem procedimentos localizados em outras máquinas. Quando um processo rodando em uma máquina A chama um procedimento em uma máquina B, o processo que chamou é suspenso, e a execução do procedimento é levada a efeito na máquina B. A informação pode ser

transportada do processo que chama para o procedimento chamado através de parâmetros, e voltar para o processo como resultado da execução do procedimento.

Os Agentes Móveis diferem do mecanismo de RPC quanto ao fato deste limitar-se ao envio de dados para execução remota. Agentes transportam código e dados (BIESZCZAD, 1998).

Para o desenvolvimento de Agentes Móveis deve-se utilizar linguagens de programação que possibilitem o transporte de código e dados, e a definição do comportamento dos agentes durante a migração e testes de segurança.

Bieszczad (1998) apresenta aspectos da aplicação de Agentes Móveis na gerência de redes. Agentes Móveis podem encapsular *scripts* de gerência para que estes executem sob demanda onde for necessário. Um agente pode ser enviado para coletar dados entre os elementos de uma rede e voltar com os dados filtrados e coletados. Se o tamanho do agente permanecer pequeno haverá economia no uso da banda. Se o agente puder tomar decisões durante suas operações mais banda será economizada.

Uma característica também importante a ser observada, é a forma de operação dos agentes. Segundo Kendal e outros (1998), as formas de operação dos agentes móveis podem ser divididas em várias categorias, são elas:

a) **forma de ataque**: solitário, quando o agente é programado e migra por todos os pontos da rede executando uma tarefa ou em bandos, quando são enviados vários agentes concomitantemente para vários locais para realização de uma mesma tarefa;
b) **forma de movimentação**: alfândega, o agente deve possuir uma espécie de registro para usufruir do local; trilha, são definidos preliminarmente os percursos para atingir o destino; sinais, não se sabe previamente o trajeto, mas cada um é redirecionado para cada novo ponto ao se chegar nas marcas específicas;
c) **forma de cooperação**: não cooperativa, os agentes não trocam informações entre si; compartilhada, trocam informações com outros na intenção de executar determinada tarefa; controlada, quando se têm certos agentes que agem como gerentes de grupo e controlam as tarefas executadas por um grupo;

d) **forma de aprendizado**: processo em que o agente processa as informações obtidas ao longo de seu ciclo de vida, adaptando seu curso de ação em função das novas informações.

A aplicação de Agentes Móveis deve ser examinada com atenção. Questões relativas à segurança, a real utilização de banda e a eficiência da solução devem ser consideradas quando da aplicação desta tecnologia (BERNADES, 1999).

2.2 PRINCIPAIS PLATAFORMAS DE DESENVOLVIMENTO

Como principais plataformas de desenvolvimento de agentes temos:

a) **Aglets -** módulo para o desenvolvimento de agentes da empresa IBM (CHANG,1996). Foi desenvolvido em Java. Considerado atualmente a principal plataforma de desenvolvimento de aplicações com agentes móveis;
b) **Concórdia -** esforço do laboratório de sistemas Horizonte da empresa Mitsubishi Electric Information Center America. É desenvolvido inteiramente em Java e apresenta versões para rodar na plataforma Windows (95, 98 e NT) e para o sistema Solaris. Possui classes específicas de agentes permitindo o desenvolvimento dos mesmos de forma simples (WONG, 1997);
c) **Inferno -** camada de software da Lucent Technologies, o laboratório de pesquisa dos laboratórios Bell. É uma linguagem para aplicações distribuídas que incluí protocolos e APIs, isto é, um sistema completo (INFERNO..., 1997);
d) **Mole -** módulo de desenvolvimento de agentes que utiliza a linguagem Java. Está sendo desenvolvido pelo grupo de Sistemas Distribuídos da universidade de Stuttgart (Alemanha). O módulo permite a escrita e execução de agentes móveis em Java (STRABER, 1997);
e) **Mozart -** sistema de suporte a computação distribuída aberta e a inferência baseada em restrições. Implementa a linguagem concorrente e objeto orientada Oz, para o desenvolvimento de sistemas multiagentes. O projeto é uma colaboração entre os departamentos de Ciências Computacionais das universidades DFKI (Alemã) e Católica de Louvain (Bélgica) e o Laboratório de Sistemas Inteligentes do Instituto Sueco de Ciências Computacionais (ROY, 2005);

f) **MSAgent -** camada de software desenvolvido pela empresa Microsoft que é integrada ao sistema operacional Windows (95, 98 e NT). Utiliza a tecnologia COM e ActiveX da empresa. Pode ser utilizado diretamente pelos navegadores através de uma marcação <OBJECT> ou então inserido como componente nas linguagens que suportam a tecnologia COM.

g) ***TronsØ And Cornell Moving Agents* (TACOMA) -** camada de software para o suporte de processos (agentes), escrito em C e que utiliza o protocolo TCP. Suporta as linguagens C, C++, Perl, Python, Scheme e Visual Basic e as plataformas Unix e Windows (95 e NT). Está sendo desenvolvido como um esforço conjunto entre os departamentos de Ciências Computacionais das Universidades de Cornell (EUA), Tronso (Noruega) e da Califórnia em São Diego (JOHANSEN, 1999).

Na implementação de agentes, uma opção é o desenvolvimento sobre camadas de software criadas para facilitar o seu desenvolvimento e a sua implementação e que rodam sobre um sistema operacional, ou seja, são extensões do sistema operacional com o propósito específico de gerenciamento, desenvolvimento, coordenação e comunicação entre agentes. A utilização de um destes ambientes permite alguma flexibilidade na escolha da linguagem de programação.

A grande vantagem desta forma de implementação é a facilidade de criação e manutenção dos agentes, pois grande parte do trabalho de programação já foi realizado, sendo necessário apenas uma adaptação às características do ambiente que está sendo utilizado. Esta opção, por um lado viabiliza e reduz consideravelmente o ciclo de desenvolvimento de alguns projetos, mas por outro lado limita a flexibilidade do projetista. A limitação decorre do fato de que a maioria destes ambientes exige a programação em uma linguagem específica e o sistema resultante só poderá rodar em plataformas que possuem esta camada adicional de software instalada ou nas quais este ambiente também rode.

O sistema denominado "*Aglets*" da empresa IBM (AGLETS..., 2002) é uma destas camadas de software que foi desenvolvida em *Java* e requer que os agentes também sejam programados nesta linguagem (CHANG, 1996). Situação semelhante ocorre com o Concordia (WONG, 1997) da empresa Mitsubishi Electric. Já o

ambiente Mozart (ROY, 2005) desenvolvido por um consórcio de instituições requer que os agentes sejam programados em uma linguagem própria denominada de Oz.

Na plataforma proposta pela IBM, um agente é instalado em cada máquina sendo a gerência destes, centralizado em um servidor de agentes denominado *Aglet Server*. O *Aglet Server* fornece aos usuários um aplicativo gráfico, denominado Tahiti (Figura 4), que possibilita a criação, transferência e destruição de agentes, além da configuração de privilégios de acesso e segurança no servidor. Em outras palavras: o Tahiti (Figura 4) é uma aplicação construída no nível mais alto da *API Aglet*, que possibilita a manipulação de um agente em todas as etapas de seu ciclo de vida (LANGE, 1998, p.13). Trata-se de um visualizador do contexto de execução dos agentes, ou ainda, uma interface gráfica que gerencia os serviços oferecidos pelo *Aglet Server*.

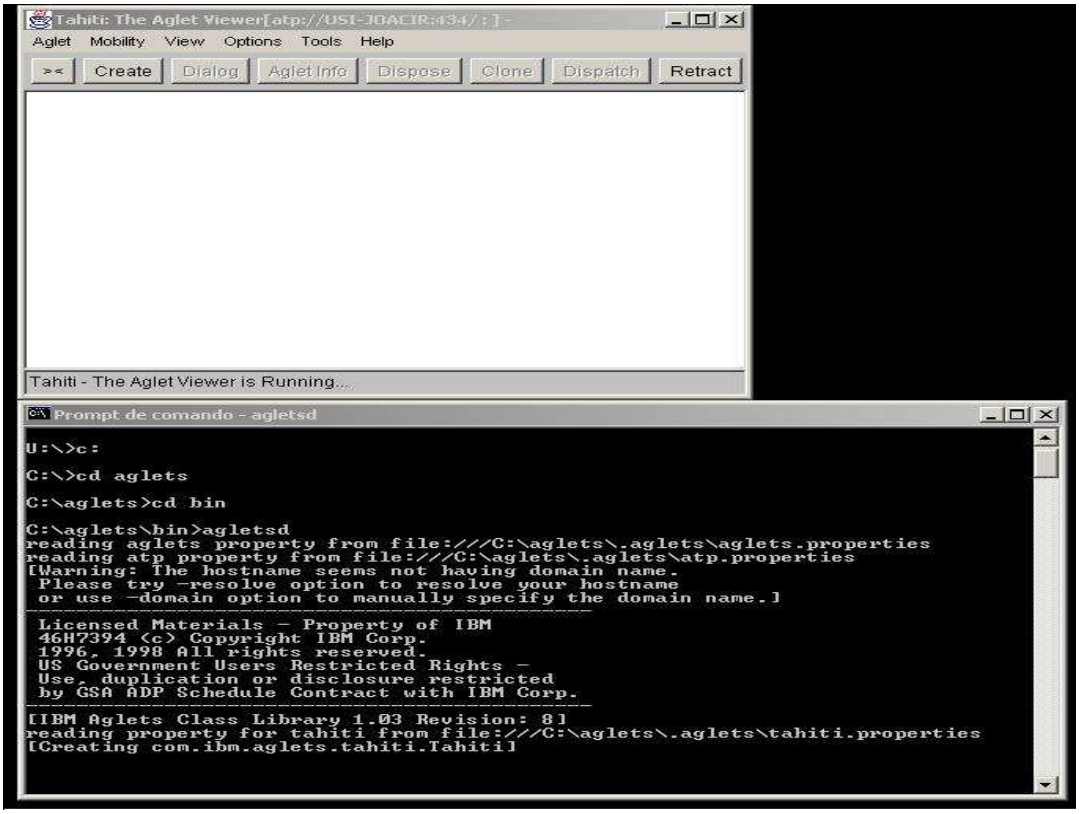

Figura 1 - Servidor Tahiti
Nota: Autoria própria (2007)

Uma das principais funções do *Aglet Server* se refere ao controle de acesso aos recursos dos hospedeiros (hosts) onde os agentes são executados. Em cada hospedeiro, múltiplos usuários podem ter acesso à plataforma e, a cada usuário, podem corresponder várias instâncias do servidor para a execução dos agentes. A

utilização de vários servidores é feita através da configuração de diferentes portas de execução para cada um deles. Em cada servidor podem ser criados múltiplos contextos para a execução dos agentes, embora cada agente possa estar associado apenas a um único contexto em um dado instante.

A Figura 5 ilustra um exemplo de ambiente de execução para agentes, onde existem duas instâncias do Tahiti configuradas nas portas 434 e 12000 para a visualização de contextos de execução fornecidos aos agentes pelos respectivos servidores, em única máquina da rede denominada "*root*".

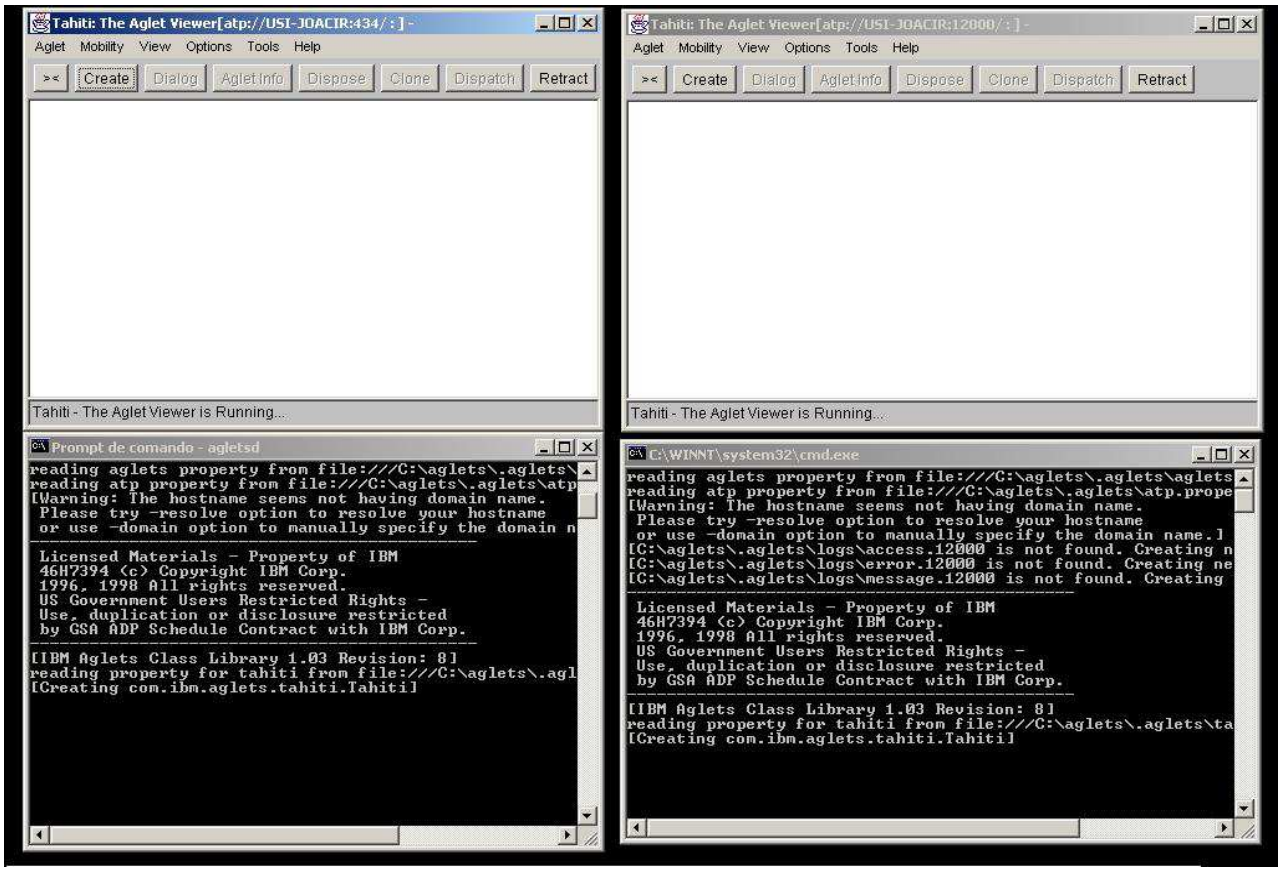

Figura 2 - Instâncias Servidor e Cliente do Thaiti
Fonte: Autoria própria (2007)

Os exemplos citados anteriormente são bastante restritivos, pois fornecem apenas uma linguagem como opção de programação, o mesmo não ocorre com o projeto TACOMA que oferece suporte para seis diferentes linguagens, entre elas o C++ e o Visual Basic e roda nas plataformas Windows 98, NT e Unix (JOHANSEN, 1999).

De qualquer forma se por um lado estes ambientes agilizam parte da

programação, por outro lado será necessário estudá-los e entendê-los e possivelmente aprender a linguagem suportada para se poder fazer uso deles.

2.3 MODELO CONCEITUAL

A plataforma *Aglet* foi a escolhida para a implementação do protótipo desenvolvido no trabalho.

As razões que levaram a tal escolha foram, em resumo, as seguintes:

a) existência de uma biblioteca de classes escritas em JAVA feitas para dar suporte a aplicações com agentes móveis;
b) a plataforma disponibiliza um ambiente gráfico onde é possível a criação, transferência e destruição de agentes, além da configuração de privilégios de acesso e segurança no servidor;
c) documentação disponível para implementação dos recursos disponíveis.

A criação de um aplicativo de agentes móveis é feita através de especializações das classes fornecidas no pacote e na implementação de funções virtuais específicas. A plataforma Aglets pode ser descrita a partir do modelo conceitual de um sistema de agentes Java. Adaptado de Staneva (2002, p.105), este modelo é ilustrado na Figura 3.

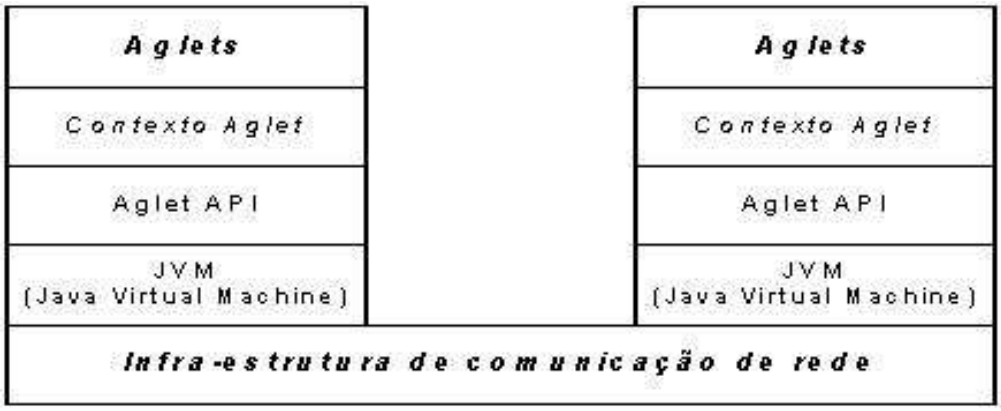

Figura 3 - Modelo Conceitual da Plataforma Aglets
Fonte: Staneva (2002, p.105).

Por este modelo, a plataforma *Aglets* fornece uma API com a funcionalidade básica para os processos de gerenciamento, migração e comunicação de agentes.

O servidor de *Aglets* é multitarefa (*multithread*) e funciona no topo da máquina virtual.

O sistema e suas classes se dividem em 3 grupos principais que são executados sobre um servidor que executa o sistema em cada máquina.

a) **os Agentes**, ou *Aglets*, são uma classe em java com todos os métodos para movimentação, clonagem, persistência e troca de mensagens entre agentes. Um agente específico que desejemos construir deve especializar um *Aglet* e implementar os métodos abstratos de criação, clonagem, mobilidade, tratamento de mensagens, etc.

b) **os *Proxys*** ou *AgletProxy*, são classes de objetos que não são especializados, eles são criados junto com os *Aglets* e funcionam como uma interface de manipulação (*handle)* para os agentes. Todas as ordens ou mensagens a serem dadas ao agente externamente devem ser feitas através do objeto desta classe. Sua finalidade é gerar segurança e transparência. Um *Aglet* pode já ter caminhado para outro nó da rede, contudo pode ser comandado remotamente através de seu *Proxy.*

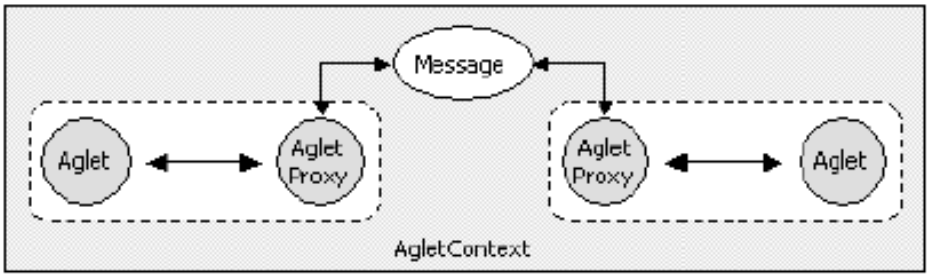

Figura 4 - Contexto Aglet
Fonte: Autoria própria (2007)

c) **os Contextos**, ou *AgletContext* (figura 4), são sub-plataformas que rodam dentro do servidor em cada máquina ou nó da rede. Eles constituem a verdadeira plataforma para cada agente. Eles funcionam como locais abstratos, os quais os agentes realmente enxergam. Uma máquina pode rodar vários contextos, porém para os agentes é indiferente se eles estão rodando em múltiplas máquinas ou em uma máquina só.

O paradigma de programação dos *Aglets* é orientado a objetos e também orientado a eventos. O sistema principal chama automaticamente certos métodos abstratos que são implementados para executar a tarefa a que o agente se destina.

No ciclo de vida (figura 5) podemos ter:

a) **criação** - O método de criação (não o construtor). Implementa as tarefas de inicialização do agente. Só é executado uma vez.

b) **execução** - É o método executado após a criação do agente ou após a migração do agente para outro contexto. Pode ser usado para designar as tarefas que serão feitas após a migração.

c) **adaptadores para captura de eventos -** os agentes possuem ponteiros para objetos que serão usados para captura dos eventos de clonagem, movimento e persistência. Os métodos abstratos destes objetos é que serão invocados nestes respectivos eventos. Eles funcionam como *plug-ins* para os agentes e são acrescentados na medida da necessidade de tratar com as classes de eventos.

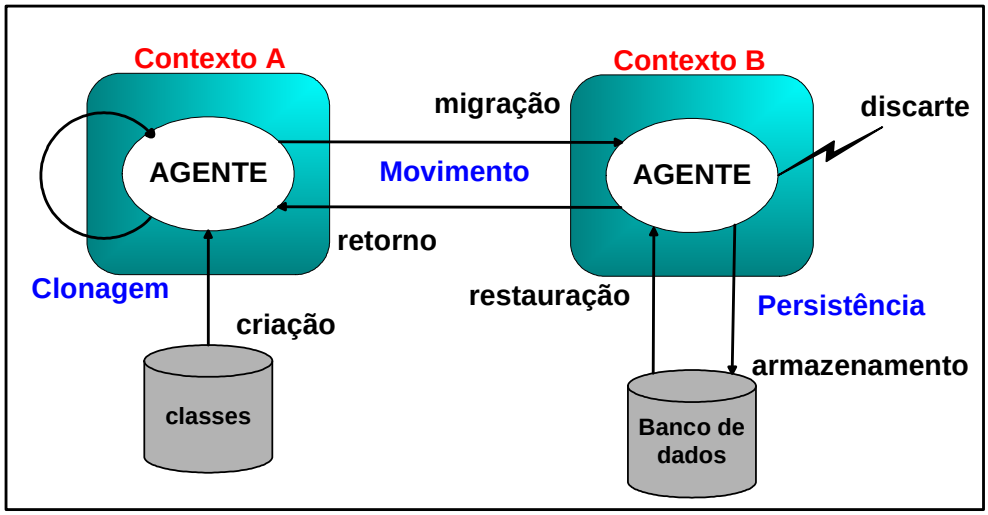

Figura 5 - Modelo do ciclo de vida de um agente
Fonte: Autoria própria (2007)

d) **clonagem** - Os adaptadores de clonagem capturam os eventos que ocorrem na clonagem de um agente*:*

- ***OnCloning*** - É disparado quando alguém requer a clonagem do agente*;*

- ***OnClone*** - É disparado antes do início da clonagem. Usado para inicializar

a clonagem;

- ***OnCloned*** - Chamado após o fim da clonagem.

e) **movimento:**

- ***OnDispatch*** - É chamado antes do inicio de uma migração do agente.
- ***OnArrival*** - É chamado quando o agente chega no seu destino.
- ***OnReverting*** - É chamado quando um contexto requer o retorno no agente.

f) **persistência** - Os agentes podem ser provisoriamente armazenados num banco de dados. Quando o sistema ordena o armazenamento ou a restauração do agente, o dois métodos abaixo são invocados.

g) ***onActivation*** e ***onDeactivating*** - invocado sobre o agente antes e depois de seu estado de execução ser movido entre a memória e um meio de armazenamento permanente, respectivamente.

Um dos mecanismos interessantes e particulares dos agentes é o mecanismo de retração ou retorno. Um contexto (ou base) pode "chamar" um agente de volta bastando para isso conhecer o local onde ele está e o seu identificador. Este mecanismo é bastante útil para o caso, por exemplo, de *firewalls*, onde o agente não pode entrar sozinho no hospedeiro, por isso ele pode marcar um "ponto de encontro" e o hospedeiro chamá-lo de volta.

Outro exemplo é para computação móvel. Onde o dono do agente pode mudar de local e deseja chamar o seu agente para o novo local onde ele estiver.

Os agentes também possuem dois mecanismos flexíveis para tratamento de mensagens. O primeiro deles permite o tratamento de mensagens de resposta de modo assíncrono, ou seja, o programa continua seu fluxo normal mesmo que a resposta das mensagens não tenha voltado. O outro é um mecanismo de *multicast* e subscrição de mensagens. Podemos inscrever agentes numa lista e quando necesitamos, podemos enviar mensagens para todos na lista. É importante se notar que esse mecanismo só funciona dentro de cada contexto.

Também encontra-se formas de busca de agentes usando um tipo de registro que o agente pode fazer a cada nova migração. O problema com esse método é que se houver muitos agentes fazendo registros constantes num servidor, isso pode gerar congestionamentos. Outro problema é que para acharmos o agente, devemos saber seu contexto de origem, pois é lá que ele irá fazer os registros constantes.

Os *Proxies* fornecem segurança para o agente, pelo menos do ponto de vista de implementação, contudo a transparência de localização somente se faz

interessante em alguns casos, pois se tivermos um agente remoto em constante comunicação com o *Proxy* local estaremos voltando ao ponto de comunicação de rede convencional, onde perderemos as vantagens oferecidas pelos sistemas de agentes móveis.

2.4 AGENTES MÓVEIS: CONSIDERAÇÕES ADICIONAIS

Conforme afirmado anteriormente, a utilização de agentes móveis confere às aplicações de sistemas distribuídos algumas vantagens principais, como redução do tráfego de dados, redução da latência de rede, integração de sistemas heterogêneos e, talvez o mais importante benefício, execução autônoma e assíncrona, que possibilita o desenvolvimento de aplicações mais robustas e tolerantes a falhas (ARIDOR, 1998). Para que estas vantagens possam ser verificadas na prática, é necessário que as aplicações de agentes móveis sejam desenvolvidas com base em princípios da engenharia de *software*, assim como qualquer outro sistema computacional. Entretanto, as aplicações de agentes se caracterizam pela natureza experimental de seus projetos (CAZZANIGA, 1998). Grande parte da modelagem de agentes feitas de maneira *ad hoc* resultou em restrições quanto à utilização desta tecnologia em aplicações de porte industrial: duplicação de esforços em projetos, inabilidade para a satisfação de requerimentos de aplicações industriais, dificuldade de identificação de abstrações no nível de agentes individuais, apresentação de pesquisas restritas a objetivos e soluções de problemas (DEUGO, 1999).

Dada a importância atual da tecnologia de agentes para as mais diversas áreas de pesquisa da computação, é fundamental concentrar esforços no sentido de documentar as abstrações referentes a agentes, de forma que pesquisadores possam compartilhar conhecimento a respeito desta tecnologia. A adoção da técnica de padrões de projetos (DEUGO, 1999) oferece uma excelente maneira de documentar as abstrações referentes a agentes em um formato largamente aceito pela comunidade de engenharia de *software*.

O conceito de “padrões de projeto” é atualmente reconhecido como uma das inovações mais significativas do campo da orientação a objetos (ARIDOR, 1998). Através da utilização de padrões, torna-se possível identificar os principais

elementos recorrentes em sistemas de agentes móveis e, ao mesmo tempo, formalizar a experiência das pessoas que projetam tais sistemas (ARIDOR, 1998). Além disso, uma vez que muitas plataformas de agentes foram implementadas utilizando técnicas de orientação a objetos, grande parte dos padrões de projetos que existem também pode ser aplicada a tais sistemas.

Entretanto, deve-se ressaltar o fato de que sistemas de agentes móveis possuem maior capacidade de adaptação ao ambiente do que sistemas puramente orientados a objetos.

No que se refere à plataforma *Aglets*, um número considerável de padrões de projetos recorrentes em aplicações de agentes móveis foi reconhecido já no início de seu desenvolvimento (ARIDOR, 1998). Estes padrões foram implementados em Java e incluídos na primeira versão da plataforma *Aglets Workbench*, possuindo nomes com significado intuitivo, tais como Mestre-escravo (*Master-Slave)*, Mensageiro (*Messenger)*, e Notificador (*Notifier)*. Os primeiros padrões mostraram-se altamente eficientes e capazes de suportar usuários iniciantes com *Aglets* e com o paradigma de agentes móveis (ARIDOR, 1998).

Na implementação de sistemas baseados em agentes móveis, o padrão Mestre-escravo (Master-slave) geralmente é o mais usado, pois, o mesmo é um dos padrões mais simples e fundamentais do projeto de agentes móveis (GRIMSHAW, 2001). Através deste padrão, um agente mestre pode delegar uma tarefa específica a um agente escravo, que é então enviado a um local destino onde pode executar a tarefa que lhe foi atribuída, a fim de, posteriormente, retornar ao local original com os resultados encontrados (ARIDOR, 1998).

Existem pelo menos duas situações onde este padrão pode ser aplicado: quando, por questões de desempenho, é necessário que as tarefas dos agentes mestre e escravo sejam executadas simultaneamente, possivelmente em locais distintos, ou quando um agente estacionário precisa executar em um local remoto uma tarefa que lhe foi solicitada. Neste último caso, um agente mestre estacionário delega a tarefa a um agente escravo que possui a capacidade de migrar para o local onde a tarefa deve ser executada, procedimento este adotado no modelo de validação deste trabalho.

A idéia principal é utilizar as classes abstratas, mestre e escravo, para

especificar os papéis dos agentes no cumprimento de uma tarefa: retornar um agente escravo, ou enviá-lo a um outro local, iniciar a execução da tarefa, tratar exceções que ocorrem durante a execução, etc.

O diagrama da Figura 6 ilustra a colaboração entre as classes do padrão Mestre-Escravo (Master-Slave).

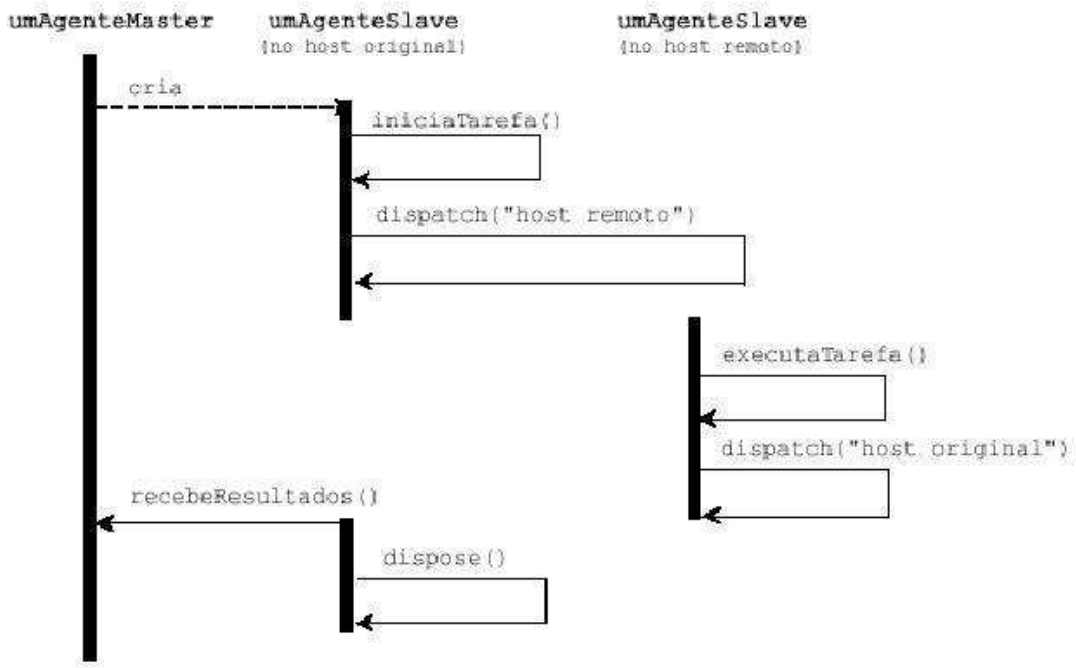

Figura 6 - Padrão de comunicação mestre-escravo
Fonte: Aridor (1998).

3 RECUPERAÇÃO DE DADOS E GESTÃO DO CONHECIMENTO

3.1 O PROCESSO DA RECUPERAÇÃO DE DADOS (DATAMINING)

A humanidade tem, ao longo do tempo, armazenado uma grande quantidade de informações. Nas grandes empresas este fenômeno não é diferente. Era de se esperar que este acúmulo de dados trouxesse às mesmas, uma fonte de informações úteis. Entretanto, o que se constata é que a busca por estas informações necessita de auxilio de técnicas efetivas.

Neste contexto estão situados algoritmos que visam extrair de grandes bases de dados as informações que não podem ser encontradas manualmente (a exemplo do *Knowledge Discovery in Databases (KDD)*, (FAYYAD, 1996). Inserido neste processo de descoberta, temos uma etapa que denominamos *Datamining* (Mineração) onde são aplicadas as técnicas para buscar as informações.

Datamining ou Mineração de Dados é um conjunto de técnicas computacionais, matemáticas e estatísticas para a descoberta de conhecimento novo, não trivial, que possa estar presente em grandes massas de dados, normalmente armazenadas de forma digital em bancos de dados empresariais. A mineração de dados vem ganhando importância cada vez maior em várias disciplinas, auxiliando no processo de extração de relações escondidas em dados experimentais, administrativos, de produção, corporativos e financeiros.

Datamining pode ser utilizado com os seguintes objetivos:

a) explicativo: explicar algum acontecimento ou medida observada, tal como porque é que o consumo de cerveja aumentou na Orla de Salvador;
b) confirmativo: confirmar uma hipótese. Uma companhia de concessão de crédito, por exemplo, pode querer examinar os registros dos seus clientes para determinar se as microempresas do setor de imóveis têm mais probabilidade de adquirir um financiamento do que microempresas do setor agrário;
c) exploratório: analisar os dados procurando novos padrões, aquisições, agrupamentos e regras de comportamento dos dados. Uma companhia de cartões de crédito pode analisar seus registros históricos para determinar que fatores estão associados a pessoas que representam risco ao crédito.

Devido ao alto custo envolvido, estas ferramentas eram unicamente utilizadas, até o momento, por grandes corporações e instituições governamentais. A maior parte das atividades de Datamining ficava restrita a especialistas, empresas que ofereciam os seus serviços de análise, mas sem entregar aos clientes os seus métodos e ferramentas.

Com o grande aumento do volume de dados nas empresas e com o crescimento do uso de tecnologia de bases de dados, especialmente de Datawarehouse, as técnicas de Datamining assumiram papel importante no suporte aos processos de tomada de decisão e devem, aos poucos, ganhar mercado entre as empresas de menor porte.

Várias empresas bem sucedidas estão a optar por *Datamining* no auxílio a tomadas de decisões mais eficientes. A partir do uso de técnicas analíticas exatas, o *Datamining* possibilita a transformação dos dados em informações importantes ao desenvolvimento de estratégias para aumentar a vantagem competitiva de mercado.

Ao dedicar tempo, esforço e recursos na construção de bases de dados, deve-se objetivar obter maior retorno dos investimentos com o uso de Mineração de dados. O KDD faz parte de um ciclo contínuo - um processo que combina dados acumulados sobre os seus clientes com as interações que se realiza com eles. Com o uso da mineração de dados, é possível identificar novas oportunidades num determinado negócio, transformando os dados contidos nas bases de dados em informações importantes para o relacionamento com os clientes. Ao analisar os resultados obtidos através deste processo, pode-se resolver problemas complexos e tomar decisões mais inteligentes e eficazes.

O processo de um KDD (figura 7) pode ser dividido nas seguintes fases:

a) seleção – selecionando ou segmentando os dados de acordo com algum critério, por exemplo, todas as pessoas que possuem um carro, deste modo subconjuntos dos dados podem ser determinados;
b) pré-processamento – Esta é a fase de limpeza dos dados onde certas informações são removidas ou re-configuradas, desde que sejam consideradas desnecessárias ou que se necessite de uma melhor consistência das mesmas. Informações não pertinentes ou mal formatadas podem reduzir a velocidade de "*queries*" (consultas), resultados e

desempenho dos algoritmos de mineração;

Figura 7 - Fases do processo de descoberta do conhecimento
Fonte: Piatetsky-Shapiro (1991).

c) transformação - Os dados não são somente transferidos, mas transformados por normalização, redução ou expansão de escala, amostragem ou combinação para que se tornem mais adequados aos algoritmos de mineração.
d) mineração de dados (*datamining*) - Pode ser definida como a extração de padrões ou modelos de dados observados. A mineração de dados combina métodos e ferramentas das seguintes áreas: aprendizagem de máquina, estatística, banco de dados, sistemas especialistas e visualização de dados;
e) interpretação e avaliação - Os padrões identificados pelo sistema são interpretados e transformados em conhecimento que pode então ser usado para apoiar decisões humanas como, por exemplo, classificação de tarefas, resumo de conteúdos de uma base de dados ou explicação fenômenos observados.

Os principais motivos que tem levado as empresas a investir nessa tecnologia tem sido a obtenção de uma melhor visão sobre a extensão base de dados e a revelação de relações implícitas e padrões entre os dados que nem sempre são visíveis através da simples observação. Há três razões principais para se

desenvolver um projeto mineração de dados:

a) melhor apresentação e utilização dados: As empresas necessitam dar significado a uma quantidade cada vez maior de informações nas suas Bases de Dados. Antes de realizar qualquer análise, o objetivo é qualificar e armazenar os dados a serem trabalhados e encontrar novas formas de apresentá-los de forma mais natural e transparente para os seus usuários;
b) descoberta de novos conhecimentos: A maior parte das aplicações atuais de DM enquadram-se nesta tecnologia cujo objetivo é explicitar relacionamentos ocultos, padrões e correlações entre os diferentes dados existentes na BD da empresa;
c) qualidade dos dados: muitas vezes através da mineração de dados, as empresas descobrem que os seus dados são incompletos, errados ou contraditórios. É, então necessário obter dados cada vez mais consistentes para processamento e análise futura.

Como principais técnicas usadas para a implementação do *Dataming*, podemos citar: técnicas estatísticas; visualização; árvores de decisão; regras de associação; algoritmos genéricos, redes neurais, técnicas de aglomeração, entre outras.

3.2 IMPORTÂNCIA DO CONHECIMENTO

O conhecimento foi, é e sempre será um trunfo competitivo de grande importância, sendo considerado atualmente mais precioso do que os recursos naturais, meios de produção ou até mesmo recursos financeiros (DAVENPORT, 2001).

Há, porém, que se atentar para a importância não só da sua aquisição, mas também para a necessidade da sua criação e transferência, lembrando que conhecimento sem ação é nulo, ou melhor, nem pode ser considerado como tal. O conhecimento tornou-se um dos recursos fundamentais para as organizações, desde o momento em que ocorreu a troca de uma economia industrial baseada em linha de montagem e controle hierárquico para uma economia global, descentralizada. Devido a esta economia global, as organizações agora trabalham, competem e

cooperam em escala mundial.

A constatação da importância do conhecimento para a sobrevivência e prosperidade organizacional tem gerado nas empresas a preocupação de gerenciar esse precioso recurso de um modo mais adequado. A Gestão do Conhecimento é uma disciplina que se propõe a oferecer instrumentos que auxiliem as empresas a transformar o conhecimento em uma fonte de vantagem competitiva (HEMEL, 1998).

Stewart (1998), Sveiby (1998), Davenport e Prusak (1998), entendem que o conhecimento da organização também chamado de capital intelectual, competência, habilidade e inteligência empresarial é reconhecido como um ativo intangível de inestimável valor.

> Conhecimento é uma mistura fluida de experiência condensada, valores, informação contextual e insight experimentado, a qual proporciona uma estrutura para a avaliação e incorporação de novas experiências e informações. Ele tem origem e é aplicado na mente dos conhecedores. Nas organizações, ele costuma estar embutido não só em documentos ou repositórios, mas também em rotinas, processos, práticas e normas organizacionais. (DAVENPORT 1998, p.7).

O conhecimento tem um sentido duplo, estando associado a um corpo de informações que se constitui de fatos, opiniões, idéias, teorias, princípios e modelos e, por outro lado, podendo também se referir à situação ou estado de uma pessoa em relação àquele conjunto de informações. Este estado pode ser de ignorância, consciência, familiaridade, entendimento ou habilidade (BARCLAY; MURRAY, 1997).

Conhecimento é um componente importante das organizações e tem trazido uma nova percepção, uma nova consciência de que o conhecimento organizacional é algo ligado às pessoas, regras e fórmulas que entram de diferentes formas e dinamicamente se transformam em outras (DRUCKER, 1994; NONAKA; TAKEUCHI, 1997; SVEIBY, 1998; TOFFLER, 1994).

De acordo com a visão de (TURTHIL, 1990 apud REZENDE, 2002), conhecimento, ou sua aquisição, é um processo interno de compreensão das informações recebidas, que ocorre de forma diferente em cada indivíduo devido à existência de modelos mentais individuais que podem resultar em ações e decisões completamente diferentes como resultado de um mesmo conjunto de dados. Tofller (1990) afirma que o conhecimento corresponde à fonte de poder de mais alta qualidade e a chave para a futura mudança de poder. Sveiby (1998) coloca que a

economia da era do conhecimento oferece recursos ilimitados porque a capacidade humana de gerar conhecimentos é infinita. Ao contrário dos recursos físicos, o conhecimento cresce quando é compartilhado. A ênfase dada ao conhecimento como recurso econômico é tratado por Toffler (1996, p. 74), ao afirmar que:

> O conhecimento passou de auxiliar do poder monetário e dá força física á sua própria essência e é por isso que a batalha pelo controle do conhecimento e pelos meios de comunicação, está se acirrando pelo mundo inteiro.[...] o conhecimento é substituto definitivo de outros recursos.

Davenport e Prusak (1998) afirmam que neste novo contexto de negócios as organizações reconhecem que o conhecimento é a única fonte capaz de gerar uma vantagem competitiva sustentável. Os autores constatam que atividades baseadas no conhecimento, como o desenvolvimento de novos processos e produtos, estão se tornando primordiais para as empresas. As corporações estão se diferenciando umas das outras pelo que sabem.

Para a criação de conhecimento contínuo, Senge (2000), apresentou perspectivas sobre a necessidade de se desenvolver organizações que aprendem, referindo-se ao capital intelectual, à gestão do conhecimento e a organizações de aprendizagem. Introduz um novo padrão de comportamento e uma nova maneira de olhar a organização, seus integrantes e demais partes interessadas em seu sucesso.

Nesse sentido, Chiavenato (1999, p. 457), observa que: “A competitividade imposta requer organizações de aprendizagem capazes de gerenciar a mudança a seu favor. A intensidade com que as pessoas e as organizações aprendem vai tornar-se fundamental para a criação desta vantagem competitiva”.

Segundo Drucker (1995), o conhecimento era para os antigos freqüentemente percebido como algo que servia ao engrandecimento do ser, do intelecto e do espírito. Sob a concepção de Platão, filósofo da Grécia Antiga. ″O conhecimento é a crença verdadeiramente justificada". (NONAKA; TAKEUCHI, 1997, p. 24).

O Papa João Paulo II (*apud* STWEART, 1998, p.11) reconheceu a importância do conhecimento, escrevendo em sua encíclica Centesimus Annus (1991): "Se antes a terra, e depois o capital, eram os fatores decisivos da produção [...] hoje, o fator decisivo é cada vez mais o homem em si, ou seja, seu conhecimento".

Apesar da percepção geral, que coloca o conhecimento nas organizações

relacionado a ativos intangíveis, o termo que vem sendo utilizado por autores como Edvinsson (1998), Davenport (1998), Stewart (1998) e Sveiby (1998) para abranger tais ativos ou mesmo caracterizar e discutir a questão, é o conceito de capital intelectual o qual requer um aprofundamento.

Dentre as variadas correntes de estudos que existem sobre o tema do conhecimento nas organizações, há uma que o enfoca enquanto capital intelectual. O principal argumento desta corrente pode ser resumido através de uma observação relativamente simples, que é a de que existem ativos, chamados genericamente de intangíveis (SVEIBY, 1998; EDVINSSON, 1998), que proporcionam o desenvolvimento e valorização da organização, e que não são aqueles necessariamente presentes no patrimônio físico da empresa.

3.3 A FERRAMENTA EXPERTEXT

Nas Organizações modernas, a produção de documentos impressos aumenta na medida que mais sistemas são incorporados ao cotidiano destas, com o processamento eletrônico de documentos, tanto na captura e arquivamento quanto na recuperação, significará uma forte redução de custos no processo de manipulação deste documento em comparação com os procedimentos tradicionais, resultando em uma forte redução de custos muitas vezes difíceis de mensurar de forma direta e pontual.

Em todas as áreas do conhecimento, observa-se que os esforços em agregar potenciais facilitam de certa forma a execução de tarefas, desde que, estes sejam combinados e organizados. Aplica-se também, esta regra, às atividades relacionadas ao aprendizado de algoritmos baseados em inteligência artificial, com o intuito de classificação de conteúdo.

De forma semelhante ao processo ocorrido com os seres humanos, a *expertise* adquirida por um programa em particular, inserida no paradigma de inteligência artificial, pode ser distribuída, manipulada e mesclada a outras, a fim de compor bases sólidas de qualidade e menos tendenciosas a erros.

Segundo Oliveira (2004, p.45):

> a ferramenta Expertext, fornece uma forma de acompanhar todo o processo de combinação de *expertise* de classificadores de textos, baseados no teorema de Bayes, desde a seleção de conjuntos de documentos para treino e testes, passando pelo treinamento específico de cada algoritmo e sua posterior combinação.

A ferramenta *Expertext*, apresenta como funcionalidade, a análise de textos através da sua aplicabilidade da combinação de classificadores e a precisão dentro de um processo de categorização de documentos. O trabalho de Oliveira (2004) teve como hipótese norteadora, a metodologia de combinação do aprendizado de vários classificadores de textos, construídos seguindo o paradigma de aprendizado de máquina e teorema de *Bayes*, em uma base única de conhecimento. Oliveira (2004) explorou a aplicabilidade da combinação de classificadores a fim de obter maior precisão dentro de um processo de categorização de documentos

Segundo Oliveira (2004), o processo de classificação de documentos pode ser melhorado quando se combinam classificadores. Uma estratégia comumente utilizada é a combinação de hipóteses de classificação, que, entretanto, traz alguns efeitos colaterais indesejados, tais como o consumo de tempo e memória. A operação do conjunto de classificadores, nesta abordagem, exige uma avaliação individual de hipóteses de cada classificador do conjunto, que pode exigir um esforço computacional pequeno, quando avaliado isoladamente, mas substancial quando visto como empenho conjugado.

A ferramenta *Expertext* realiza uma outra estratégia, caracterizado pela substituição do processo de combinação de hipóteses de um conjunto de especialistas sobre um determinado assunto, pela soma da *expertise* individual de cada um a fim de compor um único especialista. Segundo Oliveira (2004), esta estratégia trouxe algumas peculiaridades interessantes a observar, tais como a possibilidade de incremento paulatino das habilidades de classificação, através da introdução de novos classificadores que podem ser treinados isoladamente, e a implementação de um mecanismo de avaliação de cada *expertise* combinada.

O *Expertext*, apresenta um modelo de implementação definido em três pacotes distintos de classes. O conjunto de apresentação, codificado pelo pacote *Desktop*, é composto pelas classes que compõem a interface homem-máquina do aplicativo. O conjunto mestre, representado pelo pacote *Bayes*, é formado por classes que implementam todo o mecanismo relacionado ao processo de

treinamento, categorização e combinação de *expertise* dos classificadores de textos. Por fim, o conjunto de acesso a dados, inserido no pacote *Database*, é responsável por recuperar e atualizar os dados armazenados.

A arquitetura da ferramenta *Expertext* foi definida levando em consideração uma modularização do aplicativo em quatro partes inter-relacionadas: o módulo de entrada de documentos, o módulo de configuração de pesos e filtros, o módulo de treinamento, classificação e combinação de classificadores e o módulo de visualização de dados. Estes módulos estão associados às seguintes funções básicas da ferramenta.

a) cadastro das referências de documentos com fins de treinamento e teste;
b) criação de uma interface para entrada das opções de filtro e atribuição de pesos;
c) treinamento e classificação de textos juntamente com a combinação de classificadores;
d) representação e visualização de características e resultados do processo de classificação de textos.

A arquitetura do aplicativo é mostrada na Figura 7.

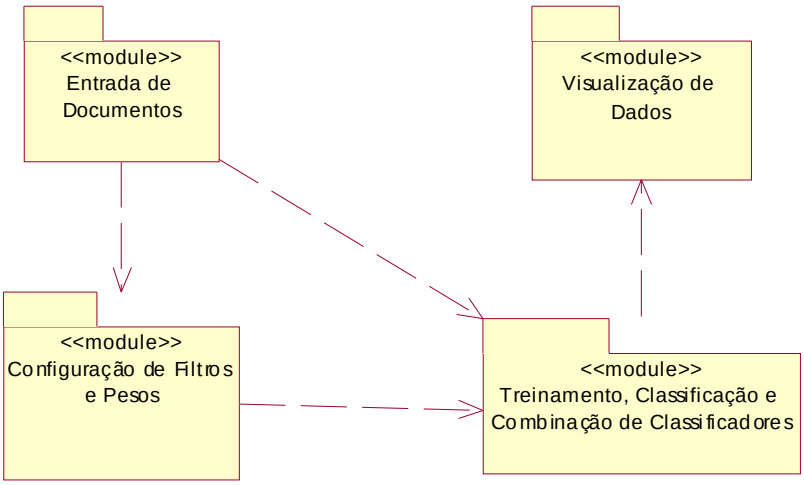

Figura 8 - Arquitetura da ferramenta ExperText
Fonte: Oliveira (2004).

O módulo de entrada de documentos é responsável pelo cadastro das amostras e informações acerca da localização e tipo de categoria rotulada de cada arquivo de texto enquadrado com a finalidade de treino ou testes. O módulo de configuração de filtros e pesos tem a finalidade de fornecer os parâmetros

necessários para a realização do processo de combinação de *expertise* entre os classificadores.

O módulo de treinamento, classificação e combinação trata exclusivamente das funções referentes ao aprendizado de máquina baseado na fórmula *naive bayes*. Neste caso, são utilizados os arquivos cadastrados no módulo de documentos para obtenção de conhecimento em cada classificador, além dos parâmetros utilizados no processo de combinação. Finalmente, o módulo de visualização de dados é responsável pela exibição dos resultados obtidos pelo módulo de treinamento, classificação e combinação. Nesta oportunidade, são empregados gráficos em duas e três dimensões, além do uso de tabelas para ilustrar os dados computados pela ferramenta.

A ferramenta *Expertext* está preparada para trabalhar com vários tipos diferentes de banco de dados. Para que esta facilidade fosse possível, foi implementado um conjunto de classes obedecendo ao padrão de projeto *Abstract Factory* que fornece uma interface para criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas (GAMMA, 2000). O efeito deste padrão de projeto é a possibilidade de uso de diferentes classes especializadas em um tipo de banco de dados específico, quando necessário, através de uma interface única.

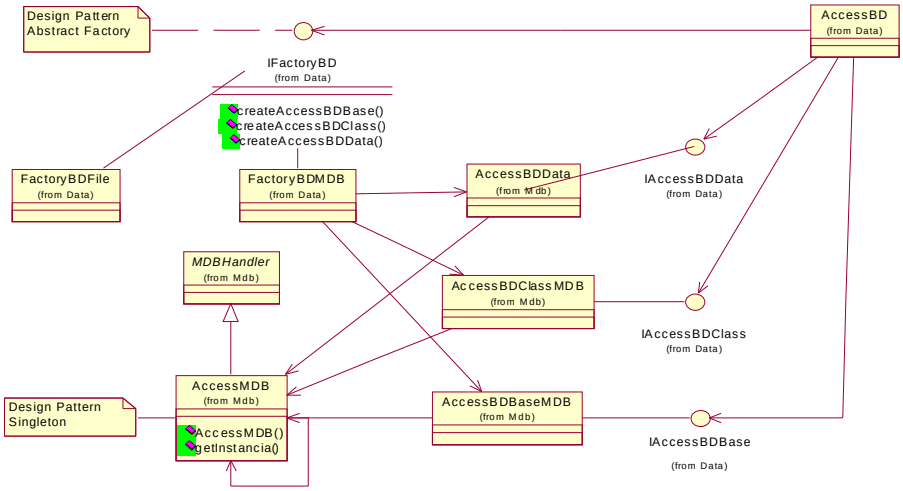

Figura 9 - Abstract Factory utilizado no acesso a diferentes tipos de banco de dados
Fonte: Oliveira (2004).

4 ARQUITETURA DO SISTEMA

O ***Sistema Automatizado de Coleta de Informações (*SACI)** propõe como funcionalidade, através do uso de agentes móveis, recuperar dados de arquivos armazenados em local previamente determinado, transferindo informações sobre os mesmos, para uma base de dados em um servidor central. Através desta base de dados gerada é possível análises quantitativas, financeiras, de produção individual, de localização geográfica de dados em arquivos ou de arquivos em uma rede, como veremos na seção 4.1 deste trabalho. As análises são efetuadas com o auxílio da ferramenta Expertext (OLIVEIRA, 2004) que irá gerar então, um conjunto de relatórios pré-definidos ou configurados, necessários à tomada de decisão envolvendo as informações coletadas inicialmente.

A principal justificativa para a escolha de uma arquitetura de agentes móveis para implementação do sistema SACI está ligada ao próprio conceito de agente móvel, que é uma entidade capaz de executar tarefas autônomas. Nessa proposta isso se concretiza visto que o usuário não irá se preocupar em buscar essas informações na própria ou em outras máquinas, já que o agente fará isso por ele. Outra justificativa diz respeito à diminuição de tráfego na rede. Agentes móveis podem compactar a conversação inteira em uma única transmissão de agente, que pode reduzir o tráfego de rede sensivelmente.

A arquitetura do sistema SACI (figura 10) adota um modelo baseado na plataforma *Aglet,* por se apresentar a mais viável para a realização do protótipo experimental, conforme foi descrito na seção 2.2, destacando-se características como: integração com a linguagem de programação Java, interface gráfica para gerência dos serviços oferecidos, documentação técnica disponível e devido também, ao seu intensivo uso no contexto de aplicações de rede.

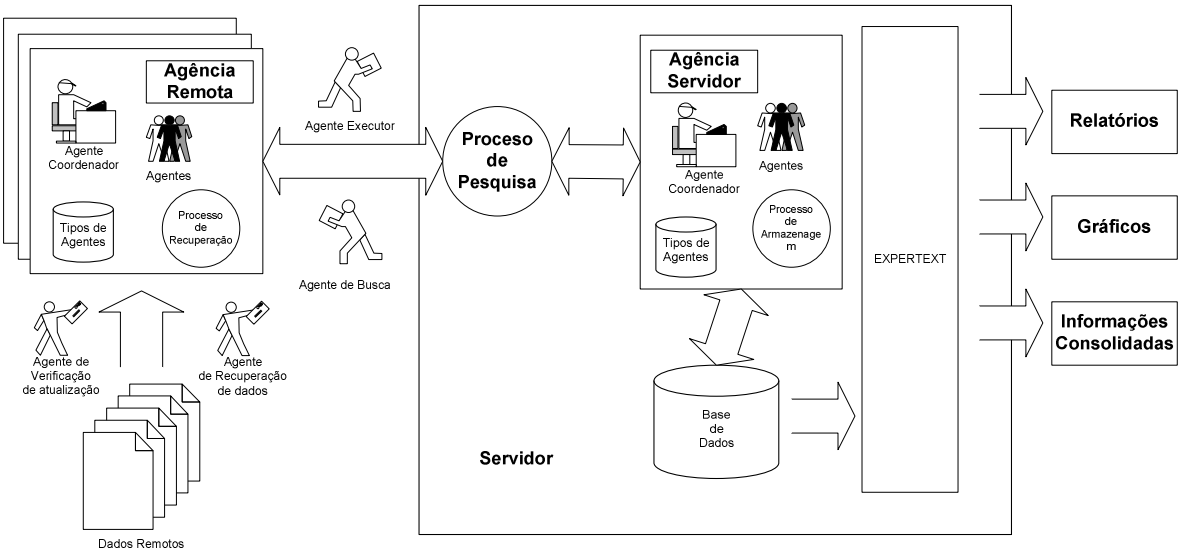

Figura 10 - Arquitetura do Sistema SACI
Fonte: Autoria própria (2007).

O sistema SACI é composto pelos seguintes componentes funcionais:

a) agências
   - Agência servidor
   - Agência remota
b) Agentes
   - Agente coordenador
   - Agente executor
   - Agente de busca
   - Agente de verificação de atualização
   - Agente de recuperação de dados

As Agências, compostas pela Agência Servidor e pela Agência Remota (figura 10), são responsáveis pela criação, transferência e destruição de agentes além da configuração de privilégios de acesso e segurança. A Agência Servidor é instalada e configurada no servidor de informações, de onde todo o processo de busca será controlado. A Agência Remota é instalada e configurada em todas as máquinas clientes pertencentes à rede.

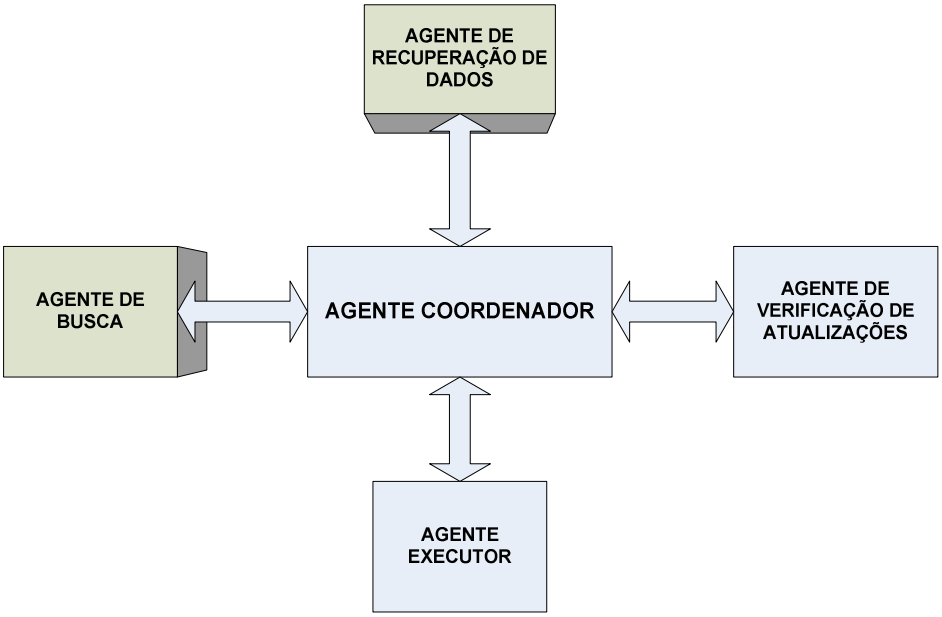

Figura 11 - Arquitetura dos Agentes – Agente Coordenador
Fonte: Autoria própria (2007).

Para a execução das tarefas, teremos os seguintes agentes:

a) o agente coordenador irá controlar toda a ação dos demais agentes na Agência, controlando sua criação, transferência e destruição, servindo também como interface (figura 11) entre os agentes de busca e os agentes de recuperação de dados e os agentes de verificação de atualizações durante os processos de recuperação e pesquisa. Na agência servidor (figura 10), o agente coordenador controla também o armazenamento e leitura dos dados contidos na base de dados do sistema SACI;
b) o agente executor é responsável pela execução efetiva do envio de informação de atualização para a base de dados do sistema SACI, através do já mencionado Agente Coordenador;
c) os agentes de busca são responsáveis pelas pesquisas de informações baseadas em palavra-chave previamente definidas pelo usuário, pesquisas estas que alimentam a base de dados do sistema SACI. A periodicidade destas pesquisas deve ser configurada de acordo com as características de inclusão de novos arquivos de cada cenário estudado;
d) os agentes de verificação de atualizações de dados são responsáveis pela informação de atualização cadastral dos dados armazenados;
e) o agente de recuperação de dados alimenta a base de dados do sistema

SACI, localizada no servidor, através de incursões efetuadas nas estações remotas, em pastas de arquivos previamente determinadas. Estes agentes estarão atrelados as Agências, que irão conter os métodos necessários à sua operação.

A interação entre os componentes da arquitetura do sistema SACI que resulta na sua funcionalidade é baseada na comunicação não cooperativa, ou seja, os agentes não trocam informações entre si. Toda e qualquer comunicação é controlada pela agência, servidor ou remota, tendo sua gerência sendo efetuada pelo agente coordenador. A interação com a ferramenta *Expertext* é efetuada através do conjunto de arquivos textos, com informações colhidas com base em palavras-chaves pré-configuradas (como veremos na seção 5.2), que servirão como base de informações da ferramenta *Expertext*.

4.1 FUNCIONALIDADES DO SISTEMA SACI

O sistema SACI, através de sua funcionalidade de pesquisa seletiva e com o auxílio da ferramenta *Expertext (*OLIVEIRA, 2004), através da sua aplicabilidade da combinação de classificadores e a precisão dentro de um processo de categorização de documentos (visto na seção 3.3), possibilita análises tais como:

a) análises quantitativas, financeiras ou classificatórias através da classificação de termos encontrados em arquivos pertinentes, por valor ou por palavra-chave;
b) análises de produção individual de informações por pessoa física através da pesquisa e classificação de arquivos por data de criação e/ou alteração dos mesmos;
c) análises de ocupação física de arquivos através da pesquisa e classificação de características estruturais tais como tipo, tamanho e localização.;
d) Análises geográficas, percentuais e/ou numéricas, baseadas na localização dos arquivos em uma rede.

5 IMPLEMENTAÇÃO DO PROTÓTIPO

Neste capítulo é descrita a implementação da solução proposta para atingir o objetivo principal do trabalho.

5.1 CENÁRIO

Conforme descrito na seção 4.1, o sistema SACI pode ter uma funcionalidade diversificada que tem como base o potencial da ferramenta Expertext. Visando a validação efetiva da proposta de arquitetura do sistema apresentado, faz-se necessário focar a funcionalidade do protótipo desenvolvido num cenário de aplicação específico (figura 12).

Figura 12 - Sistema SACI – Funcionalidade do Protótipo. Processo de Financiamento
Fonte: Autoria própria (2007).

Do universo de informações corporativas, foi escolhido como cenário de implantação, o processo de aprovação de projetos financiados por uma agência de fomento governamental (figura 12). Através da análise de informações contidas em arquivos espalhados por toda a instituição, arquivos estes que armazenam informações de propostas de projetos financeiros que estejam em qualquer uma das fases do processo (cadastro, análise, negociação ou em andamento), será possível a geração de relatórios que indiquem exatamente a situação atual da Organização em relação aos cumprimentos das metas por ela instituídas.

O sistema SACI será configurado para recuperar informações pertinentes ao protótipo e para gerar relatórios através das funcionalidades da ferramenta Expertext

que venham a dar subsídios na análise pretendida.

O cenário em que o sistema SACI foi experimentado consistiu de uma rede (figura 13), onde foram usados como base três computadores, Servidor_1, Estação 1 e Estação 2, simulando 6 usuários em rede.

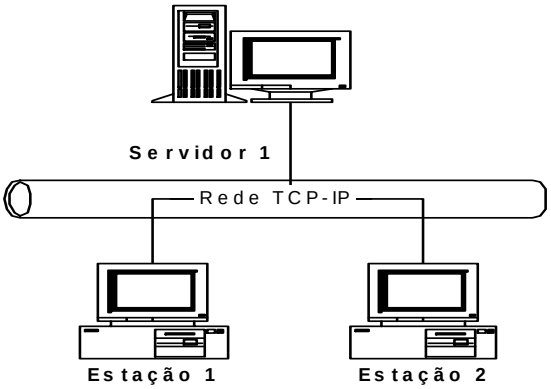

Figura 13 - Rede (física) de Implementação do Protótipo
Fonte: Autoria própria (2007).

O primeiro destes simulou a máquina Servidor e os outros dois simularam máquinas dos usuários virtuais. Como para os dois últimos, foram utilizados, simultaneamente, três instâncias do cliente de agentes em cada máquina, pode-se concluir facilmente que estes simularam seis usuários virtuais. Estes computadores estavam equipados com os sistemas operacionais Windows 2000 (SP4), bem como versões diferentes do Java Development Kit, JDK 1.3 (Servidor) e 1.4 (Estações). Em todas as máquinas o ASDK 2.0 do SourceForge foi utilizado e na rede de comunicação foi utilizado o protocolo TCP/IP com configurações básicas.

Desta forma, o Servidor_1 serviu como ponto central de gerenciamento simulando a base de informações, onde foi instalado a instância Servidor do Sistema SACI, bem como a ferramenta Expertext. Nesta máquina também foi instalado todo o ambiente de desenvolvimento e o ambiente de geração de relatórios técnicos e gerenciais (planilhas, textos, etc). Nas outras duas máquinas (Estação 1 e 2) foram instalados as instâncias clientes do sistema SACI servindo como base de testes para o protótipo.

No que se refere aos bancos de dados implantados no modelo de validação, apenas a máquina Servidor_1 possuiu o banco de dados *db_servidor*, com informações sobre os projetos disponíveis para análises. O banco de dados *db_agencia*, com informações sobre os usuários contidos no itinerário do agente de

busca, foi criado somente nas máquinas clientes, que simularam os computadores dos usuários. Nas três máquinas foram criados ainda, o banco de dados db_agentes contendo informações sobre os agentes disponíveis no sistema.

O modelo de dados do sistema SACI descrevendo a estrutura da base de dados e os seus relacionamentos, é mostrado na figura 14.

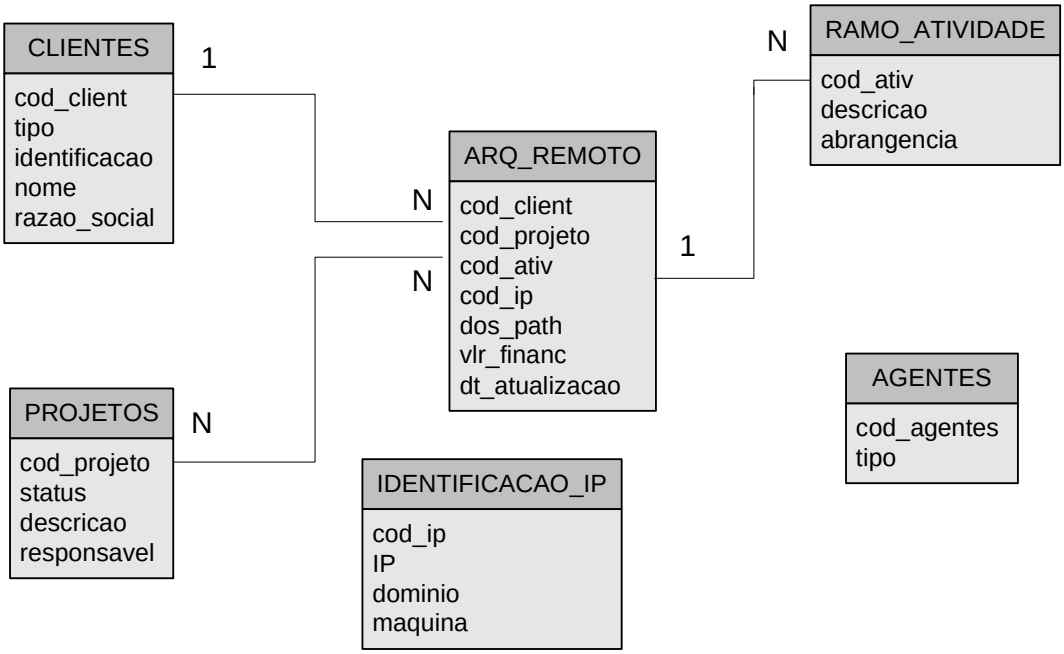

Figura 14 - Modelo de dados utilizado pelo sistema SACI
Fonte: Autoria própria (2007).

A tabela CLIENTES contém todos os dados de identificação dos clientes atendidos pela Organização nos diversos projetos identificados na tabela PROJETOS. As informações dos clientes e dos projetos associados serão usadas tanto para identificação e complementação dos arquivos texto a serem analisados pela ferramenta Expertext quanto para a localização cruzada (cliente por projetos, por exemplo) efetuada pelos agentes. A tabela RAMO_ATIVIDADE contém as informações dos vários ramos de atividades apoiados pela instituição. É importante salientar, que a busca pelos agentes pode ser efetuada nas 3 dimensões: clientes, projetos e ramo de atividade.

A tabela ARQ_REMOTO contém todas as informações dos arquivos textos pesquisados na Organização, informações estas recuperadas pelos agentes. Com base em informações contidas nesta tabela, como por exemplo, valor financiado (vlr_financ), é que será possível uma análise de acompanhamento econômico.

A tabela AGENTES contém o código e os respectivos tipos de agentes, tais como agente coordenador, agente executor, etc. Esta identificação é necessária quando da criação do agente e das suas particularidades associadas como nível de segurança por exemplo.

A tabela IDENTIFICAÇÃO_IP contém todas as informações de rede de uma determinada máquina, necessárias ao sistema SACI. Estas informações serão úteis na identificação e localização de todos os equipamentos envolvidos com o sistema.

Na máquina Servidor_1, também foi instalada a ferramenta *Expertext* que apresenta o modelo de dados da figura 15, modelo este descrito no capítulo 3, seção 3.3. O processo de cadastro de documentos na ferramenta é iniciado apontando-se para um diretório de arquivos específicos, criados previamente conforme a classificação dos projetos em estudo. Os rótulos de classe em cada arquivo de texto são criados então, utilizando-se o nome do diretório onde o documento se encontra. Assim, por exemplo, se um documento estiver no diretório *c:\projetos\financeiros\microcredito*, ele receberá o rótulo da classe *microcredito* como categoria em que está enquadrado.

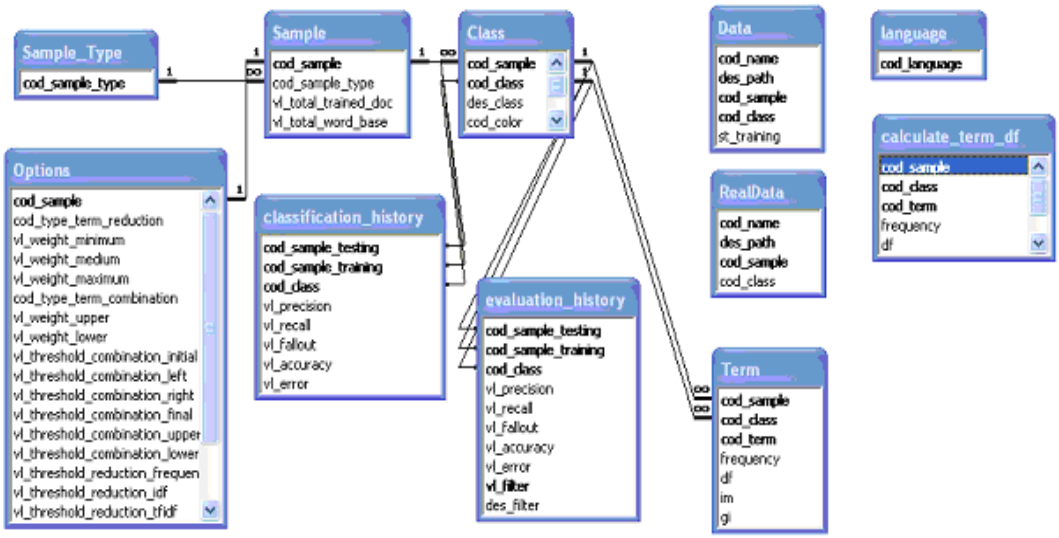

Figura 15 - Modelo de Dados utilizados pela ferramenta Expertext
Fonte: Oliveira (2004).

Utilizando-se a funcionalidade de otimização da ferramenta Expertext (capítulo 3, seção 3.3), a carga de vários documentos localizados em diretórios distintos pode ter a sua a transferência de referências feita apontando-se para um diretório maior

que englobe os subdiretórios em que estes se encontram. A própria ferramenta se encarregará de capturar o nome dos últimos diretórios encontrados e rotular os documentos como sendo de classes homônimas a estes diretórios.

5.2 OPERAÇÃO DOS AGENTES

A forma de operação dos agentes, segundo definições já mencionadas anteriormente na seção 2.2, caracteriza-se como forma de ataque em bandos, forma de movimentação em trilha e forma de cooperação não cooperativa.

O processo de busca e recuperação de informações contidas nos arquivos escolhidos para pesquisa no modelo de validação representa a funcionalidade geral do sistema SACI. Uma vez disparado, segue o fluxo mostrado na figura 16.

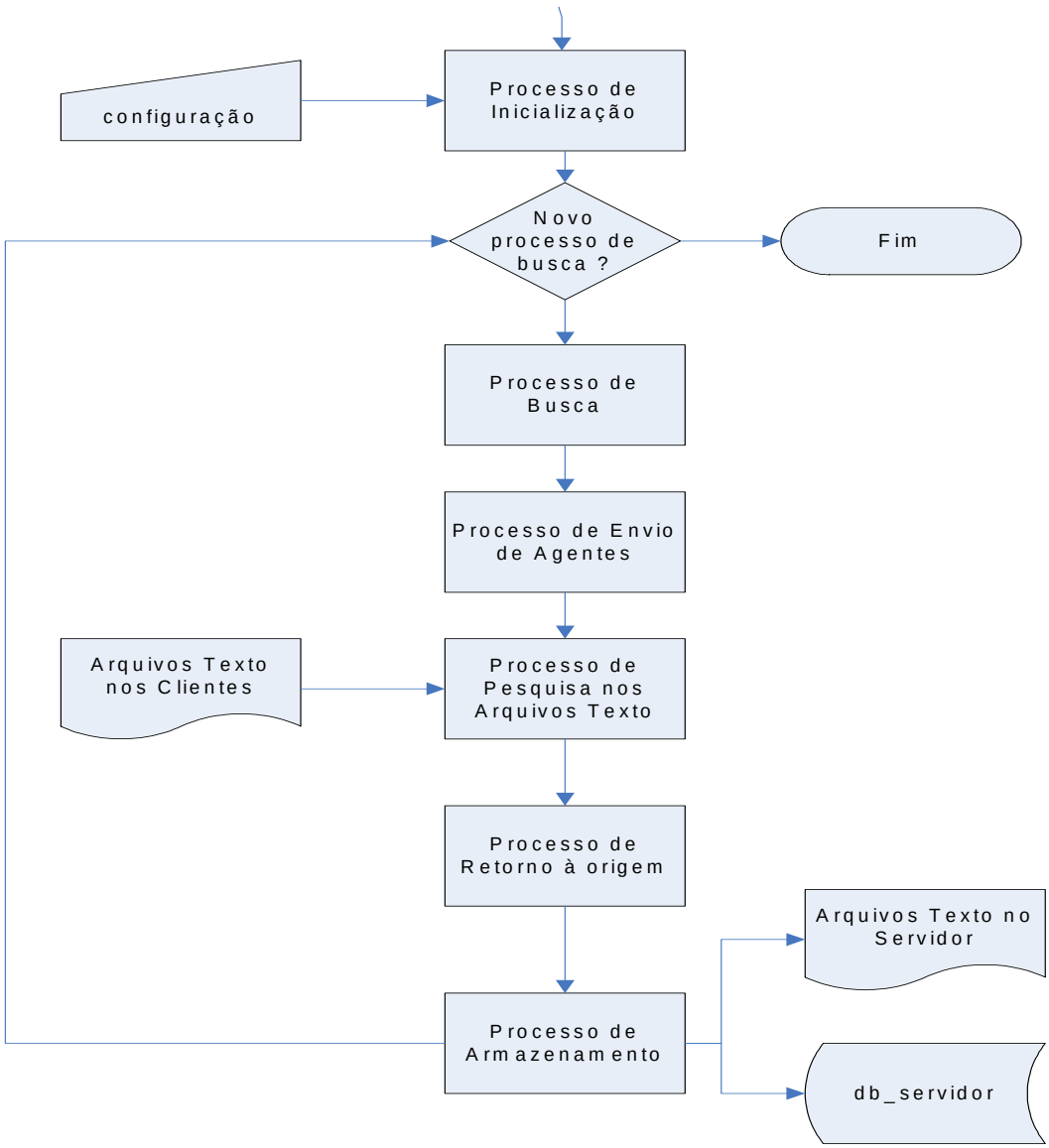

Figura 16 - Fluxo de Recuperação e Busca - Sistema SACI
Fonte: Autoria própria (2007).

Como processo de inicialização, a base de dados do sistema é alimentada com todos os dados de identificação dos arquivos alvos, arquivos estes que foram previamente colocados nas pastas base, também previamente criadas, nas estações clientes. As informações colhidas, conforme foi visto no modelo de dados (figura 14) são informações referentes ao conteúdo e a aspectos de localização e identificação dos arquivos envolvidos nos processo.

Após a fase de inicialização da base de dados, e uma vez cadastrada as palavras chaves que servirão como parâmetros de pesquisa, é efetuada uma busca

em todos os arquivos-texto que contenham estas palavras-chave, como projeto, ramo de trabalho, valor do projeto, etc.

O processo de busca dispara então, o processo de envio dos agentes, agentes estes que são definidos na fase de configuração do sistema e que uma vez disparados seguem um intinerário criado a partir de dados coletados e armazenados durante o processo de inicialização.

Uma vez na estação cliente e, através da agência instalada no mesmo, é disparado o processo de pesquisa nos arquivos textos disponíveis. Todo este processo é gerenciado pelo agente coordenador.

Após a análise do(s) arquivo(s) selecionado(s), o agente coordenador irá gerenciar o retorno do agente de pesquisa com as informações coletadas, informações estas que podem ser algum dado novo ou apenas uma negativa informando a ausência de informação.

Uma vez na agência servidor, é disparado o processo de armazenamento que tem como função: atualizar os dados no banco de dados db_servidor bem como, nos arquivos textos, com informações colhidas com base nas palavras-chaves pré-configuradas, que servirão como base de informações da ferramenta Expertext.

No processo de retorno à origem, cada agente também levará informações sobre data de atualização dos arquivos pesquisados. Essas informações serão armazenadas em uma base de dados no servidor de dados e serão úteis nos próximos passos. A intenção é executar esse processo com uma periodicidade média, pois informações sobre projetos financeiros não são atualizadas com freqüência.

No intuito de facilitar a pesquisa nas execuções seguintes do processo de envio dos agentes, a investigação não será realizada em todos os documentos da estação, ou seja, cada agente levará consigo as informações sobre os documentos encontrados na pesquisa anterior em determinada estação de trabalho.

Por exemplo, ao pesquisar novamente a estação 3, o agente levará consigo as informações da pesquisa anterior na estação 3; isso fará com que o agente pesquise somente o(s) documentos(s) específico(s) e só atualize as informações quando houver diferenças na data de atualização do arquivo. No entanto, outros documentos podem ser criados nessa estação. Por causa disso, existe também um

algoritmo de pesquisa programado para executar a pesquisa em toda a base de documentos também periodicamente (vide figura 17 abaixo).

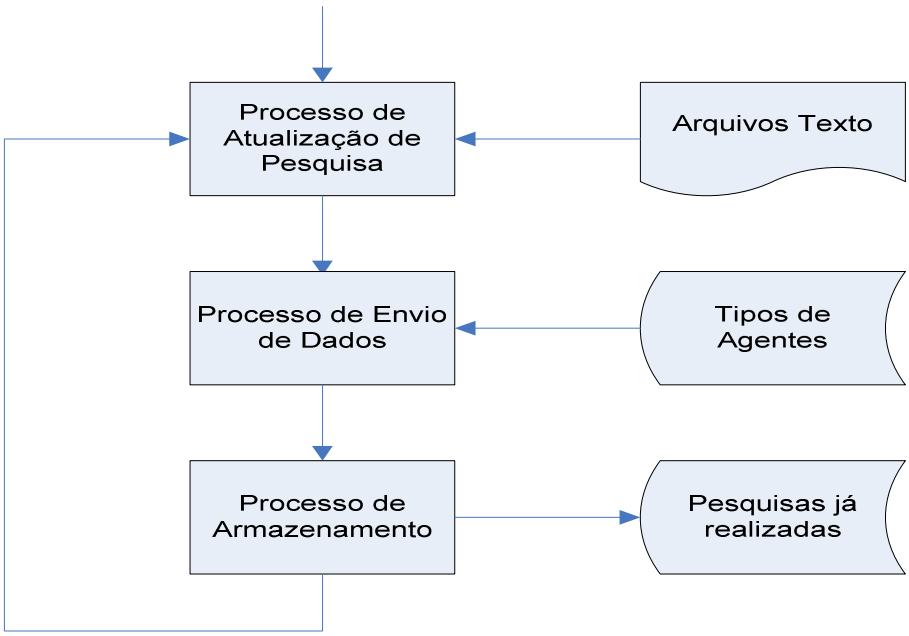

Figura 17 - Fluxo de Atualização de Dados Pesquisados – Sistema SACI
Fonte: Autoria própria (2007).

5.3 OPÇÕES NO USO DA FERRAMENTA

A ferramenta Expertext foi construída com o objetivo de fornecer uma forma de acompanhar todo o processo de combinação de *expertise* de classificadores de textos, baseados no teorema de Bayes, desde a seleção de conjuntos de documentos para treino e testes, passando pelo treinamento específico de cada algoritmo e sua posterior combinação (OLIVEIRA, 2004).

Na customização adotada para a implantação do sistema SACI neste trabalho, a ferramenta *Expertext* foi utilizada para o desenvolvimento de heurísticas aplicadas à classificação e filtragem de projetos financeiros de uma instituição, fruto do conhecimento obtido com o uso da ferramenta. Neste tipo de abordagem, o processo de classificação e seleção de cada projeto analisado depende do aprendizado e prévio conhecimento do conteúdo dos exemplos fornecidos por seu usuário, podendo ser muitas vezes de longa duração e inexato. Com as técnicas e simulações possíveis de serem realizadas com o Expertext, é possível encurtar ou eliminar o tempo de aprendizado, ao fornecer de antemão para o usuário, uma base

de dados contendo uma possível experiência otimizada na filtragem de documentos relativos aos projetos em estudo. Neste caso, a base de dados recebida é fruto de um processo de combinação de várias outras bases, cujos mentores são usuários que desprenderam tempo e dedicação na transferência de seu conhecimento tácito do tipo de informação desejada para o algoritmo classificador.

A principal utilização do Expertext nos testes realizados é o auxilio da mesma no processo de análise das amostras, características coletadas e resultados obtidos durante as fases de aprendizado e classificação de documentos relativos aos projetos financeiros a serem analisados.

5.4 COMO FUNCIONA O SISTEMA

A pesquisa e recuperação de dados é feita através de um processo de extração de informação válida, previamente desconhecida e de máxima abrangência a partir de bases de dados de arquivos textuais. O uso de agente móvel, através da sua capacidade de seleção e mobilidade em um ambiente distribuído, fornece à solução proposta subsídios na resposta ao objetivo do protótipo.

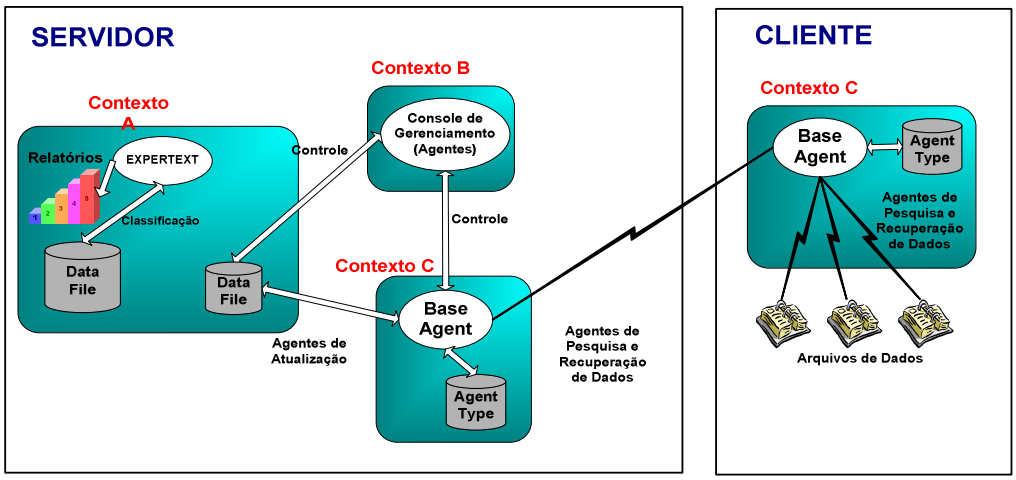

Figura 18 - Funcionamento Geral do Sistema
Fonte: Autoria própria (2007).

Como podemos observar através da figura 18, os agentes coletam informações dos arquivos da rede, conforme características previamente determinadas e geram arquivos imagens na base de diretórios, sendo então estes

arquivos analisados pela ferramenta Expertext. Estes arquivos imagens, não são uma cópia dos arquivos originais e sim, um arquivo contendo todas as informações relevantes para o projeto, sobre os arquivos originais. Através da ferramenta *Expertext., é* gerado então relatórios pré-definidos que servirão como base para a elaboração de relatórios gerenciais que darão auxilio a tomadas de decisão Organizacionais, podendo ser solicitado por um determinado departamento ou pessoa dentro da Organização.

Os documentos da base textual do servidor com informações sobre projetos financeiros apoiados pela Organização poderão auxiliar no processo de gestão financeira, na medida em que ali serão encontradas informações que caracterizam o perfil dos projetos aprovados e em via de aprovação, permitindo uma tomada de decisão para um melhor gerenciamento financeiro e conseqüentemente do conhecimento gerado na Organização.

O processo de operação no uso da ferramenta basicamente seguirá os passos descritos abaixo, que são visualizados na figura 19:

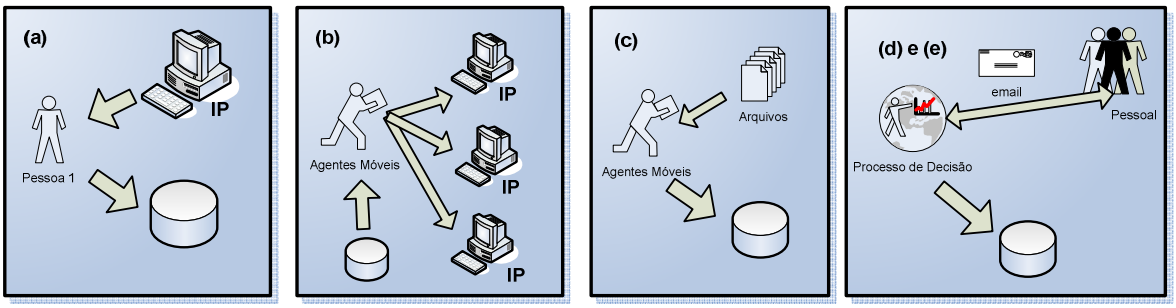

Figura 19 - Fluxo de atividades básicas da ferramenta
Fonte: Autoria própria (2007).

a) cadastro das estações, através de seu número IP, que irão servir como base para o trabalho;
b) envio dos agentes responsáveis por descobrir a informação às estações ativas – os agentes serão criados e enviados para os endereços previamente cadastrados. É importante destacar que em cada estação estará ativa uma "*agência*", base para a operação com a plataforma de agentes móveis;
c) execução da tarefa pelos agentes – cada agente numa estação de trabalho terá o objetivo de verificar todos os documentos do tipo não estruturado, ou

seja, textuais, e armazenar informações sobre aqueles que atendam ao critério de busca.;

d) solicitação do envio dos documentos – de posse da informação sobre os documentos encontrados nas estações, e caso seja necessário o arquivo na íntegra, uma aplicação solicitará automaticamente aos usuários proprietários o envio dos documentos via *e-mail*;
e) recepção dos documentos – os documentos serão recebidos através de *e-mail* e armazenados, adicionando-se então à base de dados textual com subsídios para pesquisa de informações sobre o tema pesquisado.

5.5 TESTES E RESULTADOS

Como pré-requisitos para a realização de testes, foram adotadas as seguintes premissas:

a) várias instâncias do Aglet foram executadas em diferentes portas de comunicação, sendo uma usada pelo servidor e o restante pelas máquinas clientes;
b) todos os arquivos a serem usados no teste, deverão estar em uma pasta padrão previamente identificada. Esta ação visa diminuir o tempo de processamento e busca de informações nos arquivos selecionados;
c) somente arquivos texto não estruturados e sem formatação serão inicialmente usados no teste, devido ao tamanho e a melhor acessibilidade dos dados contidos nos mesmos;
d) os arquivos escolhidos são todos referentes a informações de projetos financeiros apoiados pela instituição;
e) os dados dos arquivos selecionados serão atualizados na base de pesquisa, com uma periodicidade média, pois informações sobre projetos financeiros não são atualizadas com freqüência;
f) tanto a aplicação servidor quanto a aplicação cliente do *aglet* terão portas distintas e únicas possibilitando assim a implantação em uma mesma máquina.

5.5.1 Testes realizados com o uso da ferramenta

Seguem abaixo, os testes realizados com o uso da ferramenta proposta:

5.5.1.1 Processo de escolha e análise dos projetos financeiros

Para este processo de classificação, foram selecionados projetos financeiros apoiados pela instituição, no interior do Estado. Três profissionais inseridos no corpo técnico da área financeira, forneceram informações com o objetivo de compor amostras de treinamento e testes. Os projetos gerenciados por cada profissional foram selecionados e separados por área técnica de origem. Desta forma, a quantidade de arquivos textos obtidos foi o demonstrado na Tabela 1.

Tabela 1 - Quantidade de projetos obtidos por área de atuação.

Classe	Quantidade de Documentos
Indústria	90
Comércio	183
Microcrédito	230
Rural	152
Serviços	175

Fonte: Autoria própria (2007).

Baseados na metodologia de definição de amostra fornecida por Oliveira (2004), foram criadas quatro amostras de treinamento e uma amostra de teste para o processo de combinação. Como resultado foi observado a ocorrência de um alto índice dos termos ***pendências*** e ***serasa*** (freqüência igual a 615 e 512) no conjunto de termos que compõe a *expertise* combinada derivado da classe MICROCRÉDITO, bem como um alto índice de ocorrência dos termos ***pendências*** e ***documentação*** (freqüência igual a 834 e 714). Por a área de Microcrédito ser a mais requisitada por pessoas físicas de baixa renda, nada mais normal que estes termos e suas variações apareçam em demasia devido ao alto índice inadimplência registrado no mercado, e por a área de Comércio ser a mais requisitada por pequenos comerciantes, muitas vezes ainda na informalidade, nada mais normal que estes

termos e suas variações apareçam também em demasia devido a burocracia da regularidade de documentação aplicados por Órgãos governamentais.

5.5.1.2 Operacionalização do processo de cadastro e instalação do sistema

Foi efetuado o cadastramento de máquinas clientes com o sistema operacional Windows 2000. Em todas as máquinas clientes foi usado a sub-rede ip 192.168.0.0/24 com barramento Ethernet a 10 Mbps. A implantação da ferramenta se mostrou estável, tendo como ponto de atenção a instalação do Java Development Kit 1.3 no Servidor e 1.4 nas Estações clientes e plataforma *ASDK for Windows* para correto funcionamento. A instalação na plataforma windows apesar da exigência de uma versão específica se mostrou fácil de instalar devido a sua interatividade de interface com o usuário. Como o sistema SACI é desenvolvido com recursos de Java 2, o sistema só deve funcionar corretamente com versões recentes do interpretador. É provável a ocorrência de incompatibilidade se a versão do ASDK da IBM for utilizado, pois a versão do SourceForge utiliza recursos de Java 2.

5.5.1.3 Compatibilidade da ferramenta com outros aplicativos

Foi testado a compatibilidade da ferramenta com outros aplicativos do pacote Microsoft Office, quando da geração e análise de relatórios baseados na base de dados alimentada pela pesquisa dos agentes móveis. Para realização deste teste utilizou-se API fornecidas pela microsoft através do site e através do pacote MSDN. A integração se mostrou possível, sendo uma ferramenta de apoio ao Expertext. O uso destes aplicativos serviram como forma de análise da quantidade e da qualidade das informações contidas no banco de dados bem como dos arquivos da base de dados textual.

5.5.1.4 Operacionalização com grande quantidade de arquivos de rede

Foi inicialmente testado o sistema com os dados da figura 20 abaixo, variando-se o numero de arquivos por máquina e com projetos diferentes. Utilizou-se como sistema base a plataforma Windows 2000. Observou-se, conforme esperado um tempo de pesquisa mais alto no inicio devido a maior quantidade de informação

trafegada sendo esta diminuída com o aumento do numero de arquivos, devido a maior informação já armazenada em banco de dados.

Qtd de Arquivos p/máquinas	Tempo de conclusão da Pesquisa (em segundos)
10	1,51
50	1,83
100	2,17
150	2,21
200	2,35

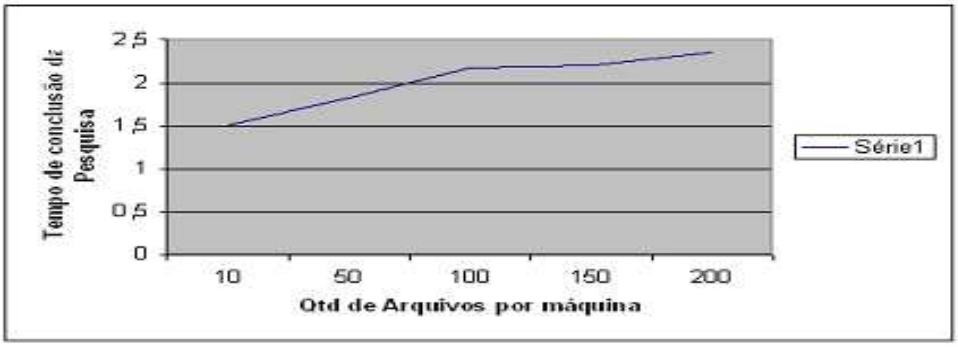

Figura 20 - Operação do sistema com grande quantidade de arquivos
Fonte: Autoria própria (2007).

5.5.1.5 Performance da rede de comunicação de dados

Para realização deste teste, foi montada uma rede ethernet com um Hub 3com de 10Mbps. Foi instalado uma estação de monitoramento com um Snnifer de Rede (ethereal 0.99.0 for windows), ferramenta esta que serviu para medição do performance da rede antes e depois do uso da ferramenta em teste, baseado no número de pacotes trafegados. Podemos observar que o uso é mais intenso no inicio das operações devido ao alto número de pesquisas realizadas pelos agentes. Ao longo do tempo a tendência é diminuir na intensidade devido ao armazenamento de informações dos arquivos já cadastrados, mantendo-se apenas praticamente o processo de atualização de arquivos novos ou modificados.

Sem o uso da Ferramenta para realização de pesquisas

Tempo (min)	Pacotes Trafegados (Kbytes)
5	98315
10	98451
15	98312
20	98418
25	98461
30	98364
35	98369
40	98482
45	98432
50	98514
55	98349
60	98618

Com o uso da Ferramenta para realização de pesquisas

Tempo (min)	Pacotes Trafegados (Kbytes)
5	99512
10	99470
15	99360
20	98756
25	98883
30	98673
35	98578
40	98582
45	98558
50	98739
55	98731
60	98767

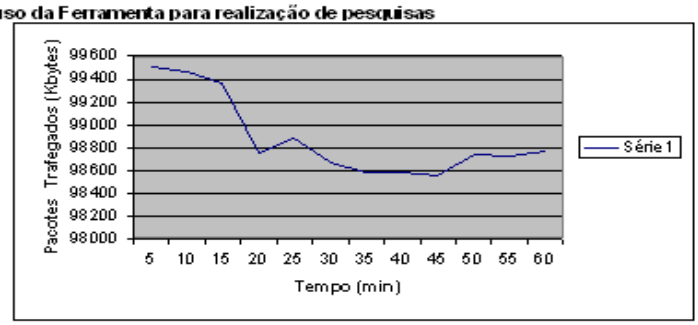

Figura 21 - Performance da rede de comunicação de dados
Fonte: Autoria própria (2007).

5.6 ANÁLISE

Foi constatado através dos testes efetuados, ser possível a análise das atividades e gerenciamento de projetos financeiros apoiados pela instituição, através da classificação de termos encontrados nos arquivos pertinentes a cada projeto. Constatou-se também, através do protótipo realizado, uma área de atuação carente de investimentos (Microcrédito e Comércio), com particularidades antes percebidas apenas empiricamente e após a realização dos testes através de números e gráficos.

Foi observado também, que ao manipular arquivos texto com tamanhos acima de 100 *kilobytes*, há uma queda acentuada de performance no processo de classificação. Parte deste comportamento é provocado pelo volume de informação demandada pela ferramenta Expertext, e à própria natureza do algoritmo de classificação *naive bayes* que trabalha com um número razoável de computações. O uso de arquivos acima de 100 *kilobytes*, bem como o aumento da quantidade de arquivos manipulados, também provoca um aumento na carga trafegada pela rede de comunicação de dados bem como uma queda na performance da atuação dos agentes móveis nas pesquisas realizadas remotamente.

6 CONSIDERAÇÕES FINAIS

Nas organizações, a escrita é um ato continuamente exercitado nas comunicações internas, documentos e qualquer outra forma de informação compartilhada. Com a introdução do computador, atualmente, é de certa forma difícil desassociar o raciocínio e a ação sobre o trabalho, sem a presença de artefatos digitais. A combinação da escrita e o uso de computadores acarretaram em uma avalanche de informações, onde muitas vezes se sabe que um determinado documento existe, mas não se sabe onde e como pesquisá-lo.

Tendo como um dos motivadores o cenário indicado, foi implementado o protótipo de uma ferramenta de busca, recuperação e análise de dados distribuídos em uma Instituição, idéia esta apoiada na utilização de agentes móveis com o apoio da ferramenta *Expertext*. O uso dos agentes móveis combinado com a linguagem de programação *JavaTM* (SUN, 2003), proporcionou uma maior portabilidade e facilidade no uso da ferramenta.

Do ponto de vista de sua funcionalidade e com relação aos usuários finais, o sistema implementado apresentou resultados efetivos tais como: uma grande quantidade de projetos financeiros na área denominada microcrédito, com pendências e irregularidades dos proponentes, junto ao Serasa e a outras instituições de controle do crédito, demonstrando a carência de um maior investimento e atuação pró-ativa nas áreas de controle e cadastro de propostas. Foi identificado também, que a área denominada de comércio, apresentou uma maior ocorrência da irregularidade de documentação junto a órgãos governamentais, dos proponentes dos projetos analisados. Estas duas constatações anteriormente citadas serviram de base para reformulação de políticas de financiamento na Desenbahia, instituição financeira pesquisada.

A aplicação da ferramenta *Expertext* como apoio à solução proposta de classificação e análise envolvendo arquivos de projetos financeiros, foi importante para a percepção da técnica de classificação dentro do paradigma de mineração de textos e sua utilidade no paradigma de gestão do conhecimento. A implementação do protótipo cumpriu sua função e o sistema de agentes funcionou adequadamente permitindo, através das informações recuperadas na rede e dos gráficos e críticas disponíveis, a análise dos melhores indicadores de avaliação no gerenciamento dos

projetos financeiros apoiados pela Instituição.

As considerações finais sobre a ferramenta implementada são de que a mesma atingiu o objetivo proposto na medida em que, através do seu uso, foi possível a observação e conseqüente atuação nas áreas determinadas para pesquisa. Com base no trabalho realizado conclui-se que o uso de agentes móveis para recuperação de informação distribuída e utilização no processo de gerenciamento do conhecimento é uma abordagem factível e viável tecnicamente. Além disso, no contexto empregado esses agentes tendem a ser muito importantes para as Organizações em geral e financeiras em particular, visto que podem ajudar a construção de uma base de informações com as informações de projetos apoiados pela Instituição, facilitando, assim, a busca por informações que propiciem a criação de novos projetos e linhas de financiamento Organizacionais, tornando a organização mais competitiva.

6.1 CONTRIBUIÇÕES

O processo de recuperação de informações em um ambiente distribuído nas Organizações modernas, baseados em agentes móveis, destinados à atividade de busca e recuperação de textos, pressupõe a existência de caminhos e documentos rotulados por algum agente humano. Na maioria das vezes, este procedimento é custoso em tempo e recursos, caso o volume a ser tratado seja substancial. Seguindo este raciocínio, uma contribuição importante deste trabalho é disponibilizar uma discussão sobre o tema, propondo uma solução e fornecendo uma estrutura básica para a implementação de soluções que auxiliem no processo de recuperação de informações específicas em um ambiente distribuído e volátil, e contribua para a tomada de decisão em contextos variados.

Dentre as principais contribuições deste trabalho podem ser destacadas:

a) Uma proposta para a implementação de um processo de recuperação e análise de informações de projetos financeiros, concluídos e em andamento, projetos estes que contam com o Apoio da Organização em questão;
b) A definição e implementação de uma proposta de implementação de apoio ao processo de recuperação de informações distribuídas através do uso de

agentes móveis. Além de permitir o gerenciamento dos critérios de busca e recuperação, a proposta sugere uma integração com a ferramenta *Expertext*, o que permite uma classificação e análise dos textos recuperados.

Apesar da solução proposta ser direcionada para a busca e recuperação de informações de projetos financeiros apoiados por uma determinada Instituição, a base da solução proposta pode ser utilizada com o objetivo de coletar expertise de maneira remota e descentralizada sobre a categorização de textos em tópicos específicos, como lixo eletrônico. A idéia neste caso é criar uma ferramenta poderosa na detecção de SPAM, proposta esta também sugerida por Oliveira (2004) quando da implementação da ferramenta *Expertext*.

Para a realização deste trabalho, foi adotada uma estratégia de pesquisa envolvendo também pesquisas bibliográficas e revisão da literatura sobre os temas, recuperação de dados e gestão do conhecimento, mineração de textos, padrões de projetos, e técnicas de desenvolvimento com agentes móveis direcionados ao propósito da automação no processo de busca e recuperação de informações. Esta estratégia foi importante para a definição das características da proposta de solução sugerida, bem como ao longo do desenvolvimento da implementação de teste efetuada.

6.2 PERSPECTIVAS

Os estudos efetuados no decorrer deste trabalho apontaram novas funcionalidades que podem ser acrescentadas à solução em trabalhos futuros. A principal extensão sugerida foi que a solução permita o controle do nível de cooperação e comunicação entre os agentes de busca e recuperação, através da manipulação das características intrínsecas dos agentes móveis.

Uma outra necessidade observada durante os estudos de caso, foi a necessidade de uma implementação de uma console de gerenciamento gráfica para controle da integração entre o gerenciamento dos agentes móveis e a ferramenta *Expertext*.

Além disso, podem ser listados alguns outros trabalhos futuros relacionados à evolução:

a) Melhoria no desempenho da solução de busca e recuperação de informações textuais ao trabalhar com arquivos de grande tamanho.
b) Possibilidade de trabalhar com arquivos de natureza diferente, sem necessidade de conversão para arquivos em formato texto.

REFERÊNCIAS

AGLETS Software Development Kit. Disponível em: <http://www.trl.ibm.com/aglets/index_e.htm>. Acesso em: 16 dez. 2004.

ARIDOR, Y.; LANGE, D. B. Agent design patterns: Elements of agents application design. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS, 2., 1998, Minneapolis. **Proceedings…** Minneapolis: IEEE, 1998. p. 108 – 115.

BARCLAY, R. O.; MURRAY, P. C. **What is Knowledge Manangement?** Austin: Knowledge Management, 1997.

BERNADES, M. C. **Avaliação do uso de agentes móveis em segurança computacional**. 1999. 112 f. Dissertação (Mestrado em Ciência da Computação) – Universidade de São Paulo, São Paulo, 1999.

BIESZCZAD, A.; PAGUREK, B.; WHITE, T. Mobile agents for network management, **IEEE Communications Surveys**, New York, vol. 1, n.1, fourth quarter 1998. Disponível em:<http://www.comsoc.org/pubs/surveys>. Acesso em: 5 jun. 2005.

BITTENCOURT, G. **Inteligência artificial:** ferramentas e teorias. Florianópolis: Editora da UFSC, 1998.

CAZZANIGA, F.; FORONI, G.; RIOS, J. F. Agentes de software inteligentes na prática. In: SISTEMAS inteligentes na internet, 1998. Disponível em: http://black.rc.unesp.br/IA/cintiab/busca/cap3.html. Acesso em: 21 fev. 2004.

CHANG, D. T., LANGE, D. B. Mobile Agents: a new paradigm for distributed object computing on the WWW. In: CONFERÊNCIA ANNUAL SOBRE PROGRAMAÇÃO ORIENTADA A OBJETO, SISTEMAS, LINGUAGENS E APLICAÇÕES, 11., 1996, Califórnia. **Proceedings**… Califórnia: SIGPLAN, 1998. Disponível em: < http://www.informatik.uni-trier.de/~ley/db/conf/oopsla/>. Acesso em: 16 jun. 2005.

CHIAVENATO, I. **Introdução à teoria geral da administração**. 6. ed. Rio de Janeiro: Campus, 2000.

CHIAVENATO, I. **Gestão de Pessoas:** o novo papel dos recursos humanos nas organizações. Rio de Janeiro: Campus, 1999.

DAVENPORT, T. H.; PRUSAK, L. **Conhecimento empresarial:** como as organizações gerenciam o seu capital intelectual. Rio de Janeiro: Campus, 1998.

DAVENPORT, T. H. **Ecologia da informação**. São Paulo: Futura, 2001.

DEUGO, D.; WEISS, M. **A case for mobile agent patterns**, 1999. Disponível em:<http://fusion.scs.carleton.ca/~deugo/Patterns/Agent/Papers/DeugoMAC3-99.pdf>. Acesso em: 13 mar. 2004.

DRUCKER, P. The age of social transformation. **The Atlantic Monthly,** Washington, DC, v. 274, n. 5, p. 53-80, nov. 1994.

DRUCKER, P. F. **Administrando em tempos de grandes mudanças**. São Paulo. Pioneira, 1995.

EDVINSON, L.; MALONE, M. S. **Capital intelectual**. São Paulo: Makron Books, 1998.

FAYYAD, U.; SHAPIRO, G. P.; SMITH, P. The KDD process for extracting useful knowledge from volumes of Data. **Communications of the ACM**, New York, v. 39, n. 11, p. 27-35 , Nov. 1996.

FERBER, J.; GASSER, L. Intelligence artificielle distribuée. In: INTERNATIONAL WORKSHOP ON EXPERT SYSTEMS E THEIR APPLICATIONS, 10., 1991, France. **Proceedings…** France: Avignon Cours, 1991.

GAMMA, E. et al. **Padrões de projeto:** soluções reutilizáveis de software orientado a Objetos. Porto Alegre: Bookman, 2000.

GASPAR, G. Boundaries, identify and aggregation: plurality issues in multiagent systems. In: WERNER, E.; DEMAZEAU, Y. (Ed.). **Decentralized artificial intelligence**. Amsterdan: Elservier Science, 1992. p. 199-212.

GIL, A. C. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2002.
GODOY, A. S. Pesquisa qualitativa: tipos fundamentais. **Revista de Administração de Empresas,** São Paulo, v. 35, n. 3, mai/jun. 1995.

GLASS, G. **Objectspace voyager core package technical overview**. New York: ACM Press/Addison-Wesley Publishing Co, 1999.

GRANVILLE, L. Z. **Agentes móveis no gerenciamento de redes**. Porto Alegre: UFRGS, 2000.

GRIMSHAW, D. **CPS 720 artificial intelligence topics with agents**, 2001. Disponível em http://www.ryerson.ca/~dgrimsha/courses/cps720/index.html. Acesso em 13 fev. 2004.

HAMEL, G.; PRAHALAD, C.K. **Competindo pelo futuro**. 15. ed. Rio de Janeiro: Campus, 1998.

HORVAT, D. et al. Mobile agents and java mobile agents toolkit. In: ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 33., 2000, Hawaii. **Proceedings**... Hawaii: IEEE Computer Society, 2000. Disponível em <http://www.computer.org/proceedings/hicss/0493/04938/04938029.pdf>. Acesso em: 02 fev. 2003.

INFERNO Network Software Solutions, Implementation of SNMP Agent by Inferno, **Lucent Technologies**, 1997. Disponível em: <http://www.lucent-inferno.com/Pages/Developers/Documentation/White_Papers/snmp.html>. Acesso em: 05 abr. 2005.

JOHANSEN, D.; SCHNEIDER, F. B.; RENESSE, R. V. **What TACOMA taught us**. New York: ACM Press Addison-Wesley Publishing, 1999.

KENDAL, E. A. et al. Patterns of intelligent and mobile agents. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS, 2., 1998. Minneapolis. **Proceedings**… Minneapolis: ACM Press, 1998. Disponível em: <http://autonomousagents.org/>. Acesso em: 04 abr. 2004.

LANGE, D. B.; OSHIMA, M. Mobile agents with Java: the Aglet API. **World Wide Web Journal,** New York, v. 1, n.3, p. 111-121, 1998. Disponível em:<http://www.moe-lange.com/danny/wwwj.pdf>. Acesso em: 16 nov. 2003.

LANGE, D. B; OSHIMA, M. **Programming and deploying java mobile agents with aglets**. London: Addison Wesley Longman, 1998.

LEMAY, L.; CADENHEAD, R. **Aprenda em 21 dias Java 2**: professional reference. 2. ed. Rio de Janeiro: Campus, 2001.

KOCH, F. L. ; WESTPHALL, C. B.. Decentralized Network Management using Distributed Artificial Intelligence. **Journal of Network and Systems Management**. USA, v. 9, n. 4, p. 357-388, dez. 2001.

MITSUBISHI ELETRIC ITA. HORIZON SYSTEMS LABORATORY. **Concordia**: Java mobile Agent Technology. Disponível em: <http://www.merl.com/projects/concordia/WWW/Concordia-at-a-glance.html>. Acesso em: 11 nov. 1999.

NONAKA, I.; TAKEUCHI, H. **Criação de conhecimento na empresa:** como as empresas japonesas geram a dinâmica da inovação. Rio de Janeiro: Campus, 1997.

OLIVEIRA, G. L. **Categorização automática de documentos por múltiplos classificadores Naive Bayes**. 2004. 136 f. Dissertação (Mestrado em Redes de Computadores) - Universidade Salvador, Salvador, 2004.

PIATETSKY-SHAPIRO, G.; FRAWLEY, B. (Ed.). **Knowledge discovery in databases.** Cambridge: AAAI Press/The MIT Press, 1991.

ROY, P. V. (Ed.). **Multiparadigm programming in Mozart/OZ.** New York: Springer, 2005.

SENGE, P. **A quinta disciplina**: a dança das mudanças: os desafios de manter o crescimento e o sucesso em organizações que aprendem. Rio de Janeiro: Campus, 2000.

SILVA, F. M. A. **A Transaction model based on mobile agents**. 1999. 167 f. Tese (Doutorado em Informática) - Universidade Técnica de Berlim, Berlim, 1999.

STANEVA, D.; GACHEVA, P. Building distributed applications with Java mobile agents. In: INTERNATIONAL WORKSHOP ON NEXT GENERATION NETWORK TECHNOLOGIES, 3., 2002. Bulgária. **Proceedings**... Bulgária: Rousse, 2002. Disponível em: http://saturn.acad.bg/bis/pdfs/16_doklad.pdf. Acesso em: 26 fev. 2004.

STEWART, T. A. **Capital intelectual:** a nova vantagem competitiva das empresas. 2. ed. Rio de Janeiro: Campus, 1998.

STRABER, M.; BAUMANN, J.; HOHL, F. **Mole:** a java based mobile agent system. Stuttgart: University of Stuttgart, 1997.

SVEIBY, K. E. **A nova riqueza das organizações:** gerenciando e avaliando patrimônios de conhecimento. Rio de Janeiro: Campus, 1998.

TOFFLER, A. **Powerschift:** as mudanças do poder. Rio de Janeiro: Record, 1994.

WEISS, G. **Multiagent systems:** a modern introduction to distributed artificial intelligence. Massachusetts: MIT PRESS, 1999.

WONG, D.; PACIOREK, N.; WALSH, T. **Concordia:** an infrastructure for collaborating mobile Agents. Waltham, MA: Horizon Systems Laboratory, Mitsubishi Electric ITA, 1997.

APÊNDICE A – Instalação do ambiente AGLET

Escolha os **drives\diretórios** utilizados para desenvolvimento e instalação. A tabela abaixo mostra os 4 diretórios e suas respectivas referências.

Descrição	**Referência no texto**	**Sugestão**
Pacote Java	(JDK_HOME)	C:\JDKx.x.x
Pacote do Aglets	(AGLET_HOME)	C:\Aglets
Programa Instalado	(PROG_INSTALL)	C:\Proj
Programa em desenvolvimento	(PROG_DEVEP)	F:\Proj\Aglets

Passos de instalação

Instalar JDK:	(JDK_HOME)
Descompactar AgletsXXX.zip Trocar diretório principal no intuito de manter portabilidade com versões subsequentes(XXX)	(AGLET_HOME)
Criar os diretorios de desenvolvimento e instalação	(PROG_INSTALL) (PROG_DEVEP)
Transfira o conteúdo do diretório (AGLET_HOME)\public para os diretórios de instalação(PROG_INSTALL) e desenvolvimento (PROG_DEVEP) para que possamos usar os exemplos para testar o sistema.	
Se estiver instalando o sistema pronto, coloque-o no diretório (PROG_INSTALL).	
Criar Arquivo para configurar e desconfigurar as variáveis de ambiente	**SetPaths.bat** **UnSetClassPath.bat**
Criar o arquivo de propriedades para desenvolvimento.	**Devep.props**
Criar o arquivo de propriedades de instalação e coloque no diretório de instalação.	**Install.props**
Modificar o arquivo para inicializar o servidor. Copiar o agletsd.bat para tahiti.bat e acrescentar o ítem 0 - colocar no diretório de desenvolvimento e de instalação. Criar os "shortcuts" adequados	**tahiti.bat**

Variáveis de Ambiente

SetPaths.bat

Para poder compilar e executar os programas em desenvolvimento é necessário configurar algumas variáveis de ambiente. A variável CLASSPATH deve conter os diretórios e pacotes de classes JAVA utilizadas na compilação dos programas, contudo não devem ficar lá depois da operação.

```
set JDK_HOME = (JDK_HOME)
set AGLET_HOME = (AGLET_HOME)
set PROG_INSTALL = (PROG_INSTALL)
set PROG_DEVEP = (PROG_DEVEP)
set PATH = %PATH%;%JDK_HOME%\bin;
set CLASSPATH = %CLASSPATH%;C:\jdk1.1.8\lib\classes.zip;
        %AGLET_HOME%\lib\aglets.jar;
        outros pacotes...
        %AGLET_HOME%\public;
        %PROG_DEVEP%;
```

UnSetClassPath.bat

Retira os diretórios extras colocados apenas para compilação.
set CLASSPATH = C:\jdk1.1.8\lib\classes.zip;

Compila.bat

É sempre bom colocar todos os arquivos a serem compilados num .bat para não precisar escrever tudo um milhão de vezes.

javac **<Arquivo1.java> <Arquivo1.java>...<Arquivon.java>**

Arquivo de Propriedades

Alem das opções colocadas aqui, as principais configurações do arquivo de propriedades se referem aos diretórios onde serão colocadas as classes. É importante ter um arquivo de propriedades para o processo de desenvolvimento e outro para a versão final do sistema. O desenvolvimento deve ser feito em diretórios separados e depois a versão final posta no diretório de instalação.

Devep.props

aglets.home = (AGLET_HOME)

```
# O sistema irá criar o diretório .aglets sob este
# diretório contendo as informações do usuário e
# atributos do servidor.
user.home = (PROG_DEVEP)
maf.port=4434
verbose = true
```

 # aqui devem ficar as classes para rodar em local
aglets.class.path= (PROG_DEVEP);(AGLET_HOME)\\public;

 # O diretório a seguir deve conter todas as classes que
 # não estão no (AGLET_HOME) e que irão ser TRANSFERIDAS.
 # Os exemplos fornecidos pelo pacote também devem ficar
 # neste diretório.
aglets.public.root = (PROG_DEVEP)

aglets.secure = true
#aglets.startup=(Aglet inicial)

 # se não colocar Resolve=true ambientes em rede não vão
 # conseguir transferir aglets
atp.resolve=true

#atp.domain=trl.ibm.co.jp
atp.offline=false
atp.authentication=false
#atp.secureseed=true - default
#atp.secureseed=false - é mais rápido

Servidor

Alterar o inicio do arquivo abaixo para os diretórios escolhidos em
tahiti.bat
...
set JDK_HOME = (JDK_HOME)
set AGLET_HOME = (AGLET_HOME)
...
Crie um "shortcut" para chamaro servidor e passar o arquivo de propriedades.
%PROG_DEVEP%\tahiti -f Devep.props

Operação

Arquivo de Propriedades

Install.props
aglets.home = (AGLET_HOME)
user.home = (PROG_INSTALL)
maf.port=4434
verbose = false

 # aqui devem ficar as classes para rodar em local
aglets.class.path=(AGLET_HOME)\\public;(PROG_INSTALL);

 # O diretório a seguir deve conter todas as classes que
 # não estão no (AGLET_HOME) e que irão ser TRANSFERIDAS.
 # Os exemplos fornecidos pelo pacote também devem ficar
 # neste diretório.

```
aglets.public.root = (PROG_INSTALL)
aglets.secure = true
aglets.startup=(Aglet inicial)

# se não colocar Resolve=true ambientes em rede não vão
# conseguir transferir aglets
atp.resolve=false

#atp.domain=trl.ibm.co.jp
atp.offline=false
atp.authentication=false
#atp.secureseed=true   -  default
#atp.secureseed=false  -  é mais rápido
```

Servidor

Alterar o inicio do arquivo abaixo para os diretórios escolhidos em
tahiti.bat

```
...
set JDK_HOME = (JDK_HOME)
set AGLET_HOME = (AGLET_HOME)
...
```

Criar "shortcut" e chamar:

Crie um "shortcut" para chamaro servidor e passar o arquivo de propriedades.
%AGLET_HOME%\bin\tahiti -f Install.props

Testes de Navegação

Exemplo 1 - Hello Aglet

Após rodar o servidor nos computadores ao longo da rede, rode o servidor de instalação e tente rodar o Aglet "HelloAglet" entre os nos da rede e verifique se eles estão indo e voltando.

Repita o mesmo procedimento para os servidor de desenvolvimento mas mantenha os de instalação nas máquinas remotas.

Implementação

Devemos agora implementar, executar e navegar um Agente simples para verificar se nossa plataforma de desenvolvimento e operação está ok.

Para compilar um programa java você deve criar um arquivo *.class. Isto é feito através do compilador javac ou, se você estiver usando algum ambiente como o visual J++, os comandos de compilação do mesmo.

Se você estiver usando o compilador javac, não se esqueça de configurar as

variáveis de ambiente:

SetPaths

javac nomedoarquivo.java

Depois de terminar de compilar, antes de rodar os programas, reconfigure as variáveis de ambiente:

UnSetPaths

O arquivo *.class criado deve estar no diretório de desenvolvimento: PROG_DEVEP, porem, se você quiser colocá-lo num subdiretório, o que é aconselhável, utilize uma cláusula package no inicio do arquivo com o nome do subdiretório.

Para rodar o programa dentro do servidor Tahiti basta colocar o nome do diretório depois ponto (.) e o nome da classe:

Ex. Projeto1.Aglet1

A seguir mande o sistema criar a classe.

Exemplo 1 - Ida

O exemplo abaixo irá saltar para o endereço definido pela urldest. A saída e a chegada do agente poderá ser observada pela tela do servidor Tahiti.

```
package Projeto1; //Subdiretório onde o arquivo vai ficar

import com.ibm.aglet.Aglet;
import java.net.URL;

        public class Aglet1 extends Aglet
        {
        // Variáveis do agente //
        private int Inicio=1;

        public void run()
        {
                try{
                  // Definir o endereço de destino //
                   URL urldest = new URL("atp://192.168.0.2");

                   if(Inicio==1)
                           {
                           Inicio = 0;
                           dispatch(url);    //Comando de movimentação
                           }
                }

                catch(Exception e)
                {
                System.out.println("Erro:"+e.getMessage());
                }
        }
}
```

Exemplo 2 - Ida e Volta e "Ping-Pong"

Como pode-se observar o método run é chamado a cada vez que o agente chega a algum lugar ou quando ele é criado. As variáveis do agente irão manter seu conteúdo entre os saltos. O exemplo abaixo fará o agente dar um salto e retornar ao endereço original. A urldest irá conter o destino e a urlback deve conter o endereço de volta do agente.

A condição da linha if(Inicio==0) irá ser aplicada apenas quando o agente chegar ao destino. Nesse momento podemos repor o valor da variável inicio para 0 e assim faremos ele ir outra vez para urldest, ou podemos colocar o valor 2 e assim o agente irá terminar ao retornar.

```
package Projeto1; //Subdiretório onde o arquivo vai ficar

import com.ibm.aglet.Aglet;
import java.net.URL;

        public class Aglet1 extends Aglet
        {
        // Variáveis do agente //
        private int Inicio=1;

        public void run()
        {
                try{
                  // Definir o endereço de destino //
                   URL urldest = new URL("atp://192.168.0.2");
                  URL urlback = new URL("atp://192.168.0.1");

                   if(Inicio==1)
                            {
                            Inicio = 0;
                            dispatch(url);     //Comando de movimentação
                            }
                   if(Inicio==0)
                            {
                            Inicio = 2; // vai e volta
                             //Inicio = 1;// faz ping pong                    dispatch(urlback);
                                                                    }
                    if(Inicio==2)
                            dispose();
                  }

                catch(Exception e)
                {
                System.out.println("Erro:"+e.getMessage());
                }
        }
}
```

Acesso aos recursos

Devemos configurar as permissões dos aglets e da máquina virtual JAVA para os recursos que desejarmos utilizar. Isto pode ser feito através da interface gráfica ou

alterando o arquivo aglets.policy que se encontra no subdiretório security do diretório .aglets.
Este arquivo é composto de seções "grant", seguido do "codebase" e a lista de permições. Os "codebases" indicam a partir de onde o sistema irá encontrar os programas dos agentes. Existem 3 tipos de "codebases", os que estão locais na máquina(seção codeBase "file:///-/")', os que estão em outro ponto da rede(codeBase "atp://*:*/") e os que são obtidos através de um servidor WEB(codeBase "http://*:*/").

Cada tipo de permissão tem os seus valores possíveis e isto é explicado na documentação anexa do aglets e no arquivo de exemplo sample.policy:

```
grant
codeBase "atp://*:*/"
{
...
permission java.io.FilePermission "c:\\-", "read,write";
};
grant
 codeBase "http://*:*/"
{
...
permission java.io.FilePermission "c:\\-", "read,write";
};
grant
 codeBase "file:///-/"
{
...
permission java.io.FilePermission "c:\\-", "read,write";
}
```

Arquivos

Para ler ou escrever em arquivos devemos configurar suas permissões. Isto pode ser feito através da interface gráfica ou alterando o arquivo aglets.policy que se encontra no subdiretório security do diretório .aglets.
A permissão para ler e escrever num diretório tem o seguinte formato:

```
permission java.io.FilePermission "Diretório\arquivo", "read,write,delete";
```

No exemplo damos permissão para escrita e leitura em todo o disco C:
permission java.io.FilePermission "c:\\-", "read,write";

```
File
.exists()

RandomAccessFile
  .writeBytes(..)
  .readLine();
```