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

Dorival Magro Júnior

Uma Arquitetura de Aplicação Utilizando Agentes Móveis para um Sistema de Imobiliárias

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

Prof. Rosvelter J**.** Coelho da Costa.

Florianópolis, Junho de 2002.

Uma Arquitetura de Aplicação Utilizando Agentes Móveis para um Sistema de Imobiliárias

Dorival Magro Júnior

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Fernando A. Ostuni Gauthier, Dr.
Coordenador do Curso

Banca Examinadora:

_____

Prof. Rosvelter João Coelho da Costa, Dr.
Presidente da Banca (orientador)

_____

Prof. Vitório Bruno Mazzola, Dr.
Membro da Banca

_____

Prof. Murilo Silva de Camargo, Dr.
Membro da Banca

_____

Prof. João Bosco Mangueira Sobral, Dr.
Membro da Banca

Sumário

Lista de Figuras

Lista de Tabelas

Abstract

Mobile agent architectures enable a decrease of information flow on the network by providing processes to move themselves over the network. In this dissertation, it is proposed a mobile agent architecture of a wab real estate to broker the sale/rent of residence to a client, no matter where they are.

A prototype was designed and implemented by using the aglet mobile agent platform developed by IBM of Tokyo. UML (Unified Modeling Language) was used for system analysis and modeling. The mobility infrastructure provided by aglets technology was explored as a basic tool for system integration over the Internet.

KEY WORDS: agents, mobility, web application, distributed systems.

Resumo

Arquitetura de agentes móveis possibilita uma diminuição no fluxo de informações na rede de modo que os processos movem-se através da rede. Nesta dissertação, está proposto uma arquitetura de agentes móveis de uma imobiliária para compra/venda de imóveis a um cliente não importando onde ele está.

Um protótipo foi desenhado e implementado usando-se a plataforma aglet de agente móvel desenvolvida pela IBM de Tóquio. A UML (Unified Modeling Language) foi utilizada para análise e modelagem. A infra-estutura de mobilidade proporcionada pela tecnologia aglets foi explorada como uma ferramenta básica para o sistema de integração na Internet.

Palavras chaves: Agentes, mobilidade, aplicações Web, sistemas distribuído.

Capítulo I

Introdução

Após o advento da Internet e da globalização, a utilização de sistemas distribuídos tornou-se ao mesmo tempo banal e problemática. Os paradígmas utilizados para desenvolver este tipo de sistema não o fazem de maneira inteiramente satisfatória devido, principalmente, ao alto grau de complexidade e especialidade exigido.

Um dos principais problemas associado ao desenvolvimento de sistemas distribuídos complexos é a integração entre as diferentes partes de um sistema ou entre sistemas diferentes. Para tanto, inúmeros modelos de sistemas distribuídos têm sido estudados e experimentados. Boa parte tenta imitar no contexto distribuído o mesmo mecanismo utilizado nos ambientes centralizados. É assim, por exemplo, no caso de modelo de objetos distribuídos tais como CORBA e RMI/JAVA. Neste caso, o mesmo princípio de interação entre objetos residindo em um mesmo computador é generalizado para a interação entre objetos residindo em diferentes computadores da rede.

Do ponto de vista de projeto, essa homogeneização semântica da interação é bastante positiva, pois permitirá uma uniformização do projeto local com o projeto distribuído, pois soluções de um poderão ser facilmente estendidas para o âmbito do outro. Mais infelizmente tais modelos não conseguem capturar toda a complexidade do fenômeno da distribuição. Citamos, por exemplo, o problema do tratamento de referências remota, intrínseco nesse tipo de modelo.

Há os modelos orientados a mensagens que por acomodarem melhor o fenômeno da distribuição, exigem uma infra-estrutura relativamente simplificada para funcionar.

Pecam, entretanto, no que diz respeito a reusabilidade e flexibilidade. Existe um forte acoplamento entre os tipos de mensagens que podem circular nos sistemas e os que podem respondê-las.

O modelo de agentes móveis permite um melhor compromisso entre os aspectos conjunturais de cada projeto com a estrutura que o sistema distribuído oferece. Ao movimentar-se um agente simula a passagem de mensagem, mas carrega geralmente consigo parte essencial da receita – eventualmente toda – para fabricar a resposta no ambiente que o recebe.

O objetivo principal deste trabalho de dissertação é estudar o desenvolvimento de software complexos utilizando a abordagem orientada a agentes móveis. Procura-se mostrar a viabilidade do uso dessa abordagem no desenvolvimento de uma aplicação distribuída consistindo na integração de uma rede de imobiliárias. Através da Internet, este sistema permite a pesquisa, reserva e liberação de reserva de imóveis das imobiliárias não importando onde os futuros clientes e nem os imóveis estejam. O protótipo desse sistema foi desenvolvido utilizando a plataforma Java 2 da Sun Microsystem e os aglets versão 1.1 Beta 3 da IBM que seria o ambiente para programação de agentes móveis em Java. As bases de dados acessadas neste sistema são do tipo Access 2000 da Microsoft. O sistema foi modelado segundo uma adaptação da UML (Unified Modeling Language) para melhor representar a realidade dos agentes móveis.

Este dissertação está organizado da seguinte maneira:

O Capítulo 2 propõe-se a fazer um estudo bibliográfico sobre a computação distribuída, que seria a camada mais externa da localização dos sistemas que utilizarão agentes móveis;

O Capítulo 3 aborda o conceito de agentes de software, a questão da mobilidade e como tudo isso relaciona-se com o desenvolvimento de sistemas complexos de software.

O Capítulo 4 estuda o framework dos aglets.

O Capítulo 5 apresenta a análise e projeto do protótipo de um sistema de pesquisa e reserva de imóveis que, utilizando a abordagem de software orientado a agentes, expõe a viabilidade de sua realização.

Finalmente, o Capítulo 6 expõe as conclusões do trabalho e suas perspectivas de continuação.

Capítulo II

Computação Distribuída

2.1 Introdução

A computação distribuída divide a carga de processamento ou serviços entre dois ou mais computadores, efetuando um processamento em paralelo, permitindo desta forma um maior desempenho e a possibilidade de utilização de um maior número de recursos que se encontram pela rede.

Este capítulo tem por objetivo mostrar o grande panorama onde a computação baseada em agentes encontra-se inserida.

2.2 Características da computação distribuída

A computação distribuída compreende a utilização de técnicas que permitem construir soluções distribuindo uma aplicação em agentes computacionais individuais com capacidade para resolver uma tarefa específica que, por sua vez, serão distribuídos ao longo da rede de computadores, compartilhando recursos e realizando tarefas cooperativas [JF98].

Algumas vantagens das aplicações distribuídas são apresentadas abaixo:

- Distribuitividade
  Compartilhamento de recursos e melhor distribuição da carga de processamento.

- Paralelismo

  Permite dividir um grande problema em pequenos problemas sem recorrer ao uso de soluções complexas e dispendiosas, cada computador processa a solução de um pedaço do problema.

- Confiabilidade

  Se uma máquina falhar, automaticamente o processo pode ser encaminhado para uma outra.

- Transparência
  - Localização

    Os recursos são acessados apenas pelo nome, não sendo necessário indicar a localização.

  - Migração

    Os recursos podem ser remanejados para um outro local sem necessitar alterar o seu assinalamento.

- Replicação

  Para melhorar o desempenho, os dados podem ser replicados em vários locais.

- Concorrência

  Os recursos podem ser acessados por vários usuários concorrentemente.

- Redundância

  Agentes processando informação redundante em diferentes pontos da rede podem ser utilizados em sistemas tolerantes a falta. Se uma máquina ou um agente falha, a informação é obtida por redundância.

- Crescimento incremental
  Conforme a necessidade, novos computadores podem ser facilmente adicionados.

Segue agora algumas desvantagens das aplicações distribuídas:

- Complexidade
  A complexidade inerente aos sistemas distribuídos representa um sério entrave ao desenvolvimento de sistemas deste tipo - Programação concorrente e/ou distribuída, bem mais complexa do que a seqüencial.

- Experiência
  Existe ainda pouca experiência em projeto, implantação e uso de sistemas distribuídos.

- Segurança
  Necessidade de cuidados especiais que garantam a segurança e a privacidade, quanto maior a facilidade de acesso, menor a segurança.

Na computação distribuída a arquitetura Cliente/Servidor é a mais utilizada, consistindo em uma parte servidor, responsável por prover informações ou acesso a recursos e a parte cliente que, através de uma rede se conecta ao servidor para extrair informações ou utilizar-se de serviços.

2.3 Camadas de uma aplicação distribuída

Uma aplicação pode ser construída utilizando-se várias camadas, cada uma disponibilizando serviços para a camada imediatamente superior e cada uma com um ambiente de trabalho definido.

A primeira camada - a de mais baixo nível - é a de rede que através de protocolos de comunicação permite a comunicação entre um grupo de computadores. As aplicações são processadas na última camada - a de mais alto nível - utilizando os serviços, protocolos de rede e o sistema operacional para coordenar suas tarefas através da rede.

A aplicação encontrando-se na última camada, constitui-se de várias partes. Por exemplo, processos, fluxos de controle (threads), objetos e agentes.

- Processos

  Um processo é especificado por um programa desenvolvido em alguma linguagem de programação, compilado e executado utilizando recursos tais como CPU, dispositivos de E/S e memória. Geralmente, sistemas operacionais podem executar vários processos concorrentemente.

- Fluxos de Controle (Threads)

  Todo processo tem pelo menos um thread de controle. Cada thread pode ser executado independentemente dos outros threads do programa com ou sem sincronização entre si. Por exemplo, um thread poderia monitorar a fila de impressão e outro poderia estar de prontidão a espera de um evento de mouse ou teclado.

- Objetos

  Um objeto é um conjunto de atributos relacionados com operações específicas para consultá-los, alterá-los ou para tomar decisões. Um processo pode ser composto de um ou mais objetos que podem ser acessados por um ou mais threads. Através da introdução da tecnologia de objetos distribuídos tais como RMI e CORBA, um objeto pode ser propagado de forma lógica através de vários processos ou de vários computadores [FG96].

- Agentes

  São entidades de software que realizam um conjunto de operações em nome do usuário e, diferentemente de um objeto, possuem mais recursos e autonomia. Uma aplicação distribuída pode ser composta por um ou mais agentes cooperando para alcançar um objetivo comum. Cada agente pode envolver vários processos ou hospedeiros remotos e podem consistir de vários objetos e threads. Uma aplicação pode ter vários agentes assim como um agente pode, ao mesmo tempo, atuar em várias aplicações.

2.4 Modelos de sistemas distribuídos

2.4.1 Sistema com memória distribuída

Sistemas com memória distribuída são sistemas onde cada processador tem sua própria memória local e que pode ser acessada diretamente só pela própria CPU. A transferência de dados de um processador para outro é realizada via rede. Difere dos sistemas de memória compartilhada nos quais vários processadores têm acesso ao mesmo espaço de memória via um barramento de memória.

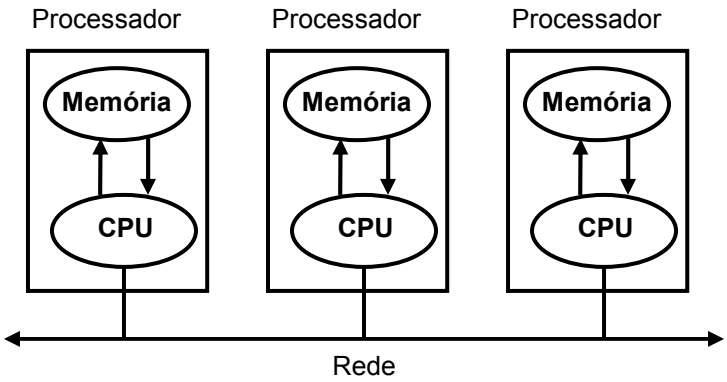

Figura 2.1 - Sistemas de Memória Distribuída

2.4.2 Sistemas com passagem de mensagem

O mecanismo pelo qual os dados da memória de um processador é copiado para a memória de outro chama-se passagem de mensagem. Nos sistemas de memória distribuída, os dados são geralmente enviados pela rede como pacotes de informação de um processador para o outro. Uma mensagem pode consistir de um ou mais pacotes e, habitualmente, inclui a informação de roteamento e controle.

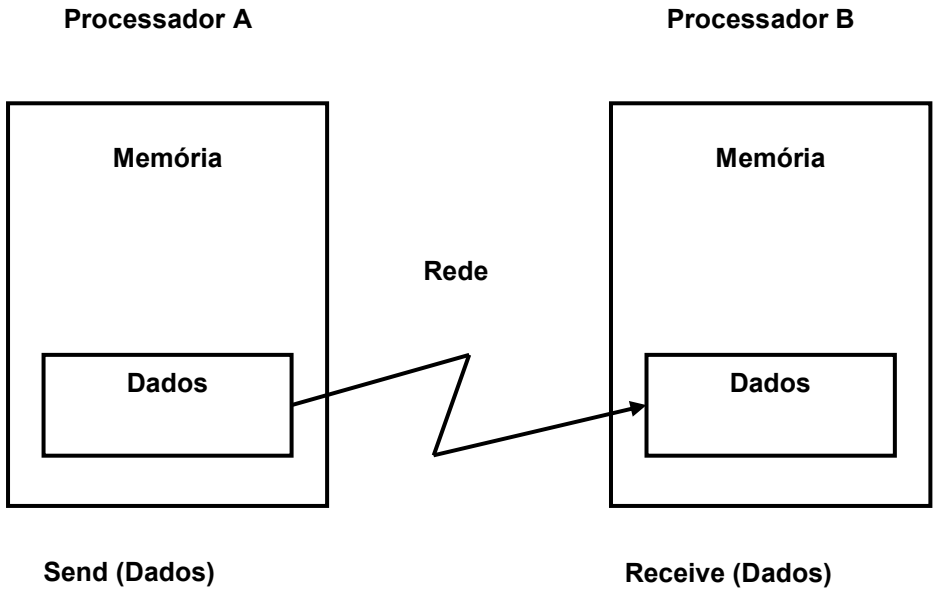

Figura 2.2 – Sistemas com passagem de mensagem

Nos sistemas de passagem de mensagem, todos os processos comunicam-se entre si através da troca de mensagem. O envio e a recepção de mensagens são operações cooperantes. Serão síncronas quando desejamos obter uma comunicação coordenada e segura, ou assíncronas, quando desejamos priorizar o desempenho. Para a comunicação assíncrona, é necessário utilizar um tampão para armazenar as mensagens que são recebidas para serem posteriormente copiadas pela aplicação.

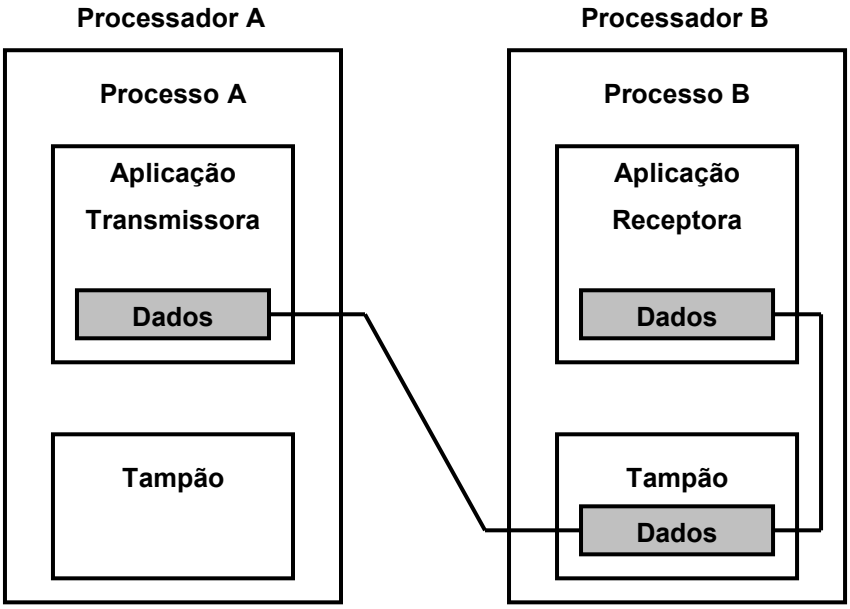

Figura 2.3 – Utilização de um tampão para receber mensagens

Freqüentemente, é utilizada uma interface (MPI - Message Passing Interface) projetada para ser um padrão prático e flexível para construção de sistemas de memória distribuída com passagem de mensagem.

2.4.3 Sistemas de Objetos Distribuídos

Os sistemas de objetos distribuídos constituem-se de um conjunto de objetos remotos que são disponibilizados através de rede para que possam ser acessados pelas aplicações distribuídas de forma remota e transparente [RO98]. Os objetos remotos são tratados como se fossem objetos locais. Outra característica é a possibilidade de construir um objeto em um hospedeiro e enviá-lo a um outro.

A Fig 2.4 ilustra algumas das características principais dos sistemas de objetos distribuídos. O skeleton é usado pelo servidor para gerar novas instâncias das classes e rotear chamadas remotas aos métodos desses objetos. O stub é usado pelo cliente para rotear as transações (os quais são invocação de métodos remotos) para o objeto

localizado no servidor. No servidor, as realizações das classes são passadas para um serviço de registro, que registra a nova classe no administrador de objetos utilizando o serviço de nomes. O objeto que é registrado e armazenado no servidor é disponibilizado para as aplicações clientes através do serviço de nomes e do administrador de objetos.

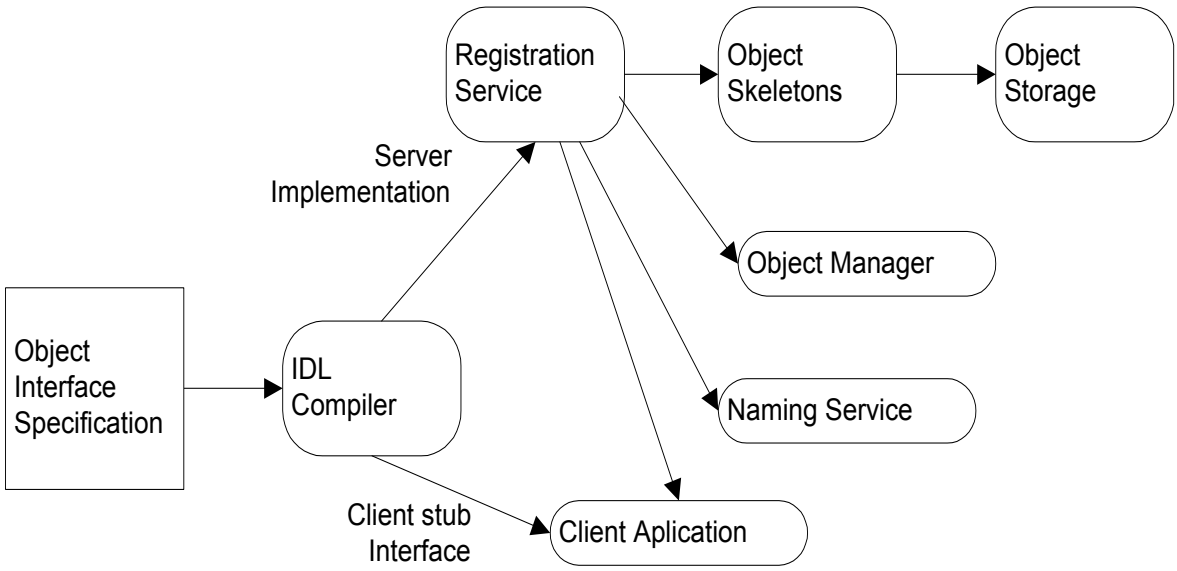

Figura 2.4 – Componentes de um sistema de objetos distribuídos [JF98]

A interface de especificação do objeto especifica quais são os métodos do objeto que são acessíveis remotamente e permite aos clientes acessá-los sem conhecer sua realização e ao servidor realizar a dita classe em alguma linguagem orientada a objetos. A plataforma Java possui a declaração de interface própria, CORBA utiliza o IDL (Interface Definition Language, uma linguagem independente de plataforma) e DECOM da Microsoft utiliza o COM (Component Object Model).

O administrador de objetos gerencia os skeletons dos objetos e suas referências no servidor. Este papel é realizado pelo ORB (Object Request Broker) no sistema CORBA e pelo serviço de registro (RMI registry) no sistema Java.

O serviço de registro e o serviço de nomes é o intermediário entre o objeto cliente e o administrador do objeto. Uma interface para um objeto é registrado para que ele seja acessível ao cliente. Este, por sua vez, utiliza o serviço de nomes para criar e utilizar remotamente os objetos que precisa.

O protocolo de comunicação de objeto gerencia os pedidos remotos e deve suportar pelo menos o envio e recepção de referências aos objetos, referências aos métodos e aos dados na forma de objeto ou tipo básico.

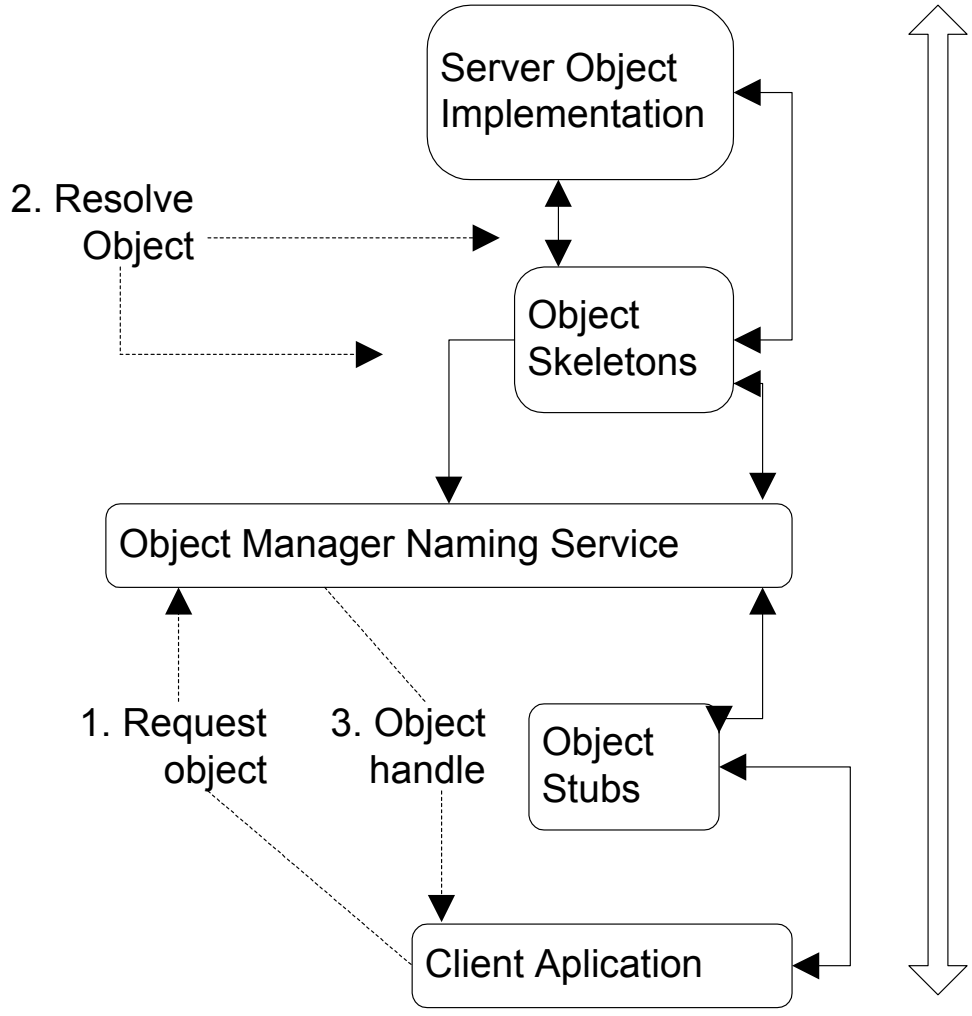

Figura 2.5 - Transações em tempo de execução [JF98]

A Fig. 2.5 mostra as transações em tempo de execução. O cliente solicita ao administrador de objetos e ao serviço de nomes, uma nova instância de uma classe. Por conseqüência, uma referência (handler) ao objeto gerado é entregue para a aplicação cliente. Com esta referência o cliente pode interagir com o objeto.

2.4.4 Sistemas cooperativos

Um sistema cooperativo é composto de um conjunto de agentes locais ou remotos compartilhando informação, enviando petições e interagindo para a obtenção de uma meta comum. Um sistema cooperativo possui os seguintes elementos:

- Agentes autônomos ou conduzidos por usuários;
- Servidores de operações e serviços;
- Repositório de dados dinâmicos e persistentes;
- Transações entre agentes, servidores e dados.

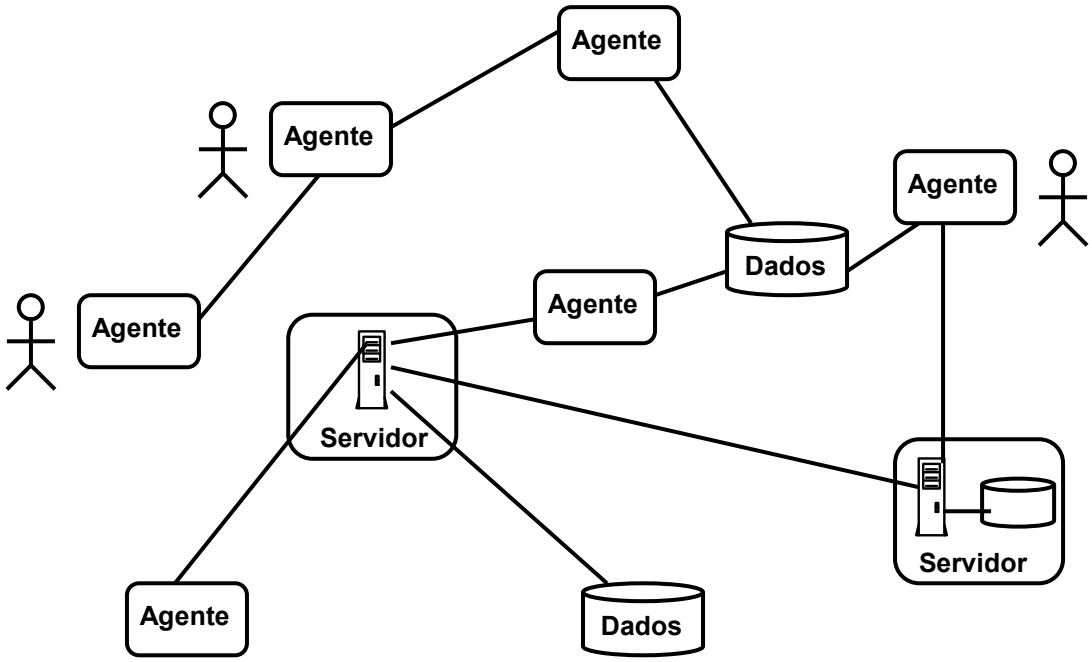

Figura 2.6 - Estrutura geral de um sistema cooperativo

Todos eles ocorrem em qualquer sistema distribuído, mas o fato de realizarem transações para a obtenção de uma meta comum faz com que o sistema seja cooperativo. Exemplos de sistemas cooperativos típicos:

- Tela compartilhada;
- Conversação interativa;
- Máquinas de computação paralela e distribuída;
- Agentes de pesquisa coordenada de dados.

Num sistema cooperativo, os agentes interagem dinamicamente. Por isso, a comunicação deve ser flexível e com capacidade para rotear as transações. Dependendo da aplicação, deve suportar mensagens ponto-a-ponto entre agentes, difusão de mensagens para toda a comunidade ou para um grupo específico de agentes participantes. Além de disso, deve contar com alguma forma de identificação para diferenciar um agente do outro e para o correto envio e recepção de mensagens e também com alguns mecanismos de autenticação.

A colaboração entre agentes é expressa normalmente por um conjunto de dados que necessitam ser compartilhado. Um aspecto importante no projeto de tais sistemas é manutenção da integridade e consistência do estado da informação compartilhada. Normalmente um agente atua como um intermediário para tratar os eventos que afetam o estado do sistema. O mediador recebe as notificações por parte do agente e leva cabo as atualizações de estado necessárias. Isto permite uma seqüência correta e segura das modificações dos estados compartilhados.

2.4.5 A Arquitetura de Sistemas Distribuídos

Há quatro tipos de elementos em um sistema que podem estar distribuídos: hardware, dados, programas e controle. Os três últimos referem-se ao software. A

distribuição de dados levou ao desenvolvimento de uma área bastante extensa, envolvendo sistemas de arquivos distribuídos e sistemas de banco de dados distribuídos.

Os sistemas distribuídos permitem que uma aplicação seja dividida em diferentes partes que se comunicam através de linhas de comunicação e cada parte podendo ser processada em um sub-sistema independente.

2.4.5.1 Características

Há algumas características que fazem com que os sistemas distribuídos sejam indicados em muitos projetos de sistemas, são elas:

- Crescimento incremental

  Um sistema distribuído apresenta facilidades de expansão, diferente de outros sistemas que só permitem a expansão por repetição. A repetição apresenta o inconveniente de produzir dois ou mais sistemas distintos e não um sistema maior.

- Confiabilidade

  Sistemas distribuídos podem ser potencialmente mais confiáveis devido à multiplicidade e a um certo grau de autonomia de suas partes. É notório que a distribuição física não é tão importante quanto a distribuição lógica. Esta última pode ser realizada tanto a um único processador quanto a vários processadores localizados num mesmo ambiente ou em ambiente distintos. A distribuição física está mais ligada a questões de desempenho, tempo de resposta, organização do sistema e principalmente à possibilidade de obter-se um controle explicito sobre o processamento e informações locais.

- Estrutura

  Sistemas distribuídos podem refletir a estrutura organizacional à qual eles servem.

- Proteção

  Sistemas distribuídos podem oferecer mais segurança do que sistemas centralizados. A segurança resulta muito mais da distribuição lógica do sistema do que da sua distribuição física.

A estrutura de um sistema distribuído pode ser visualizada em camadas. A camada mais interna corresponde ao hardware que é quem suporta todas as outras camadas de software. Em seguida vem o núcleo que é a primeira camada de software a envolver o hardware. Sobre esse núcleo, situa-se o sistema operacional propriamente dito que fornece o suporte necessário à execução de programas. E em seguida, tem-se a camada mais externa, correspondente ao nível de aplicação.

2.4.5.2 Desempenho de um Sistema Distribuído

O desempenho, considerado fator primordial de um sistema computacional distribuído, é analisado sob o ponto de vista do usuário através dos tempos de resposta obtidos a partir de uma requisição. Entre os prévios elementos que afetam o desempenho do sistema, destacam-se: a estação de trabalho, o servidor e a rede de comunicação.

2.4.5.3 Elementos que afetam o desempenho de um Sistema Distribuído

Os elementos, estação de trabalho, servidor e rede de comunicação, são considerados os mais relevantes para análise de desempenho de um Sistema Distribuído.

O desempenho individual de cada elemento é diretamente influenciado pelos demais, como pode ser observado nas seguintes situações:

- Servidores com grande capacidade de processamento, meio de comunicação de alta velocidade e estações de trabalho com baixa capacidade de processamento.

  As estações de trabalho não conseguem processar as informações na mesma velocidade em que estas chegam e, com isso, o tempo de resposta observado pelos usuários não é satisfatório, considerando-se as boas características dos servidores e do meio de comunicação.

- Estações de trabalho com grande capacidade de processamento, meio de comunicação de alta velocidade e servidores com baixa capacidade de processamento em relação à carga de trabalho.

  As estações de trabalho precisam esperar as informações requisitadas aos servidores para continuar o processamento e, com isso, o tempo de resposta observado pelo usuário também não é satisfatório apesar das boas características das estações de trabalho e do meio de comunicação.

- Servidores e estações de trabalho com grande capacidade de processamento e meio de comunicação de baixa velocidade.

  O tempo gasto na transmissão das informações degrada o tempo de resposta, tornando-o pouco satisfatório a despeito das características dos servidores e das estações de trabalho.

2.4.5.4 Processamento Distribuído (Cliente/ Servidor)

Uma tecnologia bastante utilizada em processamento distribuído é a de cliente/servidor. Os servidores, oferecem serviços a processos de usuários, ou seja,

executam a tarefa solicitada e enviam uma resposta ao cliente que se traduz nos dados solicitados. Por sua vez os clientes, solicitam um determinado serviço, através do envio de uma mensagem ao servidor. Enquanto o processo servidor está trabalhando sobre solicitação, o cliente está livre para realizar outras tarefas.

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

A arquitetura cliente/servidor, que corresponde à forma como os aplicativos são estruturados, pode ser dividida de duas maneiras [INF96].

- Aplicação duas camadas

  Corresponde a organização em duas camadas, ou seja, o cliente comunica-se diretamente com o servidor. Nesta situação a base de dados fica no servidor e as regras e a lógica da aplicação no cliente, dependendo da situação também poderão estar com o banco de dados. Uma conseqüência negativa desta alternativa diz respeito à manutenção, toda vez que uma aplicação for alterada, tanto bancos de dados como aplicações clientes precisam ser alteradas. Além disso, a aplicação cliente precisa ser distribuída para todos os usuários.

- Aplicação três camadas

  É uma arquitetura em multicamadas, nela uma camada intermediária é criada entre o servidor e o cliente. A função deste servidor intermediário é de armazenar as regras do negócio e a lógica da aplicação. O cliente fica

responsável apenas pela interface com o usuário. Qualquer alteração na camada intermediária é imediatamente assumida por todas as aplicações e pelo banco de dados.

Ainda em relação aos sistemas de cliente e servidor, esses são constituídos por três camadas básicas. A camada de serviços do sistema inclui o sistema operacional, componentes de rede e a interface com o usuário. Todo o software usado para controlar o hardware encontra-se nessa camada. A camada de serviços do sistema também deve oferecer um mecanismo de IPC (Interprocess Communication - Comunicação entre Processos) para passagem de mensagens. A camada de aplicação é constituída pelos processos cliente ou servidor e quaisquer outros processos da aplicação. A camada de mais baixo nível é a camada de hardware.

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

No que tange o equilíbrio de processamento, pode-se dizer que o mesmo corresponde à quantidade de trabalho realizado em cada lado cliente e servidor [REN94]. A maior parte do processamento da aplicação pode ser organizada em três segmentos: interativo, aplicação e banco de dados. Em uma aplicação centralizada no servidor o cliente realiza apenas parte do processamento interativo. Se o servidor realizar apenas parte do processamento do banco de dados a aplicação é centralizada na estação de trabalho. O ponto de equilíbrio em um processamento cliente/servidor ocorre entre o segmento de processamento interativo e processamento do banco de dados.

2.5 Conclusão

Neste capítulo, foi explorado um tipo de computação em evidência e promissor. A tendência é que tudo torne-se distribuído e é justamente neste contexto de aplicação que

o papel dos agentes se insere. Como será visto no próximo capítulo, os agentes vão além de simples coadjuvantes neste cenário. Em alguns casos eles podem se tornar os atores principais.

Capítulo III

Agentes de Software

3.1. Introdução

Desenhar e desenvolver sistemas altamente comercializáveis e de alta qualidade não é uma tarefa fácil. Neste sentido foram diversos paradígmas de engenharia de software tais como, programação procedural, programação estruturada, programação orientada a objeto, padrões de projeto e frameworks têm sido propostos..

Cada novo paradígma promete tornar o processo de desenvolvimento de software mais fácil e abranger problemas mais complexos. Mas novos problemas e situações vão surgindo a cada dia. Muitas vezes, a solução de um problema acaba gerando outros problemas. A necessidade/oportunidade de aprimoramento contínuo também é outro fator importante. Para tanto é necessário que se continue pesquisando e experimentando novas técnicas de engenharia de software [JNR&WM02]. É neste contexto que os agentes de software aparecem como uma das mais promissoras.

Agentes de software, embora não seja considerado por muitos um paradigma realmente novo, seu uso, em certos casos, apresenta algumas vantagens sobre os paradígmas convencionais. Este capítulo tem por objetivo introduzir a noção de agentes e alguns conceitos sobre software orientado a agentes.

3.2. A natureza de sistemas complexos de software

Sistemas altamente comercializáveis e de alta qualidade são complexos por natureza. Isto é tipicamente caracterizado pelo grande número de partes interagindo no sistema. A engenharia de software tendo o papel de prover estruturas e técnicas para facilitar a manipulação de sistemas complexos. Neste sentido, ferramentas tem sido propostas para gerenciar esta complexidade e geralmente, baseiam-se nos seguintes mecanismos:

- Decomposição
  A técnica mais básica para resolver um grande problema é dividí-lo em pedaços menores.

- Abstração
  Definir um modelo simplificado do sistema dando ênfase em alguns detalhes ou propriedades e suprimindo outros.

- Organização
  O processo de identificação e gerenciamento dos inter-relacionamentos entre os vários componentes do sistema.

A natureza e a maneira precisa pela qual essas ferramentas são utilizadas depende enormemente dos paradígmas de software. Em virtude disso, quando caracterizamos um novo paradígma, como o software orientado a agentes, este uso precisa estar claro, mais do que isso, quando avaliamos o poder do paradígma, precisamos discutir como ele pode ajudar os desenvolvedores a construir sistemas mais eficazes.

3.3 Software orientado a agentes

Um agente é um sistema computacional encapsulado que está situado em algum ambiente e que tem a capacidade de ações autônomas e flexíveis neste ambiente para alcançar os objetivos para qual ele foi projetado.

Uma outra definição de agente é, um sistema de hardware e/ou software satisfazendo as seguintes propriedades:

- Autonomia
  Agentes possuem controle sobre suas próprias ações e sobre seu comportamento.

- Habilidade social
  Agentes têm a capacidade de interação com o ambiente local de execução, bem como com outros agentes e usuários.

- Reatividade
  Agentes devem ser capazes de perceber o ambiente externo e responder a mudanças ocorridas de forma satisfatória.

- Pró-ativismo
  Agentes não reagem simplesmente em resposta ao ambiente, mas são capazes de exibir um comportamento baseado em metas, tomando a iniciativa, podendo inclusive, oportunamente adotar novas metas para satisfazer os objetivos de sua construção.

Quando adotamos o ponto de vista do mundo orientado a agentes, logo torna-se aparente que o uso de um simples agente é insuficiente. A maioria dos problemas requer ou envolve múltiplos agentes. Deve-se isto a necessidade de representar a natureza descentralizada do problema, multiplicidade do controle, múltiplas perspectivas, ou

interesses conflitantes. Além disso, os agentes precisarão interagir para alcançar seus objetivos individuais ou senão para gerenciar as dependências que resultam por estarem situados em um ambiente comum. Essas interações vão desde uma interoperação semântica (a habilidade de trocar comunicações de forma compreensiva), passando pelo tipo tradicional de interação cliente/servidor (a habilidade de requisitar que uma certa ação seja executada), até as interações sociais ricas (a habilidade de cooperar, coordenar e negociar o curso das ações).

Contudo, qualquer que seja a natureza do processo, existem dois pontos que diferenciam qualitativamente as interações dos agentes daquelas que ocorrem em outros paradígmas de engenharia de software. Primeiramente, interações orientada a agentes geralmente ocorrem através de uma linguagem de comunicação de agentes de alto nível. Segundo, os agentes são solucionadores de problemas flexíveis, operando em um ambiente na qual eles não controlam totalmente, as interações precisam ser manipuladas de maneira similarmente flexíveis.

Na maioria dos casos, agentes agem para alcançar objetivos em nome de indivíduos (usuários) ou softwares de terceiros. Deste modo, sempre que os agentes interagirem, existirá tipicamente algum contexto organizacional subjacente. Este contexto ajuda a definir a natureza do relacionamento entre os agentes.

Agrupando tudo isto, podemos notar que a adoção de uma abordagem orientada a agentes por parte da engenharia de software, significa decompor o problema em vários pequenos problemas e interagir componentes autônomos (agentes) que têm objetivos particulares para alcançar. O modelo chave de abstração que define o sistema orientado a agentes são agentes, interações e organizações.

3.4 Mobilidade

Mobilidade é uma característica pela qual os agentes podem “viajar” pela rede para cumprir os objetivos para qual o agente foi projetado. Os agentes não precisam

necessariamente exercer esta mobilidade, principalmente se eles se encontram no mesmo ambiente onde a computação ocorrerá.

Agentes móveis também podem ser definidos como programas que podem ser disparados de um computador (cliente) a fim de serem executados remotamente em outro (servidor). Além disso, esses elementos possuem características inerentes ao conceito de multiagentes, que proporcionam um bom desempenho em sistemas de objetos distribuídos. Assim, características como cooperação, autonomia e representatividade foram herdadas da sua própria origem, entretanto, foram acopladas outras tantas a fim de suprir as necessidades exigidas para o bom funcionamento de modelos que utilizam esse paradígma, a saber [HAR96]:

- Objetos passantes
  Quando um agente móvel é transferido, todo o objeto é movido, ou seja, o código, os dados, o itinerário necessário para chegar ao servidor, o estado de execução, etc.

- Assincronismo
  O agente móvel possui seu próprio *thread* de execução e esse não precisa ser executado sincronamente.

- Interação local
  O agente móvel interage com outros agentes móveis ou objetos estacionários locais. Se necessário, um agente mensageiro é despachado para facilitar a interação com agentes remotos.

- Operações sem conexão
  O agente móvel pode executar tarefas mesmo com a conexão fechada. Quando se faz necessária transferência de agentes, aguarda-se até que a conexão seja restabelecida.

- Execução paralela

  Múltiplos agentes podem ser disparados para diferentes servidores a fim de executar tarefas em paralelo.

Os agentes móveis possuem várias qualidades que aumentam os benefícios de sua utilização[LDA98]:

- Redução do tráfego de rede

  Como os agentes são despachados para o local onde ocorre a computação, o único tráfego de rede existente é o do agente móvel indo para o sítio destino e o resultado do seu processamento, naquele local, retornando para o sítio de origem.

- Superação da latência da rede

  Alguns sistemas de tempo real não suportam uma grande latência da rede. Precisando de respostas imediatas. O uso de agentes móveis é muito cômodo, pois o agente pode ser despachado para agir localmente.

- Protocolos encapsulados

  Num sistema distribuído, cada hospedeiro possui um protocolo responsável por receber e enviar os dados. Isso pode se tornar um problema quando for necessária a implantação de novas regras de segurança ou otimização. Os agentes, por sua vez, podem se mover para um hospedeiro e estabelecer um canal baseado nas propriedades do protocolo.

- Execução assíncrona e autônoma

  Uma vez criados e despachados, os agentes percorrem a rede em busca dos objetivos para a qual eles foram projetados. Eles o fazem de maneira autônoma (sem intervenção do usuário) e sem a necessidade do originador da execução, no caso o agente origem, estar conectado.

- Adaptação dinâmica

  Os agentes móveis podem, de maneira autônoma, reagir ao ambiente se adaptando ao mesmo.

- Naturalmente heterogêneo

  Podendo operar em quase qualquer plataforma, os agentes móveis podem se deslocar através de redes com configurações, acesso a dados e até sistemas operacionais diversos.

- Robustez e tolerância a faltas

  A habilidade dos agentes móveis de reagir dinamicamente em situações e eventos desfavoráveis permite que se construam sistemas distribuídos robustos e tolerantes a falta.

3.5 Conclusão

Neste capítulo foram introduzidos alguns conceitos sobre software orientado a agentes e noções de agentes e sua mobilidade. Estes conceitos são necessários para o próximo capítulo onde são discutidas as formas de realização dos agentes móveis utilizando os Aglets.

Capítulo IV

Aglets

4.1 Introdução

Os aglets surgem como o próximo salto na evolução de conteúdo executável na Internet, introduzindo agentes com a capacidade de se mover de uma máquina para outra através da rede. A migração inicia com a interrupção da execução do aglet na máquina origem, seu despacho para uma máquina remota e o reinício da execução após sua chegada ao destino. Mecanismos de segurança impedem que aglets não autorizados tenham acesso a determinados recursos do sistema [LDB&OSH98].

Aglets foram introduzidos originalmente por pesquisadores da IBM de Tókio e têm os seguintes objetivos:

- Fornecer um modelo compreensivo e simples de programação utilizando agentes móveis sem, no entanto, implicar em modificações na máquina vitual Java ou no seu código nativo;

- Disponibilizar mecanismos de comunicação e interação que permitissem agentes se comunicarem entre si;

- Projetar uma arquitetura de agentes móveis extensível e reutilizável;

- Prover um framework coerente com o modelo Web/Java.

Aglets relacionam-se com applets de acordo com a seguinte equação:

aglet = agent + applet

A plataforma Aglets Workbench [IBM98] é uma das mais utilizadas, no desenvolvimento de softwares orientados a agentes móveis e serviu de base para especificação do modelo MAF/OMG (Mobile Agent Facility) da organização Object Management Group.

4.2 Aglets workbench

Aglets compreendem uma biblioteca arquitetural de classes e programas de suporte operacional, todos existentes em código Java. Aglets Workbench pode ser livremente utilizado sem fins comerciais. A biblioteca contém classes para descrever agentes, mensagens, itinerários, identificadores de agentes e contextos.

Aglets Workbench utiliza-se da capacidade multiplataforma de Java para criar uma arquitetura simples suportando a migração e execução de código entre computadores interligados através da Internet. Os principais elementos desta plataforma são comentados a seguir:

Aglets

São programas assíncronos e autônomos. Sabem o que fazer (que computação devem realizar) e para onde ir (qual o itinerário a seguir) em qualquer momento durante a sua existência.

Proxies e Contextos

Para permitir a execução segura de agentes quaisquer em hospedeiros quaisquer é necessário garantir a integridade tanto dos agentes como dos

hospedeiros. Agentes podem estar hospedados em um hospedeiro hostil, defeituoso, ou que hospeda outros agentes maliciosos. Por outro lado, um hospedeiro que hospeda um agente malicioso tem que se proteger de ações não permitidas que um agente queira executar.

Estes problemas inerentes à segurança de sistemas multi-agentes são solucionados por duas maneiras:

- Proteção de hospedeiro
  Execução dos agentes dentro de um contexto de agentes (um sandbox), que confere proteção ao hospedeiro.

- Proteção de agentes
  Interposição de proxies entre os agentes e o contexto de modo que cada agente esteja associado a um proxy que o protege dos outros agentes;

Transferência de Aglets entre hospedeiros

O código e o estado de execução dos agentes são transferidos entre hospedeiros através de um protocolo seguro chamado Agent Transfer Protocol – ATP.

Execução de Aglets

Aglets são objetos computacionais cuja execução é suportada dentro de uma máquina virtual Java. Aglets executam as tarefas para as quais foram incumbidos através do envio de mensagens para os outros agentes e para o contexto de execução onde estão hospedados.

A utilização de aglets segue o mesmo princípio dos frameworks[1] applets e servlets. Em conformidade com o princípio de desenvolvimento de frameworks, o programador que desenvolve aglets realiza uma série de métodos com nomes pré-

[1] Frameworks são esqueletos de uma arquitetura de sistema suportados por uma biblioteca (API) orientada a objetos.

definidos, que respondem às ocorrências de eventos durante o ciclo de vida dos aglets. Os principais métodos de definição de um aglet são:

- `onCreate()` ou `onClone()` - invocado sobre o agente imediatamente após sua criação;

- `onDisposing()` - invocado sobre o agente antes de sua destruição;

- `onDispatching()` e `onArrival()` - invocado sobre o agente antes e depois da sua migração entre hospedeiros, respectivamente;

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

- `handleMessage()` - invocado sobre o agente quando ele recebe uma mensagem de outro agente;

- `run()` - invocado sobre o agente no momento da sua criação ou reativação.

4.3 Objetivos dos aglets

- Fornecer um modelo fácil e compreensivo de programação de agentes móveis sem requerer modificações da máquina Java ou código nativo.

- Suporte de comunicação dinâmica que permite aos agentes se comunicarem com agentes desconhecidos tão bem quanto com agentes conhecidos.

- Projetar uma arquitetura proporcionando reusabilidade e extensibilidade.

- Projetar uma arquitetura harmoniosa com a tecnologia Web/Java existente.

- Fornecer mecanismos de segurança que são compreensíveis e simples o bastante para permitir que usuários finais possam confiar em agentes móveis.

4.4 Avaliação de Aglet API

O Aglet API define a funcionalidade fundamental de agentes móveis. A figura 4 mostra as principais interfaces e classes definidas no Aglet API e suas relações.

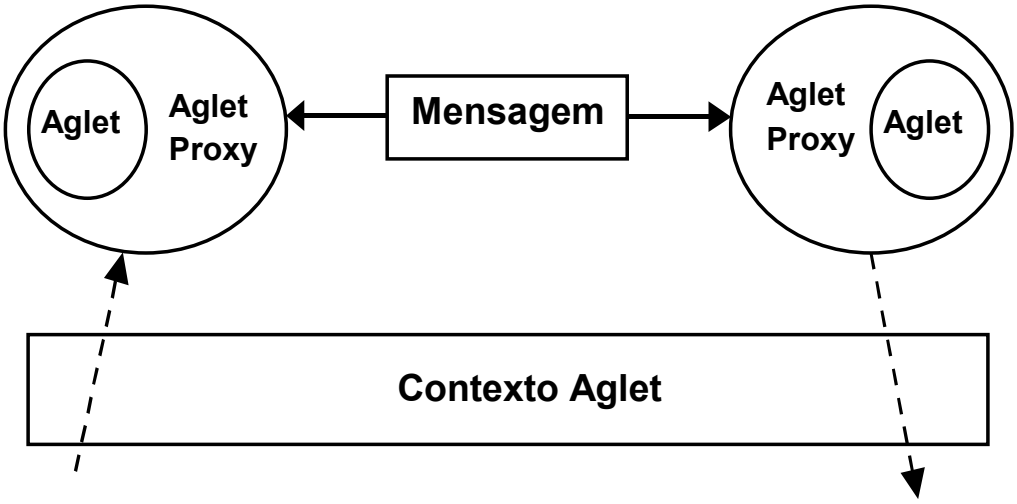

Figura 4.1 – Interfaces e classes definidas no aglet API

com.ibm.aglet.Aglet

A classe abstrata `Aglet` define os métodos fundamentais (por exemplo, `dispatch (URL)`) usados para controlar os ciclos de mobilidade e a vida dos agentes móveis. Todos os agentes móveis definidos em `Aglet` realizam esta classe abstrata. A função `Aglet.dispatch(URL)` induz um aglet a mover-se da máquina local até o destino especificado no parâmetro. A função `Aglet.deactivate (duração)` permite um aglet ser persistentemente armazenado e a função `Aglet.clone()` gera

uma nova instância do aglet que tem o mesmo estado que o aglet original. O objeto retornado pela função `clone` não é um objeto `Aglet`, mas um objeto `AgletProxy`.

A classe Aglet é também usada para acessar os atributos associados a um aglet. O objeto `com.ibm.aglet.AgletInfo` que pode ser obtido pela função `Aglet.getAgletInfo()`, contém atributos inerentes a um aglet, como seu tempo de criação, codebase e também seus atributos dinâmicos: tempo de chegada e o endereço de seu contexto atual.

A tabela seguinte alguns métodos primários com suas semânticas.

Método	**Comportamento**
dispose()	Finaliza o aglet.
dispatch(URL)	Despacha o aglet para o destino especificado no endereço URL.
deactivate(duração)	Instrui o aglet para se armazenar em uma mídia persistente.
getAgletInfo()	Fornece informações sobre o aglet.

Tabela 4.1 – Métodos primários e suas semânticas

com.ibm.aglet.AgletID

Todas instâncias de aglet têm suas identidades próprias e únicas que permanecem imutável durante o seu ciclo da vida. Uma identidade pode consistir de vários atributos como um "user ID" e o tipo de um sistema de agente. O `AgletID` é um objeto que mantém um único identificador para um dado agente, enquanto faz o encapsulamento de seus detalhes de representação.

com.ibm.aglet.AgletProxy

Os objeto de interface `AgletProxy` age como um manipulador (handle) de aglets fornecendo uma interface comum de acesso ao aglet. Já que aglets possuem

vários métodos públicos que não deveriam ser diretamente acessados por outro aglets para razões de segurança, qualquer aglet que quer se comunicar com outros aglets tem que primeiro acessar os seus objetos proxies para então interagir com eles. Em outras palavras, o aglet proxy age como um objeto de escudo que protege um agente de agentes maliciosos. Quando invocado, o objeto proxy consulta o `SecurityManager` para determinar se é permitida ao visitante a execução do método. Outro papel importante da interface `AgletProxy` é prover o aglet com transparência de localização. Se o aglet reside em um hospedeiro remoto, ele remete os pedidos para o hospedeiro remoto e devolve o resultado para o hospedeiro local.

O `AgletProxy` pode ser obtido das seguintes maneiras:

- Obtendo uma enumeração de proxies em um contexto:
  `AgletContext.getAgletProxies()`

- Obtendo um `AgletProxy` para um `AgletID` via
  `AgletContext.getAgletProxy(AgletID)`
  ou
  `Aglets.getAgletProxy(String contextName, AgletID)`

- Obtendo um objeto `AgletProxy` por mensagem. Um objeto `AgletProxy` pode ser posto no objeto `Message` como um parâmetro, e enviado para o aglet localmente ou remotamente.

- Pondo um objeto `AgletProxy` na propriedade de contexto,
  `AgletContext.setProperty(String,Object).`
  e compartilhando o objeto `proxy`.

A biblioteca runtime é responsável por fornecer uma realização da interface `AgletProxy`.

com.ibm.aglet.AgletContext

A classe `AgletContext` fornece uma interface para o ambiente runtime que se insere os aglets. Qualquer aglet pode obter uma referência para seu objeto `AgletContext` atual via `Aglet.getAgletContext()` e utilizá-lo para obter informações locais como o endereço do contexto do hospedeiro e a enumeração dos `AgletProxies`, ou para criar um novo aglet no contexto. Uma vez que o aglet tenha sido despachado, o objeto de contexto utilizado não é mais disponível.

Os programadores aglet não precisam programar esta interface, pois já é fornecida pela biblioteca runtime.

com.ibm.aglet.Message

Os objetos de Aglet se comunicam através da permutação de objetos da classe `Message`. Um objeto `Message` possui um objeto String para especificar o tipo da mensagem e objetos arbitrários como parâmetros. Uma mensagem pode ser enviada para o aglet chamando `AgletProxy.sendMessage(Message msg)`,

`FutureReplyAgletProxy.sendAsyncMessage(Message msg)`,

ou

`void AgletProxy.sendOnewayMessage(Message msg)` e é recebido por um aglet através de `Aglet.handleMessage(Mensage msg)`.

com.ibm.aglet.Ticket

Um objeto `Ticket` é usado para especificar um destino e a qualidade de transferência. Em outras palavras, define a via pela qual um aglet é transferido. Pode incluir o destino, o protocolo que será usado, a qualidade como um intervalo de tempo e o nível de integridade ou confidencialidade que devem ser obtidas.

com.ibm.aglet.FutureReply

Um objeto da interface `FutureReply` é retornado como resultado do envio assíncrono de mensagens. É utilizada para determinar se a resposta encontra-se disponível associando ou não um timeout.

4.4 Aglets e seus ciclos de vida

A classe `com.ibm.aglet.Aglet` fornece as funcionalidades básicas dos aglets. Para criar um aglet, precisamos primeiramente instanciá-la. Existem dois caminhos para isso. O primeiro invocando o método `AgletContext.createAglet(URL codebase, String name, Object init)`. Esta função cria uma nova instância dentro do contexto especificado e a inicializa se necessário. Invoca então o `Aglet.onCreation(Object init)` do objeto com o parâmetro que foi passado para o método `createAglet`. O outro modo é criar uma cópia de um aglet existente usando a função `Aglet.clone()`. O aglet clonado tem a mesma situação do original, mas tem um objeto AgletID diferente, portanto, uma identidade distinta.

Uma vez criado, um objeto aglet pode ser despachado para e/ou retirado de um servidor remoto, desativado e colocado em armazenamento secundário para reativação mais tarde.

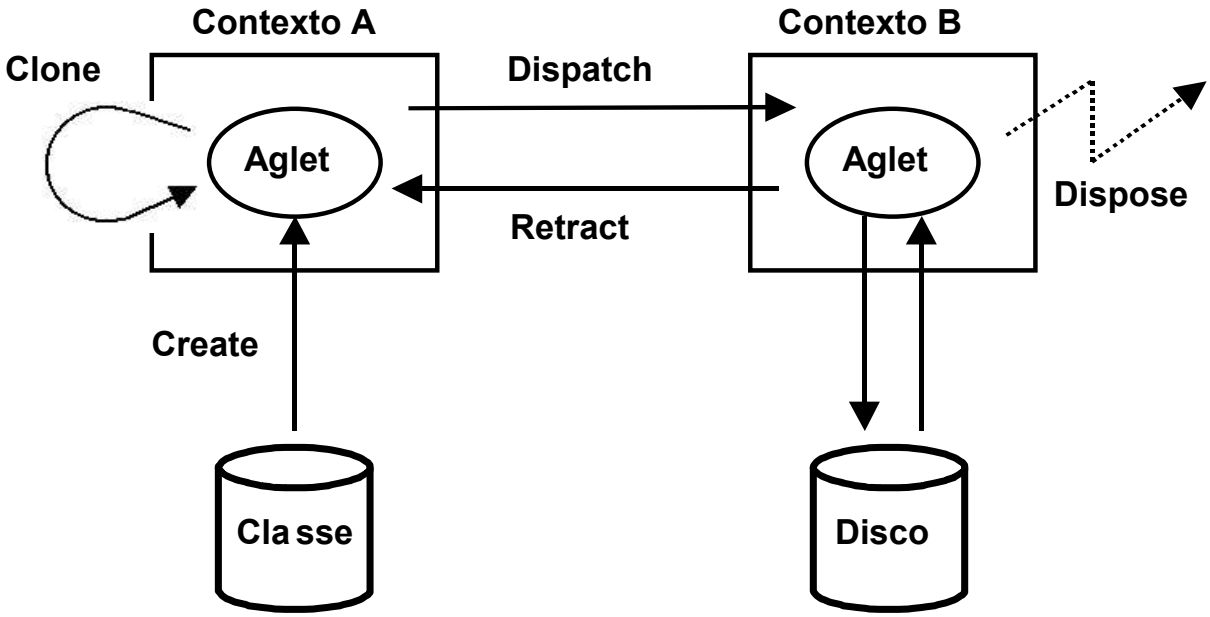

Figura 4.2 – Aglets e seus ciclos de vida [IBM98]

Um aglet pode despachar-se para um servidor distante invocando o método Aglet.dispatch(URL dest). Na realidade, de um contexto aglet para outro.

Um servidor aglet pode incluir múltiplos contextos em uma mesma máquina virtual Java e um hospedeiro pode incluir múltiplos servidores. Para que isso funcione, contextos possuem três níveis de identificação:

- o endereço do hospedeiro, tipicamente um endereço IP.
- o número da porta associada ao servidor.
- o nome do contexto dentro do servidor.

```
Por exemplo,
   atp://aglets.ibm.com:1434/nome_do_contexto
```

O envio de um aglet causa a suspensão de sua execução, a serialização da sua condição interna e a recuperação do seu bytecode na forma padrão para então ser transportado para o destino. No lado do receptor, o objeto de Java que é recebido é reconstruído de acordo com os dados recebidos e um novo thread é designado e executado.

Aglets podem ser persistentes. A função `Aglet.deactivate(long timeout)` faz com que um aglet seja armazenado em um armazenamento secundário e que "durma" um número determinado de milissegundos. Depois de passar o tempo estipulado ou outro programa pedir sua reativação, o aglet é ativado dentro do mesmo contexto onde ele foi desativado.

Diferentemente de objetos Java normais que são automaticamente descartados por um coletor de lixo (`garbage collector`), um objeto aglet, desde que ativo, pode decidir se morre ou não. Se o método `dispose()` for chamado para matar o aglet, o `onDisposing()` é chamado para executar a finalização satisfatória para a condição atual do aglet. Observando-se que este comportamento é diferente do método `finalizer()` de objetos Java ordinários, que é invocado quando o objeto for destruído pelo coletor de lixo. Os programadores de aglets são responsáveis pela liberação de recursos alocados ao aglet como arquivos ou conexões de BD, pois tais recursos não podem ser automaticamente liberados.

Uma vez descartado, a instância torna-se uma entidade inválida e qualquer tentativa para operá-la resultará em exceção do tipo `SecurityException`.

4.4 A arquitetura aglet

A arquitetura aglet consiste de duas camadas para as quais duas APIs são associadas. A camada Runtime define o comportamento de componentes tais como o `AgletProxy` e o `AgletContext` e fornece as funções fundamentais para os aglets serem criados, administrados e despachados para hospedeiros distantes. A camada de comunicação é responsável principalmente pela transferência de aglets de um destino para outro. Também suporta a comunicação aglet-aglet e proporciona facilidades para administração de aglets.

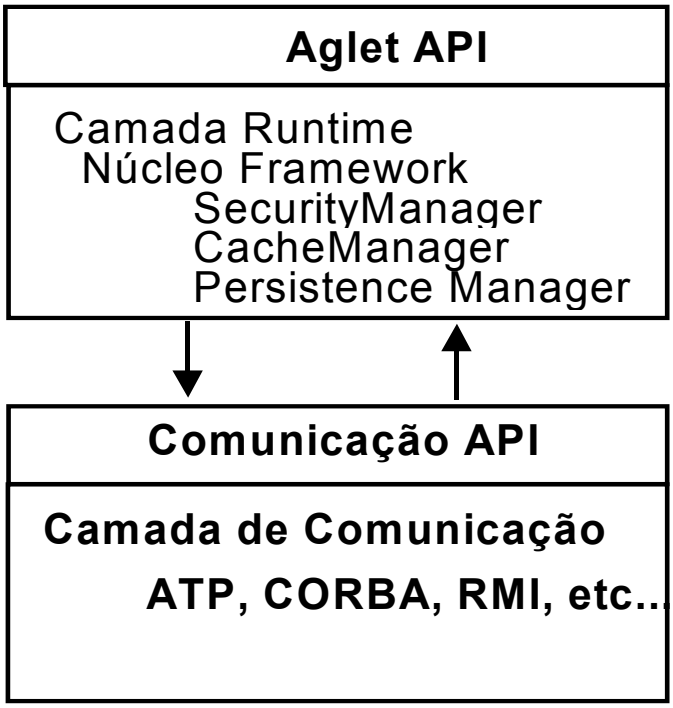

Figura 4.3 – Arquitetura Aglet

4.4.1 A arquitetura da camada de comunicação

A Fig. 4.4 mostra a arquitetura da camada de comunicação onde `com.ibm.maf.MAFAgentSystem` é uma classe abstrata que define um conjunto de métodos equivalentes para a interface MASIF.

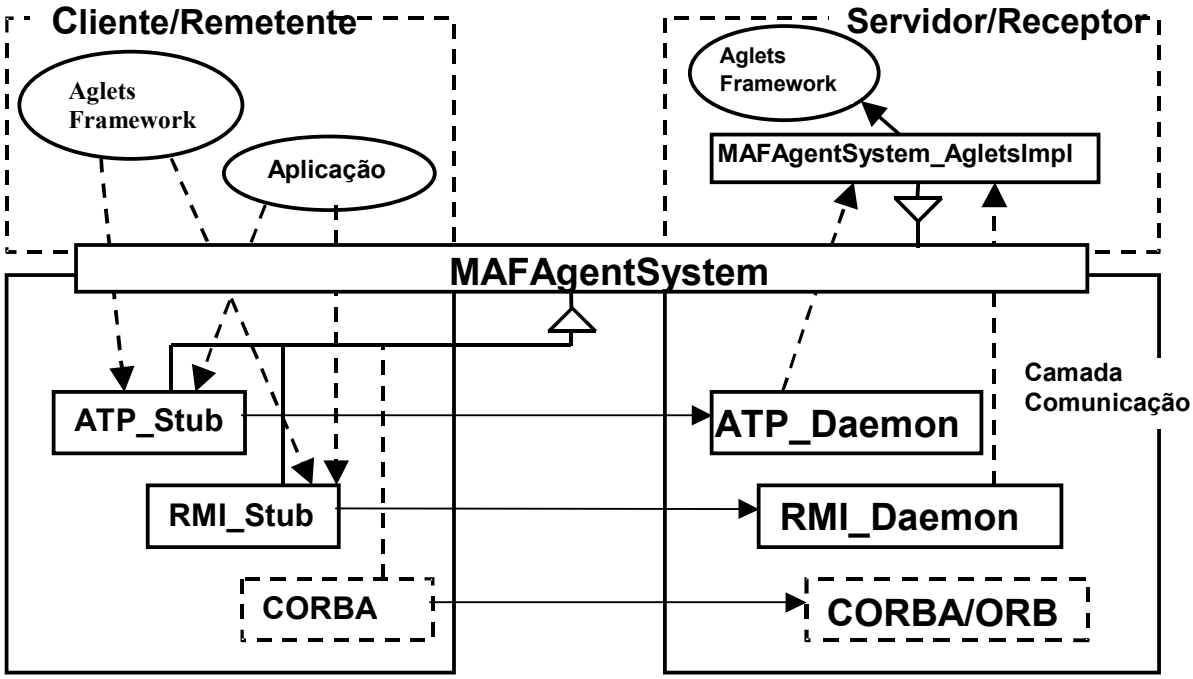

Figura 4.4 – Arquitetura da camada de comunicação [IBM98]

Um detalhamento maior dessa arquitetura foge dos objetivos deste manuscrito. Segue-se apenas alguns comentários.

Um aplicativo ou cliente usa um objeto stub para enviar uma mensagem para um objeto remoto. O sistema deve prover uma classe stub para cada protocolo que ele suporta. A instanciação e gerência de objetos stubs é de responsabilidade do sistema. A versão 1.1 Beta 3 da IBM suporta os protocolos ATP e RMI.

Por outro lado, um servidor aglet pode prover uma realização para a interface `MAFAgentSystem` que realmente responde os pedidos. A classe `com.ibm.aglets.MAFAgentSystem_AgletsImpl` provê uma realização para a interface Aglet.

4.4.2 A Estrutura de objetos aglets

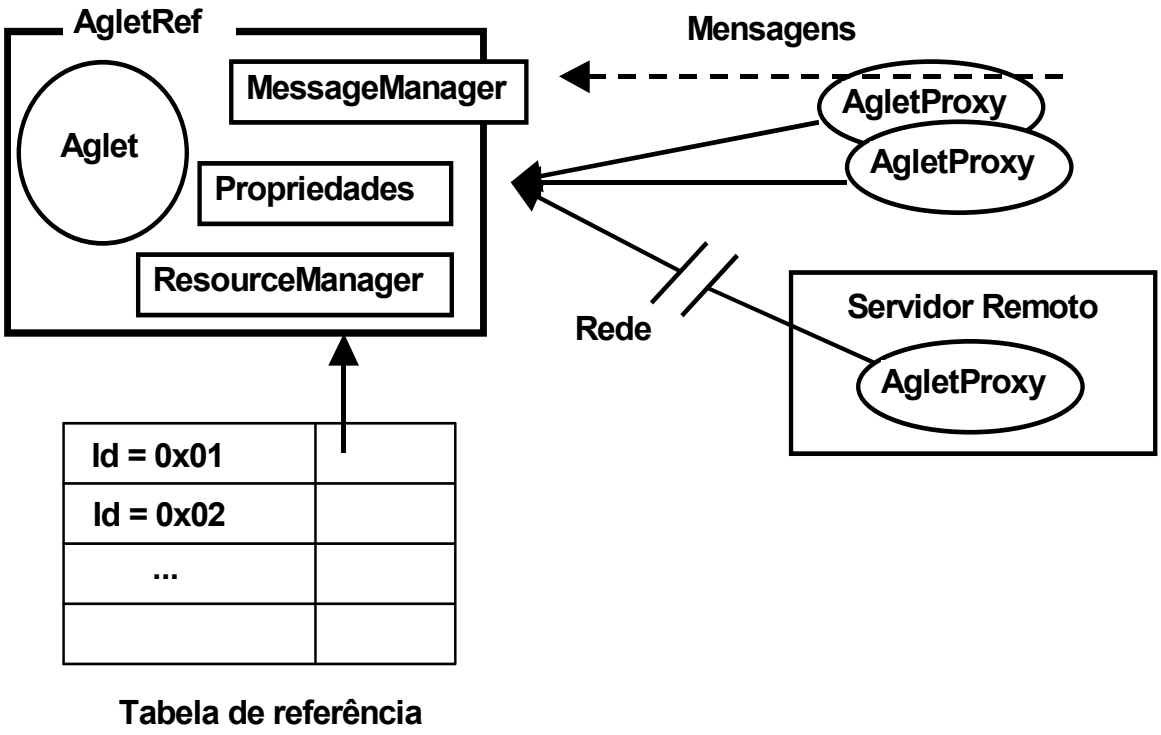

Figura 4.5 – Estrutura do Objeto Aglet. [IBM98]

Um objeto `AgletRef` é a representação interna de um objeto aglet. Tem todos os componentes necessários para o aglet funcionar. Por exemplo, um `MessageManager` para controlar mensagens que chegam e um `ResourceManager` para administrar recursos utilizados pelo aglet e para administrar recursos relacionados com informações de segurança e com o objeto AgletInfo. Ele realiza a maioria das funcionalidades definidas na classe `com.ibm.aglet.Aglet`.

4.5 O protocolo ATP – Agent Transfer Protocol

Um pedido ATP compreende uma linha de pedido, um campo de cabeçalho e um conteúdo. A linha de pedido especifica o método do pedido enquanto que o campo de cabeçalho contém os parâmetros do pedido. O Protocolo ATP define os seguintes métodos padrões de pedidos:

Dispatch

O método `Dispatch` pede a um sistema de agente destino para reconstruir um agente e iniciar sua execução. Se o pedido foi bem sucedido, o remetente deve terminar o agente e liberar os recursos por ele utilizados.

Retract

O método `Retract` pede a um sistema de agente destino para enviar de volta ao remetente um agente especificado. O receptor é responsável por reconstruir e reiniciar o agente. Se o agente for transferido com sucesso, o receptor deve terminar o agente e liberar quaisquer recursos utilizados por ele.

Fetch

O método `Fetch` é semelhante ao método GET em HTTP; ele pede a um receptor para receber e enviar qualquer informação identificada (normalmente, bytecode de classes).

Message

O método `Message` é usado para passar uma mensagem a um agente identificado por um agente-id (que é a identificação do agente dentro do contexto) e para retornar uma resposta. Embora o protocolo adote uma forma

request/reply, ele não anuncia quaisquer regras para o esquema de comunicação entre agentes.

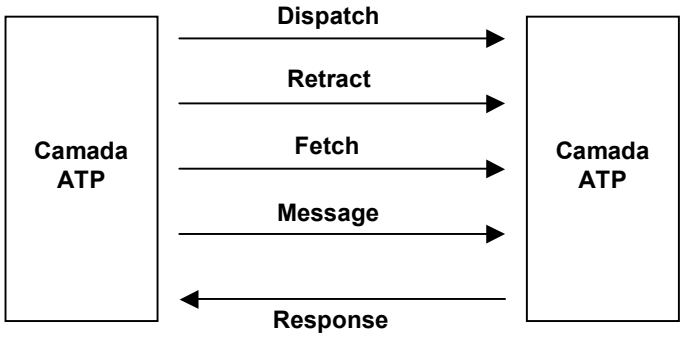

Figura 4.6 – Protocolo ATP. [IBM98]

4.6 Conclusão

Neste capítulo foram apresentadas noções básicas da arquitetura dos aglets. Isto é necessário para o próximo capítulo onde os aglets foram utilizados para a elaboração de um protótipo de aplicação distribuída.

Capítulo V

Um Sistema utilizando agentes móveis para busca e reserva de imóveis

5.1 Introdução

Neste capítulo será demonstrado o desenvolvimento de um protótipo de um estudo de caso envolvendo agentes móveis. Trata-se de um sistema que permite a busca e reserva de imóveis através da internet.

O objetivo é mostrar que agentes móveis representam uma ferramenta viável para a integração de sistemas em ambiente distribuídos.

5.2 Análise

A busca de um imóvel para locação responde a uma necessidade bastante banalizada nos nossos dias, principalmente, neste nosso mundo cada vez mais populado por nômades. O futuro inquilino tenta escolher, entre várias opções, aquela que seria a mais adequada para ele. Este procedimento é extremamente falho e quase que totalmente dependente de sorte já que a maneira como isso é feito (jornal ou visita a imobiliárias) depende do cliente estar na hora certa no local certo. A sua presença física é imprescindível para o processo de escolha acarretando demora e, portanto, perda de oportunidades.

Uma maneira interessante de resolver esta questão seria uma pesquisa na base de dados da imobiliária. Mas esta é uma solução no mínimo simplista ou um tapa buraco para um problema mais complexo. Esta solução seria parcialmente satisfatória se o inquilino morasse na mesma cidade onde está o imóvel e a imobiliária responsável pelo mesmo.

O problema poderia entretanto, ser mais complexo. Por exemplo, um cliente de uma cidade do interior procura uma casa para passar as férias no litoral. Informações como data de disponibilidade do imóvel e preço seriam mais suscetíveis a mudanças devido a uma gama maior de variáveis (condições do tempo, demanda, etc).

Isto posto, este estudo de caso procura demonstrar um protótipo para a solução deste problema, utilizando-se de uma arquitetura baseada em agentes móveis.

5.2.1 Descrição do Sistema

Nome do Sistema

Imobiliária Virtual

Descrição do sistema

O sistema deverá controlar a pesquisa, a reserva, a liberação das reservas e a locação de imóveis de uma rede de imobiliárias interligadas (não necessariamente matriz e filiais), que atua em várias cidades, sendo que cada imobiliária é responsável por todos os imóveis de clientes localizados em sua região.

Cada imobiliária é responsável, também, pela inclusão, exclusão e modificação de dados dos imóveis de seus clientes, podendo utilizar para isso tipos de bancos de dados diferentes.

A aplicação deverá permitir que sejam feitas pesquisas, reservas, liberação de reservas e locações não importando onde os futuros clientes e nem os imóveis estejam.

Fatores de Riscos

- A rede pode estar fora do ar, o que acarretaria à impossibilidade, naquele momento, da localidade com problemas participar do processo solicitado.

Descrição dos atores

Cliente – Pessoa que está à procura de um imóvel.

Controle de Imóvel – Sistema que controla todas as inclusões, exclusões e reservas, cada imobiliária possui seu próprio sistema e método de acesso às bases de dados.

Aglet – Sistema responsável pela mobilidade dos agentes entre as imobiliárias.

5.2.2 Diagrama de Caso de Uso

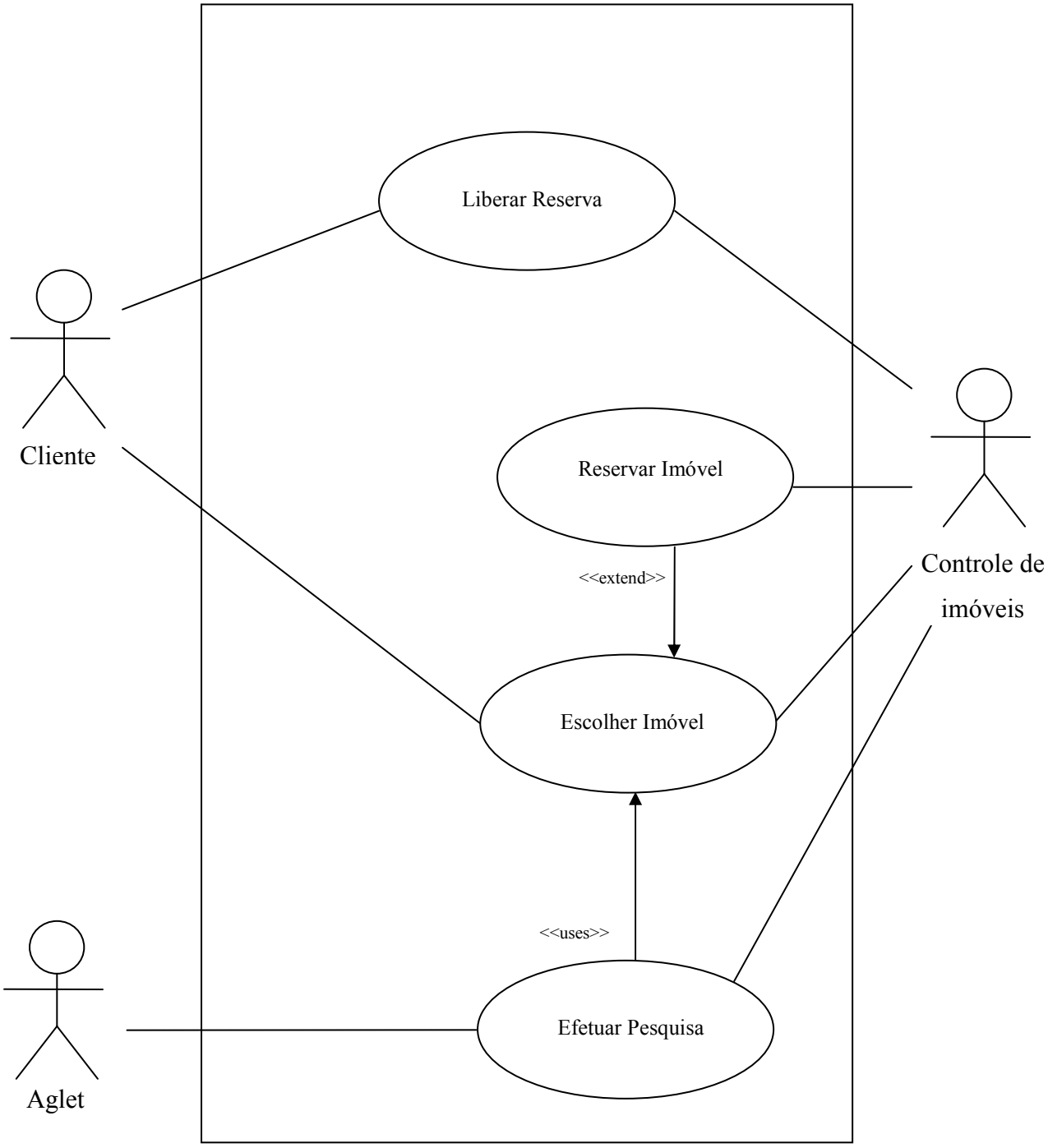

Figura 5.1 – Diagrama de caso de uso

5.2.2 Descrição detalhada dos casos de uso

Escolher Imóvel

Este caso descreve o processo pelo qual o usuário pesquisa um tipo de imóvel nas imobiliárias da rede.

Atores

- Cliente

Fluxo de Eventos

Fluxo básico

1. O case inicia quando o servidor da interface usuário do sítio onde a pesquisa será efetuada for colocado no ar.
2. O cliente preenche os campos Nome e CPF.
3. O sistema irá mostrar a tela de Pesquisa.
4. O cliente preenche o corpo palavra chave para pesquisa.
5. O cliente clica no botão Pesquisar.
6. O sistema cria os aglets necessários, passando-lhes o destino, a referência para o Aglet de Comunicação e o texto para a busca.
7. Cada aglet vai se despachar para a localidade (imobiliária) que lhe foi passada como destino.
8. O sistema vai disponibilizando as informações sobre os imóveis encontrados assim que as recebe.
9. Mediante a lista retornada o usuário irá escolher o imóvel e o local desejados e clicar no botão RESERVAR.
10. O sistema despacha o imóvel desejado, o nome e CPF do cliente em forma de mensagem para o aglet que se encontra no hospedeiro que hospeda a imobiliária respectiva ao imóvel desejado que efetua a Reserva e fim do caso.

Cenários Secundários

- Determinadas imobiliárias podem estar fora do ar, com isso teremos um menor número de opções, porém sem comprometer o resultado de busca nas outras imobiliárias.
- O usuário pode sair do sistema antes de clicar o botão Reservar.

Efetuar Pesquisa

Este caso descreve o processo pelo qual o sistema cria e mantém o ambiente da imobiliária na qual torna possível ao aglet a solicitação de serviços (reservas e liberação de reservas) e também acesso direto aos imóveis instanciados por ela.

Atores

- Controle de Imóveis

Fluxo de Eventos

Fluxo básico

1. Este caso inicia quando o aglet chega ao sítio.
2. O ambiente imobiliária reconhece o aglet e o mesmo é inserido no ambiente.
3. O aglet solicita a listagem de imóveis que atende a busca digitada pelo usuário.
4. O ambiente imobiliária fornece uma lista com todos os imóveis que atendem à busca, instanciando-os e deixando-os disponíveis para que se efetue operações diretamente sobre eles.
5. O aglet repassa as informações (ID do aglet, URL do hospedeiro e nome da imobiliária) para o aglet de comunicação do sítio onde a busca foi efetuada.

Cenários Secundários

- O sítio que solicitou as informações pode estar fora do ar para receber as respostas.

Reservar Imóvel

Este caso descreve o processo pelo qual o cliente reserva determinado imóvel em qualquer imobiliária da rede.

Atores

- Controle de Imóveis

Casos Estendidos

- Escolher Imóvel

Fluxo de Eventos

Fluxo básico

1. O caso inicia quando o aglet recebe a mensagem de solicitação de reserva
2. O aglet solicita ao Controle de Imóveis o serviço de reserva.
3. O aglet envia uma mensagem para o Aglet de Comunicação, devolvendo o status do pedido efetuado pelo cliente.
4. O Aglet de Comunicação repassa para a interface o resultado e esta informa ao cliente e fim do caso.

Cenários Secundários

- O imóvel já foi locado.

Liberar Reserva

Este caso descreve o processo pelo qual o cliente libera a sua reserva de um determinado imóvel por ter locado outro ou desistido da locação.

Atores

- Cliente
- Controle de Imóveis

Fluxo de Eventos

Fluxo básico

1. O caso inicia quando o cliente seleciona de uma lista de imóveis escolhidos anteriormente qual o imóvel que ele irá liberar a reserva efetuada por ele.
2. O cliente clica no botão Liberar Reserva.
3. O Aglet de Comunicação envia uma mensagem para o aglet responsável pela reserva do imóvel escolhido determinando sua destruição.
4. O aglet envia resposta para o Aglet de Comunicação, se retira da lista de reserva e se destrói.
5. O Aglet de Comunicação repassa para a interface o resultado, esta informa ao cliente e fim do caso.

5.3 Arquitetura do sistema

A seguir é apresentada a forma como o sistema e, por conseqüência, a mobilidade dos agentes, vai funcionar. Esta arquitetura é dividida em vários subsistemas, Fig. 5.2 , cada um responsável por uma parte da aplicação.

5.3.1 Estrutura

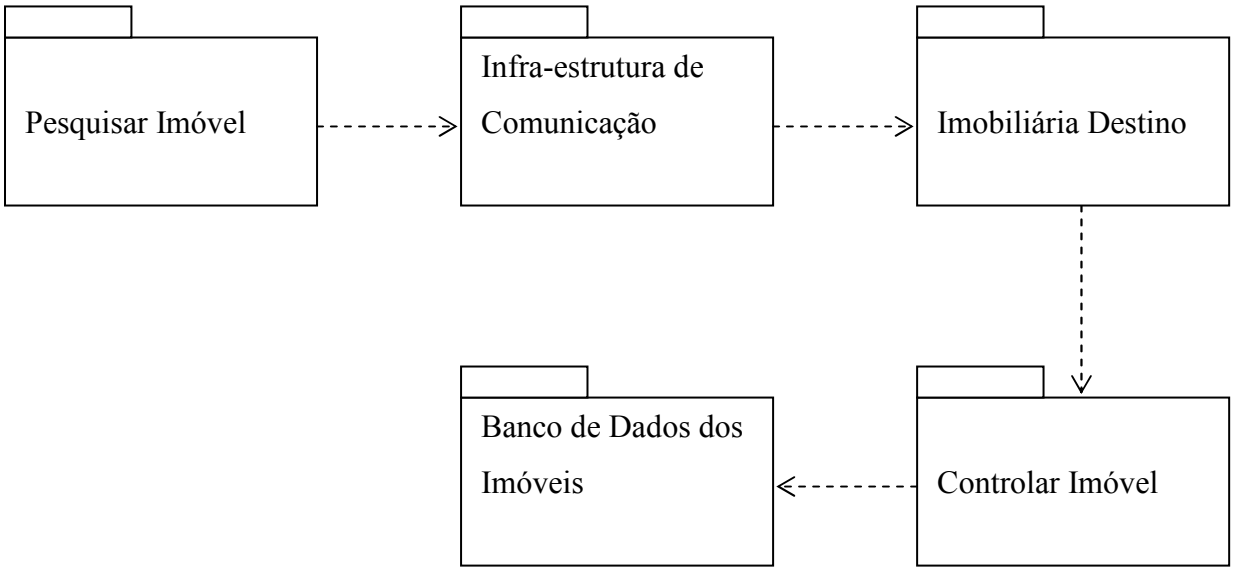

Figura 5.2 – Subsistemas.

Pesquisar Material

Este subsistema é responsável por obter uma lista de imóveis das imobiliárias que satisfaça as condições de busca fornecida pelo cliente.

Infra-estrutura de Comunicação

Este subsistema é responsável pela comunicação entre os aglets residindo nas localidades do cliente e do imóvel. No projeto do sistema seção 5.4, este subsistema é realizado por um agente fixo localizado em cada imobiliária do sistema.

Imobiliária Destino

Este subsistema permite com que o aglet enviado possa requisitar lista, reserva e liberação de reserva de imóveis.

Controlar Material

Este subsistema controla as reservas e liberação de reservas feitas no Imóveis.

Banco de Dados dos Imóveis

Banco de dados que contém informações sobre os imóveis.

5.3.2 Comportamento

Pesquisa de imóveis

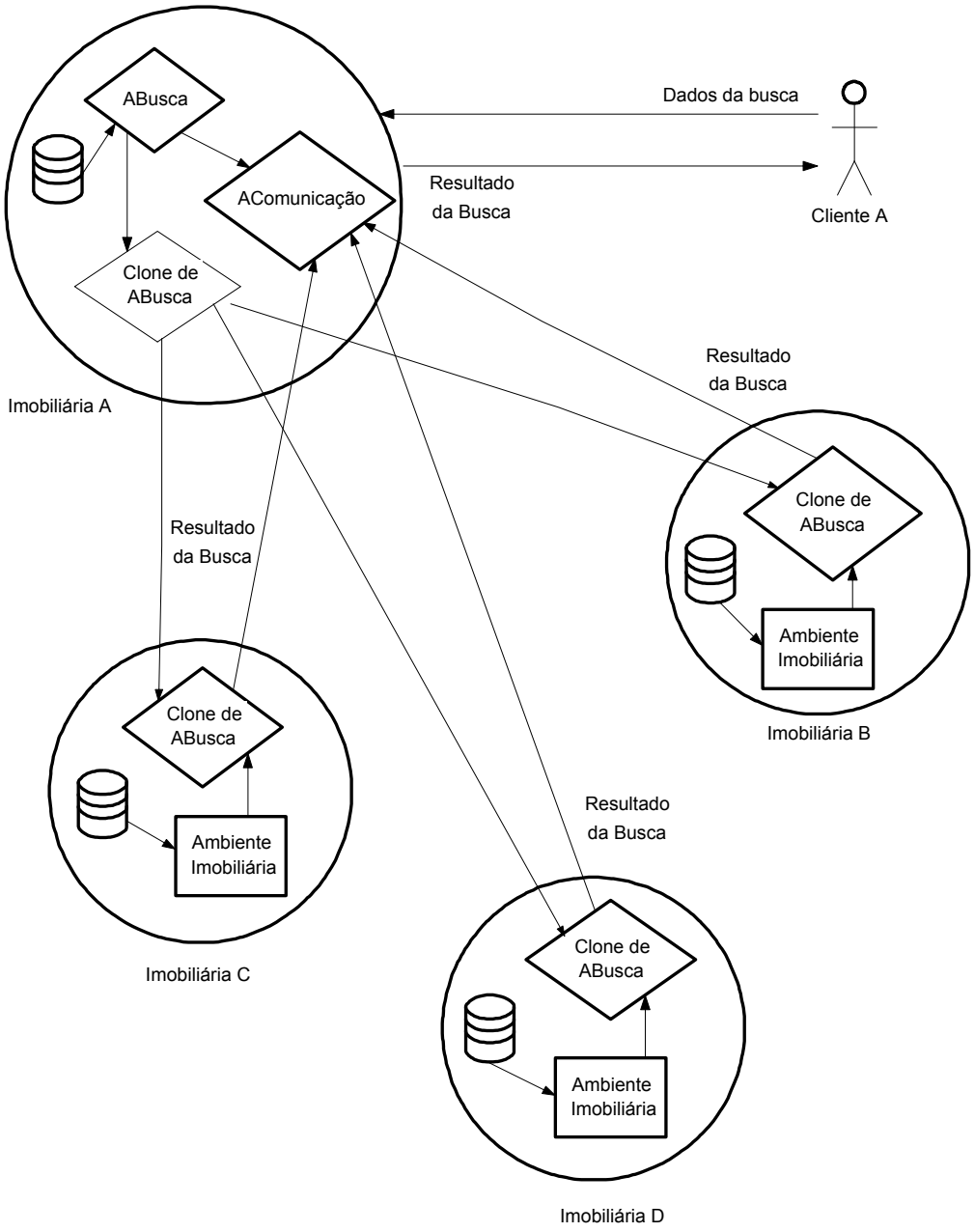

Figura 5.3 – Pesquisa de imóveis.

O diagrama da Fig. 5.3 ilustra o comportamento associado à pesquisa de imóveis onde um cliente, através de uma interface, preenche as especificações do imóvel desejado. O sistema cria o aglet ABusca que é clonado e cada clone se move para os

sítios especificados para efetuarem a busca do imóvel desejado. Lá chegando eles interagem com o ambiente imobiliária que torna possível a pesquisa. De posse dos resultados o aglet remete de volta os resultados para o sítio de origem e este por sua vez disponibiliza ao cliente.

Reserva de imóveis

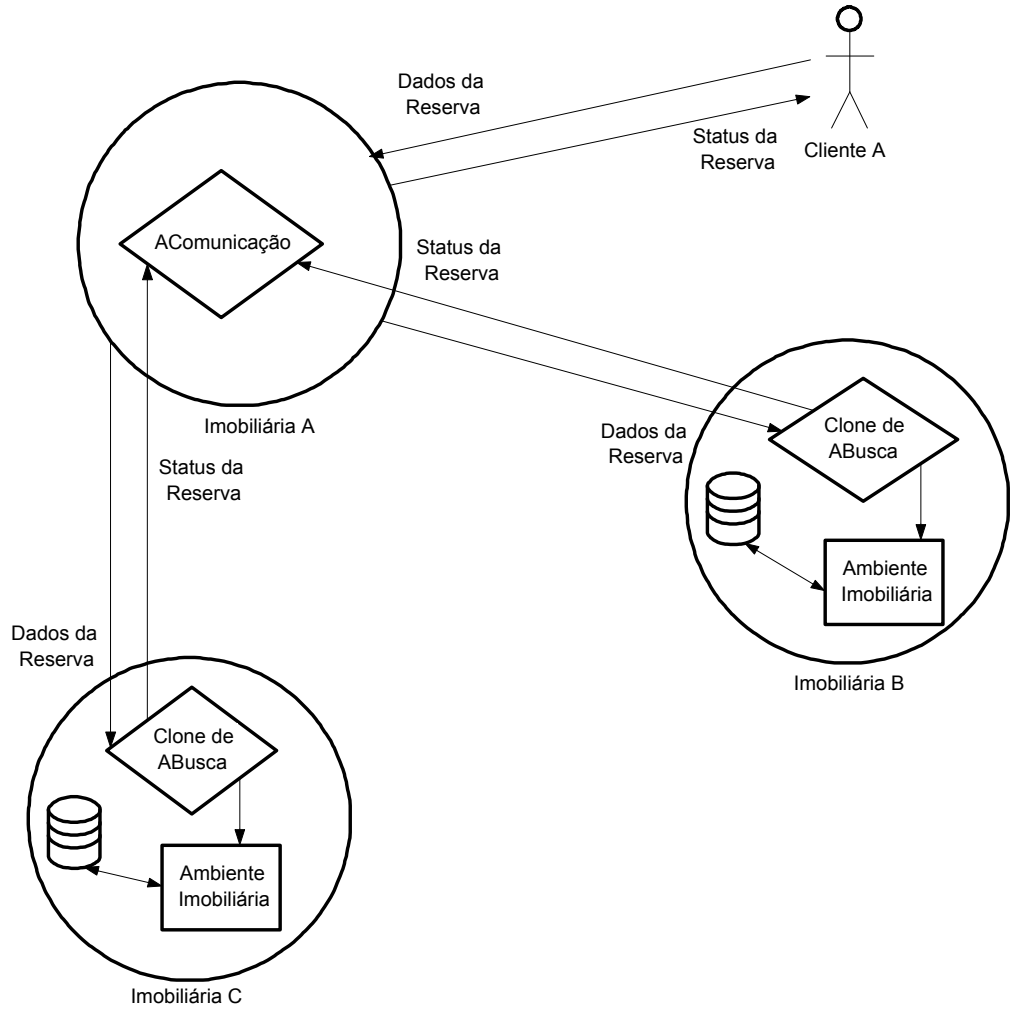

Figura 5.4 – Reserva de imóveis.

No comportamento associado a reserva de imóveis da Fig. 5.4 o cliente passa para a interface cliente qual o imóvel que ele deseja. Uma mensagem contendo a reserva do cliente é passada para o aglet ABusca que continua no sítio onde foi localizado os imóveis. Este devolve para o aglet Acomunicação o resultado da reserva que por sua vez repassa para a interface cliente para que seja divulgado ao cliente.

Liberar reserva de imóveis

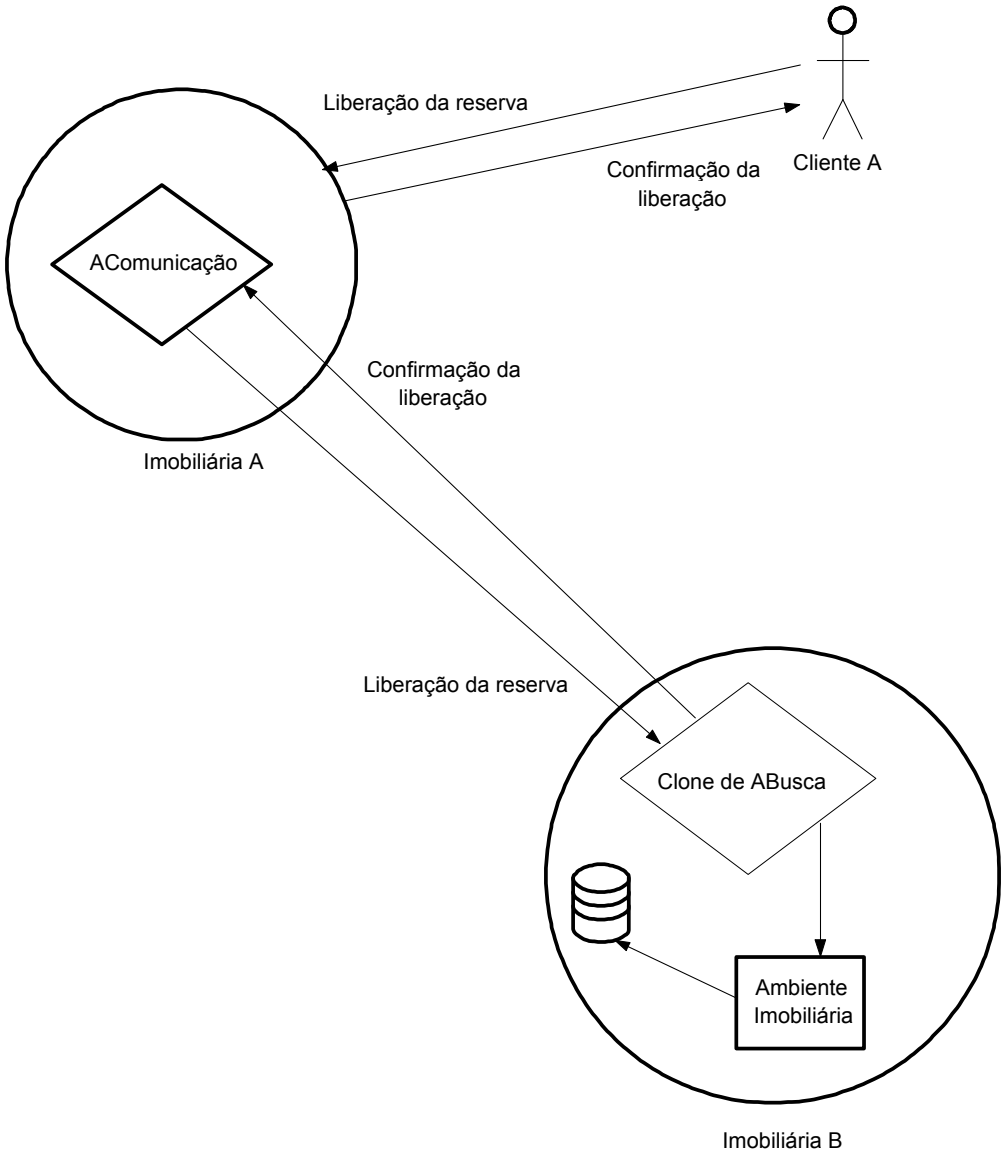

Figura 5.5 – Liberar reserva de imóveis.

Quando um cliente não deseja mais um imóvel ou se já efetuou a locação de outro imóvel deve-se liberar a sua reserva. A Fig. 5.5 ilustra esse comportamento. O cliente escolhe, na interface cliente, qual o imóvel que ele deseja liberar. Uma mensagem contendo a liberação da reserva do cliente é enviada, através do aglet Acomunicação, para o aglet ABusca que continua no sítio onde foi localizado os imóveis. Este devolve para o aglet Acomunicação a confirmação da liberação e se destrói. O aglet Acomunicação repassa para a interface cliente para divulgação ao cliente.

5.4 Projeto

O protótipo do sistema de Imobiliária foi desenvolvido utilizando a plataforma Java 2 da Sun Microsystem e os aglets versão 1.1 Beta 3 da IBM que seria o ambiente para programação de agentes móveis em Java. Este protótipo acessa algumas bases de dados que foram criadas, por comodidade, não necessidade, através do Access 2000 da Microsoft. Em uma situação real o sistema poderia trabalhar, sem problemas, com diversos tipos de banco de dados e seus acessos.

5.4.1 Organização da aplicação

A aplicação está organizada em servidores de contexto, interface gráfica e aglets. A Fig. 5.6 apresenta o diagrama de classes que demonstra esta organização e seus relacionamentos.

Temos então, distribuídas em todas as imobiliárias participantes do sistema, um servidor de contexto responsável por abrigar todos os aglets que chegam de outros sítios.

Para se fazer uma pesquisa de imóvel é necessário uma interface gráfica `GUI_Imobiliária`. Esta interface, além de tornar possível a inserção de dados no sistema, instancia um aglet de comunicação – `AComunicacao` – que funcionará como uma espécie de antena e servirá de referência para os aglets que forem criados e enviados a outros sítios. Desta forma será possível, por exemplo, que os aglets de busca – `ABusca` – possam enviar mensagens ao cliente que encontra-se no sítio de origem da pesquisa.

É a interface `GUI_Imobiliária` que requisita ao cliente seu nome e o número do CPF (cf. Fig 5.7). É efetuado uma validação do CPF através do cálculo do dígito verificador. Ocorrendo a tentativa de preenchimento de um CPF inexistente, o sistema emite uma mensagem (cf. Fig. 5.8). Uma vez disponibilizado esses dados corretamente, a interface permite o preenchimento do campo de características do imóvel para se efetuar a busca (cf. Fig 5.9).

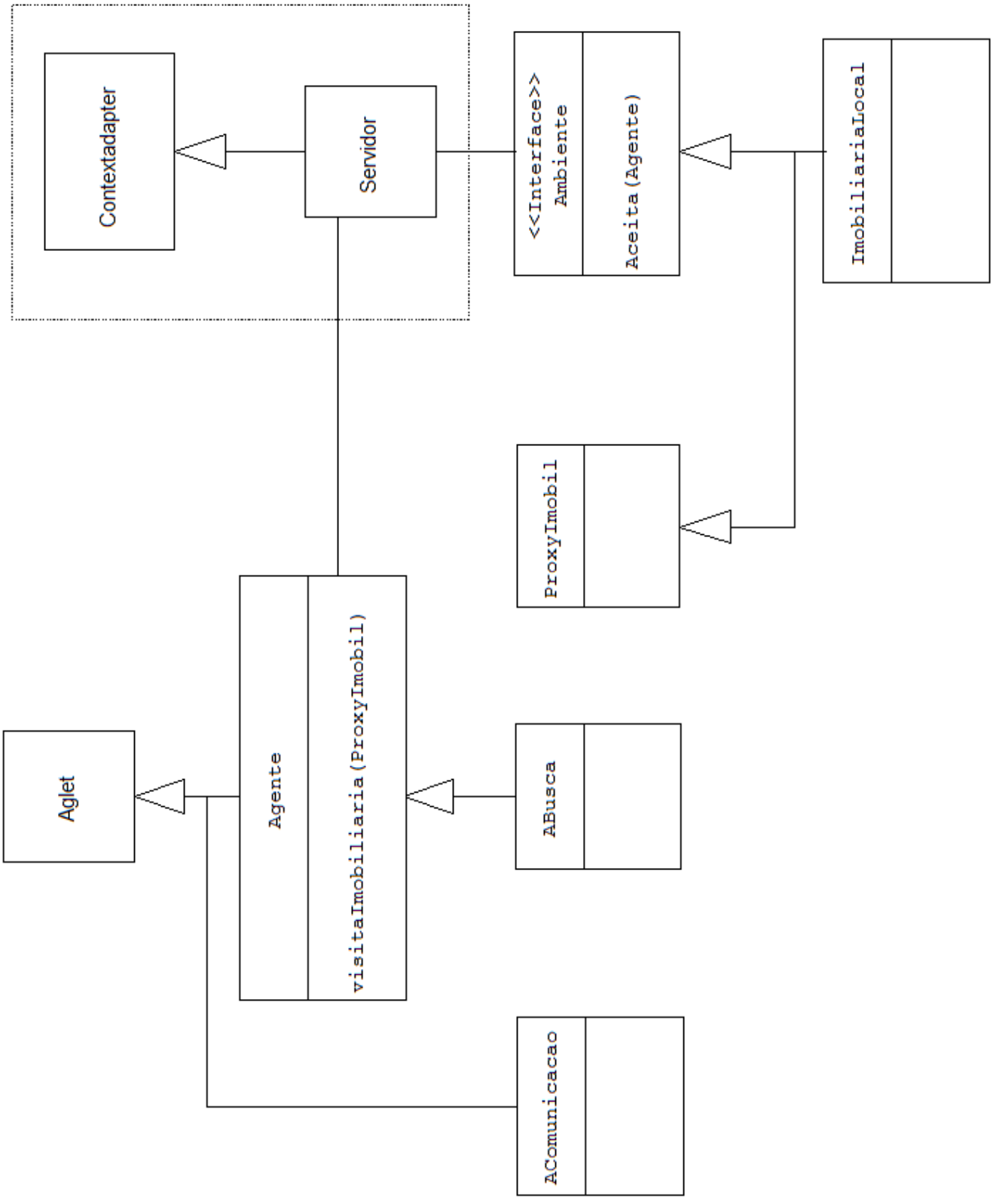

Figura 5.6 – Diagrama de Classes.

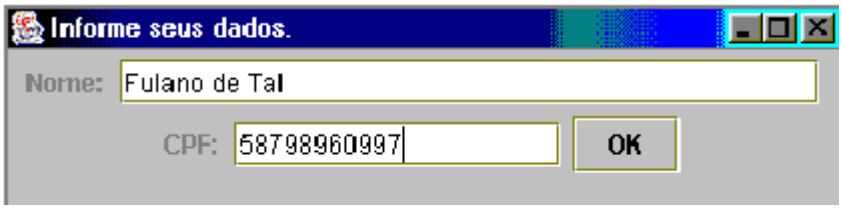

Figura 5.7 – Interface da Imobiliária – informações do cliente.

Figura 5.8 – Interface da Imobiliária – CPF inválido.

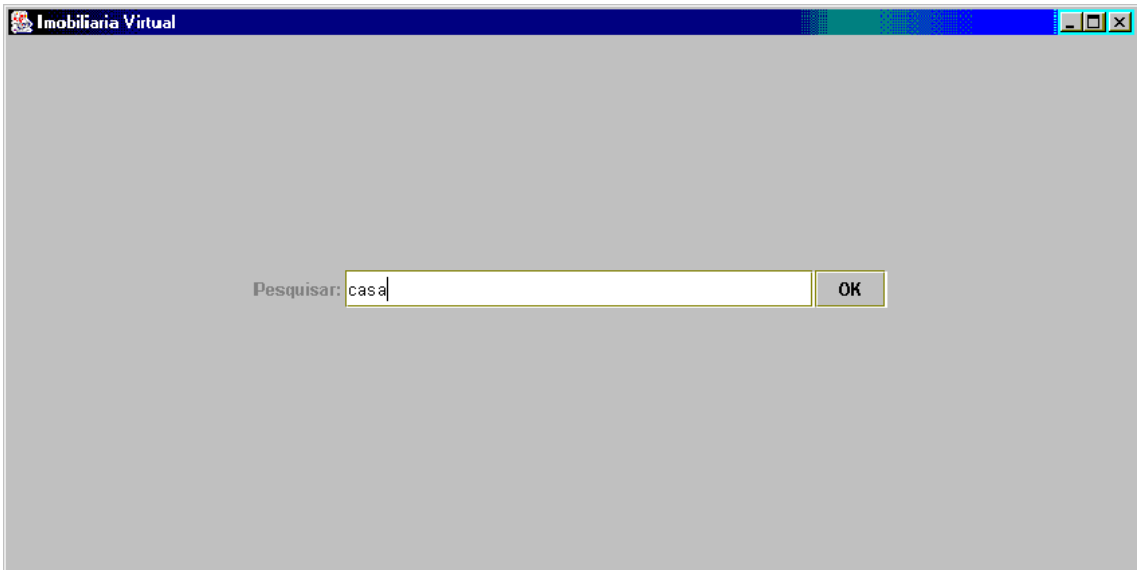

Figura 5.9 – Interface da Imobiliária – Pesquisar.

Como não faz parte do escopo deste trabalho, a formatação dos dados necessários para a pesquisa dos imóveis foi simplificada. Para um funcionamento otimizado seria necessário mudar a interface e aumentar a descrição dos imóveis. Nesse ponto pode-se ainda agregar um sistema especialista para a escolha dos imóveis.

A interface gráfica `GUI_Imobiliária` de posse das informações do tipo de imóvel desejado pelo cliente, consulta uma base local para verificar as imobiliárias que compõem a rede de imobiliárias e seus respectivos endereços e envia, para cada uma delas uma instância do aglet `ABusca` que leva consigo o endereço de destino, a string de busca e o endereço do `AComunicação` do sítio onde a pesquisa foi iniciada.

Chegando ao seu destino, o aglet `ABusca` é recebido pelo servidor de contexto local que providencia a sua inserção no ambiente `ImobiliariaLocal`. Uma vez aceito e inserido no ambiente, o aglet solicita o serviço de instanciação de imóveis que condizem com a string de busca e recebe uma lista dos imóveis encontrados.

De posse da lista, o aglet `ABusca` envia mensagens para o objeto `AComunicacao` do sítio onde a busca se originou contendo, ID do imóvel, sua descrição, a próxima data em que o imóvel estará disponível e o endereço do `ABusca` responsável pela informação.

No sítio de origem das pesquisas, o objeto `AComunicacao` recebe as mensagens de todos os aglets `ABusca` e as entrega para a interface `GUI_Imobiliária` que mostra os imóveis retornados para o usuário (Fig. 5.10).

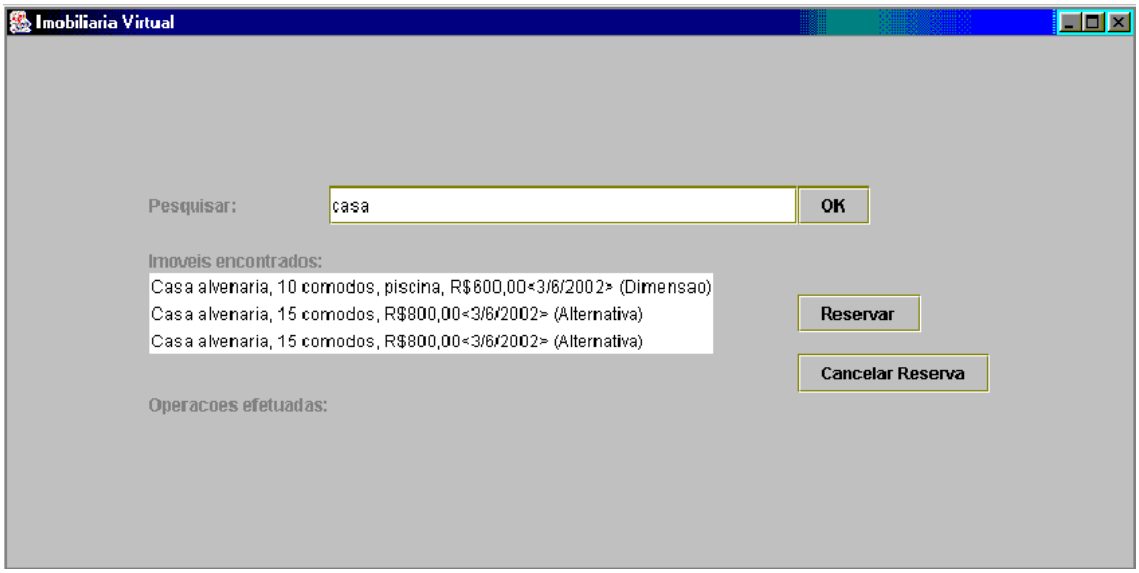

Figura 5.10 – Interface da Imobiliária – Retorno da pesquisa.

O Cliente faz a(s) escolha(s) desejada(s) e solicita a reserva – botão reservar da Fig. 5.10 – e a interface envia uma mensagem diretamente para o `ABusca` responsável pela informação do imóvel escolhido.

O `ABusca`, recebendo a mensagem, identifica o imóvel escolhido por seu ID e reserva o imóvel registrando o CPF do cliente interessado na base de reservas.

O cliente também poderá escolher a opção de cancelar uma solicitação de reserva efetuada enviando uma mensagem específica para o `ABusca` que irá providenciar o cancelamento da reserva (Fig. 5.11).

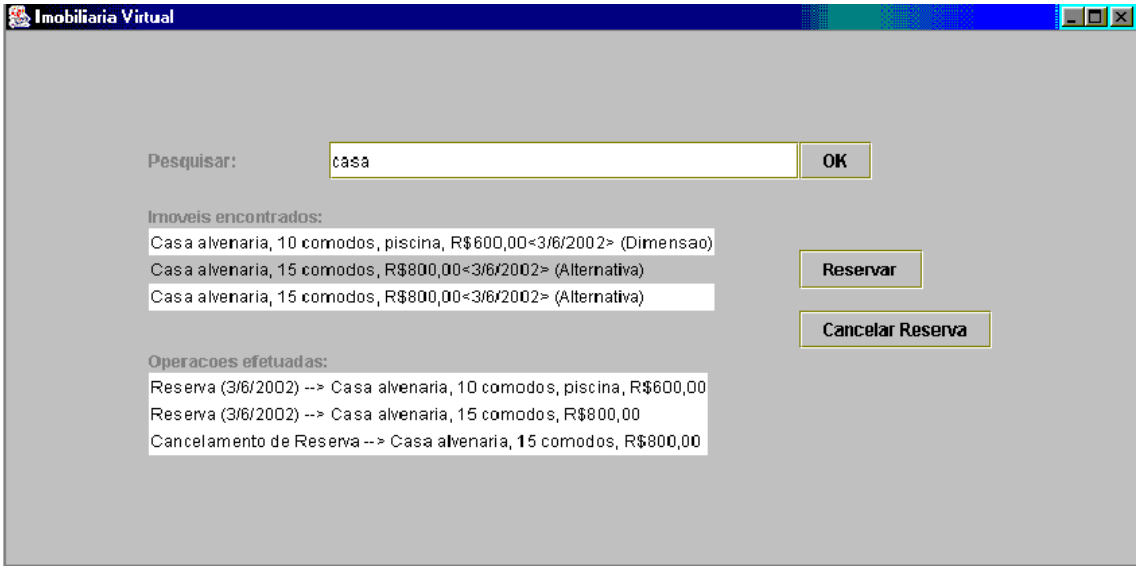

Figura 5.11 – Interface da Imobiliária – Cancelamento de reserva.

5.4.2 Particularidades do protótipo

A formatação dos dados necessários para a pesquisa dos imóveis foi simplificada. Para um funcionamento otimizado seria necessário mudar a interface e aumentar a descrição dos imóveis. Nesse ponto pode-se ainda agregar um sistema especialista para a escolha dos imóveis.

Parte-se do princípio de que as imobiliárias terão suas próprias bases de dados e, assim sendo, caberá a elas que as informações necessárias para que os agentes móveis possam realizar suas tarefas estejam disponibilizadas. Desta forma apenas os servidores de cada imobiliária terão acesso direto às bases de dados para cadastrar novos imóveis, alterar dados dos já cadastrados, e realizar outros serviços pertinentes.

5.5 Conclusão

Foi ilustrado neste capítulo o protótipo de um sistema de imobiliária para mostrar a viabilidade da utilização dos agentes móveis, como ferramenta, na integração de sistemas em ambientes distribuídos.

Capítulo VI

Conclusão

Este trabalho abordou a utilização do paradígma de softwares baseados em agentes como uma forma viável de desenvolvimento de sistemas complexos, utilizando agentes móveis como uma ferramenta de integração entre os sistemas distribuídos ao longo da rede. Vimos que os agentes fazem parte de um contexto de computação distribuída maior e que a mobilidade pode ser vista como ponto chave na solução de diferentes problemas que ocorrem em ambientes distribuídos.

Para dar ênfase a este tipo de paradígma, foi desenvolvido o protótipo de um sistema de busca e reserva de imóveis utilizando a Internet como meio de acesso. Para isso, foram utilizados a plataforma Java 2 da Sun Microsystem e os aglets versão 1.1 Beta 3 da IBM que seria o ambiente para programação de agentes móveis em Java. As bases de dados acessadas neste sistema foram do tipo Access 2000 da Microsoft mas somente por comodidade e sem prejuízo da funcionalidade já que, pelas características dos aglets, os tipos do banco de dados acessados seria indiferente, mesmo em sistemas heterogêneos. Aglets fazem isto de modo nativo enquanto outras formas de soluções a fazem de modo muito mais complexo.

Na análise do sistema desenvolvido, utilizou-se uma adaptação da UML como forma de representação da modelagem do sistema. Algumas das formas de representações foram repensadas para poderem melhor exprimir o conceito de mobilidade dos agentes.

Como resultados observados poderemos destacar os seguintes:

- Através de simulações no protótipo foram verificadas que todas as transações foram efetuadas e que, desta forma, podemos concluir que o modelo reagiu adequadamente a esse novo paradigma de desenvolvimento sugerindo a viabilidade do mesmo.

- A problema da complexidade do tipo de sistema/situação escolhida foi solucionada de forma mais natural do que as soluções convencionais.

Este trabalho apresenta muitas possibilidades de continuação como por exemplo:

- Desenvolvimento de uma metodologia de desenvolvimento que possa melhor refletir os aspectos de mobilidade dos agentes.

- Aspectos sobre segurança poderão ter uma investigação mais profunda, pois a mobilidade traz novos desafios ao projeto de sistemas seguros.

- Estudo de novas técnicas de cooperação entre agentes.

Referência

[CAR99] Luca Cardelli, “Abstractions for Mobile Computation”, Microsoft Research, http://www.research.digital.com/SRC/personal/Luca_Cardelli/ (Jun/2000) , 1999.

[JF98] Jim Farley, “Java Distributed Computing “, O´REILLY, 1998.

[FG96] Franklin, Stan & Graesser, Art – Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. 1.996. Obtido do endereço eletrônico: http://www.msci.memphis.edu/~franklin/AgentProg.html em 15/06/2000.

[RO98] Robert Orlafi Dan Harkey, “Client/Server Programing with Java and ORBA”, John Wiley & Sons,Inc, Second Edition, 1998.

[INF96] A Caminho da Linguagem Distribuída. Informática Exame. Junho de 1996 p. 94-98

[REN94] RENAUD, Paul. E. Introdução aos sistemas Cliente/Servidor: Guia Prático para profissionais de sistemas. Rio de Janeiro: Infobook, 1994

[JNR&WM02] JENNINGS Nicholas R. & WOOLDRIDGE Michael. Agent-Oriented Software Engineering. 2002

[IBM98] Fiji Kit documentation. Obtido do endereço eletrônico: http:// aglets.trl.ibm.co.jp/infrastructure/fiji/fiji.html em 09/04/2001

[HAR96] Harrison, C.; Chess, D.; Kershenbaum, A. Mobile agents: are they a good idea? 1996

[LDB&OSH98] LANGE, Danny B. & OSHIMA, Mitsuru, Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley,1998.

[LDA98] Lange, Danny B. – Mobile Objects and Mobile Agents: The Future of Distributed Computing? Obtido do endereço eletrônico: http://www.genmagic.com /asa/danny/Ecoop98.pdf em 20/01/2001.