Um Modelo para Portais Móveis baseado em Middleware Reflexivo e Agentes Móveis

Marcos Vinícius Gialdi

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Um Modelo para Portais Móveis baseado em Middleware Reflexivo e Agentes Móveis

Marcos Vinícius Gialdi[1]

Fevereiro de 2004

Banca Examinadora:

- Edmundo R. M. Madeira
  IC - UNICAMP (Orientador)
- Fabio Moreira Costa
  INF - UFG
- Maria Beatriz Felgar de Toledo
  IC - UNICAMP
- Paulo Lício de Geus
  IC - UNICAMP (Suplente)

[1]**Trabalho com suporte parcial do CNPq, processo 133199/2002-9**

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Gialdi, Marcos Vinícius

G347m Um modelo para portais móveis baseado em middleware reflexivo e agentes móveis / Marcos Vinícius Gialdi -- Campinas, [S.P. :s.n.], 2004.

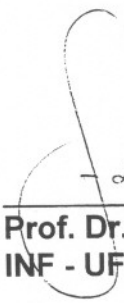
Orientador : Edmundo Roberto Mauro Madeira

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

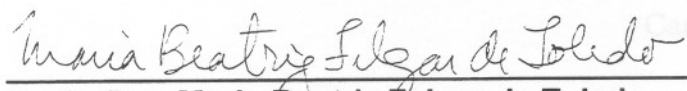
1. Computação móvel. 2. Agentes inteligentes (Software). 3. Processamento distribuído. I. Madeira, Edmundo Roberto Mauro. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

TERMO DE APROVAÇÃO

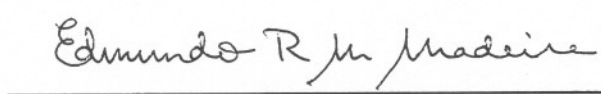
Tese defendida e aprovada em 26 de março de 2004, pela Banca examinadora composta pelos Professores Doutores:



Prof. Dr. Fábio Moreira Costa
INF - UFG



Profa. Dra. Maria Beatriz Felgar de Toledo
IC - UNICAMP



Prof. Dr. Edmundo Roberto Mauro Madeira
IC - UNICAMP

Um Modelo para Portais Móveis baseado em Middleware Reflexivo e Agentes Móveis

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Marcos Vinícius Gialdi e aprovada pela Banca Examinadora.

Campinas, Fevereiro de 2004.

Edmundo R. M. Madeira
IC - UNICAMP (Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Resumo

A Computação Móvel possibilita o acesso a informação de qualquer lugar e a qualquer momento. Contudo, as preferências do usuário e as limitações geradas no ambiente móvel trazem a necessidade de personalizar o acesso aos serviços. Este trabalho descreve o ICoMP (*I-centric Communication Mobile Portal*), um modelo para Portal Móvel que oferece serviços aos usuários baseado no seu perfil (preferências, recursos disponíveis, localização e contexto). O ICoMP baseia-se na abordagem I-centric, onde os sistemas adaptam-se dinamicamente às necessidades do usuário e às mudanças no ambiente móvel. A infra-estrutura do ICoMP integra um *middleware* reflexivo (*ReMMoC*) e uma plataforma de Agentes Móveis (*Grasshopper*) para dar suporte a dois tipos de acesso: a serviços oferecidos por agentes móveis e serviços implementados por tipos de middleware heterogêneos. Além disso, uma avaliação do modelo ICoMP é feita em termos de espaço de armazenamento e no suporte a aplicações móveis.

Abstract

Mobile Computing enables information access anytime, anywhere. However, the needs of users and the constraints generated by the mobile environment raise the necessity to personalize access to services. This work describes ICoMP (I-centric Communication Mobile Portal), which is a mobile portal model that offers services based upon user profiles (these contain information including: preferences, available resources, location and environment context). Notably, ICoMP follows an I-Centric approach, whereby the system dynamically adapts to manage both user requirements and changes in the mobile environment. ICoMP's infrastructure integrates a reflective middleware (ReMMoC) and a mobile agent platform (Grasshopper) to support two types of access to information and application services, i.e. it interoperates with both services offered by mobile agents and also services implemented upon heterogeneous middleware. Moreover, we evaluate our implementation of the ICoMP portal in terms of memory footprint cost and support for mobile applications.

Agradecimentos

A Deus, pelo Dom da vida.

Aos meus pais, pelo eterno apoio, amor e compreensão.

Ao professor Edmundo, pela orientação durante o desenvolvimento deste trabalho.

A minha família, pelo incentivo e carinho.

Aos amigos de mestrado, pelos momentos de alegria e descontração.

Ao CNPq, pelo apoio financeiro.

À Microsoft, por permitir a utilização do laboratório .NET para desenvolvimento deste projeto.

A todos que contribuiram para a realização deste trabalho.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A computação móvel vem possibilitando o acesso sem fio aos serviços de Internet e Telecomunicações. Desta forma, tem-se a possibilidade de acesso a qualquer informação, a qualquer tempo e de qualquer lugar, abrindo a possibilidade para criação de novas tecnologias bem como novos serviços e produtos. Além disso, os dispositivos computacionais móveis com as mais diferentes características estão se tornando populares pelo mundo. Celulares com acesso à Internet, assistentes pessoais (PDAs), *laptops* e vários tipos de *handhelds* proliferam-se em diversos setores e avitividades.

Existem muitas projeções sendo feitas e expectativas sendo criadas com base nestas novas tecnologias. De acordo com o *DC Research* e *The Yankee Group* [1], em 2005 o número de usuários de telefones com acesso à Internet somarão aproximadamente 1 bilhão de pessoas no mundo. Por outro lado, o *Gartner Group* [2] prevê que tecnologias sem fio irão tornar os trabalhadores móveis cerca de 30 vezes mais produtivos. Existem muitos outros números semelhantes a estes, o que parece apontar para, no mínimo, uma grande massificação dos dispositivos móveis.

Contudo, o ambiente móvel traz algumas limitações. Estas limitações vão de dispositivos móveis com CPUs menos poderosas, telas menores, menos memória até redes com largura de banda reduzida e menor estabilidade de conexões.

Logo, as limitações impostas pela computação móvel e os interesses do próprio usuário trazem a necessidade de personalização no acesso aos serviços. Neste caso, é necessário prover infra-estruturas de software apropriadas ao ambiente de computação móvel, onde as aplicações são adaptadas às limitações deste ambiente.

1.1 Objetivos

O objetivo deste trabalho é especificar e desenvolver um modelo para Portais Móveis baseado em *middleware* reflexivo, que é configurável e reconfigurável dinamicamente,

possibilitando uma melhor utilização dos recursos disponíveis no ambiente, e nos serviços oferecidos por uma plataforma de agentes móveis como assincronismo, distribuição do processamento, dentre outros, de grande utilidade dentro de um ambiente móvel [38, 39]. Este modelo é chamado de ICoMP (*I-centric Communication Mobile Portal*), onde Portais Móveis podem oferecer serviços aos usuários baseando-se no seu perfil (preferências, recursos disponíveis, localização, ambiente, dentre outros), isto é, adaptando-se às necessidades do usuário e às mudanças num ambiente móvel. O ICoMP baseia-se na abordagem *I-centric*, onde os sistemas de comunicação se adaptam ao espaço de comunicação, ambiente e situação do indivíduo [8]; na utilização de um *middleware* reflexivo como infra-estrutra base dos provedores de serviços; e numa plataforma de agentes móveis. Desta forma, o ICoMP permite dois tipos de acesso: a informações e serviços oferecidos pelos provedores de serviços, e a serviços oferecidos pela plataforma de agentes móveis como despachar e receber agentes.

1.2 Contribuições

Este trabalho traz como contribuições principais:

- A revisão de modelos e infra-estruturas de software para a computação móvel [3, 4, 5, 6, 7], apontando suas vantagens e desvantagens.
- Um Modelo de Portal Móvel que tem em sua infra-estrutura *middleware* reflexivo e agentes móveis.
- A implementação de um protótipo do Portal Móvel permitindo o acesso a informações e serviços oferecidos pelos provedores de serviços, e a serviços oferecidos pela plataforma de agentes móveis.
- Avaliação de uma aplicação em termos de funcionalidades e espaço de armazenamento.

1.3 Estrutura da dissertação

Este trabalho está dividido da seguinte forma:

- O Capítulo 2 apresenta os conceitos básicos: computação móvel, contexto e I-Centric.
- O Capítulo 3 apresenta as tecnolgias utilizadas para o desenvolvimento deste trabalho como *Middleware* Reflexivo e Agentes Móveis.

- O modelo proposto é apresentado no Capítulo 4.
- O Capítulo 5 aborda questões relativas a implementação e avaliação do modelo.
- Por fim, o Capítulo 6 conclui o trabalho, tecendo comentários finais e propondo algumas extensões.

Capítulo 2

Conceitos Básicos

Neste Capítulo serão abordados os conceitos relativos a Computação Móvel, Contexto e *I-Centric*.

2.1 Computação Móvel

Esta seção define computação móvel, onde ela se aplica bem como suas limitações. Além disso, o termo mobilidade é caracterizado.

2.1.1 Definição

Computação móvel está relacionada à mobilidade de *software*, *hardware* e dados num ambiente computacional. Neste caso, os dispositivos móveis, tais como computadores portáteis e PDAs, têm a capacidade de se comunicar com a parte fixa da rede e possivelmente com outros dispositivos móveis, não tendo uma posição fixa dentro da rede. A computação móvel representa um novo paradigma computacional, ampliando o conceito tradicional de computação distribuída, podendo ser considerada uma configuração de sistema distribuído com computadores móveis.

Além disso, a computação móvel tem como objetivo prover ao usuário acesso permanente a uma rede fixa ou móvel independente de sua posição física, possibilitando acessar informações, aplicações e serviços de qualquer lugar e a qualquer momento

A computação móvel está se tornando uma área madura e parece destinada a se tornar o paradigma computacional dominante. Dispositivos móveis, também chamados genericamente de *handhelds*, estão aparecendo de diversas formas. Por exemplo, PDAs (*Personal Digital Assistants*), telefones celulares e vários outros tipos de dispositivos. Além disso, dispositivos móveis estão sendo fabricados com outras facilidades, funcionalidades e interfaces como GPS (*Global Positioning System*), tocadores de áudio e câmeras fotográficas

digitais, jogos eletrônicos e placas de comunicação sem fio multiprotocolos, que facilitarão a comunicação entre diferentes tipos de dispositivos e infra-estruturas de comunicação. O mercado desses dispositivos está crescendo e sendo usado em aplicações que envolvem negócios, indústria, escolas, hospitais, laser, etc.

Dentre as diversas infra-estruturas de comunicação sem fio existentes, as mais utilizadas são a comunicação celular de segunda geração chamada de 2G (baseada nos padrões TDMA, CDMA e GSM), a geração 2,5 que é uma solução intermediária baseada em comunicação de pacotes, a 3G, que nos próximos anos promete velocidades na faixa de Mbps, redes locais sem fio baseadas no padrão IEEE 802.11, redes pessoais baseadas no padrão Bluetooth e IEEE 802.15 e 802.16, redes de sensores sem fio, e RFID (*Radio Frequency Identification*) [41].

A Figura 2.1 ilustra o cenário da computação móvel.

Figura 2.1: Cenário da Computação Móvel

A arquitetura genérica de uma plataforma móvel consiste em uma arquitetura distribuída, na qual diversos computadores, geralmente conhecidos como Hosts Fixos (FS - *Fixed Hosts*) e Estações de Base (BS - *Base Stations*), são interligados através de uma rede com fio de alta velocidade. Hosts fixos são computadores de finalidade genérica que não são equipados para gerenciar unidades móveis, mas podem ser configurados de forma a fazê-lo. Estações de Base são equipadas com interfaces para as redes sem fio e podem comunicar-se com as unidades móveis para suportar o acesso a dados. Unidades Móveis (*Mobile Units*) são computadores portáteis movidos à bateria, que se movimentam livremente em um domínio geográfico de mobilidade, uma área que é restringida pela limitada amplitude de banda dos canais de comunicacão sem fio. Para gerenciar a mobilidade dessas unidades, o domínio de mobilidade geográfica é dividido em domínios

menores chamados de células. A computação móvel requer que o movimento de unidades seja irrestrito dentro do domínio de mobilidade geográfica (movimento intercelular), e ao mesmo tempo em que possuir acesso contínuo durante o movimento, garante que o movimento de uma unidade móvel através dos limites das células não terá nenhum efeito sobre o processo de recuperação de dados.

Unidades Móveis e Estações de Base se comunicam através de canais sem fio que possuem larguras de banda significativamente menores do que aquelas de uma rede com fio. Um canal de conexão do tipo *downlink* é utilizado para enviar dados das Estações de Base para as Unidades Móveis, e um canal de conexão do tipo *uplink* é utilizado para enviar dados das Unidades Móveis para as Estações de Base.

Como principais requisitos impostos pelas características da computação móvel aos sistemas de software destacam-se:

- A capacidade de localizar/endereçar elementos móveis;
- A capacidade de perceber mudanças no ambiente de execução;
- A capacidade de se autoconfigurar;
- A capacidade de migrar funcionalidade;
- O uso eficiente dos recursos no elemento móvel;
- O uso eficiente dos recursos do canal de comunicação sem fio;
- O alto grau de tolerância à falhas; e
- Uso de mecanismos para autenticação e criptografia de dados.

2.1.2 Aplicações

Com a arquitetura e a infra-estrutura disponíveis surge uma nova classe de aplicações, a qual combina a computação pessoal e o ambiente de computação móvel. Para que essas novas tecnologias de computação móvel possam trabalhar em conjunto, é necessário que as mesmas dêem suporte a aplicações que sejam necessárias e úteis a uma comunidade de usuários.

A tecnologia sem fio dever ir além de servir a um pequeno número de aplicações verticais (de âmbito específico de empresas) e alcançar um grande número de usuários. Entre as principais características que podem levar essas tecnologias a se popularizarem destacam-se:

- A possibilidade de acessar grandes bancos de dados de qualquer lugar e a qualquer momento;
- A versatilidade de formas de comunicação e a cooperacão que poder existir entre seus usuários;
- A possibilidade de notificação a uma comunidade de usuários a respeito de fatos ou eventos críticos e úteis para seus usuários móveis;
- A maior facilidade de integração de pontos distantes de empresas.

Entre os muitos exemplos de aplicações para o ambiente de computação móvel destacam-se diversas aplicações:

- Vendas de produtos por vendedores com acesso a estoques em tempo real;
- Controle de transportes de carga e localização geográfica de caminhões através de rastreamento;
- Acompanhamento e orientação de procedimentos médicos em ambulâncias;
- Envio e recebimento de arquivos e *e-mails* em *notebooks* através de modem de celular;
- Acesso à Internet em *notebooks* e PDAs;
- Computação cooperativa entre usuários móveis, como por exemplo alertas de acidentes em rodovias, catástrofes naturais, situações de perigo, etc;
- Comércio eletrônico num ambiente móvel.

2.1.3 Limitações

Diversas características novas e limitações inerentes aos sistemas de computação móvel precisam ser reavaliadas. Estas limitações podem ser explicadas pelas seguintes restrições:

- Elementos móveis são relativamente pobres em recursos em relação aos elementos fixos, estáticos;
- A conectividade no ambiente móvel é altamente variável em seu desempenho e na confiabilidade da conexão;
- Os elementos móveis contam com uma fonte de energia finita para seus equipamentos.

As características e limitações dos sistemas de computação móvel, podem ser classificadas de acordo com as seguintes propriedades:

- Mobilidade dos hosts - A localização dos elementos móveis e conseqüentemente seus pontos de conexão à rede fixa se alteram com o seu movimento. As principais conseqüências da mobilidade são:
  - A configuração dos sistemas que incluem elementos móveis não é estática.
  - Os algoritmos de processamento distribuído tradicionais precisam ser reprojetados devido à ausência de uma topologia fixa da rede com os elementos móveis;
  - A distribuição de carga em uma rede com elementos móveis pode mudar de maneira muito rápida, em função da possibilidade de mudança de localização dinâmica desses elementos.
  - O custo para localizar um elemento móvel contribui de forma significativa para o custo de cada comunicação;
  - Estruturas de dados, algoritmos e planos de consultas eficientes devem ser planejados e desenvolvidos para representar, gerenciar e consultar a localização dos elementos móveis, uma vez que os dados de localização podem se alterar de forma muito rápida.
  - O número de serviços disponíveis para um elemento móvel pode variar de uma localização para outra, em função das características da célula e da estação de base na qual ele se conecta;
  - Os recursos disponíveis nos elementos móveis variam, como por exemplo a sua capacidade de memória, o tamanho de sua tela, o tempo de duração de sua bateria, etc.

- Interface de comunicação sem fio - O fato da comunicação entre um elemento móvel e uma estação de base ser através de uma rede de comunicação sem fio, tem como principais conseqüências:
  - Conectividade fraca e intermitente.
  - As redes sem fio são mais caras;
  - Oferecem menores larguras de banda;
  - Possuem uma latência maior do que as redes com fio;
  - São menos confiáveis que as redes com fio.

  - A qualidade da conexão pode variar abruptamente em função de possíveis interferências na comunicação, da distância do elemento móvel em relação à estação de base na qual está conectado ou devido ao compartilhamento da estação de base por vários elementos móveis;
  - Facilidade para transmissão de dados (broadcast).
  - Possibilidade para disseminação de informações a grupos de clientes móveis específicos;
  - Possibilidade de prestação de serviços específicos para certos grupos de clientes móveis
  - Em algumas redes o acesso à rede é pago pelo tempo de conexão, como por exemplo nas comunicações via telefones celulares com comutação por circuitos. Em outras redes o acesso à rede é pago por mensagens, como por exemplo nas comunicações via pacotes de rádio.

- Portabilidade dos Elementos Móveis - Por serem portáteis, os elementos móveis precisam ser pequenos e leves, tornando-os inferiores, em suas capacidades de processamento, aos computadores convencionais. Assim sendo:

  - Elementos móveis são pobres em recursos quando comparados com os elementos estáticos;
  - São equipados com pouca memória do tipo RAM (Random Access Memory);
  - Seus processadores são mais lentos;
  - Possuem pouca memória não volátil (não possuem disco rígido);
  - A interface do usuário é mais limitada;
  - O monitor é menor, o tamanho da tela é reduzido;
  - Os dispositivos de entrada de dados também são menores e limitados (por exemplo, teclados de aparelhos celulares).
  - Elementos móveis contam com pouca capacidade em suas baterias;
  - Dependem da energia fornecida por baterias;
  - Normalmente as baterias possuem capacidades limitadas;
  - Em alguns lugares, dificuldade para recarga da bateria.
  - Elementos móveis são poucos robustos
  - Fáceis de serem danificados;
  - Facilmente perdidos ou roubados.

2.1.4 Mobilidade

A questão principal na computação móvel é a mobilidade, que introduz restrições inexistentes na computação tradicional formada por computadores estáticos. As restrições são geradas pela variação da velocidade do canal, passando por interferências do ambiente e localização da estação móvel, até duração da bateria desta estação. No caso de desconexões temos: desconexão voluntária, variações na taxa de sinal/ruído, energia disponível na bateria e conhecimento da distribuição da largura de banda. É necessário o tratamento das operações no modo desconectado: quando o usuário se reconecta com a rede fixa, as modificações que foram feitas em arquivos durante o modo desconectado devem ser enviadas para o servidor apropriado. As características do computador móvel, como peso e tamanho, limitam a disponibilidade de recursos como memória e capacidade de processamento.

Além disso, devido às diferenças estruturais de um sistema móvel, assim como as variações de tráfego, o ambiente de operação do usuário passa a ser altamente dinâmico. Um usuário pode desfrutar de uma taxa de transmissão numa área e com o seu deslocamento alterar essa taxa de transmissão.

Num ambiente móvel podemos caracterizar dois tipos de mobilidade:

- Mobilidade pessoal é a habilidade dos usuários móveis acessarem serviços através de diferentes tipos de dispositivos e redes, onde quer que os usuários estejam. Os aspectos chave são: identificar os usuários globalmente e adaptação ao terminal em uso.

- Mobilidade de terminal é a capacidade de um terminal móvel mudar seu ponto de ligação à rede e ainda reter o endereço de rede antigo, bem como manter quaisquer sessões de comunicação ativas.

Dentro deste contexto, o objetivo principal da computação móvel é prover para os usuários um ambiente computacional, ou seja, um conjunto de serviços comparáveis aos existentes num sistema distribuído, permitindo a mobilidade.

2.2 Contexto

Quando nós humanos interagimos com outras pessoas e com o ambiente, nós fazemos uso de informações da situação que estão implícitas. Nós podemos, intuitivamente, deduzir e interpretar o contexto da situação corrente e reagir de maneira apropriada. Por exemplo, uma pessoa discutindo com outra pessoa automaticamente observa os gestos e tom de voz da outra parte e reage de maneira apropriada.

Os computadores não são tão bons como os seres humanos para deduzir informações da situação de seus ambientes e usá-las nas interações. Eles não podem tirar proveito de tais informações de maneira transparente, eles necessitam que ela seja fornecida de maneira explícita. Isto é um desafio para a interação humano-computador. Por exemplo, as interfaces de apresentação ao usuário raramente podem monitorar e adaptarem-se automaticamente à intensidade de luz e ao nível de ruído sem que o usuário forneça esta informação. Outro exemplo vem da computação móvel. Não seria interessante se pudéssemos obter serviços e informações de acordo com nossa localização e atividade que estivermos fazendo? Por exemplo, se estamos num estádio assistindo uma partida de futebol, poderíamos obter informações adicionais sobre os jogadores, estarmos aptos a participar de apostas, e verificarmos a situação de tráfego fora do estádio. Há várias maneiras pelas quais a informação do contexto poderia ser usada para que sistemas computacionais e aplicações se tornem mais amigáveis, flexíveis e adaptáveis. O uso de informação do contexto é especialmente importante em um ambiente móvel, onde o ambiente de interação, execução e necessidades de uso mudam rapidamente [35].

Contexto significa informação da situação ou como [9] define "qualquer informação que pode ser usada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, lugar, ou objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo os próprios usuários e aplicações".

Praticamente qualquer informação disponível no tempo de uma interação pode ser vista como informação do contexto. Alguns exemplos são:

- Identidade
- Informação espacial - localização, orientação e velocidade.
- Informação temporal - hora do dia, data e período do ano.
- Informação do ambiente - temperatura, qualidade do ar, luz e nível de ruído.
- Situação social - com quem você está e pessoas que estão próximas.
- Recursos que estão próximos - dispositivos acessíveis.
- Disponibilidade de recursos - bateria, *display*, rede e largura de banda.
- Medidas da parte física da pessoa - pressão sanguínea, batida do coração, taxa de respiração, atividade muscular e tom de voz.
- Atividade - conversando, lendo, andando e correndo.
- Agendas

Ciência do contexto significa estar apto a usar a informação do contexto. Um sistema está ciente do contexto se ele pode extrair, interpretar e usar a informação do contexto e adaptar suas funcionalidades ao contexto corrente. O desafio para tais sistemas recai na complexidade de capturar, representar e processar dados contextuais [10].

Para capturar informação do contexto geralmente alguns sensores e/ou programas adicionais são necessários. Para transferir a informação do contexto para aplicações e para diferentes aplicações estarem aptas a usarem a mesma informação do contexto, um formato de representação comum para tal informação deve existir. Além de estar apto a obter informação do contexto, as aplicações devem incluir alguma "inteligência" para processar a informação e deduzir o significado. Isto é, provavelmente, o assunto mais desafiador, sendo que o contexto é sempre indireto ou dedutível pela combinação de diferentes peças da informação do contexto. Por exemplo, se três pessoas se encontram em uma certa sala de escritório numa certa hora, o contexto pode indicar que está acontecendo o encontro estratégico semanal.

Há aspectos que são características das aplicações cientes do contexto [9, 11, 12]:

- A informação e os serviços podem ser apresentados aos usuários de acordo com o contexto corrente.
- Execução automática de um serviço quando em um certo contexto. Isto inclui ações disparadas pelo contexto e adaptação contextual.
- Juntar a informação do contexto para consulta posterior.

Como exemplo de sistema ciente do contexto temos o projeto CyberGuide [34] no Georgia Tech, onde o objetivo é oferecer informações ao turista baseado na sua posição e orientação. A Figura 2.2 ilustra o sistema.

2.3 *I-centric*

Observando o comportamento da comunicação dos seres humanos e o seu espaço de comunicação, é óbvio que os seres humanos interagem habitualmente com um conjunto de contextos em seu ambiente. As preferências e as necessidades de informações individuais, as pessoas com as quais interage e o conjunto de dispositivos que cada indivíduo controla definem o espaço pessoal de comunicação. Seguindo esta linha, uma nova abordagem é construir sistemas de comunicação baseados não em tecnologias específicas, mas na análise dos espaços individuais de comunicação. O resultado é um sistema de comunicação adaptado às demandas específicas de cada indivíduo. O sistema de comunicação agirá em nome das demandas dos usuários, refletindo ações recentes para permitir personalizar e auto adaptar-se aos contextos e situações.

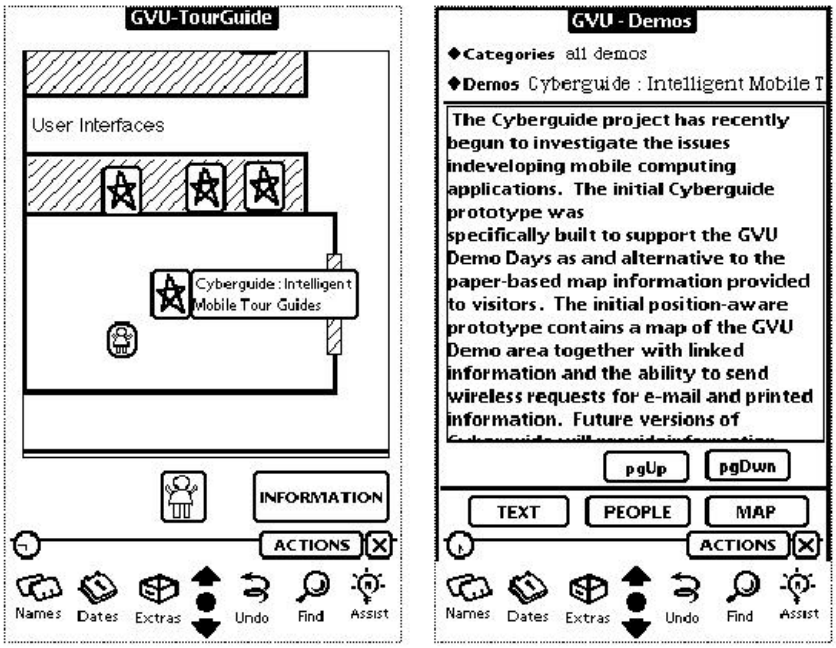


Figura 2.2: CyberGuide

O sistema de comunicação *I-centric* [8] é uma abordagem para projetar sistemas de comunicação que se adaptam ao espaço de comunicação, ao ambiente e à situação do indivíduo. Neste contexto "I" significa indivíduo, "*centric*" significa adaptáveis às exigências do I (indivíduo) e a um determinado ambiente de usuário. A Figura 2.3 ilustra o espaço de comunicação do indivíduo.

A variedade de dispositivos, tecnologias de telecomunicações, serviços, sistemas de posicionamento e detecção, e aplicações cientes do contexto ou cientes da localização podem ser vistas como tecnologias que habilitam comunicações *I-centric*.

Para comunicações I-Centric três tópicos podem ser identificados:

- Serviços I-Centric descrevem a habilidade para definir e gerenciar contextos que são ajustados para as preferências de usuários, de maneira individualizada na interação com o sistema de comunicação. Baseado na avaliação dos perfis que descrevem as preferências do usuário, potencialidade de serviços e nas informações a respeito do ambiente atual, podem ser oferecido ao usuário serviços individualizados de acordo com suas demandas.

- Interação com Ambientes inteligentes compreende a funcionalidade que é necessária em um certo contexto para sentir o ambiente de maneira a estar apto a adaptar-se a

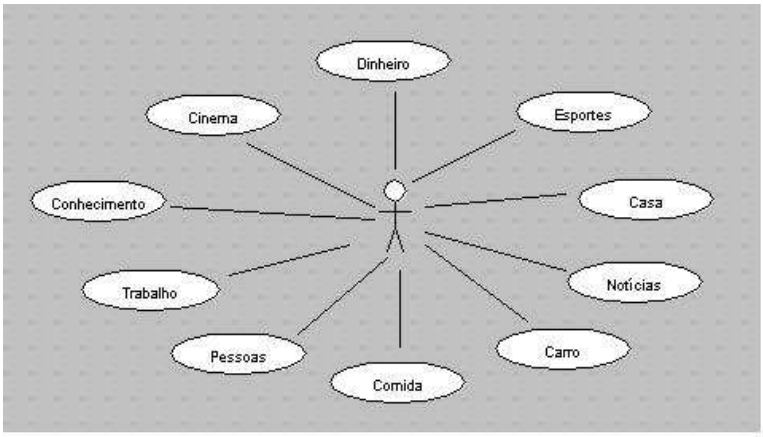


Figura 2.3: I-Centric: Espaço Pessoal de Comunicação

ele. Características temporais e espaciais são dois exemplos de informações as quais podem afetar o contexto e adaptar o sistema de comunicação para certa situação. Temperatura, nível de ruído, intensidade da luz, presença de outras pessoas e objetos na vizinhança, são parâmetros adicionais que podem ajudar a aplicação I-Centric a adaptar-se ao perfil e às necessidades do usuário. Observe que um certo ambiente pode restringir a funcionalidade fornecida por um certo contexto. Por exemplo, interagindo em um contexto de TV enquanto dirige-se um carro pode reduzir a funcionalidade para apenas "gravar o filme XYZ para assistir mais tarde".

- Plataforma de Serviço I-Centric é responsável por formar o sistema de comunicação, baseado nos contextos identificados e nas informações do ambiente monitorado. Ela ativa os objetos envolvidos no contexto, identifica causalidades entre eles baseados nos dados do ambiente monitorado, controla os serviços oferecidos por estes objetos, e converte estruturas de dados em operações para a integração de serviços. O equipamento é configurado dinamicamente, seu estado é personalizado, objetos distribuídos são controlados, a criação e a distribuição dos serivços são supervisionados, e a interação entre domínios é habilitada pela plataforma.

Capítulo 3

Tecnologias utilizadas

Este capítulo descreve as tecnologias utilizadas no modelo proposto: agentes móveis e *middleware* reflexivo, bem como o *middleware* reflexivo ReMMoC (*Reflective Middleware for Mobile Computing*) e a plataforma de agentes móveis *Grasshopper*.

3.1 Agentes Móveis

A idéia de agentes teve início com John McCarthy em 1950 e foi adotada por Oliver G. Selfridge quando os dois trabalhavam no MIT (Massachusetts Institute of Technology). Eles imaginaram um programa que pudesse executar determinada tarefa em nome do usuário de forma que este precisasse interagir o mínimo possível com o programa, ou seja, o sistema teria uma certa autonomia, bastando apenas apresentar ao usuário o resultado obtido [36]. Conseqüentemente foram surgindo várias idéias de agentes e uma delas foi a de agentes móveis. Agentes móveis são programas que podem viajar através de uma ou mais redes (WANs, Intranets, Internet), transportando tanto o código como o seu estado, e realizar tarefas em máquinas que tenham a capacidade de hospedar agentes. Isto permite que processos possam migrar de computador para computador e que também possam se replicar, de tal forma que múltiplas instâncias de um mesmo agente executem em máquinas diferentes e retornem para o ponto de origem. Diferente das chamadas de procedimentos remotos, onde um processo invoca um procedimento de um *host* remoto, a migração de processo permite que código executável, junto com o seu estado, viaje para um destino e, uma vez lá, possa interagir com bancos de dados, sistemas de arquivos, serviços de informação e outros agentes, tudo isso localmente.

Muitas questões já foram levantadas sobre as vantagens da utilização de outros mecanismos de comunicação, como o RPC (*Remote Procedure Call*), ao invés de agentes móveis, porém como visto em [25], a escolha certa do mecanismo de comunicação depende da aplicação a que se destina.

Agentes móveis são uma alternativa atraente para a construção de ambientes distribuídos, principalmente em aplicações como computação móvel [26], pois permite que dispositivos móveis enviem agentes para a rede fixa a fim de que estes executem alguma tarefa em seu lugar, de forma mais eficiente, sem gastar os recursos dos dispositivos e independente das condições de conectividade. Logo, agentes móveis tornam possíveis novos serviços e novas oportunidades de negócios através do melhor uso de recursos de comunicação (em termos de custo e desempenho) e suporte flexível à operação desconectada, dentre outros.

Nesta seção serão abordados alguns conceitos relacionados a agentes móveis, suas características, aplicações,vantagens e desvantagens. Além disso, a plataforma de agentes móveis *Grasshopper*, utilizada neste trabalho, será descrita.

3.1.1 Conceitos

Nesta seção são apresentados alguns conceitos referentes a agentes. Vale salientar que existem várias definições para estes conceitos, contudo as apresentadas a seguir foram retiradas de [37], e são as que utilizaremos nesta dissertação.

- Agentes

  Agentes são programas que podem agir de forma autônoma em nome de uma pessoa ou organização. Cada agente possui sua própria *thread* de execução, podendo desta forma executar sua tarefa de forma independente.

- Agente Estacionário

  Agente estacionário é aquele que executa no sistema onde foi gerado. Quando precisa de uma informação que esteja fora do sistema, ele utiliza mecanismos de comunicação tais como RPC ou RMI (*Remote Method Invocation*).

- Agente Móvel

  Agentes móveis são aqueles que não estão restritos aos limites do sistema onde iniciou sua execução. Eles tem a habilidade de se locomover de um sistema para outro através de uma rede. Esta habilidade permite que um agente se mova para um sistema que contenha um objeto com o qual ele precisa interagir.

- Estado do Agente

  O estado do agente pode ser tanto o seu estado de execução quanto os valores de seus atributos, que determinam o que ele deve fazer quando sua execução for iniciada no destino.

- Estado de Execução do Agente

  O estado de execução de um agente é o estado do seu *runtime*, contador de programa e pilha.

- Proprietário do Agente

  O proprietário do agente é a pessoa ou organização por quem o agente atua.

- Nome do Agente

  Agentes necessitam nomes pelos quais possam ser identificados e localizados via um serviço de nomes. A identidade de um agente é um valor único dentro do escopo de um proprietário e identifica uma instância particular de um agente. A combinação da identidade do agente e do proprietário do agente forma um nome único. Em ambientes com mais de um sistema de agentes, este nome único é formado pela identidade do agente, do proprietário do agente e do sistema de agentes. Desta forma, o nome pode ser usado como chave em operações que fazem referências à instância de um agente.

- Perfil do Agente

  O perfil do agente descreve as características deste agente, tais como a linguagem em que ele é implementado e quais métodos de serialização podem ser aplicados a ele.

- Sistema de Agentes

  Sistemas de agentes são plataformas que podem criar, interpretar, executar, transferir e destruir um agente. Assim como um agente, um sistema de agentes é associado a uma pessoa ou organização por quem aquele sistema de agentes atua. Um sistema de agentes é identificado por seu nome e endereço. Um *host* pode ter mais de um sistema de agentes.

- Agência

  Agência é um contexto dentro do sistema de agentes no qual o agente pode executar. Este contexto fornece ao agente funções tais como controle de acesso. As agências de origem e destino podem residir no mesmo sistema de agentes, ou em sistemas diferentes que suportem o mesmo perfil de agente.

- Região

  Uma região é um conjunto de sistemas de agentes que possuem o mesmo proprietário, mas não são necessariamente do mesmo tipo de sistema de agentes. O conceito de região permite que mais de um sistema de agentes represente o mesmo proprietário.

3.1.2 Características

Na literatura as características de agentes variam de autor para autor. Apresentaremos aqui as características mais relevantes e que são citadas com mais freqüência.

- Autonomia

  Esta propriedade especifica que as ações tomadas pelos agentes dependem exclusivamente das informações por eles obtidas e de seus padrões de decisão. Um agente, seja ele estacionário ou móvel, age como uma entidade independente e evolutiva, sendo capaz de coletar informações, aprender e tomar decisões sem influência externa.

- Mobilidade

  Propriedade fundamental para um agente móvel. Constitui-se na transferência dos dados, código e estado de execução de um lugar para outro.

- Identidade

  A navegação e a multiplicação dos agentes ao longo de uma rede tornam necessário que eles se diferenciem uns dos outros. No caso da clonagem (ver adiante), um agente é replicado em código e objetivo, mas cada um tem uma identidade própria permitindo que seus resultados sejam agregados.

- Ciclo de Vida

  Ciclo de vida corresponde aos processos de como os agentes são criados, executados e terminados.

  - Criação

    A criação de um agente consiste em requisitar a uma base ou plataforma de agentes que crie e inicie a execução de um agente.

  - Duração

    Uma vez criado, a execução de um agente pode ser suspensa e posteriormente reiniciada. A preservação dos dados e estado de execução do agente, nestes intervalos de suspensão, determina a sua existência.

  - Término

    A destruição do agente irá determinar o término da sua execução e eliminação dos seus dados. Um agente tem autonomia para se auto-eliminar, contudo ele também pode ser encerrado pelo usuário ou por outro agente.

- Reprodução

  Agentes podem criar outros agentes passando o seu próprio código. Esta criação copia total ou parcialmente o código do agente.

  - Clonagem

    O novo agente criado é igual em dados e estado de execução, contudo um novo identificador é criado para definir o clone como um novo agente.

  - Recombinação

    Dois ou mais agentes podem combinar seu código de forma a otimizar a criação de um agente resultante com capacidades mais aprimoradas.

- Comunicabilidade

  A comunicação entre agentes tem a finalidade de trocar dados e informações e coordenar a ação entre agentes ou grupos de agentes.

3.1.3 Vantagens

A seguir mostramos algumas possíveis vantagens no uso de agentes móveis. Porém, estas vantagens dependem da aplicação em que o agente é utilizado, pois dependendo do caso é preferível utilizar métodos de comunicação remota, tais como RPC e RMI.

- Diminuição do tráfego na rede

  Isso acontece porque não é necessária a troca de várias mensagens na rede para a realização da tarefa. Uma vez que o agente é deslocado para o *host* remoto, ele continua sua execução sem precisar comunicar com o *host* origem, assim o tráfego pesado de mensagens ocorre localmente.

- Concorrência

  A execução de uma determinada tarefa pode ser dividida em vários agentes de forma que cada um execute uma parte dela, ou uma tarefa pode ser executada, tantas vezes quanto forem o número de agentes, de forma concorrente.

- Assíncronismo

  O cliente não fica bloqueado esperando pelo retorno do agente. Ele pode executar outras tarefas enquanto isso.

- Autonomia

  O agente é autônomo. Ele não precisa de comandos externos para tomar suas próprias decisões.

3.1.4 Desvantagens

Aqui são apresentados os maiores obstáculos no uso e desenvolvimento de agentes móveis

- Desenvolvimento mais complexo

  O desenvolvimento de uma plataforma de agentes móveis é mais complexo do que outras tecnologias, tais como RPC. Isto se deve ao fato de o programador ter que se preocupar com todo o ciclo de vida do agente, sua mobilidade, sua autonomia e sua persistência.

- Segurança

  Um dos problemas de agentes móveis é com relação à segurança. Já que o agente leva o seu código e o seu estado para executar no *host* remoto, nada impede que ele execute tarefas indesejadas, tais como acessar informações confidenciais, ou mesmo carregar ou agir como um vírus. São necessários mecanismos para proteger *hosts* de agentes destrutivos e também proteger agentes de *hosts* destrutivos.

- Interoperabilidade

  Devido ao grande número de plataformas de agentes móveis, não se tem uma interoperabilidade. Uma das tentativas de se resolver isto é através da interface MASIF (*Mobile Agent System Interoperability Facility*) [37] do OMG, que é uma coleção de definições que fornecem interoperabilidade para sistemas de agentes móveis diferentes. O MASIF possui duas interfaces:

  – MAFAgentSystem

    Define operações para receber, criar, suspender e terminar a execução de um agente.

  – MAFFinder

    Define operações para registrar, desregistrar e localizar agentes, agências e sistemas de agentes.

3.1.5 Aplicações

Existem diversas aplicações para o uso de agentes móveis e a maioria delas envolve procura de informações em nome de um usuário e possivelmente a execução de uma tarefa quando determinada informação é encontrada. A seguir são apresentadas algumas aplicações:

- Coleta de dados de vários lugares

Uma das principais diferenças entre agente móvel e código móvel é o itinerário. Enquanto o código móvel viaja de um *host* X para um Y, o agente móvel possui um itinerário que ele segue para ir de um lugar a outro. Uma aplicação natural para agentes móveis é a coleta de informação em vários lugares de uma rede.

- Busca e filtragem

  Devido ao grande número de *sites* na Internet, a quantidade de informação disponível é imensa, acarretando também o aumento da quantidade de informação irrelevante. Em nome de um usuário, um agente móvel pode visitar diversos *sites*, procurar por informações disponíveis e criar um índice de *links* para trechos de informações que coincidem com determinado critério ou com as preferências do usuário.

- Monitoramento

  Novas informações são constantemente produzidas e disponibilizadas na rede. Agentes podem ser enviados para monitorar quando determinada informação estará disponível, e quando estiver, executar determinada tarefa. Um exemplo deste tipo de aplicação é a coleta de notícias personalizadas. Um agente poderia monitorar diversas fontes à espera de notícias de interesse do usuário e informar quando estas notícias ficassem disponíveis.

- Processamento paralelo

  Dado que um agente pode se mover através de uma rede e criar novos agentes, uma aplicação potencial para o seu uso é o processamento paralelo. Se um processamento requer mais tempo de CPU do que é disponível localmente, a tarefa pode ser distribuída em diversos agentes para ser executada em outras máquinas.

- Comércio eletrônico e comércio móvel

  Outro uso de agentes móveis é no comércio eletrônico. Um agente pode procurar em diversos *sites* de venda um determinado produto e comprar naquele *site* onde o encontrou por um preço menor. Ele pode interagir com outros agentes para negociar o preço de um produto, considerando-se que exista um agente de compra e um de venda.

  Uma extensão natural do seu uso em comércio eletrônico é o uso de agentes móveis em comércio móvel. E neste, o uso de agentes é naturalmente interessante, já que uma vez que o agente age de forma autônoma e independente, não é necessário que o usuário permaneça conectado durante uma transação, o que é bastante útil devido às limitações dos dispositivos usados no comércio móvel.

3.1.6 Grasshopper

A Plataforma de Agentes Móveis utilizada neste trabalho foi o *Grasshopper*.

Grasshopper [16] é uma plataforma de agentes móveis lançada pela IKV em 1998. O *Grasshopper* é implementado em Java e baseado no padrão MASIF [23], que define uma interface para interoperabilidade entre plataformas de agentes móveis. Além disso, o *Grasshopper* tem uma versão para dispositivos móveis que rodam o sistema operacional WinCE.

O *Grasshopper* é composto por agências (*agency*), regiões (*regions*) e lugares (*places*). Uma agência é o ambiente de execução para agentes móveis e estacionários. Um lugar é um agrupamento lógico de funcionalidades dentro de uma agência. Em uma agência podem existir vários lugares. As agências, bem como seus lugares, podem ser associados a uma região. A figura 3.1 ilustra os conceitos mencionados acima.

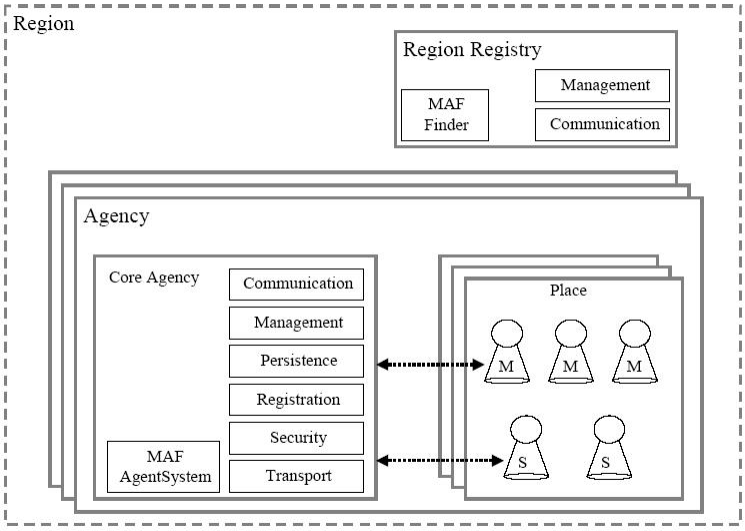


Figura 3.1: *Grasshopper*

Os agentes se comunicam utilizando a comunicação assíncrona disponível no *Grasshopper*, através da criação de *proxies* assíncronos. Foi escolhida esta forma de comunicação para evitar que um agente fique bloqueado esperando o retorno dos resultados por parte de outro agente.

Os protocolos de comunicação suportados pelo *Grasshopper* são: CORBA IIOP, RMI, RMI com SSL, Sockets e Sockets com SSL.

O *Grasshopper* fornece mecanismos de tolerância a falhas através de seu serviço de persistência. Pode-se salvar o estado do agente, interromper sua execução e depois reiniciá-

la com seu estado salvo, garantindo assim a integridade das informações. Ele também fornece um serviço de segurança com dois tipos de mecanismos:

- Segurança Externa: fornece proteção a todas as comunicações remotas que são realizadas pelo serviço de comunicação do *Grasshopper*. Entende-se por comunicação remota, toda comunicação entre o *Grasshopper* e um sistema externo.
- Segurança Interna: fornece proteção às interfaces das agências e dos agentes, bem como a certos recursos das agências (tais como o sistema de arquivos local), contra acessos não autorizados, realizados por agentes.

3.2 *Middleware* Reflexivo

Chamamos de middleware, como sendo a camada de software que fica entre o sistema operacional e a aplicação. A Figura 3.2 ilustra o middleware.

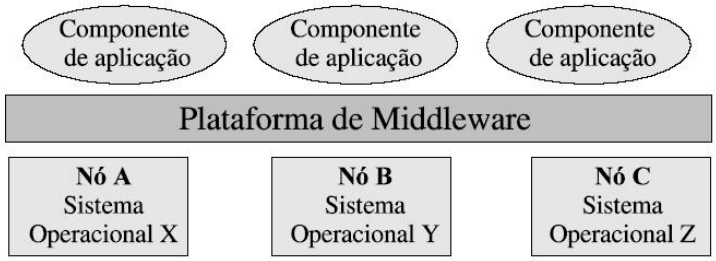


Figura 3.2: *Middleware*

O middleware é responsável por resolver problemas relacionados a ambientes distribuídos tais como: heterogeneidade de sistemas operacionais, linguagens de programação, plataformas de hardware, marshalling, unmarshalling, dentre outros. Neste caso, o desenvolvedor não se preocupa com os detalhes relacionados a distribuição e tem a possibilidade de pensar no sistema como este sendo centralizado. Como exemplos de middleware temos as tecnologias CORBA, COM e JAVA/RMI.

Contudo, o middleware funciona como uma caixa-preta, isto é, ele resolve tudo de maneira transparente. O desenvolvedor não tem conhecimento do modo como o middleware está resolvendo certos problemas.

Entretanto, para certos tipos de aplicação existe a necessidade de saber o que está acontecendo nas camadas inferiores do middleware. Elas podem obter ganhos se puderem conhecer e manipular os detalhes das camadas inferiores. Por exemplo, uma aplicação que

roda em um ambiente de computação móvel pode obter ganhos em termos de desempenho caso tenha conhecimento dos protocolos de transporte e da largura de banda para este ambiente [13].

Desta forma, surgiu a idéia do middleware reflexivo, o middleware que oferece transparência para aplicações que necessitam desta e que permite a inspeção e a manipulação dos detalhes internos do middleware.

Geralmente, o *middleware* reflexivo é implementado como uma coleção de componentes que podem ser configurados e reconfigurados dinamicamente [14].

O *middleware* reflexivo utilizado neste trabalho foi o ReMMoC (*Reflective Middleware for Mobile Computing*) [15] baseado no padrão de componentes OpenCOM [17], o qual foi desenvolvido utilizando algumas funcionalidades do padrão Microsoft COM [19]. Desta forma, a seguir, comenta-se o COM, OpenCOM e o ReMMoC.

3.2.1 Microsoft COM

Esta seção dá uma visão geral da tecnologia COM relacionada ao seu uso no OpenCOM.

A tecnologia COM é baseada em três conceitos fundamentais:

- Especificações de interfaces identificadas de maneira única e imutáveis.
- Componentes identificados de maneira única que podem implementar múltiplas interfaces.
- Um mecanismo de descoberta dinâmica de interfaces .

O COM dá suporte à unicidade atavés de GUIDs (*Globally Unique Identifiers*) de 128 bits. Além disso, o identificador único de uma interface é referenciado como um IID (*Interface Identifier*), e um tipo de componente como um CLSID (*Class Identifier*). O mecanismo de descoberta de interface constrói uma interface especial chamada *IUnknown* que deve ser implementada por todo componente COM. O propósito desta interface é:

- Permitir a consulta dinâmica de um componente (operação *QueryInterface*) para saber se ele dá suporte a uma certa interface (no caso um ponteiro para a interface é retornado).
- Dar suporte à contagem de referências (*reference counting*) automática do número de clientes utilizando a interface do componente. Isto permite que o componente seja desativado quando não tiver mais clientes.

Definições de componentes e interfaces são expressas através de uma linguagem independente chamada MIDL (*Microsoft Interface Definition Language*), sendo que uma

ferramenta chamada *midl* é usada para compilar estas definições e automaticamente gerar *templates* para uma linguagem específica. *Midl* também gera arquivos chamados de *type libraries*, que armazenam informações dos tipos relacionados aos componentes e suas interfaces.

Finalmente, o COM emprega um repositório centralizado chamado de *Registry*. Ele armazena informações a respeito dos componentes.

3.2.2 OpenCOM

OpenCOM é um modelo de componentes reflexivo, leve e eficiente, construído sobre um subconjunto do Microsoft COM. Características de nível mais alto do COM, incluindo distribuição, persitência, transação e segurança não são usadas. Aspectos do núcleo como padrão de interoperabilidade em nível binário, IDL, GUID e a interface IUnknown formam a base de implementação.

Os conceitos fundamentais do OpenCOM são interfaces, receptáculos e conexões (ligações entre interfaces e receptáculos). Uma interface expressa uma unidade de provisão de serviço e um receptáculo descreve uma unidade de requisição de serviço. OpenCOM oferece um *runtime* padrão que gerencia a criação e destruição de componentes, e age em cima de requisições de conectar/desconectar componentes. Além do mais, um grafo do sistema mostra a configuração dos componentes que estão em uso, dando suporte a introspecção da estrutura da plataforma.

O OpenCOM requer a implementação das interfaces *ILifeCycle*, *IReceptacle* e *IMetaInterface* por cada componente. A *ILifeCycle* fornece as operações de *startup* e *shutdown* que são chamadas quando um componente é criado ou destruído. A *IReceptacle* oferece métodos para modificar as interfaces conectadas ao receptáculo de um componente. A *IMetaInterface* dá suporte a inspeção dos tipos de interface e receptáculos declarados pelo componente.

Além disso, o *runtime* do OpenCOM fornece interfaces de meta-inteceptação e meta-arquitetura que podem ser usadas durante a configuração/reconfiguração do middleware.

3.2.3 ReMMoC

O projeto ReMMoC foi desenvolvido na Universidade de *Lancaster* em parceria com a *Lucent Technologies*. Este projeto propõe o uso de reflexão e tecnologia de componentes para resolver o problema de tecnologia de *middlewares* heterogêneos no ambiente de computação móvel. Aplicações móveis e serviços são implementados através de uma gama de plataformas de *middleware* (ex. RPC, orientado a mensagens, paradigmas baseado em eventos) e são anunciados usando diferentes protocolos de descoberta de serviços (SLP

(*Service Location Protocol*)[32], UPnP (*Universal Plug and Play* [42]), Jini [43], Salutation [44]). Desta forma, as aplicações móveis ganham flexibilidade através de *middleware* reflexivos no sentido de poderem interagir com diferentes tecnologias de *middleware* e protocolos de descoberta de serviços.

A Figura 3.3 ilustra este cenário de tecnolgias de *middleware* heterogêneas no ambiente móvel. No exemplo, três serviços de aplicações estão disponíveis para usuários móveis em duas localidades. As instâncias de cada serviço são implementadas utilizando diferentes tipos de *middleware* e anunciadas através de diferentes protocolos de descoberta de serviços. Por exemplo, a aplicação 1, um serviço de *jukebox* que permite aos usuários selecionar e tocas músicas naquele local, é implementada em um servidor SOAP [45] e anunciada através do UPnP no Bar Café(*Coffee Bar*). Contudo, esta mesma aplicação é implementada em um servidor CORBA e anunciada através do SLP na Casa Pública(*Public House*). Logo, o mesmo serviço está disponível para os usuários nas duas localidades mas implementados através de diferentes tecnologias de *middleware* e protocolos de descoberta de serviços. Contudo, para que o usuário possa utilizar o mesmo serviço nas duas localidades é necessário, no caso do serviço de Jukebox, que o dispositivo utilizado suporte SOAP/UPnP e Corba/SLP. Entretanto, os dispositivos móveis tem limitações em termos de recursos disponíveis para dar suporte a uma variedade de tecnologias de *middleware* e protocolos de descoberta de serviço no mesmo dispositivo. Sendo que o *middleware* num dispositivo móvel deve estar apto a mudar dinamicamente sua estrutura e comportamento, de modo que possa descobrir e interoperar com todos os serviços disponíveis no ambiente corrente.

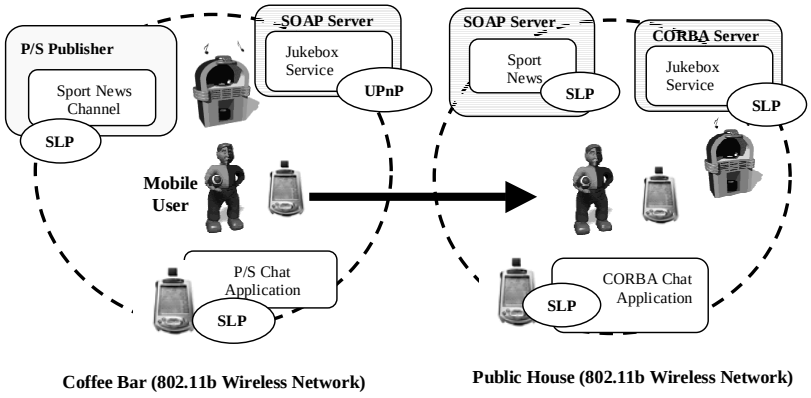


Figura 3.3: Cenário de Computação Móvel com *middlewares* heterogêneos

A plataforma ReMMoC é um *middleware* reflexivo configurável e reconfigurável dinamicamente que suporta o desenvolvimento de aplicações móveis e elimina as pro-

priedades heterogêneas de um ambiente móvel. ReMMoC usa OpenCOM como sua tecnologia de componentes e é construído como um conjunto de *frameworks* de componentes (CFs). OpenCOM é um modelo de componentes reflexivo, eficiente e leve, construído utilizando um subconjunto do padrão Microsoft COM. O ReMMoC consiste de dois CFs chave: (1) um *framework* de ligação para interoperar com serviços móveis implementados através de diferentes tipos de *middleware*, e (2) um *framework* para descoberta de serviços anunciados através de uma gama de protocolos de descoberta de serviços. Na verdade, o ReMMoC é uma instância do OpenCOM para computação móvel. A Figura 3.4 nos mostra uma visão da plataforma ReMMoC.

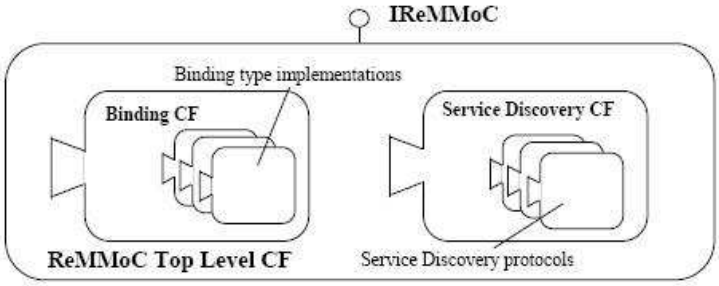


Figura 3.4: Visão da plataforma ReMMoC

A função do *framework* de ligação é interoperar com serviços móveis heterogêneos. Consequentemente, ele pode ser configurado como um cliente IIOP e fazer um número de requisições IIOP, ou mudar sua configuração para um cliente SOAP. Paradigmas diferentes de *middleware*, síncronos e assíncronos, podem ser plugados no *framework* de ligação se eles foram implementados usando componentes OpenCOM. A estrutura do *framework* de componentes gerencia a configuração e reconfiguração dinâmica destas ligações e assegura o tipo de ligação correta antes que a operação occorra. Além disso, podem ser feitas mudanças na configuração para atender requisitos como qualidade do serviço e da própria aplicação. Por exemplo, uma aplicação pode requisitar a funcionalidade IIOP no lado do servidor, além do lado cliente já existente. Neste caso, componentes implementando a funcionalidade no lado servidor são adicionados.

O *framework* de descoberta de serviços permite que serviços anunciados através de diferentes protocolos de descoberta de serviços sejam encontrados. O componente é configurado para a tecnologia de descoberta usada no ambiente corrente. Por exemplo, se o ambiente usa SLP, a configuração do *framework* será para SLP. Contudo, se SLP e UPnP estão sendo utilizados, a configuração do componente será para descobrir ambos. Isto inclui um serviço de descoberta genérico que retorna informações de diferentes protocolos

de descoberta de serviços num formato genérico.

É interessante ressaltar que novas tecnologias podem ser adcionadas aos *framework*s. Para isto, é necessário que estejam implementadas através do padrão OpenCOM.

Capítulo 4

Modelo

O modelo ICoMP é baseado na abordagem *I-centric*, na utilização de um *middleware* reflexivo para construírmos os provedores de serviços e nos serviços oferecidos por uma plataforma de agentes móveis. Além disso, o ICoMP permite dois tipos de acesso: a informações do Portal e serviços oferecidos pelos provedores de serviços; e a serviços oferecidos pela plataforma de agentes móveis como a procura e recebimento de informações ou serviços que não estão no Portal.

A abordagem *I-centric* indica que o Portal oferece serviços ao usuário baseado no seu perfil. A infra-estrutura de um *middleware* reflexivo (provedores de serviços) possibilita que o Portal Móvel se adapte dinamicamente às mudanças no ambiente. Além disso, o Portal Móvel pode oferecer serviços que necessitam de agentes móveis como despachar agentes para procurar informações ou serviços que não estão no Portal, receber os agentes que retornam após a execução de um tarefa designada pelo usuário, dentre outros.

Quanto aos dispositivos utilizados pelo usuário para acessar os serviços do Portal, serão caracterizados por terem diferentes capacidades de processamento, memória, tamanho de tela, dentre outros. Desde telefones celulares, que têm uma pequena capacidade de processamento e limitações com relação, por exemplo, ao tamanho de tela, passando por PDAs, que têm uma capacidade de processamento maior que a dos telefones celulares mas que têm sua capacidade de armazenamento de informações limitada, até, os *notebooks* com grande capacidade de processamento e armazenamento. Logo, cada tipo de dispositivo será tratado de maneira diferenciada no que se refere ao acesso a serviços.

Este capítulo descreve o perfil do usuário bem como o modelo proposto. Além disso, as vantagens do modelo proposto são contextualizadas na seção de Trabalhos Relacionados.

4.1 Perfil do Usuário

O conceito de perfil é modelado em duas partes: parte das informações do perfil está armazenada no dispositivo móvel que o usuário esteja usando, por exemplo, a agenda do usuário, o seu histórico de *sites* acessados, seus assuntos de interesse, tais como esportes e informática, (isto caso o dispositivo tenha capacidade de armazenar esta informação) a outra parte das informações está no *home* (rede a qual o usuário pertence) do usuário, por exemplo, os detalhes de suas preferências, tais como as áreas de informática que ele tem interesse. Além disso, temos os context providers que capturam informações do contexto e as disponibilizam para as aplicações.

Conforme a Figura 4.1, o perfil do usuário é especificado através das seguintes informações: (*Context*, *Name*, *Address*, *Occupation*, *Agenda*, *Favorites*, *History*, *Preferences*), onde *Context* são informações relacionadas ao ambiente, *Name*, *Address*, *Occupation* são informações pessoais, *Agenda* são informações da agenda do usuário propriamente dito, *Favorites* são os *sites* favoritos do usuário, *History* é o histórico dos *sites* acessados e *Preferences* indica as preferências do usuário como esportes, informática e cinema.

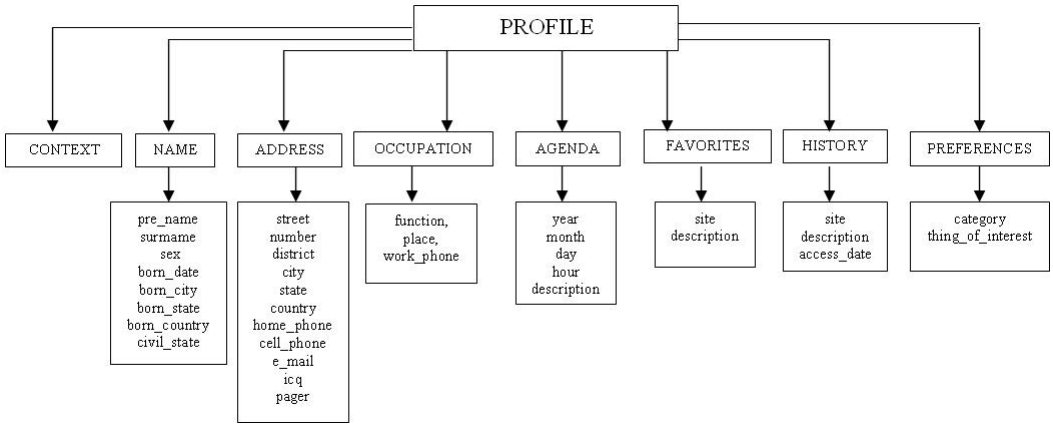


Figura 4.1: Profile

Como pode ser visto na Figura 4.2 o contexto é descrito por (*Device*, *Protocol*, *Network*, *Location*) onde *Device* descreve as características do dispositivo utilizado, *Protocol* especifica os tipos e versões dos protocolos de comunicação disponíveis no ambiente, *Network* indica a tecnologia de rede e largura de banda disponível no ambiente e *Location* indica o local onde o usuário está.

Contudo, nem sempre todas as informações que compõem o *Profile* estarão disponíveis. O usuário pode desejar disponibilizar apenas parte das informações ou os sistemas que

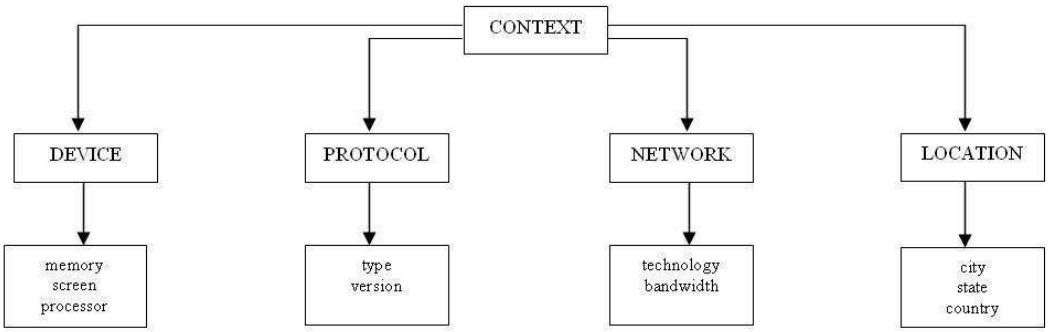


Figura 4.2: Context

fornecem informacões do contexto podem, em certo momento, disponibilizar também apenas uma parte destas informações. Além disso, o dispositivo utilizado pode não ter capacidade para armazenar as informações do *Profile*.

Por isso, o ICoMP especifica níveis com relação ao *Profile*, desde um nível com um conjunto mínimo de informações até todas as informações que compõem o *Profile*.

Inicialmente, o ICoMP especifica dois níveis:

Nível 1: *Profile*(*Preferences*(*category, object of interest*)), *Context*(*Device*(*memory, screen, processor*)), possibilitando uma adaptação baseada nas preferências do usuário e características do dispositivo utilizado.

Nível 2: *Profile*(*Preferences*(*category, object of interest*)), Agenda(*hour, day, year, month, description*), *Favorites*(*site, category, service*), *History*(*site, category, service, access date*), *Context*(*Device*(*memory, screen, processor*), *Location*(*city, state, country*))), possibilitando uma adaptação mais refinada através das preferências do usuário, agenda, *sites* favoritos, histórico dos *sites* acessados, características do dispositivo utilizado e localização.

Contudo, o ICoMP possibilita que mais níveis sejam especificados.

4.2 Modelo

O modelo ICoMP utiliza uma plataforma de agentes móveis e um middleware reflexivo em sua infra-estrutura. Além disso, o ICoMP especifica um mecanismo de adaptação ao perfil do usuário. A Figura 4.3 ilustra a infra-estrutura do modelo.

Componentes

Os componentes são descritos a seguir:

Service Providers: Provedores de Serviços baseados num *middleware* reflexivo. São os

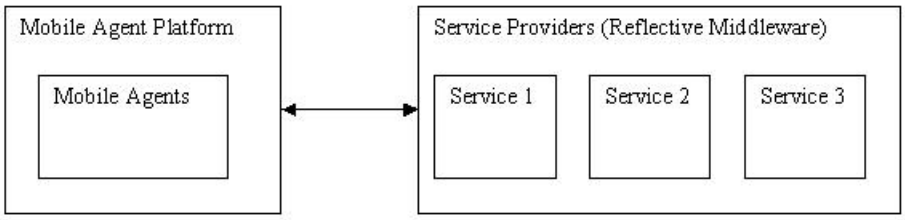


Figura 4.3: Infra - Estrutura do Modelo

componentes que implementam os serviços oferecidos pelo Portal. Por exemplo, na figura 4.3 temos Service1, Service 2, dentre outros.

Mobile Agents: É a parte do Portal responsável por oferecer serviços que se utilizam de agentes móveis através de uma plataforma de agentes móveis.

A figura 4.4 ilustra os componentes do modelo relacionados a adaptação.

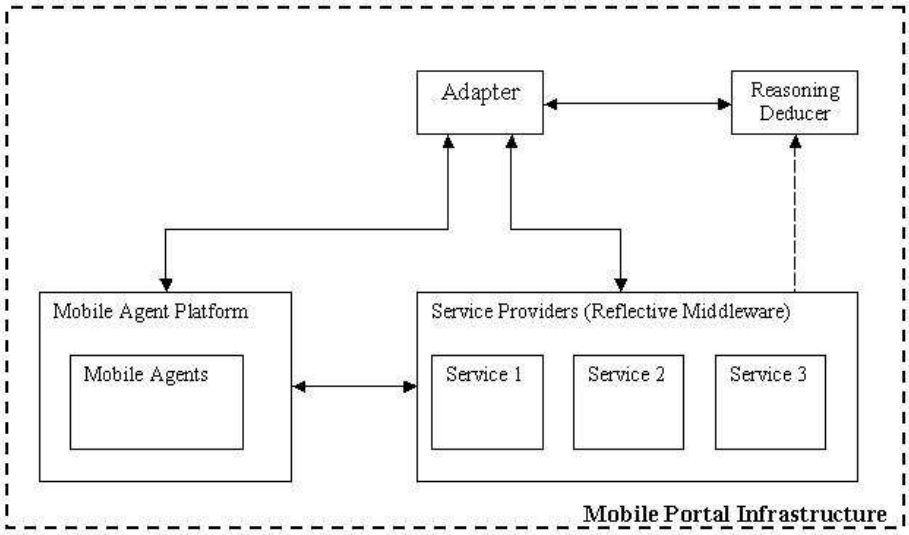


Figura 4.4: Componentes de adaptação do Modelo

Reasonig Deducer: Baseado nas informações do *Profile* e dos *Service Providers*, realiza as deduções que definem que tipo de adaptação deve ser feita.

Adapter: Utiliza os serviços do *Reasoning Deducer* para decidir que tipo de adaptação será feita, tais como: oferecer serviços ao usuário, contactar o provedor de serviços e despachar um agente. Utilizando o Adapater é possível oferecer serviços adaptando-se ao conteúdo e a forma de apresentação. O *Adapter* é o "cérebro" do modelo, ele indica o que

cada componente deve fazer para executar a adaptação.

Profile: Armazena as informações relativas ao perfil do usuário. É composto pela parte das informações armazenadas no *home* do usuário, parte das informações que ficam no dispositivo e pelas informações que são fornecidas pelos *Context Providers* como o *Context Toolkit* [3, 4].

Context Providers: Sistemas que capturam informação do contexto e as disponibilizam. Caso o ambiente tenha sensores, estes sistemas capturam as informações dos sensores e as disponibilizam para que o ICoMP utilize este tipo de informação no momento de adaptação às necessidades do usuário.

A figura 4.5 ilustra o modelo com todos os seus componentes.

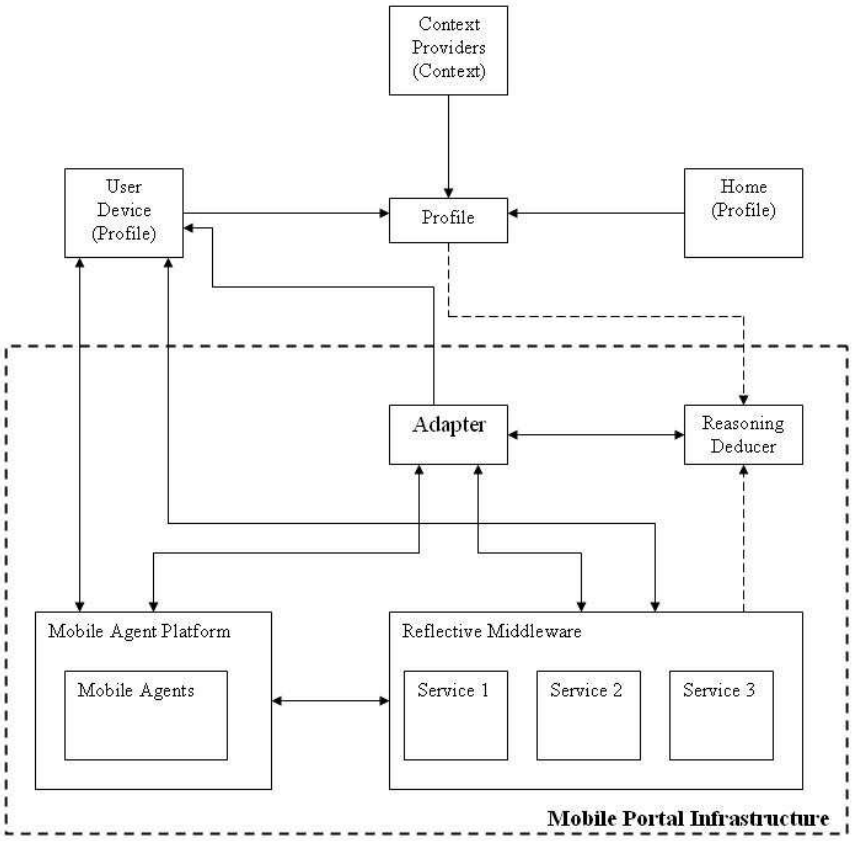


Figura 4.5: Modelo

Interfaces

As interfaces dos principais componentes do modelo ICoMP são:

- *Reasoning Deducer*:

  - IReasoningDeducer - esta interface oferece os seguintes métodos:
    * obterperfil() - recupera as informações relativas ao perfil do usuário.
    * fazerdedução() - baseado nas informações do perfil do usuário este método faz a deduções, indicando que tipo de adaptação pode ser feita.

- *Adapter*:
  - IAdapter - esta interface oferece os seguintes métodos:
    * fazeradaptação() - método que faz a adaptação baseado nos resultados do *ReasoningDeducer*.

- *Mobile Agents*:
  - IMobileAgents - esta interface deve oferecer métodos como:
    * despachar() - método que faz o agente se mover.
    * retornar() - método que faz o agenter retornar ao seu lugar de origem.
    * clone() - método que cria um clone do agente com o mesmo estado do original.
    * ativar() - método que faz com que a execução de um agente seja iniciada ou retomada.
    * desativar() - método que congela a execução de um agente;
    * criar() - método que cria um agente.
    * destruir() - método que destrói um agente e libera seus recursos.

- *Middleware Reflexivo*(Provedores de Serviço) - o *middleware* reflexivo basicamente deve ter estas interfaces:
  - IMetaArquitetura - interface que possibilita a identificação de todas as conexões entre os componentes.
  - IMetaInterface - interface que possibilita a inspeção de todas as interfaces declaradas pelo componente.
  - IMetaInterceptação - interface que possibilita associar/desassociar interceptadores a uma interface em particular.

Serviços

De maneira geral, o usuário pode solicitar serviços ao Portal Móvel, bem como o Portal pode oferecer serviços ao usuário baseando-se no seu perfil, o que caracteriza o *I-centric*, sendo que o atendimento à solicitação ou ao oferecimento de serviços é feito através dos

Provedores de Serviço. Além disso, o usuário pode trazer seus agentes e lançá-los neste novo ambiente ou criar novos agentes neste novo ambiente.

Desta forma, é possível demonstrar como os componentes do ICoMP se relacionam:

Oferecimento de Serviços: Ao se conectar, o ICoMP obtém o *Profile* do usuário. Este *Profile* é enviado para o *Reasoning Deducer* que fará as inferências para descobrir que tipos de serviços podem ser oferecidos ao usuário baseando-se no seu perfil e no que pode ser oferecido pelos *Service Providers*. O *Adapter* analisa as respostas do *Reasoning Deducer* para decidir que tipo de adaptação será feita para o oferecimento de serviços. A Figura 4.6 ilustra esta ação.

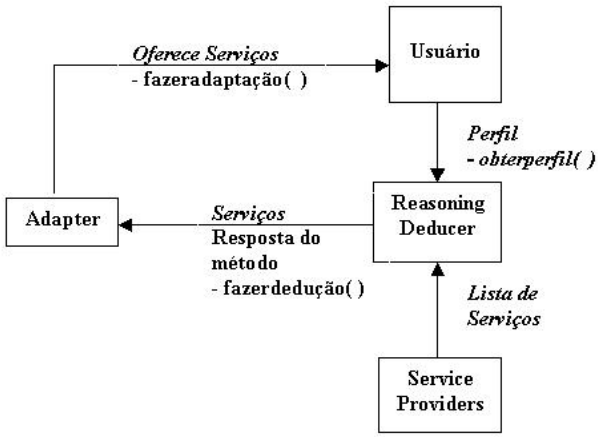


Figura 4.6: Oferecimento de Serviços

Despachar Agentes (1): O usuário requisita ao ICoMP que crie um agente para realizar certo tipo de tarefa. O ICoMP recebe o pedido e o envia à plataforma de agentes móveis que é responsável pelos agentes móveis. A Figura 4.7 ilustra esta ação.

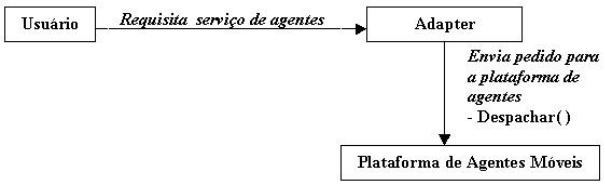


Figura 4.7: Despachar Agentes (1)

Despachar Agentes (2): Ao analisar as respostas do *Reasoning Deducer*, o *Adapter*

pode requisitar a criação de agentes para oferecer algum tipo de serviço ou realizar alguma tarefa para o usuário. A Figura 4.8 ilustra esta ação.

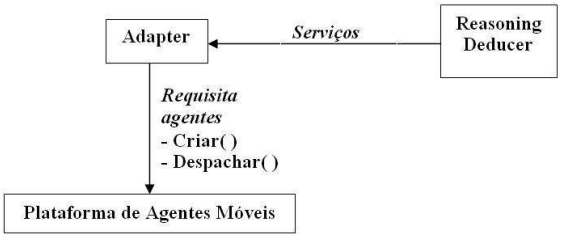


Figura 4.8: Despachar Agentes (2)

Plataforma de Agentes Móveis interage com o Provedor de Serviços: O agente pode requisitar informações sobre um serviço ou até mesmo utilizar algum serviço do provedor de serviços. A Figura 4.9 ilustra esta ação.

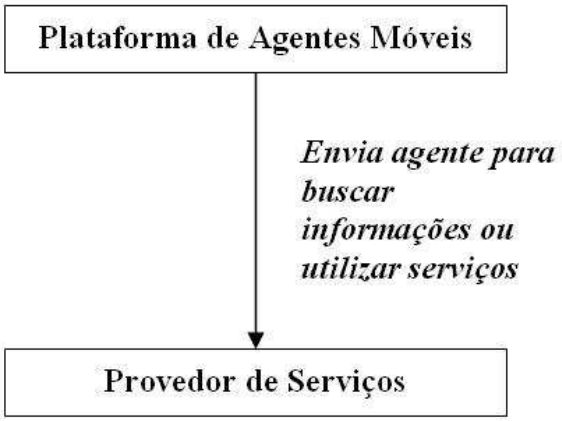


Figura 4.9: Plataforma de Agentes Móveis interage com Provedor de Serviços

Agente retorna ao Usuário: Após realizar alguma tarefa para o usuário, o agente pode retornar o resultado ao usuário dependendo da tarefa executada. Além disso, o sistema pode desejar entrar em contacto com o usuário através de seu agente. A Figura 4.10 ilustra esta ação.

Usuário utiliza serviços: O usuário interage com o *Service Provider*(Provedor de Serviço) para fazer uso do serviço oferecido por este. A Figura 4.11 ilustra esta ação.

Neste caso, temos o seguinte cenário:

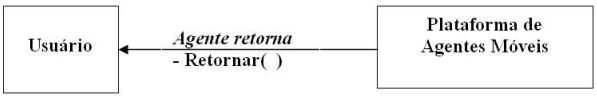


Figura 4.10: Agente Retorna ao usuário

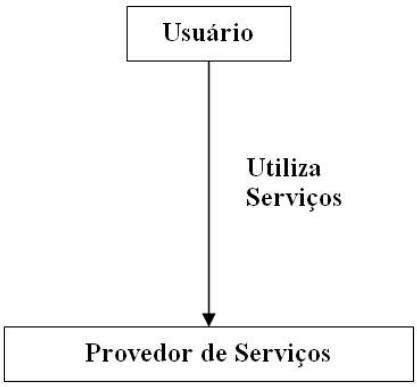


Figura 4.11: Usuário utiliza serviços dos Provedores de Serviços

- O usuário se conecta ao ICoMP através de seu dispositivo móvel. Ao se conectar, o *Adapter* é inicializado. Ele recupera o *Profile* do usuário e o envia para o *Reasoning Deducer*.

- O *Reasoning Deducer* consulta as informações do *Profile* e dos serviços oferecidos pelos *Service Providers* para fazer as inferências e deduções que serão enviados ao *Adapter*.

- O *Adapter* analisa as respostas do *Reasoning Deducer* para decidir que tipo de adaptação será feita para o oferecimento de serviços. Neste caso, o *Adapter* pode oferecer serviços ao usuário, fornecendo toda a infra-estrutura necessária para que o usuário possa acessar os serviços oferecidos.

- Caso o usuário queira utilizar serviços oferecidos pelos *Service Providers e Mobile Agents*, o *Adapter* disponibilizirá os componentes, interfaces e métodos para esta interação, adaptados aos recursos que o usuário dispõe naquele momento.

4.3 Trabalhos Relacionados

Atualmente, alguns projetos tem investigado modelos para adaptação num ambiente de computação móvel:

MyCampus: A Semantic Web Environment for Context - Aware Mobile Services [7] é um ambiente *web* semântico para serviços cientes do contexto, os quais estão em processo de desenvolvimento e validação no campus da Universidade *Carniege Melon*. O ambiente gira em torno de uma coleção de agentes customizáveis, capazes de descobrir e acessar automaticamente serviços de Intranet e Internet, bem como dar assistência aos usuários nas suas diferentes tarefas. O poder e a escalabilidade do ambiente derivam diretamente de ontologias para descrever atributos contextuais, preferências do usuário e serviços *web*, tornando-o possível acomodar facilmente novos agentes para tarefas específicas e novos serviços *web*.

Gaia [6] é uma infra-estrutura desenvolvida pela Universidade de *Illinois* que trata um espaço ativo (*Active Space*) e seus dispositivos de forma análoga a um SO tradicional, fornecendo serviços básicos, incluindo eventos, presença de entidades (dispositivos, usuários e serviços), notificação de contexto, descoberta e sistema de nomes, etc. O Espaço Ativo corresponde a qualquer ambiente de computação ubíqua que pode ser gerenciado pela infra-estrutura. Gaia utiliza uma nova abstração para computação que é chamada de espaço virtual do usuário (*User Virtual Space*). Um espaço virtual do usuário é composto por dados, tarefas e dispositivos que estão associados a um usuário; ele é permanentemente ativo e indenpende de dispositivo; move-se com o usuário e mapeia dados e tarefas no ambiente de computação ubíqua do usuário de acordo com seu contexto atual. Gaia converte um espaço físico e seu dispositivos de computação ubíqua em um sistema de computação programável. Além disso, o Gaia possui uma variedade de agentes [24] para executar tarefas como: descoberta, sentir o contexto e distribuição de eventos.

CAMP: *A Context-Aware Mobile Portal* [5] é uma arquitetura definida pela combinação da tecnologia de Portal Internet com mobilidade pessoal, adaptação de terminal, mobilidade de terminal e conceitos de ciência do contexto, permitindo o desenvolvimento e customização eficiente de serviços. A idéia central é fornecer aos usuários uma variedade de serviços, os quais são automaticamente adaptados ao contexto do usuário. O CAMP compreende um *framework* para hospedar e/ou combinar serviços, um *middleware* para autenticar usuários, gerenciar e acessar informações do contexto e um processo de adaptação que permite várias etapas de adaptação baseadas na informação do contexto.

No *Georgia Tech*, um projeto tem sido conduzido para criar uma arquitetura que dê suporte à criação de aplicações cientes do contexto, esta arquitetura é formada por um conjunto de ferramentas chamado de *Context Toolkit*[3, 4]. A arquitetura usa uma abordagem orientada a objetos que contém três tipos de objetos: *widgets*, *servers* e *in-*

terpreters. Um *widget* do contexto é um componente de software que fornece aplicações com acesso a informação do contexto de seu ambiente operacional. Eles isolam aplicações no que diz respeito à aquisição do contexto. Isto significa que eles atualmente escondem a complexidade dos sensores usados pelas aplicações e abstraem informação do contexto para ajustá-las às necessidades das aplicações. Eles são apontados como blocos reusáveis e customizáveis de sensibilidade do contexto. O *widget* é definido por seus atributos e *callbacks*. Atributos são peças do contexto que ele torna disponível para outros componentes via *polling* ou assinantes (componentes que se registraram para receber informações sobre os atributos). *Callbacks* representam os tipos de eventos que o *widget* pode usar para notificar componentes assinantes.

O ICoMP é um modelo para Portais Móveis semelhante ao CAMP, possibilitando a mobilidade pessoal e de terminal. Com relação ao *Gaia* e ao *Context Toolkit*, o ICoMP dá suporte a utilização de informações do contexto para adaptação às necessidades do usuário e como no *My Campus* fornece serviços baseados em agentes. Além disso, o ICoMP tem como base da sua infra-estrutura um *middleware* reflexivo, que pode ser configurado e reconfigurado dinamicamente de acordo com o contexto, e uma plataforma de agentes móveis que pode oferecer serviços que necessitam de características como assincronismo e distribuição de processamento, dentre outras.

Capítulo 5

Implementação e Avaliação

Neste capítulo serão apresentados alguns aspectos relativos a implementação da infra-estrutura ICoMP bem como informações relativas a sua avaliação. O ambiente de implementação é baseado no sistema operacional *Windows*.

5.1 Aspectos de Implementação

Nesta seção serão apresentadas as interfaces *Grasshopper* e do ReMMoC, e a implementação dos principais componentes do modelo proposto.

Interfaces

As interfaces principais da plataforma de agentes móveis *Grasshopper* necessárias para implementar agentes móveis são ilustradas pela Figura 5.1.

A interface *IAgent* oferece métodos aplicáveis a agentes de uma maneira geral, sejam eles móveis ou não. Por exemplo, o método *getProperties()*, retorna a as propriedades de um agente.

Já a interface *IMobileAgent* oferece métodos específicos de agentes móveis. Por exemplo, o método *move()* faz com que o agente migre de um lugar para o outro.

Logo, é interessante observar que as interfaces oferecidas pelo *Grasshopper* atendem às especificações do modelo ICoMP. Por exemplo, a interface *IAgent* oferece métodos para inicializar, copiar e remover um agente. Já a interface *IMobileAgent* oferece métodos para mover um agente.

No caso do *middleware* reflexivo ReMMoC, as principais interfaces são ilustradas pela Figura 5.2.

Neste caso, é interessante observar que todos os componentes são implementados no padrão OpenCOM, através das interfaces *ICFMetaArchitecture, IConnections, ILifeCycle e IMetaInterface*.

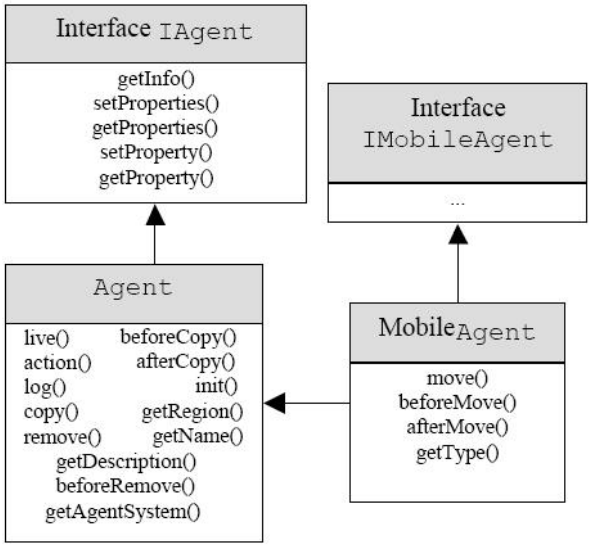


Figura 5.1: Interfaces do Grasshopper

Com relação ao ReMMoC propriamente dito temos as interfaces *IServiceCall* e *IServiceLookup*. A interface *IServiceCall* dá suporte a interoperabilidade com as diferentes tecnologias de *middleware*, por exemplo, IIOP e SOAP. Já a interface *IServiceLookup* dá suporte a descoberta de serviços anunciados através de diferentes protocolos, por exemplo, SLP e UPnP.

As interfaces oferecidas pelo ReMMoC(OpenCOM) também atendem aos requisitos de interfaces para middleware reflexivo recomendados pelo ICoMP. As interfaces *ICFMetaArchitecture, IConnections, ILifeCycle e IMetaInterface)* oferem métodos para identificar as conexões entre os componentes, inspeção das interfaces declaradas pelo componente, dentre outras.

Componentes

Baseado no modelo apresentado anteriormente, os componentes são implementados conforme ilustra a Figura 5.3.

Para exemplificar alguns componentes, usamos uma aplicação que acessa um Portal de Cinema. Neste caso, o portal oferece serviços e informações relativas a cinema, por exemplo, programação, compra de ingressos, dentre outros.

Desta forma, modelamos o *Profile* de um usuário através de suas preferências, sites favoritos e histórico de sites acessados. Baseado no *Profile*, foram gerados fatos em Prolog e uma regra foi contruída para deduzir quais filmes/gêneros podem ser oferecidos ao usuário de acordo com seu perfil. Além disso, foi implementada uma aplicação para

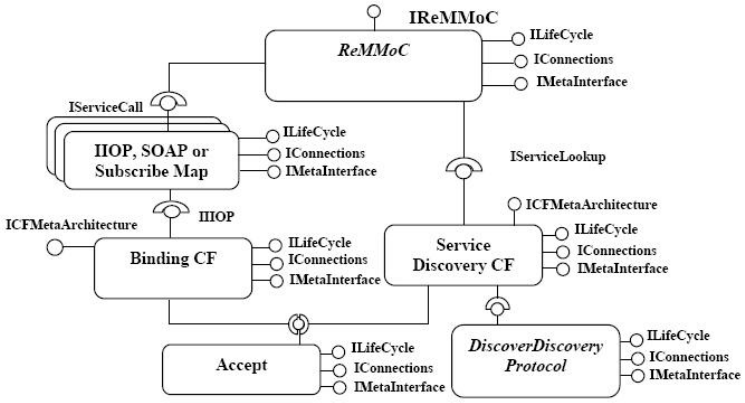


Figura 5.2: Interfaces do ReMMoC

WinCE que obtem a sinopse de um filme e o acesso a agentes móveis através de telefones celulares.

É importante notar o fato de que utilizamos esta aplicação do Portal de Cinema para mostrarmos a viabilidade do nosso modelo. Logo, utilizamos apenas parte de algumas funcionalidades especificadas pelo modelo proposto. Por exemplo, o *Profile* é composto por informações relativas ao contexto, localização, preferências, dentre outras. Porém, apresentamos a modelagem do *Profile* de maneira simplificada através das preferências, sites favoritos e histórico dos sites acessados por um usuário.

A descrição dos componentes que implementam a aplicação de Portal de Cinema segue abaixo:

- *Profile*: É armazenado através de um esquema XML [40] como exemplificado na Figura 5.4. Neste caso, as informações mostradas estão relacionadas as preferências, sites favoritos e histórico dos sites acessados pelo usuário.

  Este esquema é resultado das informações fornecidas pelo *home profile*, *context provider* e a parte do *profile* que está no dispositivo.

- *Context Providers*: sistemas que fornecem informações do contexto através de um interface XML, como o *Context Toolkit*.

- *Reasonig Deducer*: é implementado em *Prolog*, neste caso, chamado de *Prolog Deducer*. Como exemplo, temos os fatos na Figura 5.5. Estes fatos são gerados a partir do perfil do usuário. Neste caso, os fatos descrevem as preferências do usuário, seus favoritos e histórico dos sites acessados. Além disso, temos um fato que descreve o serviço oferecido por um provedor de serviços.

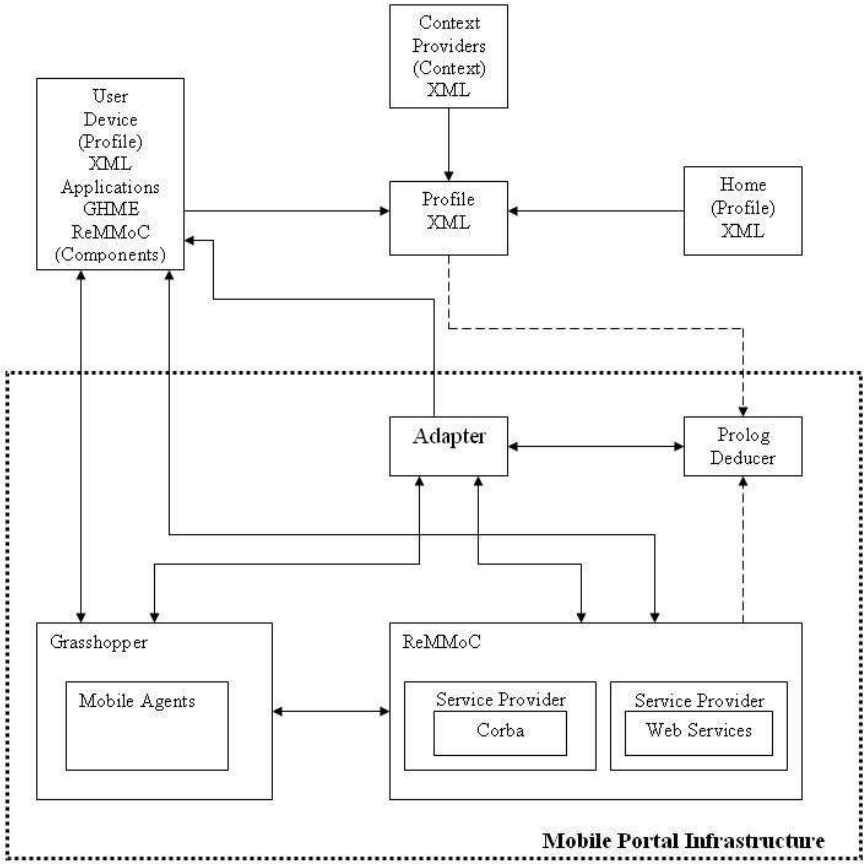


Figura 5.3: Implementação do Modelo

Na Figura 5.6 temos uma regra que faz a dedução do que pode ser oferecido ao usuário baseado nas sua preferências, favoritos, histórico e no que o provedor de serviços tem para oferecer. Neste caso, temos fatos relativos às preferências do usuário com relação a Cinema e uma regra que deduz o gênero de filme que pode ser oferecido a este usuário baseado nas suas preferências. O ambiente de desenvolvimento *Prolog* utilizado é o *Amzi Prolog + Logic Server* [18].

- *Adapter*: Utiliza os serviços do *Prolog Deducer* através da interface *IPrologDeducer* para decidir que tipo de adaptação será feita, interage com o provedor de serviços através do ReMMoC e utiliza os serviços de agentes móveis da plataforma de agentes móveis *Grasshopper*. Além disso, pode oferecer serviços ao usuário, indicando onde está o serviço e fornecendo toda a infra-estrutura em termos de interfaces e métodos para que o usuário possa utilizar os serviços oferecidos.

```
<PREFERENCES>
    <CATEGORY name="entertainment">
    <SERVICE name="movie">
</PREFERENCES>
<FAVORITES>
     <SITE name="www.kinoplex.com.br">
     <CATEGORY name="entertainment">
     <SERVICE name="movie">
</FAVORITES>
<HISTORY>
      <SITE name="www.movie.com">
      <CATEGORY name="entertainment">
      <ACCESS_DATE name="12/05/2003">
</HISTORY>
```

Figura 5.4: Exemplo do Profile em XML

```
user_preferences(entertainment, movie).

user_favorites('www.kinoplex.com.br', entertainment, movie).

user_history('www.movie.com', entertainment, movie, '12/05/2003').

provider_name('shopping', 'Shopping Catuai', '12322222', entertainment,
movie, '15 salas de Cinema', 'Campinas', 'Sao Paulo', 'Brazil', '3232-3232',
'shop@brasil.com.br', 'www.shoppingbrasil.com.br').

                                                          Fatos
```

Figura 5.5: Exemplo dos fatos no Prolog Deducer

- *Service Providers*: são implementados nos mais variados tipos de *middleware* e descobertos através de diferentes protocolos de descoberta de serviços. Contudo, as aplicações acessam os provedores de serviços através da API do ReMMoC, ilustrada pela Figura 5.7. Nesta API, o método *WSDLGet* extrai as informações utilizadas para descrever um serviço tais como, descrição do serviço, métodos e parâmetros de entrada/saída. O método *FindandInvokeOperation* invoca uma operação de um serviço desconhecido, isto é, baseado apenas na sua descrição. Já o método *InvokeOperation* faz o mesmo que o *FindandInvokeOperation* contudo, o serviço é conhecido. O método *CreateOperation* é usado para criar objetos de serviço do ReMMoC. Os métodos *AddMessageValue* e *GetMessageValue* são usados para descrever os parâmetros da operações, seus tipos, dentre outros. O uso do ReMMoC

```
service_offer_based_on_loc_pref_hist_favor(Provider, Category, Service, Description,
Provider_url, Provider_mail) :-
provider_name(_, Provider, _, Category, Service, Description, City, State, Country, _,
Provider_mail, Provider_url),
user_preferences(Category, Service),
user_favorites(_, Category, Service),
user_history(_, Category, Service, _).
```

Regra

Figura 5.6: Exemplo de regra no Prolog Deducer

através desta API torna transparente a heterogeneidade de tecnologias.

```
interface ReMMoC_ICF : IUnknown {
        HRESULT WSDLGet (WSDLService* ServiceDescription, char* XML);
        HRESULT FindandInvokeOperation (WSDLService ServiceDescription, char* OperationName,
                        int Iterations, ReMMoCOPHandler Handler);
        HRESULT InvokeOperation (WSDLService ServiceDescription, ServiceReturnEvent
                ReturnedLookupEvent, char* OperationName, int Iterations, ReMMoCOPHandler  Handler);
        HRESULT CreateOperation (WSDLService ServiceDescription, ServiceReturnEvent
              ReturnedLookupEvent, char* OperationName, int Iterations, ReMMoCOPHandler Handler);
        HRESULT AddMessageValue(WSDLService *ServiceDescription, char* OperationName,
                char* ElementName, ReMMoC_TYPE type, char* direction,  VARIANT value);
        HRESULT GetMessageValue(WSDLService *ServiceDescription, char* OperationName,
                char* ElementName, ReMMoC_TYPE type, char* direction,  VARIANT value);          }
```

Figura 5.7: ReMMoC API

Por exemplo, no caso de um Portal de Cinema, podemos abstrair as operações oferecidas pelos serviços através da linguagem WSDL [28], armazenando esta especificação no cliente. Além disso, os componentes ReMMoC que lidarão com tecnologias heterogêneas também estarão armazenados no cliente.

Nesta implementação, temos um provedor de serviços baseado em Corba Orbacus [30] e implementado em Java [31] que oferece informações sobre programação de cinema em Corba através da interface ilustrada na Figura 5.8.

De acordo com a especificação acima, podemos ter uma aplicação que utiliza os serviços deste provedor. Como exemplo, temos uma aplicação que obtem a sinopse de um determinado filme rodando num Pocket PC, como ilustra a Figura 5.9.

Alguns métodos que implementam a aplicação acima e se utilizam da API do ReMMoC são exemplificados na Figura 5.10.

Neste caso, para obter informações a respeito deste serviço, a aplicação consulta

```
interface MovieService
{
    string GetProgramming();
    // Returns the complete cinema programming
    //( movie, day, hour, ticket prices, ...) */

    string GetProgrammingByGenre(in string genre);
    // Returns the complete cinema programming
    //for a specific kind(genre) e.g. action */

    string GetSynopsis(in string movie);
    // Returns the synopsis of a specific movie*/

    string GetTraillor(in string movie);
    // Returns the Traillor of a specific movie*/

    boolean BuyTickets(in string movie, in string date);
    //  Buy the tickets for a specific Movie and
    //Date(Day, Hour) and returns the result of
    //the transaction e.g. bought or no bought*/

    string GetCinemaAddress(in string cinema );
    // Returns the Address of a specific Cinema
    //e.g. City, Street, Avenue, Number, Phone, e-mail, url*/

    string GetComments(in string movie);
    // Returns comments about a specific Movie.
    //The coments submitted by other users */

    boolean SubmitComments(in string movie, in string coment);
    // Submit comments about a specific Movie  and returns the
    // if the comment could be submitted or not */
);
```

Figura 5.8: Programação de Cinema em Corba: *Movie Service* IDL

um arquivo WSDL [29], ilustrado na Figura 5.11, onde está a descrição do método *GetSynopsis*, que retorna a sinopse de um filme.

Através do ReMMoC é possível interagir com o provedor de Serviços independente da tecnologia em que tenha sido implementado, neste exemplo, os componentes serão configurados para lidar com Corba [27] e SLP(*Service Location Protocol*)[32]. Neste caso, podemos, através da reificação, reconfigurar os componentes para que interaja com outras tecnologias de middleware e protocolos de descoberta de serviços. Por exemplo, este mesmo serviço de Cinema pode ser oferecido através de um *Web Service* e através de um protocolo de descoberta de serviços como UPnP(*Universal Plug and Play*). É importante salientar que devido aos componentes ReMMoC serem construídos baseando-se no padrão OpenCOM [17], as reconfigurações dos componentes são feitas através das interfaces *ILifeCycle*, *IConnections* e *IMetaInterface*. Além disso, o OpenCOM suporta o uso de interceptadores.

- *Mobile Agents*: Os agentes móveis são implementados na plataforma de agentes móveis *Grasshopper* e exportam seus serviços através da interface *IAgent* e *IMobileAgent*. Para o modelo ICoMP estamos implementando o acesso a agentes móveis para telefones celulares. Neste caso, a plataforma de agentes móveis *Grasshopper*

Figura 5.9: Exemplo de Aplicação no Pocket PC

fica na parte fixa da rede e a comunicação entre a aplicação, escrita em J2ME [46], e a plataforma é feita via Servlets [47]. Além disso, o *Grasshopper* tem uma versão para Windows CE rodando um *Personal Java*. Pelo fato da plataforma ter uma versão que executa sobre um *Personal Java*, assim que forem criadas máquinas virtuais Java para outros dispositivos, o *Grasshopper* poderá ser facilmente portado para estes dispositivos.

Logo, é importante observar que o ICoMP possibilita várias formas de interação ao usuário, por exemplo, acessar agentes e serviços em ambientes heterogêneos através de PDAs, telefones celulares e *laptops* como mostrado em nosso cenário. Utilizando o *Grasshopper* ou outra plataforma de agentes móveis é possível somente acessar agentes. Agentes são adequados para operações assíncronas, por exemplo, em um Portal de Cinema os agentes podem buscar opiniões a respeito de filmes e comprar ingressos para uma sessão de cinema, dentre outras. Além disto, é possível implementar agentes com algum nível de inteligência para executar tarefas mais complexas como negociar o preço de uma

```
pReMMoC_ICF->OperationCall(WServ, (unsigned char*)
                           "GetSynopsis",
                           1, &ResultCallback);

pReMMoC_ICF->GetMessageValue(&WServ,
        (unsigned char*) "GetSynopsis",
        (unsigned char*)"movie", ReMMoC_STRING,
         RequestResponse,
        (unsigned char*)"input", &var);

pReMMoC_ICF->GetMessageValue(&WServ,
        (unsigned char*) "GetSynopsis",
        (unsigned char*)"synopsis",
        ReMMoC_STRING,
        RequestResponse,
        (unsigned char*)"output", &var2 );
```

Figura 5.10: Exemplo de código da aplicação

passagem de avião e comprá-la [33]. Do mais, as APIs simples do *Grasshopper* e do ReMMoC facilitam o desenvolvimento de aplicações que lidem com diferentes tecnologias num ambiente móvel.

5.2 Exemplo de utilização

Como exemplo de utilização do ICoMP temos a seguinte situação.

Um usuário, conecta-se a uma nova rede, o Portal Móvel desta rede terá a possibilidade de oferecer serviços a este usuário. Isto será feito baseado no perfil do usuário, através das preferências do usuário, da disponibilidade de recursos e no contexto. Por exemplo, caso um usuário que seja professor universitário e more em Campinas, tenha um perfil indicando que ele gosta de assistir a filmes de humor, viaje para Porto Alegre.

Através de três casos de uso é possível verificar como se dá a interação entre o usuário e o sistema. A Figura 5.12 ilustra os caos de uso. A seguir são apresentadas breves descrições dos casos de uso e os fluxo principal e secundários de cada um, se houver.

- Conectar

  Este caso de uso representa a ação do usuário de se conectar a rede de Porto Alegre. Neste caso, o Portal de Porto Alegre verifica a conexão deste usuário e monta o seu perfil.

- Consultar Perfil/Adaptar

  Neste caso de uso, o Portal de Porto Alegre consulta o perfil do usuário para que possa fazer a adaptação às necessidades deste.

```
<service name="MovieService">
        <documentation>My Movie Portal Service</documentation>
        <port name="MoviePort" binding="MovieBinding"></port>
</service>
<operation name="GetSynopsis">
            <input message="GetMovieSynopsisInput"/>
            <output message="GetMovieSynopsisOutput"/>
</operation>
<element name="MovieSynopsisRequest">
            <complexType>
                <all>
                    <element name="movie" type="string"/>
                </all>
            </complexType>
</element>
<element name="MovieSynopsis">
            <complexType>
                <all>
                    <element name="synopsis" type="string"/>
                </all>
            </complexType>
</element>
<message name="GetMovieSynopsisInput">
        <part name="body" element="MovieSynopsisRequest"/>
</message>
<message name="GetMovieSynopsisOutput">
        <part name="body" element="MovieSynopsis"/>
</message>
```

Figura 5.11: Descrição do Serviço de Cinema: *Movie Service* WSDL

- Oferecer Serviços

  Corresponde à ação do Portal Móvel de Cinema da cidade de Porto Alegre oferecer ao usuário a programação de cinema relativa a filmes de humor adaptada ao perfil do usuário. Por exemplo, caso o usuário esteja num celular, será disponibilizado título, a hora e o local, caso esteja num *PDA* será disponibilizado, o título, a hora, o local e a sinopse do filme, caso esteja num *notebook* será disponibilizado o título, a hora, a sinopse, o local, links para se assistir ao trailer, possibilidade de se comprar os ingressos pela Internet, dentre outras. A Figura 5.13 ilustra os diferentes dispositivos que podem ser atendidos pelo modelo ICoMP.

Além disso, o usuário pode utilizar agentes móveis para comprar ingressos de uma sessão de cinema, buscar opiniões a respeito de um filme, dentre outros. É interessante salientar que utilizando agentes móveis obtemos um ganho com relação a execução das tarefa com maior autonomia e inteligência, possibilitando uma melhor utilização dos recursos disponíveis num ambiente móvel.

O diagrama de sequência ilustrado na figura 5.14 dá uma visão geral de como ocorrem as interações entre os componentes para este exemplo de utilização.

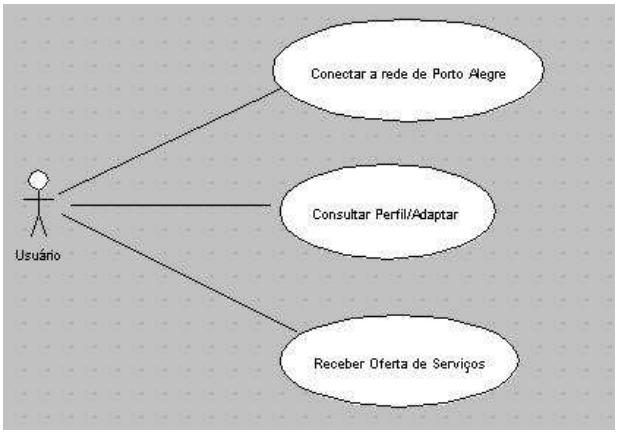


Figura 5.12: Casos de Uso

Figura 5.13: Variedade de Dispositivos: *Notebook*, PDA e telefone celular.

5.3 Avaliação

Com relação à avaliação do ICoMP, alguns dados são fornecidos a respeito dos componentes que ficam na parte cliente do Portal. Esta avaliação é feita em termos de espaço de armazenamento do ReMMoC e do *Grasshopper*.

A Tabela I exemplifica o espaço de armazenamento necessário para o ReMMoC em dispositivos StrongARM, por exemplo, pocket pc. Analisando o núcleo do *framework* e as configurações de exemplo, verifica-se que estes ocupam espaços de armazenamento menores que 300 Kbytes, adequados a dispositivos como os *handheld*, *pocket pcs*, dentre outros. Em [15] podem ser encontradas mais informações relativas ao espaço de armazenamento ocupado pelos componentes ReMMoC.

Com relação ao *Grasshopper*, foi implementada uma aplicação que acessa serviços de agentes móveis através de telefones celulares. Neste caso, o espaço de armazenamento depende da aplicação pois, a plataforma de agentes móveis (*Grasshopper*) está na parte fixa da rede. Por exemplo, o tamanho de uma aplicação que envia agentes para comprar

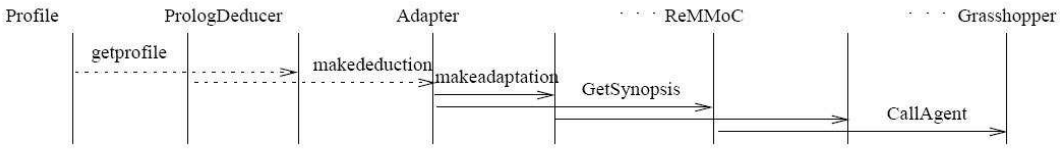


Figura 5.14: Diagrama de Sequência

Tabela 5.1: Tabela I: Espaço de Armazenamento do ReMMoC para dispositivos StrongARM

Componente	KBytes
Componentes do Núcleo ReMMoC	72.5
ReMMoC (Corba e SLP)	250.5
ReMMoC (SOAP e UPnP)	246.5
OpenCOM	27.5

ingressos para uma sessão de cinema fica em torno de 30Kbytes.

Comparamos agora nossa implementação com uma versão do *Grasshopper* para WinCE, a qual é portável para dispositivos baseados em processadores MIPS, ARM e SH-3. Nesta versão, o espaço de armazenamento necessário é de aproximadamente 700Kbytes. Este valor também é adequado para dispositivos como *handheld*, *pocket pcs*, dentre outros.

Logo, é importante observar que tanto nossa implementação quanto o *Grasshopper* para WinCE são adequados para dispositivos móveis em termos de espaço de armazenamento. Porém, nossa implementação oferece serviços através de provedores de serviços baseados em middleware reflexivo e, também, serviços de agentes móveis através de um plataforma para agentes móveis. O *Grasshopper* para WinCE só pode oferecer serviços de agentes móveis.

Capítulo 6

Conclusão

A convergência entre computação móvel e telecomunicações, com destaque para a telefonia celular, Internet e redes sem fio, criou um desafio para os desenvolvedores de infra-estruturas de software para ambientes móveis.

Embora, já existam propostas que abordam este tema, a maioria delas não lida de maneira satisfatória com a necessidade de adaptação a novas tecnologias, fazem uso de pouca informação do contexto para se adaptarem ao ambiente e não oferecem serviços baseados em agentes móveis.

Este trabalho contribui apresentando uma nova abordagem para o desenvolvimento de uma infra-estrutura para Portais Móveis que possibilita uma adaptação mais refinada às necessidades do usuário, bem como às constantes mudanças que ocorrem num ambiente móvel, e oferece serviços através de *middlewares* reflexivos e plataformas de agentes móveis. Esta abordagem é original no sentido de integrar estas duas maneiras de oferecimento de serviços. Por sua natureza distribuída o ICoMP é escalável e permite dois tipos de acesso: a informações e serviços oferecidos pelos provedores de serviços, e acesso a serviços oferecidos pela plataforma de agentes móveis como despachar e receber agentes. Além disso, o *Adapter* assume um papel de "maestro na orquestração de serviços".

Logo, o modelo proposto vem de encontro aos requisitos das aplicações em comércio móvel, pois permite aos usuários acesso a informação e serviço, na hora e lugar certos, de acordo com suas necessidades e recursos disponíveis.

Com relação à implementação, é interessante ressaltar o fato dela ser componentizada, o que a torna flexível. Neste caso, é possível acrescentar novas tecnologias, técnicas de dedução, serviços, de maneira a não comprometer o funcionamento da infra-estrutura do Portal Móvel. Através da aplicação de Cinema, é possível demonstrar as diferentes formas de interação que o ICoMP possibilita ao usuário, como o acesso a agentes e serviços implementados em diferentes tipos de tecnologias, sendo que as tecnologias utilizadas como o ReMMoC e o *Grasshopper*, atendem aos requisitos de aplicações para ambientes

móveis. Além disso, a seção de avaliação demonstra que o custo de armazenamento dos componentes que dão suporte as aplicações ser adequado a dispositivos móveis.

Os próximos passos estão sendo dados no sentido de desenvolver aplicações que utilizem todos os recursos que a infra-estrutura ICoMP pode oferecer através da implementação de diferentes *Profiles*, utilização de outras tecnologias de *middleware* e protocolos de descoberta de serviços, interação entre agentes móveis de diferentes plataformas, dentre outras.

Além disso, a infra-estrutura do Portal Móvel deverá se adequar aos padrões que estão surgindo nos sistemas adaptáveis, tais como: CC/PP (*Composite Capability/Preference Profiles*) [20], um padrão para descrever as preferências do usuário e as capacidades dos dispositivos, padrão de ontologias como o DAML+OIL (*DARPA Agent Markup Language + Ontology Interface Layer*) [22] e o RDF (*Resource Description Framework*) [21].

Referências Bibliográficas

[1] WAPMetrics. http://uolwap.uol.com.br/metrics1.php.

[2] K. Dulaney, R. Egan, e E. Purchase, How to Build a Wireless Office: The Next Wireless Revolution, a Gartner Group Research Note, Fevereiro, 2000.

[3] Dey, A.K., Salber, D., Futakawa, M. Abowd, G.D. *An Architecture to Support Context Aware Computing*, GVU Technical Report GIT-GVU-99-23. Junho 1999

[4] Salber, D., Dey A.K., Abowd, G.D. *The Context Toolkit: Aiding the Development of Context-Enabled Applications. In Proceedings of the CHI 99 Conference on Human Factors in Computing Systems: The CHI is the Limit, pp. 434-441, Pitsburgh, PA, Maio 1999, ACM Press.*

[5] Mandato, D., Kovacs, E., Hohl, F., Amir-Alikhani, H. *CAMP: A Context - Aware Mobile Portal*, IEEE Communicatios Magazine. pp. 90-97 Janeiro 2002.

[6] Roman, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R.H., Narstedt K. *"A middleware infra-structure for Active Spaces"*, IEEE Pervasive Computer, v.1, n. 4, p. 74-83, Outubro - Dezembro, 2002.

[7] Sadeh, Norman, *A Semantic Web Environment for Context-Aware Mobile Services*, Wireless World Research Forum Conference, Stockholm, Setembro, 2001.

[8] Arbanowski, St., van de Meer, S., Steglich, St., Popescu-Zeletin, R. *The Human Communication Space: Towards I-centric Communications.* v. 5, Personal and Ubiquitous Computing, Issue 1, 2001.

[9] Dey, A.K., Abowd, G.D. *Towards a better understanding of context and context - awareness.* GVU Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology, 1999.

[10] Pascoe, J. *The Stick-e Note Architecture: Extending the Interface Beyond the User.* International Conference on Intelligent User Interfaces, Orlando, Florida, Estados Unidos. ACM. pp. 261-264, 1997.

[11] Pascoe, J., Ryan, N.S. e Morse, D.R. *Issues in Developing Context-Aware Computing*. Proceedings of the International Symposium on Handheld and Ubiquitous Computing. (Karlshure, Alemanha, Setembro. 1999), Springer - Verlag, pp. 208- 221.

[12] Schilit, B.N., Adams, N.I. e Want, R. *Context - Aware Computing Applications*. Proceedings of the Workshop on Mobile Computing Systems and Applications, IEEE Computer Society, Santa Cruz, CA, pp. 85-90, 1994.

[13] Thomas Ledoux. *OpenCORBA: A Reflective Open Broker*. In Proceedings of Reflection' 99, number 1616 in LNCS, pp 197-214, St. Malo, França, Julho 1999. Springer-Verlag.

[14] Kon, F., Costa, F., Campbell, R., Blair, G. *The Case for Reflective Middleware*. Communications of the ACM, 45(6), Junho 2002.

[15] Grace, P., Blair, G., and Samuel, S. *"ReMMoC: A Reflective Middleware to Support Mobile Client Interoperability"*, In Proceedings of International Symposium on Distributed Objects and Applications (DOA), Sicily, Itália, Novembro, 2003.

[16] Grasshopper - The Agent Platform - Technical Overview , IKV++ GmbH, 1999

[17] Clarke, M., Blair, G., Coulson, G. and Parlavantzas, N. *"An Efficient Component Model for the Construction of Adaptative Middleware"*. In Proceedings of Middleware 2001, Heidelberg, Alemanha. Novembro 2001.

[18] Manual Amzi Prolog + Logic Server, Versão 6.2.14. http://www.amzi.com

[19] The Component Object Model: A Technical Overview. http://www.microsoft.com

[20] Composite Capabilities/Preference Profiles: Requirements and Architecture W3C Working Draft 21 Julho de 2000.

[21] Resource Description Framework (RDF): Concepts and Abstract Syntax W3C Working Draft. Setembro, 2003

[22] DAML+OIL Reference Description. Março, 2001.

[23] D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran, and J. White. *MASIF: The OMG mobile agent system interoperability facility*. In Proceedings of Mobile Agents, pages 50–67, Stuttgart, Alemanha, Setembro 1998.

[24] Campbell, R. H., Ranganathan, A. *A Middleware for Context-Aware Agents in Ubiquitous Computing Environments*. Middleware 2003, Rio de Janeiro, Junho 2003.

[25] Chess, D., Harrison, C., Kershenbau, A. *Mobile Agents: Are they a good idea?*. Technical Report, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, New York, Março, 1995.

[26] Gray, R. S., Kotz, D., Nog, S., Rus, D., Cybenko, G. *Mobile agents for mobile computing*. Technical Report, Dartmouth, Maio, 1996.

[27] Orfali, R., Harkey D. Client/Server Programming in Java with CORBA, 2.Ed.

[28] Web Services Description Language (WSDL) 1.1 W3C Note, Março 2001.

[29] Grace, P., Blair, G., and Samuel, S.*A Marriage of Web Services and Reflective Middleware to Solve the Problem of Mobile Client Interoperability*, International Symposium on Information and Communication Technologies, Dublin, Irlanda, Setembro 2003.

[30] Manual ORBacus for C++ and Java. http://www.iona.com

[31] Thinking in Java, Bruce Eckel. http://mindview.net/Books.

[32] RFC 2608 - Service Location Protocol, Version 2.

[33] Matos, F. M. and Madeira, E. R. M. *An Automated Negotiation Model for M-commerce Using Mobile Agents*, ICWE 2003, LNCS, vol. 2722, pp. 72-75.

[34] Long, S., et al. (1996). *Rapid Prototyping of Mobile Context-aware Applications: The Cyberguide Case Study*. 2nd ACM International Conference on Mobile Computing and Networking (MobiCom'96), Novembro 1996.

[35] Korkea-aho, M. (2000). Context-Aware Applications Survey. http://www.hut.fi/ mkorkeaa/doc/context-aware.html

[36] Kay, A. Computer Software. Scientific American, 3(251): 53-59, 1984.

[37] OMG. http://www.omg.org/docs/orbos/98-03-09.pdf

[38] Gialdi, M. V., Madeira, E. ICOMP: Um Modelo para Portais Móveis baseado em Middleware Reflexivo e Agentes Móveis. V Workshop de Comunicações Sem Fio e Computação Móvel, São Lourenço, MG, Outubro 2003.

[39] Gialdi, M. V., Madeira, E., Grace, P., Blair, G. "ICoMP: A Mobile Portal Model based on Reflective Middleware and Mobile Agents". IEEE/IFIP Mobility Aware Technologies and Applications, Outubro, 20- 22, 2004, Florianopolis, Brasil. (aceito para publicação).

[40] The World Wide Web Consortium W3C, Maio 2002, XML Schema. http://www.w3.org/TR/xmlschema-1/

[41] RFID. Radio Frequency Identification. http://www.rfid.org.

[42] Microsoft Corporation. Universal Plug and Play Device Architecture. Technical report, 2000.

[43] Sun Microsystems Inc. Jini Architectural Overview. Technical report, 2001.

[44] Salutation Consortium. White Paper. Salutation Architecture: Overview. 1998.

[45] SOAP 1.1 Specification: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[46] Sun Microsystems Inc. Java 2 Micro Edition. http://java.sun.com/j2me/

[47] Sun Microsystems Inc. Java Servlet Technology. http://java.sun.com/products/servlet/