

Universidade Federal de Uberlândia - UFU
Faculdade de Computação - FACOM

Agentes Móveis Inteligentes para Suporte à Ubiquidade de Serviços Multimídia

Autor: Juliana Oliveira de Carvalho
Orientador: Prof. Dr. Luís Fernando Faina

Dissertação de Mestrado apresentada à Faculdade
de Computação como parte dos requisitos para obten-
ção do título de Mestre em Ciência da Computação.
Área de Concentração: **Redes de Computadores.**

Dezembro - 2006

Uberlândia, MG - Brasil

Universidade Federal de Uberlândia - UFU
Faculdade de Computação - FACOM

Agentes Móveis Inteligentes para Suporte à Ubiquidade de Serviços Multimídia

Autor: Juliana Oliveira de Carvalho
Orientador: Prof. Dr. Luís Fernando Faina

Dissertação de Mestrado apresentada à Faculdade
de Computação como parte dos requisitos para obten-
ção do título de Mestre em Ciência da Computação.
Área de Concentração: **Redes de Computadores.**

Banca Examinadora

Luiz Gonzaga da Silveira Júnior, Dr. UNISINOS
Carlos Roberto Lopes, Dr. FACOM/UFU
Luís Fernando Faina, Dr. FACOM/UFU

Dezembro - 2006

Uberlândia, MG - Brasil

Dados Internacionais de Catalogação na Publicação (CIP)

C331a Carvalho, Juliana de Oliveira, 1972-

Agentes móveis inteligentes para suporte à ubiquidade de serviços de multimídia / Juliana de Oliveira Carvalho. - 2006.
117 f. : il.

Orientadora: Luís Fernando Faina.

Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.

Inclui bibliografia.

1. 1. Sistemas de telecomunicação - Teses. 2. Internet (Redes de computação) - Teses. 3. Teleconferência - Teses. 4. Agentes inteligentes (Software) - Teses. 5. Sistemas operacionais distribuídos (Computadores) - Teses. I. Faina, Luís Fernando. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 621.391

Resumo

Carvalho, Juliana Oliveira, “Agentes Móveis Inteligentes para Suporte à Ubiquidade de Serviços Multimídia”. Dissertação de Mestrado - FACOM/UFU, Uberlândia, MG. Dezembro 2006.

A convergência das telecomunicações com as redes de computadores tem permitido a disponibilização de serviços cada vez mais sofisticados e por serviços que sejam entregues com rapidez, agilidade, segurança, confiabilidade e principalmente que atendam as necessidades dos usuários. Nesse contexto os serviços serão adaptáveis e sensíveis ao contexto, ou seja, serão dependentes da mobilidade pessoal, de serviço e de terminal.

Mobilidade global requer a coordenação de uma grande variedade de provedores de serviço, garantia de compatibilidade entre as redes *backbone* (Telecom Fixa, Telecom Móvel e Internet), e obtenção de acordos entre operadoras de redes. Desta forma usuários poderão ter acesso a serviços adaptados às suas necessidades a partir de diferentes localizações e dispositivos. O uso da tecnologia de agentes móveis inteligentes proporciona autonomia e flexibilidade nos processos de personalização e disponibilização de serviços quando da necessidade de suporte à mobilidade pessoal e de serviço, e realça a tecnologia de objetos distribuídos.

Este trabalho parte de uma arquitetura que estabelece um Ponto de Referência entre Provedores, e propõe o uso de paradigmas de objetos distribuídos e agentes móveis inteligentes para integrar um ambiente de comunicação que permita o acesso a um conjunto de serviços a partir de diferentes localizações e dispositivos. Para avaliar a proposta, implementa-se um protótipo onde aspectos de mobilidade e de adaptação do serviço são investigados.

Palavras-chave: Serviços Multimídia, Agentes Móveis Inteligentes, Ubiquidade, Mobilidade Pessoal e de Serviço, Objetos Distribuídos.

Abstract

Carvalho, Juliana Oliveira, “Intelligent Mobile Agent for Ubiquity Suport of Multimedia Services”. Master Thesis - FACOM/UFU, Uberlândia, MG. December 2006.

The telecommunication convergence with the network has allowed the availability of the services more and more sophisticated and by the services that be delivered, with quickness, agility, security, reliability and mainly that take into consideration the necessities of the users. In the context the services will be adaptable and sensible to the context, so they will be dependents of the personal, service and terminal mobility.

Global mobility requires the coordination of a great variety of providers, warrant of compatibility among the backbone nets (Fix Telecom, Mobile Telecom and Internet) and acquisition of accordance among network operators. By this way users can have access to adapted services to their necessities from different localizations and devices. The use of the technology of intelligent mobile agents proportion autonomy and flexibility in the process of personalization and availability of services when it exists necessity of support to the personal and service mobility and enhance the distributed objects technology.

This work starts from the architerture that establishes provides reference point, and proposes the use of distributed objects, paradigms and intelligent mobile agents to integrate communication environment that allows the access to a localization and devices. To evaluate the proposal, it implements a prototype where aspects of mobility and adaptation services are investigared.

Keywords: Multimedia Services, Intelligent Mobile Agent, Ubiquity, Personal and Service Mobility, CORBA.

Para o meu pai Áviton (in memorian) e minha mãe Wilma
Para o meu esposo Rayner e minha filha Ana Paula

“ Não fuja de seus problemas nem se desespere. Encare-os de frente com coragem e determinação, pois se não resolvê-los no dia de hoje, certamente terá que fazê-lo no dia de amanhã porque eles continuarão existindo enquanto não forem resolvidos, prolongando o seu sofrimento.”

Fernando Toscano

Agradecimentos

A DEUS, pela condição de poder me aperfeiçoar, por nos manter firme, mesmo na dificuldade, pois sempre havia alguém para nos trazer a palavra do Senhor e então nos mostrar o caminho.

Ao professor, Luís Fernando Faina, por suas orientações e confiança no trabalho.

Ao meu querido esposo Rayner por sua compreensão, apoio e carinho em todos os momentos, principalmente nos turbulentos. A minha querida filha Ana Paula pelo carinho e amor apesar de toda falta de tempo e estresse.

A minha mãe Wilma, minhas irmãs Flávia e Cárthea, ao meus sogros, Nilta e Walter, a minha cunhada Franciane, a minha querida tia Jurany, aos meus primos Lucita, Marcélia e José por seu apoio incondicional, por suas palavras de motivação e fé.

Aos meus amigos, Lívia, Luis Reis, Viviane, Giselle, Eduardo, Joslaine, Milena por estarem presentes nos momentos difíceis.

Ao Prof. Eleri Cardozo (DCA-FEEC-Unicamp) e a Prof. Eliane G. Guimarães (CENPRA - Campinas) pela implementação da especificação MAF utilizada neste projeto.

A Faculdade de Computação da Universidade Federal de Uberlândia por proporcionar infraestrutura necessária à dissertação e ao CNPq pela concessão da bolsa de estudos.

A Faculdades Integradas de Mineiros (FIMES), pela concessão de licença do trabalho.

Sumário

Lista de Figuras	x
Acrônimos	xiii
1 Introdução	1
1.1 Objetivo do Trabalho	2
1.2 Estrutura da Dissertação	3
2 Ubiquidade de Serviços	4
2.1 Papel da Mobilidade	4
2.1.1 Características da Mobilidade	7
2.1.2 Suporte à Mobilidade de Terminal	9
2.1.3 Suporte à Mobilidade Pessoal	10
2.1.4 Suporte à Mobilidade de Serviço	11
2.2 Ubiquidade de Serviço	12
2.2.1 Ubiquidade via Mobilidade de Terminal	12
2.2.2 Ubiquidade via Mobilidade Pessoal	13
2.2.3 Ubiquidade via Mobilidade de Serviço	14
3 Sistemas Baseados em Agentes	16
3.1 Inteligência Artificial	16
3.1.1 Engenharia do Conhecimento	17
3.1.2 Aquisição do Conhecimento	18
3.1.3 Representação do Conhecimento	19
3.1.4 Formas de Representação do Raciocínio	20
3.2 Definições e Características de Agentes	22

3.2.1	Princípios de Agentes Inteligentes	25
3.2.2	Princípios de Agentes Móveis	27
3.3	Especificação <i>Mobile Agent Facility</i> (MAF)	29
4	Modelo de Agentes para Suporte à Mobilidade de Serviços	32
4.1	Visão Geral do Modelo de Suporte à Mobilidade	33
4.2	Descrição dos Componentes do PR-Prvds	35
4.3	Interações entre os Componentes do PR-Prvds	40
4.3.1	Pré e Pós-Condições nos Cenários	40
4.3.2	<i>Login</i> e <i>Logout</i> em/de um Provedor Contratado	40
4.3.3	<i>Login</i> em um Provedor Visitado	43
4.3.4	<i>Logout</i> de um Provedor Visitado	46
4.3.5	Listar Serviços Contratados e/ou Adaptáveis	46
4.3.6	Selecionar Serviço no Provedor Contratado	50
4.3.7	Selecionar Serviço no Provedor Visitado	52
4.3.8	Finalizar Serviço no Provedor Contratado	55
4.3.9	Finalizar Serviço no Provedor Visitado	57
5	Aspectos da Implementação e Resultados	59
5.1	Infra-Estrutura da Arquitetura Proposta	59
5.1.1	Modelo de Tecnologias	60
5.1.2	Infra-Estrutura do PR-Prvds	61
5.2	Aspectos de Implementação do Agente	62
5.2.1	Aspectos de Implementação da Mobilidade	62
5.2.2	Aspectos de Implementação da Inteligência	65
5.3	Cenário de Validação da Arquitetura Proposta	86
5.4	Sessão de Serviço no Provedor Contratado	87
5.4.1	Serviço Completamente Adaptável no Provedor Contratado	89
5.4.2	Serviço Parcialmente Adaptável no Provedor Contratado	90
5.4.3	Serviço Não Adaptável no Provedor Contratado	90
5.5	Sessão de Serviço no Provedor Visitado	91
5.5.1	Serviço Completamente Adaptável no Provedor Visitado	93
5.5.2	Serviço Parcialmente Adaptável no Provedor Visitado	95

5.5.3	Serviço Não Adaptável no Provedor Visitado	96
5.6	Avaliação e Comentários	98
6	Conclusão	99
	Referências Bibliográficas	101

Lista de Figuras

2.1	Entidades móveis: Usuários, <i>Views</i> , Plataformas e Provedores.	5
2.2	Serviços Centrados no Usuário.	6
2.3	Ambiente de Comunicação.	8
2.4	Integração UMTS com Redes Fixas.	9
2.5	Perfil do Usuário.	11
2.6	Aspectos de Interoperabilidade entre Redes.	12
2.7	Portabilidade do Serviço com Suporte no Provedor de Serviço.	14
3.1	O Padrão Arquitetural de Agentes em Camadas.	24
3.2	Modelo de Agente.	25
3.3	Níveis de Mobilidade de Agentes.	28
3.4	Sistema Agente.	30
4.1	Modelo Proposto para Suporte à Mobilidade.	33
4.2	Domínio Contratado de um Usuário.	34
4.3	Arquitetura do provedor para Suporte à Mobilidade.	35
4.4	<i>Login</i> e <i>Logout</i> em/de um Provedor Contratado.	42
4.5	Rede de Petri para <i>Login</i> em/de um Provedor Contratado.	43
4.6	<i>Login</i> em um Provedor Visitado.	44
4.7	Rede de Petri para <i>Login</i> em/de um Provedor Visitado.	45
4.8	Listar serviços contratados e/ou adaptáveis.	47
4.9	Rede de Petri para serviços contratados e/ou adaptáveis.	48
4.10	Listar serviços contratados e/ou adaptáveis (migração do agente de usuário pre- definida).	49
4.11	Seleção de Serviço no Provedor Contratado.	51
4.12	Rede de Petri para Seleção de Serviço no Provedor Contratado.	52

4.13 Seleção de Serviço no Provedor Visitado.	53
4.14 Rede de Petri para Seleção de um Serviço no Provedor Visitado.	54
4.15 Finalização de Serviço no Provedor Contratado.	56
4.16 Rede de Petri para Finalização de Serviço no Provedor Contratado.	57
4.17 Finalização de Serviço no Provedor Visitado.	58
5.1 Arquitetura do provedor para Suporte à Mobilidade.	60
5.2 Instância do <code>tnameserv</code>	63
5.3 Instância do <code>MAFFinder</code>	63
5.4 Instância do Provedor de Serviço PR-Prvds.	63
5.5 Criação e registro da agência beta.	64
5.6 Criação e registro do agente MyDearAgnt na agência Beta.	64
5.7 O agente MyDearAgnt migra da agência Beta para Alfa.	65
5.8 Árvore que Representa a Estrutura do Conhecimento.	66
5.9 Árvore que Representa um Serviço de Áudio.	67
5.10 Árvore que Representa um Serviço de Vídeo.	67
5.11 Suporte à Mobilidade no Serviço de Vídeo-Conferência.	86
5.12 Suporte à Mobilidade no Serviço de Conteúdo Sob Demanda.	87
5.13 Agência Beta é criada no Provedor Contratado.	88
5.14 O agente é criado na Agência Beta.	88
5.15 Árvore que Representa as Propriedades do Serviço de Áudio-Vídeo-Conferência.	88
5.16 Árvore que Representa as Propriedades da Estação de Trabalho.	89
5.17 Serviço Completamente Adaptável no Provedor Contratado.	89
5.18 Árvore que Representa as Propriedades do Palm.	90
5.19 Serviço Parcialmente Adaptável no Provedor Contratado.	90
5.20 Árvore que Representa as Propriedades do Celular.	91
5.21 Serviço Não Adaptável no Provedor Contratado.	91
5.22 Agência Beta é criada no Provedor Contratado.	92
5.23 Agência Alfa é criada no Provedor Visitado.	92
5.24 O agente é criado na Agência Beta.	92
5.25 O agente migra de Beta para a Alfa.	93
5.26 A Agência Alfa recebe o Agente.	93
5.27 Árvore que Representa as Propriedades da Estação de Trabalho.	93

5.28	Árvore que Representa as Propriedades do Provedor Visitado.	94
5.29	Árvore que Representa as Propriedades do Perfil do Usuário.	94
5.30	Serviço Completamente Adaptável no Provedor Visitado.	94
5.31	Árvore que Representa as Propriedades do Perfil do Usuário.	95
5.32	Árvore que Representa as Propriedades do Provedor Visitado.	95
5.33	Árvore que Representa as Propriedades do Palm.	95
5.34	Serviço Parcialmente Adaptável no Provedor Visitado.	96
5.35	Árvore que Representa as Propriedades do Celular.	96
5.36	Árvore que Representa as Propriedades do Provedor Visitado.	97
5.37	Árvore que Representa as Propriedades do Perfil do Usuário.	97
5.38	Serviço Não Adaptável no Provedor Visitado.	97

Acrônimos

ACAI	Agent-based Context-aware Infrastructure
ATP	Agent Transfer Protocol
CoBrA	Context-aware Broker Architecture
CORBA	Common Object Request Broker Architecture
GercRcrs	Gerência de Recursos
GPS	General Problem Solving
IA	Inteligência Artificial
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
InfoPrfl	Informação de Perfil
JESS	Java Expert System Shell
MAF	Mobile Agent Facility,
OMG	Object Management Group
ORB	Object Request Broker
PR-Prvds	Ponto de Referência entre Provedores
RPC	Remote Procedure Call
SBA	Sistemas Baseados em Agentes
SessAssServ	Sessão de Acesso e Serviço

SMA Sistemas MultiAgentes

TINA Telecommunication Information Networking Architecture

tnameserv Transient Naming Service

UMTS Universal Mobile Telecommunications System

Capítulo 1

Introdução

Nos últimos anos, o mundo tem sido palco de grandes mudanças, tais como: a revolução na comunicação e nos computadores, a ubiquidade da Internet, o uso cada vez maior da computação móvel e o aumento na banda de rede para valores onde serviços multimídia efetivos são agora viáveis. Essas mudanças nos conduzem a um cenário onde a comunicação é ubíqua, cada um de nós poderá se conectar a infra-estrutura de telecomunicações global onde quer que estejamos e pequenos dispositivos farão parte da infra-estrutura de comunicação/computação e a informação será digital e acessível através da rede [1].

Neste contexto, além da desregulamentação do setor, o crescimento do comércio internacional de equipamentos e serviços tem contribuído substancialmente para o seu crescimento e gerado pelo menos três ganhos econômicos: novos produtos e serviços contemplando principalmente comunicação multimídia e interatividade, aumento da competição entre operadoras trazendo baixa de preços e investimento adicional [2].

O mercado de telecomunicações está cada vez mais exigente, pois o papel dos serviços de telecomunicações tem aumentado diante da evolução tecnológica, principalmente pela demanda crescente de novos serviços, e por serviços que sejam entregues com rapidez, agilidade, segurança, confiabilidade e principalmente que atenda as necessidades dos usuários. Neste contexto os serviços deverão adaptar-se às necessidades dos usuários, dando a eles a possibilidade de configurar e personalizar serviços de acordo com a mobilidade pessoal, de serviço e de terminal.

Assim a indústria de telecomunicações busca soluções técnicas para os seus sistemas, tanto no desenvolvimento quanto na disponibilidade dos serviços sob essas novas condições de mercado, pois muitos desses novos serviços terão de interagir com sistemas já existentes para a completa realização de suas funcionalidades, uma vez que o alto investimento já contemplado

na indústria de telecomunicações, muito provavelmente, não será negligenciado.

A crescente popularização de dispositivos computacionais portáteis, a convergência das tecnologias de rádio, dos microprocessadores e dos dispositivos eletrônicos digitais vem tornando possível a comunicação de dados nos locais mais diversos e propiciando ao usuário acesso imediato e universal a novos serviços, de forma transparente, levando ao conceito de ubiquidade onde dispositivos inteligentes, móveis e estacionários se interagem.

Trabalhos mais recentes tratam da computação ubíqua como um novo paradigma, o qual provoca tendência de mudanças no paradigma tradicional de computação, estático e baseado em *desktops*, para uma paradigma altamente dinâmico, com constante mudanças de ambientes causadas pela mobilidade do usuário. Assim sugere o desenvolvimento de aplicações que explore a mudança de contexto dentro de um domínio dinâmico, levando em consideração não só entradas de dados explícitas, como também referentes ao contexto físico e computacional.

O uso da tecnologia de agentes para o desenvolvimento de aplicações móveis permite subdividir a aplicação em tarefas básicas, facilitando sua distribuição com base nas propriedades de autonomia, mobilidade e proatividade. Por isso alguns trabalhos propõem aplicações sensíveis ao contexto integrando dispositivos móveis e utilizando plataformas multi-agentes [3].

Uma outra abordagem, utiliza-se de informações de contexto para a implementação de espaços inteligentes, ambientes em que aplicações buscam reagir à presença de usuários, dentre eles estão *Context-aware Broker Architecture* (CoBrA) o qual é uma infra-estrutura que visa permitir a implementação de agentes, serviços e dispositivos que exploram informações de contexto em espaços ativos [4] [5] e o *Agent-based Context-aware Infrastructure* (ACAI) o qual é um projeto que visa fornecer uma infra-estrutura de apoio a execução de aplicações multi-agentes sensíveis a contexto [3]. De modo geral, ACAI procura integrar, de modo transparente, os diferentes espaços físicos utilizados por um mesmo usuário oferecendo alguns serviços básicos.

1.1 Objetivo do Trabalho

A convergência das telecomunicações com as redes de computadores tem permitido a disponibilização de serviços cada vez mais sofisticados e por serviços que sejam entregues com rapidez, agilidade, segurança, confiabilidade e principalmente que atendam as necessidades dos usuários. Nesse contexto os serviços serão adaptáveis e sensíveis ao contexto, ou seja, serão dependentes da mobilidade pessoal, de serviço e de terminal.

Este trabalho parte de uma arquitetura que estabelece um Ponto de Referência entre Pro-

vedores, proposta em “Uma Arquitetura para Suporte à Ubiquidade dos Serviços de Telecomunicações Baseada em TINA e em Agentes Móveis” [2], baseado em agentes para suporte à mobilidade pessoal e de serviços sobre redes heterogêneas fixas e móveis onde assinantes de serviços possam mudar para provedores nos quais eles não tenham serviços contratados e, ainda assim, utilizarem os seus serviços com suporte do provedor a partir do qual estabeleceram acesso. E assim o objetivo deste trabalho é:

- propor o uso de Redes de Petri para especificação formal das interações entre os componentes do Ponto de Referência entre Provedores;
- conceber/utilizar um ambiente de suporte a agentes móveis que esteja em conformidade com a Especificação *Mobile Agent Facility* (MAF);
- contemplar o estado do agente quando da migração do mesmo para outra agência;
- desenvolver e avaliar a inteligência do agente para que o mesmo possa avaliar a adaptação de serviços quando ocorrer mobilidade pessoal e de serviço;

1.2 Estrutura da Dissertação

Além deste capítulo, este trabalho está estruturado em cinco outros capítulos:

- **Capítulo 2:** são introduzidos os conceitos de mobilidade pessoal, de terminal e de serviço e ainda o conceito de ubiquidade de serviço via mobilidade.
- **Capítulo 3:** descreve um breve histórico sobre inteligência artificial, conceitos de aquisição e representação do conhecimento, formas de representação de raciocínio, descrição e características de agentes e de agentes móveis inteligentes.
- **Capítulo 4:** descreve o modelo de agentes para suporte a mobilidade de serviços.
- **Capítulo 5:** apresenta os aspectos de implementação, descrevendo a arquitetura proposta e aspectos de implementação da inteligência e mobilidade do agente.
- **Capítulo 6:** finaliza o trabalho descrevendo as conclusões tendo como base os resultados obtidos no protótipo desenvolvido e apresentado no Capítulo 5.

Capítulo 2

Ubiquidade de Serviços

Com a convergência dos mundos de Telecom (Fixa e Móvel) e Internet nos deparamos com alguns problemas, dentre eles o suporte à mobilidade pessoal e de serviço sobre a rede Internet, os quais estão à espera de definições e soluções. Deste modo tanto as operadoras de redes (fixas e móveis), quanto provedores de serviço (rede de dados) devem oferecer um leque de serviços integrados que poderão tirar proveito das tecnologias envolvidas.

Assim os serviços estão exigindo a evolução nas arquiteturas dos mesmos, em particular nas arquiteturas abertas de serviço. Um aspecto notório nesta evolução é o uso de esquemas flexíveis de comunicação, assim como o suporte para o desenvolvimento e distribuição dos serviços com base em uma separação mais nítida entre redes e serviços.

Introduzimos neste capítulo o papel da mobilidade e os conceitos de suporte à mobilidade de terminal, pessoal e de serviço. E em seqüência, abordamos a ubiquidade de serviços via à mobilidade de terminal, pessoal e de serviço.

2.1 Papel da Mobilidade

A economia digital depende dos serviços e infra-estrutura oferecidos pelos provedores na entrega de conteúdo para os usuários. Sendo assim, constitui um grande desafio para a indústria de serviços, o oferecimento de serviços que façam parte do dia a dia das pessoas. Só assim, as transações na rede serão impulsionadas, permitindo o reinvestimento pelos provedores de serviços. Nesta relação bilateral a evolução tecnológica e econômica acaba por criar novos requisitos de comunicação representados, principalmente, pela demanda de novos serviços.

Como serviço de informação é a combinação de processos relacionados ao ciclo informacional

dos setores científicos e produtivos, dependendo de pessoas e de uma plataforma tecnológica, organizados para alcançar objetivos voltados à inovação ao desenvolvimento, a produtividade e a competitividade. Esses serviços devem ser desenvolvidos de maneira a prover ao usuário meios para personalizar o uso de seus serviços.

Neste mundo onde o usuário pode personalizar seus serviços, o conhecimento é mais importante do que a informação, uma vez que a necessidade dos usuários está na informação que resolve seus problemas ou abra novas oportunidades. Assim cada vez mais os serviços estão centrados na informação e não na comunicação. O acesso aos serviços devem ser suportados por um conjunto variado de dispositivos. Isto nos leva a uma abordagem onde a informação agregada e gerada por um serviço deverá ser desacoplada de interações com o terminal. E ainda se considerarmos que a inteligência introduzida em terminais, apresentará grandes avanços, novas infra-estruturas mais consistentes serão necessárias para mapear os serviços para terminais.

Os provedores de serviços são responsáveis pelo oferecimento de serviços, conteúdo e valores aos usuários mediante negociação de contrato. A provisão de serviços envolve uma grande variedade de serviços adaptáveis às necessidades de diferentes usuários. A Fig. 2.1 ilustra três classes de entidades móveis: usuários, *views* e plataformas.

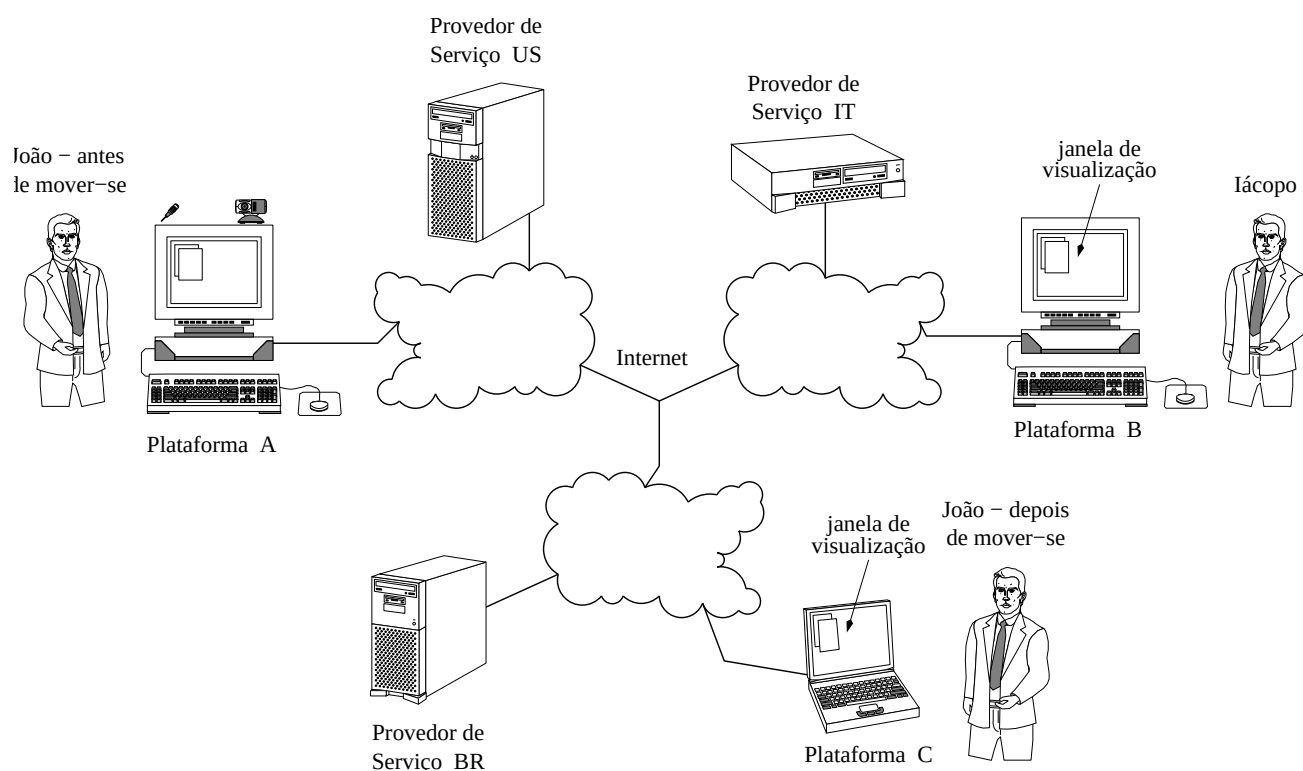


Fig. 2.1: Entidades móveis: Usuários, *Views*, Plataformas e Provedores.

Usuários podem se mover de casa para o trabalho, de cidade para cidade. Uma *view* é o que o usuário vê quando está em frente a um *display*. Usuários podem possuir várias *views*, mas usam uma por vez. Uma *view* inclui o ambiente do usuário e interfaces para aplicações, que podem ser executadas localmente ou em algum lugar na rede.

Uma plataforma é definida como uma coleção de *hardware* e *software*. O *software* da plataforma consiste em: processos ativos que implementam a *view*; o código sendo executado pelos processos; o código que implementa o *software* do ambiente e os dados representando as entidades visíveis na *view*. O *hardware* da plataforma consiste em um ambiente computacional composto de CPU, memória, dispositivos de entrada/saída como *display*, teclado e *mouse*. Em aplicações com recursos multimídia, outros acessórios precisam ser adicionados.

Se uma sessão de serviço é interrompida e retomada a partir de um outro dispositivo, precisamos do suporte à mobilidade do serviço ou da sessão responsável pelo controle e gerenciamento do serviço. A complexidade desse suporte é dependente tanto das redes como dos dispositivos envolvidos, além disso, a transformação nos dados fazer-se-á necessária para que os mesmos assumam o formato usado na rede em questão.

Os serviços de telecomunicações consistiam no transporte de *bits* de informação entre terminais, e devido a desregulamentação e a globalização as operadoras são forçadas a agregar novos serviços que possam ser adaptados às necessidades de diferentes usuários (Fig. 2.2).

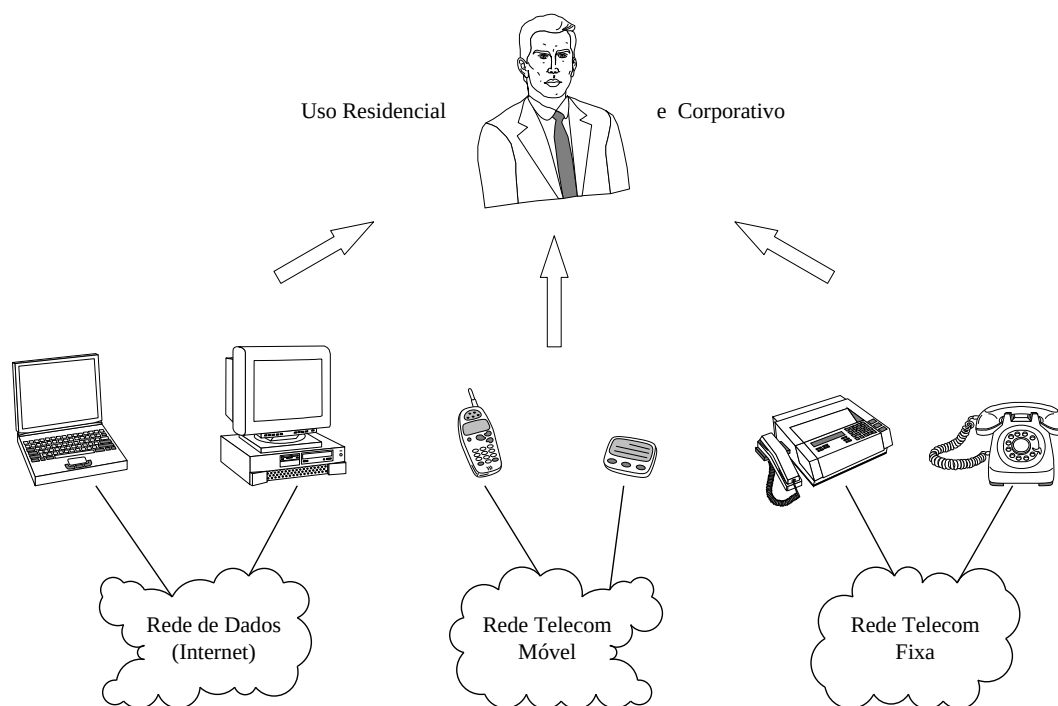


Fig. 2.2: Serviços Centrados no Usuário.

Estes serviços estão se tornando mais e mais complexos, e utilizando mais e mais a inteligência dos componentes terminais da rede. Nessa provisão de serviços, adicionalmente, serão exigidos a integração de diferentes arquiteturas de redes e flexibilidade desses serviços às necessidades de seus usuários.

Hoje em dia, a comunicação sem fio é o segmento com mais rápido crescimento na indústria das telecomunicações. A demanda de serviços básicos de voz e dados estendeu os limites da tecnologia *wireless* para o mercado cada vez maior de serviços ubíquos. Atualmente, o mercado estende ainda mais esta tecnologia, incorporando vídeo-conferência, imagens, transferência de dados e outros serviços avançados anteriormente disponíveis somente em redes fixas(*wired*) de alta velocidade. Satisfazer esta demanda requer o desenvolvimento de redes *wireless* que não somente suportam serviços multimídia integrados, mas também os entregam de maneira ininterrupta, direta e com qualidade de serviço aos terminais móveis.

A próxima geração de sistemas *wireless*, combinando as funcionalidades das redes *wireless* e fixas, irá empregar uma rede de provedores de serviço e equipamentos interoperáveis para garantir serviços multimídia em qualquer parte do globo. Um importante desafio frente a esta nova geração de sistemas será o gerenciamento da mobilidade.

Um sistema *wireless* eficiente deverá ser capaz de localizar terminais móveis a qualquer tempo para entregar seus serviços, assim como manter as conexões à medida que os terminais se movam de uma área para outra. Deste modo, para prover uma comunicação *wireless* global, será necessário o desenvolvimento de uma arquitetura que garanta a mobilidade de terminal, mobilidade pessoal e portabilidade global dos serviços.

2.1.1 Características da Mobilidade

Quando usuários movem de um provedor para o outro, suas respectivas *views* também devem se mover. Se o processamento de dados for necessário na plataforma na qual o usuário se encontra, não só as *views*, mas também os serviços deverão ter suporte à mobilidade.

Uma plataforma pode migrar com o usuário, por exemplo, quando o usuário carrega um *laptop* para um outro domínio. Quando isto ocorre, as aplicações rodando na plataforma e a *view* (ou *views*) associada(s) às aplicações também deve(m) migrar. Entretanto, a conexão de rede da plataforma precisa ser restabelecida no novo domínio, uma situação análoga ocorre na migração de um processo para outra plataforma. Quando o serviço migra, poderá usar os serviços de um provedor de serviços local do novo domínio. Assim, dois tipos de mobilidade devem ser

considerados: a mobilidade de janela sobre plataformas e a mobilidade de plataformas.

Até agora, considerou-se apenas o mundo da Internet, onde os dispositivos são dotados de razoável poder computacional, o que permite o suporte à mobilidade com diferentes soluções. Como estamos interessados em um ambiente de comunicação integrado, que permite o acesso a um dado conjunto de serviços a partir de diferentes localizações e dispositivos, o suporte à mobilidade passa a ser dependente tanto da variedade de dispositivos quanto das redes envolvidas como mostra a Fig. 2.3.

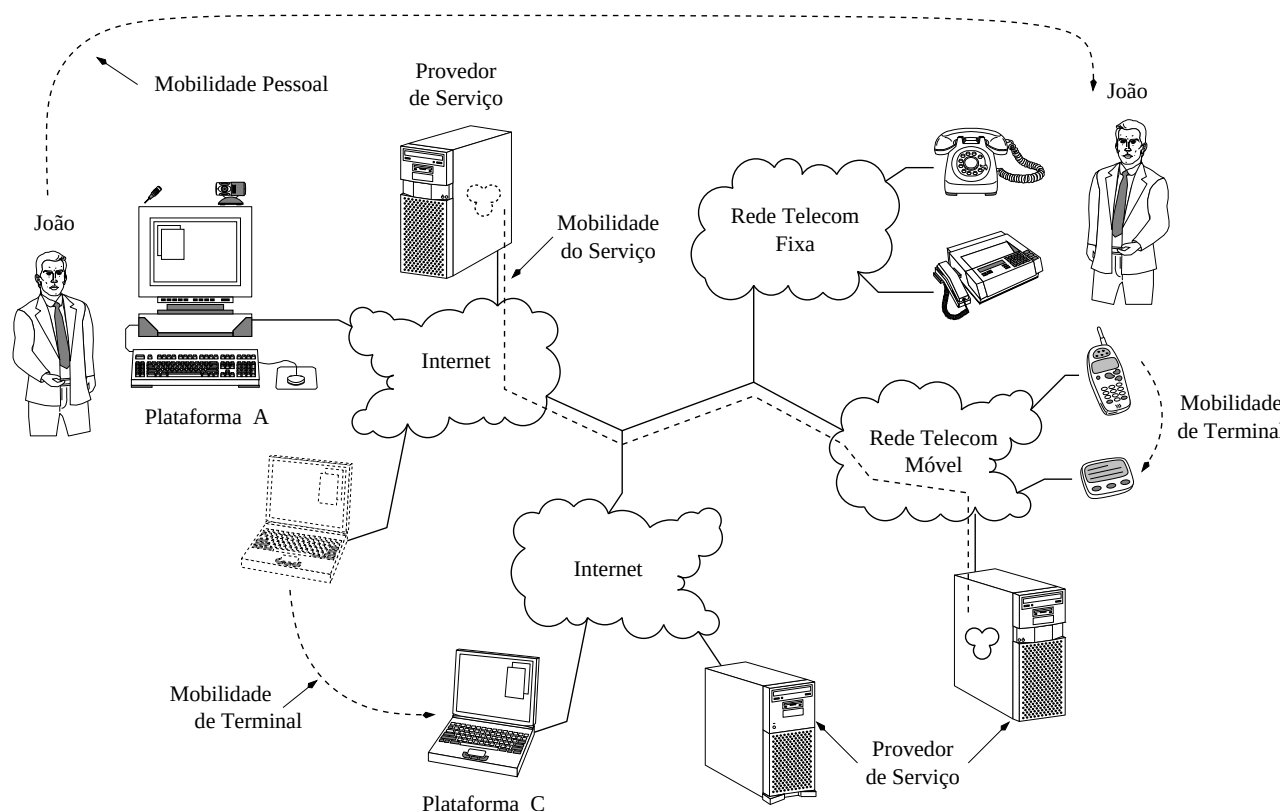


Fig. 2.3: Ambiente de Comunicação.

Neste contexto usuários poderão receber cabeçalhos de *e-mails* como mensagens em *paggers*, ler seus *e-mails* a partir do telefone celular com transformação da mensagem em áudio e serem convidados a participar de uma vídeo-conferência a partir de um celular, neste caso, com apenas informações de áudio. Em cada serviço, estão presentes mais de uma rede, sinalização, formato de dados e dispositivos, caracterizando desta maneira a necessidade da integração de serviços de diferentes redes para propiciarmos um ambiente de comunicação integrado.

2.1.2 Suporte à Mobilidade de Terminal

A mobilidade de terminal permite a mudança de localização do terminal para pontos onde a conexão possa ser mantida ou restabelecida. Quando da mudança de localização, o terminal pode ter o serviço em curso interrompido ou não, dependendo da manutenção ou não da conexão. A identificação do terminal é feita através da atribuição de um número ao mesmo. Para possibilitar a localização do terminal, uma infra-estrutura é oferecida pela rede na qual foi associado. Diferentes redes provêm diferentes mecanismos para tratar esta mobilidade.

Na Rede Telecom Móvel a mobilidade de terminal pode ser conseguida através de uma expansão geográfica da estrutura hierárquica de células. A célula corresponde à área geográfica coberta por sinais enviados e recebidos pelo equipamento transmissor e receptor presente em cada uma delas. Assim, um usuário poderia mover de uma *picocell* dentro de sua residência ou corporação, para uma célula vizinha, quando fora de sua residência ou corporação, como a Fig. 2.4, que mostra a integração *Universal Mobile Telecommunications System* (UMTS) com Redes Fixas. Em áreas de esparsa população ou onde usuários estejam se movendo em alta velocidade, eles poderiam usar serviços *wireless* oferecidos por satélites. A mudança entre esses diferentes serviços deve ser o mais direto possível através de um ambiente consistente para oferecimento e continuidade do serviço.

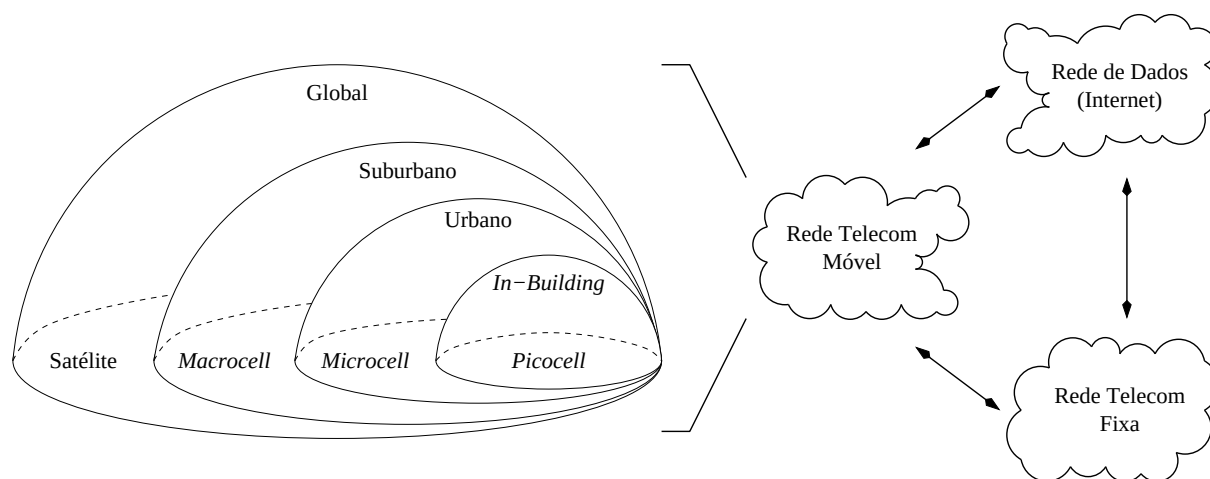


Fig. 2.4: Integração UMTS com Redes Fixas.

Nas áreas Telecom Fixa e Internet, o identificador do terminal define sua localização. Para tratar mobilidade de terminal na Internet, o IP Móvel propõe a gerenciar a entrega de pacotes ao nó móvel sem interrupção do processo de entrega dos mesmos.

2.1.3 Suporte à Mobilidade Pessoal

A mobilidade pessoal permite aos usuários o acesso a seus serviços independente do seu terminal ou do ponto de acesso, um conceito conhecido como *global roaming*. Este conceito estende-se para outras redes tais como Internet e Telecom Fixa. Para que esta integração possa acontecer, uma visão mais realista da evolução de cada uma das redes deve ser considerada pelas demais. Hoje em dia os usuários são identificados por números atribuídos aos seus terminais. Para a próxima geração será atribuído um número pessoal universal que distinguirá independente do terminal de acesso [2].

No suporte à mobilidade de terminal, as técnicas utilizadas visam a identificação do terminal e determinação da sua localização física. No entanto o usuário nem sempre fará uso de um único terminal para acesso e uso de um conjunto de serviços previamente contratado de um provedor, ainda mais, se considerarmos a integração dessas redes no oferecimento de serviços integrados e adaptáveis às necessidades dos usuários.

Um usuário pode ter diferentes identificadores para diferentes serviços na mesma rede, por exemplo, um número de telefone na Rede Telecom Fixa/Móvel; um identificador de *e-mail*, um identificador de ICQ e um endereço IP para voz sobre IP na Rede Internet. Estes identificadores são específicos dos serviços que, por sua vez, estão amarrados ao terminal e, assim, não dispomos a priori da possibilidade de oferecer serviços que não sejam intrínsecos a um dado terminal. Porquanto, a questão da disponibilização do serviço para o usuário não será direta como a que presenciamos, por exemplo nas redes Telecom Fixa e Móvel; adaptações do serviço se farão necessárias, se desejarmos mobilidade do usuário.

Com relação ao serviço, para que este seja estabelecido e disponibilizado para o usuário, informações contidas em seu perfil (p. ex., dispositivo preferencial para receber informações) devem ser consideradas. Há também a necessidade de um serviço de localização do registro de preferência do (perfil do usuário). O acesso as informações do perfil do usuário se dará mediante a autenticação de sua identificação única. É também a partir de seu perfil que serviços podem ser configurados de modo a atender requisitos específicos do usuário, sejam eles usuários residenciais ou corporativos. A Figura 2.5 apresenta o tipo de informação que pode estar contida no perfil do usuário.

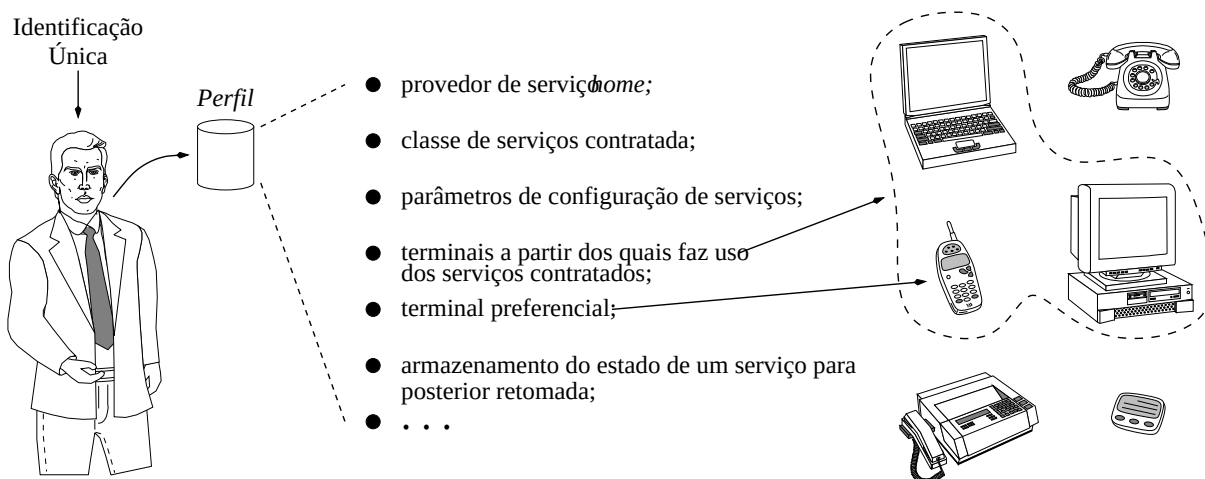


Fig. 2.5: Perfil do Usuário.

2.1.4 Suporte à Mobilidade de Serviço

A portabilidade de serviços proporciona aos usuários, que estão fora de alcance da rede contratada, o acesso e recebimento de serviços personalizados dentro dos limites estabelecidos pelas funcionalidades da rede visitada. Para tanto, a execução de *software* responsável pelas funcionalidades de um dado serviço deve ser independente do *hardware* ou da infra-estrutura mínima de suporte das aplicações de um terminal ou dispositivo. Em resumo, este termo sugere a mobilidade das aplicações e/ou dados entre diferentes localizações, independente da plataforma sobre a qual serão executados.

Nosso objetivo é propor uma arquitetura de serviços para suporte à mobilidade de serviços de informação e de telecomunicações, com a Rede Internet assumindo o papel de rede núcleo.

Assumi-se também que os *browsers*, comuns nos dispositivos da Rede Internet, já se tornaram a interface escolhida para aplicações e assim usuários farão uso dessa funcionalidade também em dispositivos portáteis de outras redes. Além disso, a ubiquidade no oferecimento de serviços é de interesse tanto das operadoras de rede quanto dos provedores de serviço. Para isso, serviços devem ser interoperáveis entre redes heterogêneas. Aspectos de interoperabilidade devem ser tratados tanto entre redes e clientes (interface de cliente), quanto entre operadoras de redes. A Fig. 2.6 mostra onde se posicionam estas interfaces para o exemplo anteriormente citado.

A liberdade da mobilidade global nesta escala irá requerer a coordenação de uma grande variedade de provedores de serviço, garantia de compatibilidade entre as redes *backbone* (Telecom Fixa, Telecom Móvel e Internet), e obtenção de acordos entre operadoras de redes.

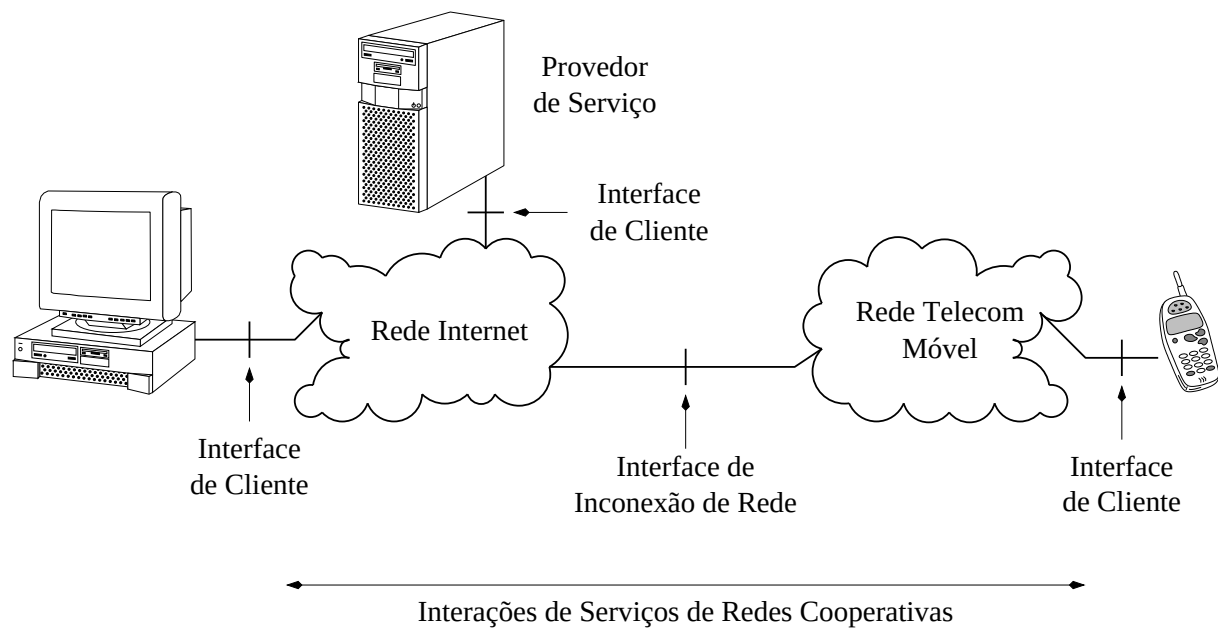


Fig. 2.6: Aspectos de Interoperabilidade entre Redes.

Enquanto acordos entre operadoras de redes são governados por contratos comerciais, a mobilidade global necessitará de acordos envolvendo países, regiões, provedores de serviço e aumento do espectro da frequência disponível.

2.2 Ubiquidade de Serviço

A ubiquidade de serviço proporciona a mobilidade pessoal, e deste modo, um ambiente computacional estará disponibilizado para que as pessoas trabalhem em qualquer lugar, a qualquer hora. Os desafios no suporte à mobilidade serviço, e por conseguinte na ubiquidade de serviço se situam em três grandes áreas:

1. provisão de serviços de comunicação *wireless* via mobilidade de terminal;
2. aplicações que lidam com a natureza arbitrária de desconexão;
3. aplicações independentes de localizações físicas e passíveis de configuração.

2.2.1 Ubiquidade via Mobilidade de Terminal

A ubiquidade do serviço através da mobilidade de terminal pressupõe que o serviço está fixo no provedor ou no terminal. Com base nos aspectos para suporte à mobilidade de terminal, a portabilidade dos serviços pode se dar via mobilidade de terminal basicamente em três casos:

1. mobilidade suportada pela camada física e enlace: o serviço deve estar disponível em todas as redes e interagir com o terminal de forma homogênea. Assim sendo, o processo de *roaming* entre redes também deverá ser suportado pelas mesmas camadas.
2. mobilidade com suporte na camada de rede: a migração do terminal é transparente, isto porque o roteador da rede contratada, agente de usuário, torna atingível o nó móvel (terminal), a partir de seu endereço contratado, até mesmo quando este nó não está conectado fisicamente à sua rede. O IP Móvel apresenta algumas desvantagens: pouca difusão do protocolo, ineficiência para um grande número de nós, e o redirecionamento pode se tornar um fator crítico em algumas aplicações que façam uso de recursos multimídia, uma vez que as informações terão que necessariamente, passar pelo roteador contratado.
3. mobilidade suportada pelas camadas superiores: a mobilidade de terminal se dá com o restabelecimento da conexão de transporte ou redirecionamento de entrada/saída. Todas as sessões de comunicação devem ser terminadas antes da migração e restabelecidas depois, o que acarreta a interrupção do serviço para uma posterior retomada. O gerenciamento de replicação de componentes não se dá, no entanto, apenas no nível de conexões, mas também nos níveis de acesso e estabelecimento do serviço. Considerando o cenário em que o usuário interrompe um serviço, muda de localização e de terminal, esta solução torna-se interessante para serviços que podem ser oferecidos de forma não contínua.

2.2.2 Ubiquidade via Mobilidade Pessoal

A ubiquidade do serviço via mobilidade pessoal também pressupõe que o serviço está no provedor ou no terminal. A portabilidade dos serviços via mobilidade pessoal se faz presente quando a disponibilização do serviço é direta, como por exemplo, nas redes Telecom Fixa e Móvel. Nestes casos, mesmo ocorrendo mudança de terminal, o serviço oferecido já está disponível. Além disso, redes Telecom Móvel estão integradas às redes Telecom Fixa.

Em se tratando de serviços na Rede Internet, quando um usuário se move de um terminal para outro, não é razoável assumir que o mesmo serviço será disponibilizado prontamente no novo terminal. Recursos de *hardware* e *software* poderão se fazer necessários para que o serviço possa ser disponibilizado. O oferecimento do serviço fica condicionado ao envio do serviço para o terminal, de tal forma a compatibilizar o terminal, a rede visitada e o próprio usuário. Neste caso, a ubiquidade via mobilidade pessoal necessita do suporte da mobilidade de serviço.

Uma outra vantagem é que o terminal pode mudar durante a migração, e neste caso, o serviço na rede visitada é enviado pelo provedor contratado de tal forma a compatibilizar o terminal, a rede visitada e o próprio usuário (Fig. 2.7). Faz-se necessário o restabelecimento das sessões de comunicação, que pode se dar de modo automático. Por outro lado, o *overhead* do serviço e a necessidade de um suporte computacional bastante forte pelo outro provedor, em decorrência da compatibilização do ambiente de suporte no oferecimento de serviços, constituem pontos desfavoráveis nesta abordagem.

Capítulo 3

Sistemas Baseados em Agentes

Segundo Nwana [6], a origem do conceito de agente vem do modelo de atores proposto por Hewitt [7] em 1977, onde atores são definidos como objetos autocontidos, interativos e de execução concorrente. Este modelo fundamentou os Sistemas MultiAgentes (SMA), uma das grandes áreas no campo da inteligência artificial que trata da solução distribuída de problemas. Já os Sistemas Baseados em Agentes emergiram dos sistemas SMAs e podem ser vistos como sistemas onde as entidades básicas de computação são os agentes, os quais trabalham cooperativamente para solução de problemas.

Neste capítulo apresentaremos conceitos de inteligência artificial, definições e características de agentes, os princípios de agentes inteligentes e princípios de agentes móveis.

3.1 Inteligência Artificial

Em 1950, Alan Turing, elaborou um teste simples para saber se uma máquina seria capaz de pensar. O teste consistia da formulação de perguntas que seriam respondidas por um computador e por uma pessoa. Se fosse confundida com o ser humano, a máquina seria considerada inteligente. Ainda hoje esse modelo é utilizado na avaliação de alguns sistemas inteligentes [8].

Em 1960, Newell e Simon propuseram a resolução de problemas gerais *general problem solving* (GPS). Contudo, é muito difícil atingir o propósito do GPS: resolver qualquer problema em qualquer domínio. O processamento do conhecimento para solucionar todo e qualquer tipo de problema é uma tarefa impossível. Em 1965, Zadeh propõe a base da lógica difusa, que lida com avaliação de expressões lógicas, contendo valores incertos. Este modelo auxilia na resolução dos problemas que apresentam imprecisões, incertezas e ambigüidades.

Por volta de 1970, a Inteligência Artificial tinha alcançado somente sucessos limitados [9]. Os sistemas eram capazes de encontrar soluções para problemas imaginários ou problemas bem construídos, como jogos.

A Inteligência Artificial (IA) é um ramo da Ciência da Computação dedicado ao estudo das técnicas que possibilitam a representação em máquinas de algum aspecto da cognição humana [10]. Seguindo essa linha Rich [11] propõe que "IA é o estudo de como fazer com que a máquina possa realizar tarefas que hoje são melhor realizadas pelo homem". Os pesquisadores em IA concordam que duas questões distintas devem ser consideradas: a questão científica, que lida com aspectos teóricos da cognição, ou seja, estuda os processos do raciocínio humano para compreender o mecanismo da inteligência, e a questão tecnológica, que se preocupa com a representação destes processos através da máquina (computadores, robôs, etc) [12].

3.1.1 Engenharia do Conhecimento

A inteligência Artificial permite construir sistemas para processamento simbólico, o que de muitas formas reproduz a forma como o ser humano resolve problemas. A partir dos anos 60 foram construídos sistemas para resolver problemas complexos. A experiência dessa primeira fase ensinou que a qualidade de solução desses sistemas não era determinada pelos mecanismos de raciocínio neles embutidos, mas sim pelo conhecimento extraído de especialistas humanos e codificado no programa.

A partir desta constatação, parte do esforço na construção de sistemas de inteligência artificial voltou-se, na segunda fase, para técnicas de extração do conhecimento de especialistas e codificação em diversos formalismos de representação. Assim nasceu a Engenharia do Conhecimento e os primeiros sistemas especialistas. Esses sistemas, onde o conhecimento de um indivíduo (o especialista) era representado em uma base de conhecimento para ser utilizado exclusivamente por um mecanismo de inferência, estabeleceram as bases para a engenharia de sistemas especialistas que orientou a área até o final da década de 80.

Uma nova revolução na Engenharia do Conhecimento aconteceu com o surgimento dos modelos administrativos de gestão de conhecimento e das plataformas distribuídas de sistemas.

Um sistema inteligente é desenvolvido através da Engenharia do Conhecimento, com o apoio de especialistas sobre o domínio da aplicação. A engenharia do conhecimento é responsável pela extração e representação do conhecimento através da utilização de linguagens de IA [13].

Os sistemas inteligentes apresentam características que os diferenciam de sistemas con-

vencionais. A principal diferença entre um sistema baseado em conhecimento e um programa convencional reside na estrutura. Num programa convencional, o domínio do conhecimento está intimamente relacionado com o *software* através de procedimentos de controle da aplicação do conhecimento. Num sistema baseado em conhecimento, as duas tarefas estão explicitamente separadas, existem dois módulos: o módulo do conhecimento, chamado base de conhecimento; e o módulo de controle, chamado motor de inferência [14].

Segundo Donald A. Waterman [15], a base de conhecimento consiste em dados e regras (ou outros tipos de representação) que o utilizam como base para o processo decisório. A sua construção é a principal tarefa da Engenharia do Conhecimento. O motor de inferência representa a forma de manipular o conhecimento já representado na base, a fim de resolver o problema. Ele determina a ordem com que serão processadas as informações, manipulando os dados a fim de inferir novos fatos, chegar a conclusões ou recomendar ações. Donald A. Waterman [15] ainda destaca alguns pontos importantes dos sistemas baseados em conhecimento: a representação do conhecimento confere maior flexibilidade e maior consistência ao sistema ao invés da representação de dados; a utilização de heurísticas busca encontrar melhor caminho para a obtenção de respostas ao invés de algoritmos. Quando são usadas heurísticas para guiar a solução de um problema num sistema especialista, isto chama-se busca heurística ou melhor caminho. Entretanto, não se pode garantir que uma solução será encontrada, mas somente que a direção escolhida para solucionar o problema é a melhor. A busca heurística é valiosa nas aplicações que requerem soluções rápidas [16]; o processo de inferência muitas vezes permite que o sistema seja interativo e mais ágil do que os sistemas convencionais que utilizam processos repetitivos, o que torna a manipulação de uma base de conhecimento mais eficaz.

3.1.2 Aquisição do Conhecimento

A aquisição do conhecimento consiste na extração do conhecimento do especialista a ser transposto em um sistema computacional para torná-lo inteligente. As fontes para a aquisição deste conhecimento são provenientes de especialistas ou de bibliografias confiáveis. Segundo Limin Fu [17], esta tarefa executada pelos engenheiros de conhecimento, é considerada o gargalo da IA. Muitos pesquisadores da área têm concentrado esforços no desenvolvimento de técnicas cada vez mais apuradas de aquisição do conhecimento [16].

Olson e Rueter (1987), propõem a classificação dos métodos de aquisição do conhecimento em dois tipos: diretos e indiretos. Os métodos diretos envolvem entrevistas e estudos de casos

realizados com o especialista do domínio em que se quer construir o sistema. Os métodos indiretos determinam o conhecimento dos especialistas através de questionários.

3.1.3 Representação do Conhecimento

O conhecimento é o destaque dos sistemas de IA. A representação do conhecimento é o componente fundamental em sistemas inteligentes. Para representar o desempenho de especialistas humanos, o sistema deve possuir não só um conjunto de informações mas, também, a habilidade de utilizá-las na resolução de problemas de forma criativa, correta e eficaz. Esta habilidade representa uma série de palpites e regras intuitivas que o especialista utiliza para resolver os problemas. Para resolver problemas, em alguns casos, é recomendado conhecer tudo sobre o problema e quais as possíveis soluções que se pretende encontrar, juntamente com algumas estratégias para solucionar cada problema. O estudo da representação do conhecimento consiste nos caminhos que podem ser trilhados para codificá-lo em um programa computacional.

Linguagens Baseadas em Lógica e suas Máquinas de Inferência

Para representar conhecimento de maneira precisa, alguns SBCs utilizam linguagens de representação de conhecimento baseadas em lógica. Diferentes tipos de lógica podem ser utilizadas, dentre eles a lógica proposicional e a lógica de primeira ordem.

Os sistemas baseados em lógica proposicional são compostos por um conjunto de regras de produção do tipo *if-then-else* e um conjunto simples de operadores (*implies*, *equals*, *not*, *and*, *or*). Embora simples de usar, os sistemas baseados em lógica proposicional não possuem muita expressividade, o que os impede de representar fatos complexos. Um exemplo de linguagem proposicional é CLIPS (*C Language Integrated Production System*) [18]. Já existe disponível uma versão de CLIPS em Java, conhecida como JESS (*Java Expert System Shell*) [19].

Os sistemas baseados em lógica de primeira ordem são compostos por fatos, objetos e relações. Normalmente as sentenças desses sistemas são expressas utilizando cláusulas de Horn. A linguagem baseada em lógica de primeira ordem mais utilizada para representar conhecimento é Prolog (Rowe, 1988). A linguagem KIF (*Knowledge Information Format*), que também utiliza lógica de primeira ordem, foi criada para servir como uma "interlíngua" entre diferentes sistemas. Foi uma das primeiras linguagens criadas especificamente para representar informação para ser transportada de um programa a outro [20].

A lógica de primeira ordem possui um grande potencial para representar conhecimento e

tem sido o instrumento preferido para a formalização de conhecimento em um SBC[21].

As linguagens de representação de conhecimento utilizam máquinas de inferência para processar o conhecimento armazenado nas bases de conhecimento. A máquina de inferência é responsável por processar a linguagem de representação usada na base de conhecimento, gerando e percorrendo o espaço de busca sempre que necessário. O tipo de inferência realizado depende do tipo de lógica da linguagem de representação de conhecimento utilizada.

Em linguagens baseadas em lógica proposicional, a inferência costuma ser baseada em um encadeamento das regras, podendo ser dos seguintes tipos [21]:

- Encadeamento regressivo ou *backward chaining*: este processo parte da suposição de que cada provável solução é verdadeira. Feito isso, tenta-se reunir evidências que comprovem ser correta a solução previamente considerada. Esta estratégia corresponde a perguntar: "É possível provar as hipóteses a partir dos dados que temos?".
- Encadeamento progressivo ou *forward chaining*: Este processo consiste em analisar os dados fornecidos pelo usuário, baseando-se no conhecimento, até uma conclusão. Isto corresponde a perguntar: "O que é possível concluir a partir dos dados que temos?".

Já as linguagens baseadas em lógica de primeira ordem costumam utilizar inferência lógica baseada apenas em algoritmos de *backward chaining*. O raciocínio disponível é então baseado em metas, que os SBCs desejam atingir. A máquina de inferência tenta provar essas metas a partir das cláusulas armazenadas na base de conhecimento. Novas sentenças são criadas de acordo com as conseqüências de um conjunto de predicados. O funcionamento de uma máquina de inferência é baseado em um algoritmo de reconhecimento de padrões, capaz de comparar os fatos com padrões, determinando quais regras da base de conhecimento são aplicáveis. Uma vez que uma regra é disparada, as ações especificadas pela regra são executadas, o que pode afetar a lista de regras aplicáveis por meio da adição ou remoção de novos fatos. Novas regras continuam sendo aplicadas até que não exista nenhuma regra que seja aplicável [18].

3.1.4 Formas de Representação do Raciocínio

O raciocínio é o encadeamento aparentemente lógico de juízos ou pensamentos. É algo tão comum e intuitivo que a maioria das pessoas não se preocupa em analisar como tal processo ocorre [22]. Para simplificar a compreensão deste estudo, do ponto de vista computacional, identificou-se alguns tipos de raciocínio em:

- **Monotônico:** o raciocínio sobre um problema, para muitas situações, se processa sobre informações estáticas. Durante o processo de resolução do problema, o estado (Verdadeiro ou Falso) dos fatos permanece constante [16].
- **Não-Monotônico:** os seres humanos tem a capacidade de manter o caminho quando as informações mudam. Se alguma coisa mudar, é possível se ajustar a outros eventos independentes [16].
- **Dedutivo:** utiliza fatos (axiomas) para deduzir novas informações. O processo inicia-se comparando o axioma com uma certa implicação para concluir novos axiomas [16]. A regra de inferência modus ponens é a forma básica de raciocínio dedutivo:

SE A é verdadeiro E A implica em B, ENTÃO B é verdadeiro.

- **Indutivo:** os seres humanos utilizam o raciocínio indutivo para alcançar uma conclusão geral de fatos limitados através de um processo de generalização. Através do raciocínio indutivo, é formada uma generalização que se acredita que possa ser aplicada a todos os casos de um certo tipo, em bases de números limitados de casos [16].
- **Abduutivo:** a dedução é exata no sentido que inferências retiradas de fatos estabelecidos e implicações válidas são logicamente corretas. Abdução é uma forma de dedução que permite inferências plausíveis. Neste caso, "plausível" significa que a conclusão pode surgir de informações disponíveis, apesar da não certeza da veracidade dessa conclusão [16].

**SE B é verdadeiro E SE A implica em B é verdadeiro,
ENTÃO A é verdadeiro?**

- **Analógico:** é o processo de generalização fundado em semelhança de relação apresentada por elementos de totalidades diferentes que consiste em passar de uma ou mais propriedades já observadas em um dos elementos à atribuição das mesmas propriedades a outros elementos de outra totalidade, no qual ainda não tenham sido observadas.
- **Senso Comum:** os seres humanos aprendem a resolver problemas de forma eficiente através da experiência. O senso comum é utilizado para encontrar uma solução rapidamente. Este tipo de raciocínio confia mais num bom julgamento do que na lógica exata.

3.2 Definições e Características de Agentes

A aplicação do paradigma de Sistemas Baseados em Agentes (SBAs) já se difundiu em muitas áreas dentro do escopo da inteligência artificial e sistemas distribuídos, mas nos últimos anos, apresentou maior importância no campo da engenharia de *software*. Dentre estas áreas, estão incluídas: assistência pessoal/interface de usuário, computação móvel, recuperação e filtragem de informação, busca de dados, mercado eletrônico, controle de serviços de telecomunicações, gerenciamento de redes e serviços. Nwana apresenta uma vasta taxonomia de agentes encontrada na literatura. Franklin e Graesser em 1996 [23], examinaram e compararam uma variedade de definições oferecidas por pesquisadores de agentes e concluíram que não havia um consenso geral no que constitui um agente ou no que agentes se diferem de programas, mas propuseram requisitos que constituem a essência para se ter um agente.

Segundo Franklin e Graesser [23], estes requisitos formalizados levam à definição do que é agente autônomo: *é um sistema que é parte de um ambiente e que é capaz de sentir e agir sobre ele, no decorrer do tempo, em busca de seus objetivos e efeitos do que sente no futuro*. Esta definição sucede na distinção entre agentes e programas, mas produz um classe de agentes tão grande que apresenta pouca utilidade.

Com base no grande conjunto de propriedades que um agente pode apresentar e no uso de uma definição menos restritiva para agentes autônomos, Franklin & Graesser [23] apresentam um conjunto de propriedades bem aceito para caracterizá-los. Apresentadas a seguir, as três primeiras são tomadas como fundamentais, enquanto a adição de outras propriedades, potencialmente, produzem classes de agentes úteis, como por exemplo, agentes móveis.

- **autonomia** - um agente pode tomar suas próprias decisões, ou seja, exercer controle sobre suas ações com base nos seus objetivos, preferências, limitações e políticas;
- **sentir e agir** - um agente é sensível às mudanças do ambientes e assim possui a habilidade de reagir em função do que percebe neste ambiente;
- **orientado a objetivo** - um agente não atua somente em reação ao ambiente, mas também em função de suas metas e objetivos estabelecidos;
- **comunicação** - um agente pode se comunicar com outros elementos em seu ambiente;
- **delegação** - usuários e outros programas podem delegar tarefas para um agente, atribuindo a ele autoridade para atuar no cumprimento destas tarefas;

- **cooperação** - agentes podem atuar em cooperação com outros agentes de modo a alcançarem objetivos não conflitantes;
- **adaptação** - também conhecida como **aprendizado**, representa a habilidade de mudar o seu comportamento baseado em experiências passadas;
- **mobilidade** - habilidade de um agente de suspender sua execução em uma máquina e se transportar para uma outra, mantendo sua integridade;
- **flexibilidade** - agentes não assumem papéis fixos, podendo atuar como clientes, servidores, observadores, atuadores, dependendo dos seus objetivos no momento.

Para Oliveira e Cardozo [24], *agentes são considerados programas autônomos que agem em benefício de seus proprietários — usuários ou outros programas e que, de maneira a acompanhar as tarefas designadas por seus proprietários, agentes podem se comunicar com outros agentes, com o seu ambiente e com seus proprietários.*

Ainda segundo Oliveira e Cardozo [24], algumas características como autonomia, comunicação, delegação, flexibilidade e eqüidade são consideradas fundamentais, enquanto cooperação, mobilidade e inteligência, embora fortemente relacionadas a agentes, não são fundamentais. A descrição de cada uma delas é apresentada a seguir:

- **autonomia, comunicação, delegação, flexibilidade** - como descritas anteriormente;
- **eqüidade** - agentes agem como pares, não há relacionamento hierárquico entre eles;
- **cooperação e mobilidade** - como descritas anteriormente;
- **inteligência** - é a habilidade de um agente de reagir e aprender a partir de interações com outros agentes e com o próprio ambiente.

Uma outra abordagem é a apresentada por Kendall: *Sistemas de agentes devem ser fortemente baseadas em padrões de software dominantes. Um padrão provê uma descrição abstrata de um problema de projeto, assim como um arranjo geral de elementos (p. ex., classes e objetos) que possa resolvê-lo. Há também um contexto que deve estar presente para que o padrão seja válido, bem como a declaração de uma solução e alguma aplicação conhecida.* Nessa abordagem, fundamentos de engenharia de *software* são considerados para caracterizar sistemas baseados em agentes. Segundo Kendall, agentes devem ser decompostos em camadas (Fig. 3.1), de tal

forma que: **i.** a camada de mais alto nível ou comportamento mais sofisticado dependa das habilidades da camada inferior; **ii.** cada camada tem relação de dependência somente com as camadas vizinhas e **iii.** há fluxo de informação nos dois sentidos em camadas adjacentes. Este modelo descreve uma arquitetura geral onde agentes se diferenciam pelo número de camadas e nas habilidades presentes em cada uma dessas camadas.

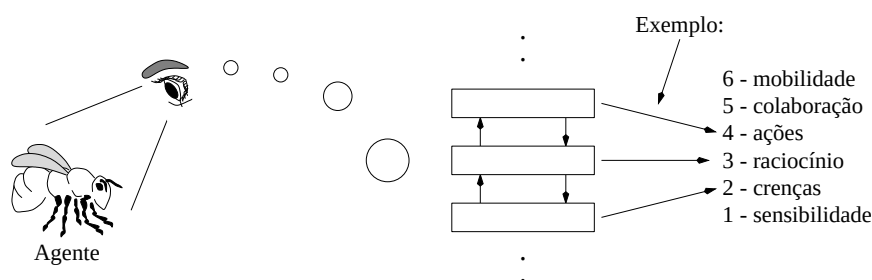


Fig. 3.1: O Padrão Arquitetural de Agentes em Camadas.

Embora, em primeira instância, seja difícil separar as características tidas como fundamentais daquelas que potencialmente os tornam classes de agentes úteis, muito provavelmente dependendo do ambiente sobre o qual atuam e dos objetivos traçados, este leque pode sofrer adições. Por exemplo, um agente representando um usuário e atuando na configuração de seus serviços segundo um perfil preestabelecido ou adquirido, terá que se mover para o local onde os serviços serão disponibilizados ou se comunicar com algum agente local ao oferecimento dos serviços. Assumindo que este agente siga a definição apresentada por Franklin e Graesser, sem uma dessas habilidades (mobilidade e comunicação), o que o agente poderia fazer? Embora seja autônomo, tenha objetivo a cumprir, percebe o ambiente e não dispõe das habilidades para agir! Aqui, parece ser imprescindível as habilidades de se mover e comunicar.

Neste trabalho, agentes são entidades de *software* autônomas, assíncronas, dotadas de habilidades e objetivos próprios e que comunicam entre si e com o ambiente através de sensores e atuadores. Este modelo de agente está ilustrado na Fig. 3.2. Habilidades ou conjunto de capacidades, que o agente pode realizar, são caracterizadas pelo leque de propriedades que a ele é atribuído, enquanto objetivos tratam das atividades a serem realizadas para o cumprimento de uma dada tarefa. Já a comunicação com o ambiente e com outros agentes se dá através dos sensores e atuadores, que são responsáveis pela emissão e recepção de mensagens.

Os vários esquemas de classificação propostos na literatura, têm sido baseados em conjuntos reduzidos de critérios. Esses critérios podem, por exemplo, refletir as características associadas aos agentes ou os papéis que eles desempenham. Algumas dessas classificações podem ser

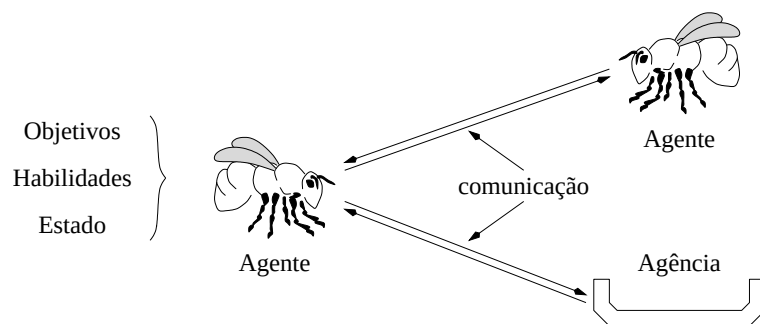


Fig. 3.2: Modelo de Agente.

encontradas em Nwana, Franklin e Graesser, Magedanz e FIPA. A maioria desses trabalhos não se preocupa em gerar uma lista extensa de tipos de agentes, mas sim em apresentar alguns tipos vistos como mais importantes sob a ótica de seus autores. Neste trabalho, nos concentramos no uso de agentes móveis inteligentes representando o usuário e atuando na configuração de seus serviços, parte do objetivo a que este trabalho se propõe.

3.2.1 Princípios de Agentes Inteligentes

Basicamente, a idéia e tecnologia de agentes inteligentes têm sido influenciadas por uma grande variedade de disciplinas e práticas. Entretanto, sua origem fundamentou-se no campo da Inteligência Artificial, e em particular da área de Inteligência Artificial Distribuída. Esta tecnologia tem sido usada desde a década de 80 para refletir a idéia de objetos inteligentes, ou seja, entidades de *software* autônomas que se comportam de acordo com uma inteligência autocontida (conhecimento e inferência).

Agentes Inteligentes são elementos de *software* autônomos, assíncronos (frequentemente disparados por eventos ou tempo) e responsáveis pelo cumprimento de parte de um processo pragmático, ou seja, suscetível de aplicação prática. Assim, o agente contém algum nível de inteligência (p. ex., objetivos, raciocínio, planejamento, aprendizado), podendo este variar de regras pré-definidas até máquinas de inferência *self-learning* baseadas em inteligência artificial. Ele age, tipicamente, em favor de um usuário ou um processo proporcionando automação de tarefas. Além disso, agentes mais avançados podem cooperar com outros agentes para suportar tarefas além de suas capacidades. Finalmente, como objetos transportáveis ou até mesmo como objetos ativos, agentes podem mover de um sistema para outro a fim de acessar recursos remotos ou encontrar e cooperar com outros agentes.

Em sistemas de um único agente (*single-agent systems*), o agente atua ou age em favor de

um usuário ou algum processo. Enquanto realiza sua tarefa, o agente pode se comunicar com o usuário assim como com recursos remotos e locais, mas nunca se comunica com outros agentes. Já em sistemas com múltiplos agentes (*multiple-agent systems*), agentes podem cooperar entre si a fim de atingir seus respectivos objetivos. Naturalmente, nestes sistemas os agentes também interagem com o usuário e com recursos do sistema.

Segundo Magedanz, em sistemas *single-agent* podemos identificar dois tipos de agentes:

- **agentes locais** - membros desta classe acessam apenas recursos locais; têm como principal objetivo a colaboração com o usuário e deste modo sua principal ênfase reside no campo da interação agente/usuário. Usualmente agentes locais agem como agentes consultores ou assistentes pessoais e por isso têm também sido chamados de *interfaces inteligentes* ou *agentes de interface*; e
- **agentes de rede** - acessam recursos locais e remotos e assim possuem um conhecimento mais ou menos detalhado da infra-estrutura e dos serviços disponíveis na rede. Muitos dos agentes de rede agem como *assistentes pessoais* e por isso não só provêem uma interface inteligente para o usuário, mas também fazem uso extensivo dos vários serviços disponíveis na rede. Em agindo assim, o agente não somente esconde a complexidade da infra-estrutura de rede, mas também torna invisível para o usuário a heterogeneidade de fontes de informação e protocolo de acesso de uma grande variedade de redes. Este tipo de agente é especialmente relevante em redes como a Internet.

Ainda segundo Magedanz, em sistemas *multiple-agent* dois tipos são identificados:

- **agentes baseados em inteligência artificial distribuída** - são desenvolvidos com técnicas de inteligência artificial onde a representação do conhecimento e mecanismos de inferência são os mesmos daqueles de sistemas não distribuídos baseados em conhecimento. Membros dessa classe são usualmente estáticos e apresentam a cooperação e negociação como ênfase. Nesse sentido, os protocolos de cooperação e negociação entre agentes são baseados na teoria que define a semântica das mensagens e em redes de contratos; e
- **agentes móveis** - podem ser transportados através da rede para locais remotos, usualmente para um ambiente de execução específico de maneira a cumprir suas tarefas.

Enfatizamos que esta taxonomia de agentes inteligentes não objetiva ser completa e deve somente ser considerada para separar em categorias os conceitos existentes para agentes. Outros

autores podem adotar uma taxonomia completamente diferente. Nossa motivação em agentes móveis inteligentes, os quais podem ser considerados como uma classe específica de agentes inteligentes, decorre do potencial impacto do uso dessa tecnologia no contexto das telecomunicações, mais especificamente, nos aspectos de provisão e personalização de serviços.

3.2.2 Princípios de Agentes Móveis

A mobilidade do agente é um atributo que possibilita aos agentes inteligentes influenciarem o modo tradicional de comunicação e realização de serviços. Por outro lado, o atributo de mobilidade requer que agentes sejam executados em ambientes heterogêneos, uma vez que terão seus códigos transportados para várias máquinas. Assim, segurança representa um aspecto chave para mobilidade de um agente. Conseqüentemente, os agentes devem ser avaliados em ambientes abertos (p. ex., na Internet) e de execução segura. Nesse sentido, *scripting languages* têm sido foco de atenção por serem altamente portáteis e, inerentemente, proverem alto grau de segurança. Além disso, o processo de migração deve ser eficiente e seguro.

Todos os aspectos do comportamento do agente incluindo execução, comunicação e mobilidade são suportados pelos serviços de suporte da agência, ambiente de execução do agente. Assim, agentes móveis podem cumprir suas tarefas localmente, ou seja, no local onde as entidades ou recursos envolvidos estão disponíveis. Frequentemente, este conceito é referenciado como programação remota, uma alternativa para o paradigma de interação cliente-servidor. No modelo cliente-servidor, componentes estáticos localizados em diferentes computadores comunicam-se remotamente via mecanismo *Remote Procedure Call* (RPC). A idéia da programação remota é antiga e, na década de 70, foi denominada *batch job processing*. No passado, as principais motivações no seu uso eram a falta de carga computacional para execução local dos programas e o desejo de compartilhar recursos e assim melhorar o balanço de carga em sistemas distribuídos. Em contraste a esses conceitos concebidos para ambientes fechados e específicos, agentes estão direcionados para ambientes abertos (p. ex., Internet).

Geralmente, dois níveis de mobilidade de agente podem ser identificados: execução remota e migração do agente, como apresentado na Fig. 3.3.

Na execução remota, um agente (código e dados) é transferido para um sistema remoto onde é ativado e executado integralmente e para isto irá interagir com recursos (p. ex., objetos, agentes) localizados no destino. Após cumprir suas tarefas, o agente poderá ser destruído ou permanecer no sistema remoto em estado dormente e ser reativado mais tarde. Já na

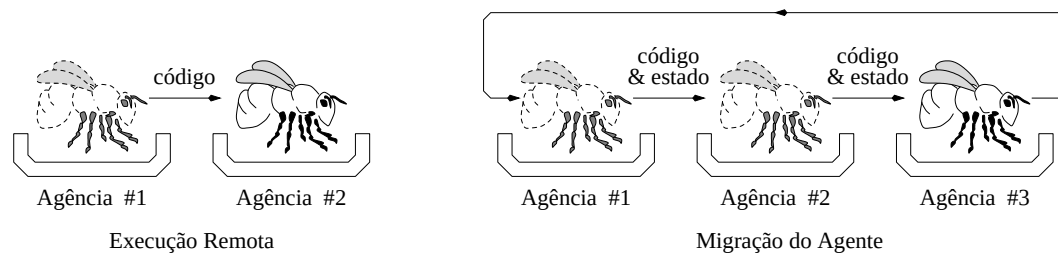


Fig. 3.3: Níveis de Mobilidade de Agentes.

migração do agente, o agente pode se mover de um nó para outro durante sua execução a fim de acompanhar progressivamente o cumprimento de sua tarefa. Uma agência que suporta a migração do agente permite ao agente ativo suspender sua execução, mover para um outro nó na rede, e retomar a execução a partir do ponto onde foi suspenso. Isto no entanto, requer a preservação do estado de execução do agente e deste modo passamos a ter migração de código, estado e dados, diferentemente do caso anterior.

A princípio, as seguintes oportunidades para o uso da tecnologia emergente de agentes podem ser identificadas no contexto do mercado de telecomunicações:

- processamento cooperativo e assíncrono de tarefas - a delegação de tarefas específicas através de agentes móveis para nós específicos ou múltiplos, possibilita alta computação paralela e dinâmica e, em particular, suporta operação desconectada de tarefas¹;
- personalização e configuração de serviços - à luz do comércio eletrônico, a tecnologia de agentes possibilita o oferecimento de novos serviços por meio de sua personalização ou (re)configuração de serviços já existentes, o que pode reduzir o número de tarefas desempenhadas pelo usuário durante a instalação, operação e manutenção de recursos através da autonomia, mobilidade e inteligência dos agentes;
- uso instantâneo de serviços e negociação ativa - o aspecto de facilitar a distribuição de clientes de serviços pode ser explorada na realização de negociação ativa;
- aumentar a flexibilidade, reuso e eficácia de soluções de problemas baseados em *software* - distribuição de *software* sob demanda e/ou provisão de serviços;
- gerenciamento descentralizado - agentes móveis representando *scripts* de gerenciamento possibilitam a distribuição das atividades de gerenciamento no tempo e no espaço (em

¹Situação onde durante a execução remota do agente, a conexão de rede é desfeita.

diferentes nós da rede), ou seja, trazer o agente de gerenciamento e controle tão próximo quanto possível ou até mesmo junto dos recursos;

- comunicação inteligente - agentes possibilitam o suporte à configuração do ambiente de comunicação do usuário, onde realizam o controle da comunicação recebida e enviada em favor do usuário. Isto inclui adaptação inteligente de serviços para arranjos de acesso a rede e dispositivos finais de usuário, assim como integração de serviços avançados;
- recuperação de informação e suporte a informações dinâmicas - agentes móveis possibilitam um meio efetivo para recuperar informações e serviços dentro de um ambiente distribuído e suportam tipos de informações dinâmicas (p. ex., dentro de correio eletrônico) em sistemas avançados de redes de informação.

Por essas características, sistemas baseados em agentes constituem uma abordagem capaz de modelar eficientemente os aspectos relacionados à personalização e provisão de serviços, à personalização da interface do usuário, ao provimento de um conjunto consistente de serviços do ponto de vista do usuário de forma independente do acesso (Telecom Fixa, Telecom Móvel ou Internet) e disponibilidade global do serviço quando fora do domínio do contrato dos serviços. Com esse intuito, nas próximas seções apresentamos um modelo que utiliza agentes para suporte à mobilidade pessoal e de serviços.

3.3 Especificação *Mobile Agent Facility* (MAF)

Os ambientes de desenvolvimento de sistemas de agentes apresentam diferenças extremas de arquitetura e de implementação, impedindo a interoperabilidade e rapidez no desenvolvimento de tecnologia de agentes móveis [25]. Várias empresas se uniram para propor uma padronização na tentativa de promover a interoperabilidade entre sistemas de agentes. Tal iniciativa chamou a atenção da *Object Management Group* (OMG), que propôs em 1998 a Especificação *Mobile Agent System Interoperity Facility* (MASIF), conhecido hoje como *Mobile Agent Facility*.

A Especificação MAF é uma coleção de definições e interfaces, que provê uma interface interoperável para sistemas de agentes móveis [26]. A interoperabilidade entre vários sistemas de agentes móveis é um dos principais objetivos da Especificação MAF, um dos aspectos da interoperabilidade é que sistemas de agentes desenvolvidos na mesma linguagem podem ser desenvolvidos por fabricantes diferentes [25].

A Especificação MAF define interface para agente de sistemas sem no entanto tratar das operações locais como interpretação, serialização e execução dos mesmos [25]. Ela basicamente suporta identificação de agentes e sistemas de agentes, gerenciamento, transferência, sintaxe de localização e tipo de sistemas de agentes [25], [26]. A Especificação MAF se propõe a ser simples e genérica quanto possível para permitir avanços futuros em sistemas de agentes. Duas são as principais interfaces da especificação: **a. MAFAgentSystem** a qual inclui operações para receber, criar, suspender e terminar agentes e **b. MAFFinder** a qual define operações para registrar, desregistrar, localizar, agentes, lugares e sistemas agentes.

A Fig. 3.4 descreve através de um diagrama a estrutura de um sistema de agentes. Um sistema de agentes, na especificação MAF é uma plataforma que serve de ambiente de execução para agentes e que pode criá-los, transferí-los e terminá-los. Um sistema de agentes deve ser unicamente identificado por seu nome e endereço. Um provedor pode conter um ou mais sistemas de agentes. Um sistema de agentes pode conter um ou mais lugares. Um lugar é o contexto de execução para um ou mais agentes e pode prover funções como o controle de acesso. Quando um agente migra, ele deve especificar um sistema de agentes e um lugar de destino. Se não for especificado o lugar de destino, o sistema de agentes colocará o agente em um lugar padrão. A localização de um agente num dado instante é o lugar onde o mesmo está executando. Cada localização deve ser identificada unicamente por um nome.

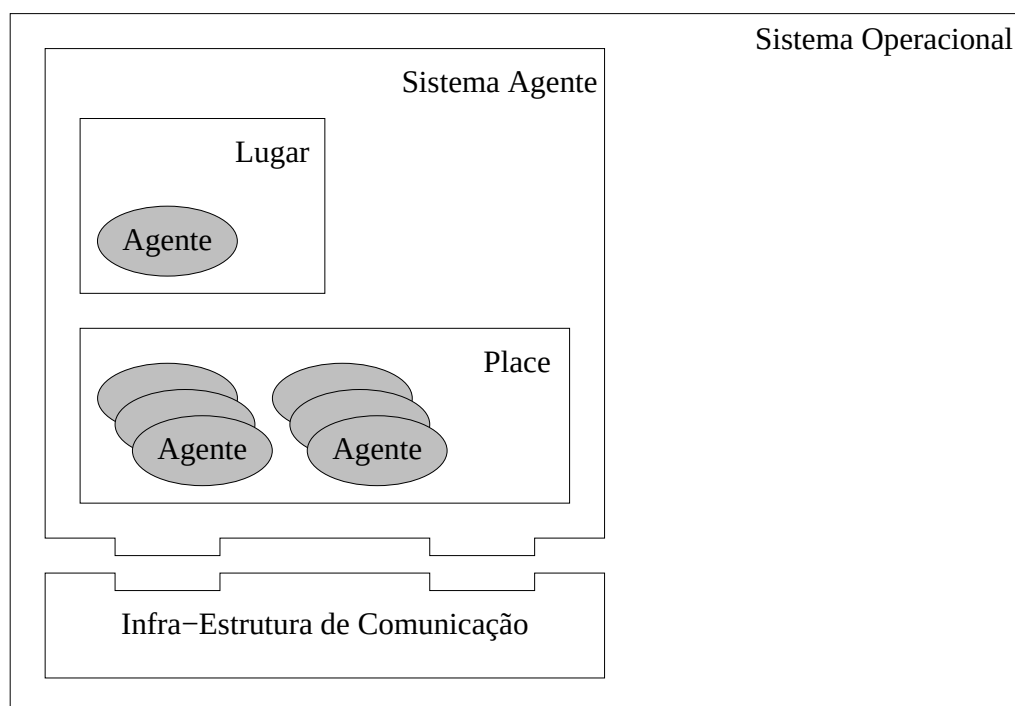


Fig. 3.4: Sistema Agente.

Um sistema de agentes é de um determinado tipo, onde o tipo define o perfil para os agentes suportados. Um tipo define características como: fabricante do sistema de agentes, linguagens que podem ser utilizadas para implementar agentes para este sistema, interface padrão utilizada para migração, padrão utilizado para serialização dos agentes. A partir do tipo do sistema de agentes, um agente pode descobrir dinamicamente se é compatível ou não com aquele sistema. O MAF padroniza também as formas de identificação única dos tipos de sistemas de agentes e a interface para descobrir o tipo de um sistema específico.

Região é um conceito suportado pelo MAF, o qual é um conjunto de sistemas de agentes subordinados a uma mesma autoridade administrativa. Os sistemas que compõem uma região não são necessariamente do mesmo tipo. A região provê um nível de abstração para os clientes externos à região. Se um cliente precisa se comunicar com um agente de uma região específica, não precisa conhecer o endereço ou a localização específica do agente. Basta saber o nome ou endereço da região e o nome do agente. A comunicação é feita entre o cliente e a região que se encarregará de localizar o agente e encaminhar a comunicação solicitada pelo cliente externo. O endereço ou nome da região, na realidade, é o nome ou endereço de um sistema de agentes desta região designando como ponto de acesso da região. Este ponto de acesso será um sistema de agentes visível externamente, como se fosse um *firewall* para a região.

Capítulo 4

Modelo de Agentes para Suporte à Mobilidade de Serviços

O uso da tecnologia de agentes no nosso modelo realça a tecnologia de objetos distribuídos e proporciona autonomia e flexibilidade nos processos de personalização e disponibilização dos serviços quando da necessidade de suporte à mobilidade pessoal e de serviço. Neste modelo, os agentes podem se mover para um outro provedor para avaliar se a nova infra-estrutura oferece suporte a adaptação de serviços do domínio e usuário que representa (agentes com certo grau de inteligência). Não obstante, nossa proposta também se utiliza da tecnologia de objetos distribuídos como forma de interoperar com sistemas legados de subscrição, acesso e serviço presentes nos provedores.

Durante o projeto e implementação de um modelo baseado em agentes para suporte à mobilidade pessoal e de serviços sobre redes heterogêneas fixas e móveis, algumas considerações são estabelecidas no sentido de garantir a interoperabilidade entre as redes Internet, Telecom Fixa e Móvel. A Fig. 4.1 apresenta os principais componentes envolvidos no provimento de serviços, considerando o suporte à mobilidade pessoal e de serviço sobre redes heterogêneas; são eles: provedor nas redes contratada e visitada, usuário e dispositivo no qual o serviço será oferecido. O suporte à mobilidade de terminal, como discutido anteriormente, se dá pela arquitetura de rede em questão, por exemplo, telefones celulares na Rede Telecom Móvel e *notebooks* em uma Rede WLAN. Ainda na Fig. 4.1, o modelo proposto estabelece um Ponto de Referência entre Provedores (PR-Prvds), além de coexistir com sistemas legados de subscrição, acesso e serviço, o que, de certo modo, contribui para preservar a infra-estrutura já existente.

Nossa proposta no uso do paradigma de Agentes Móveis e da tecnologia WWW de forma a

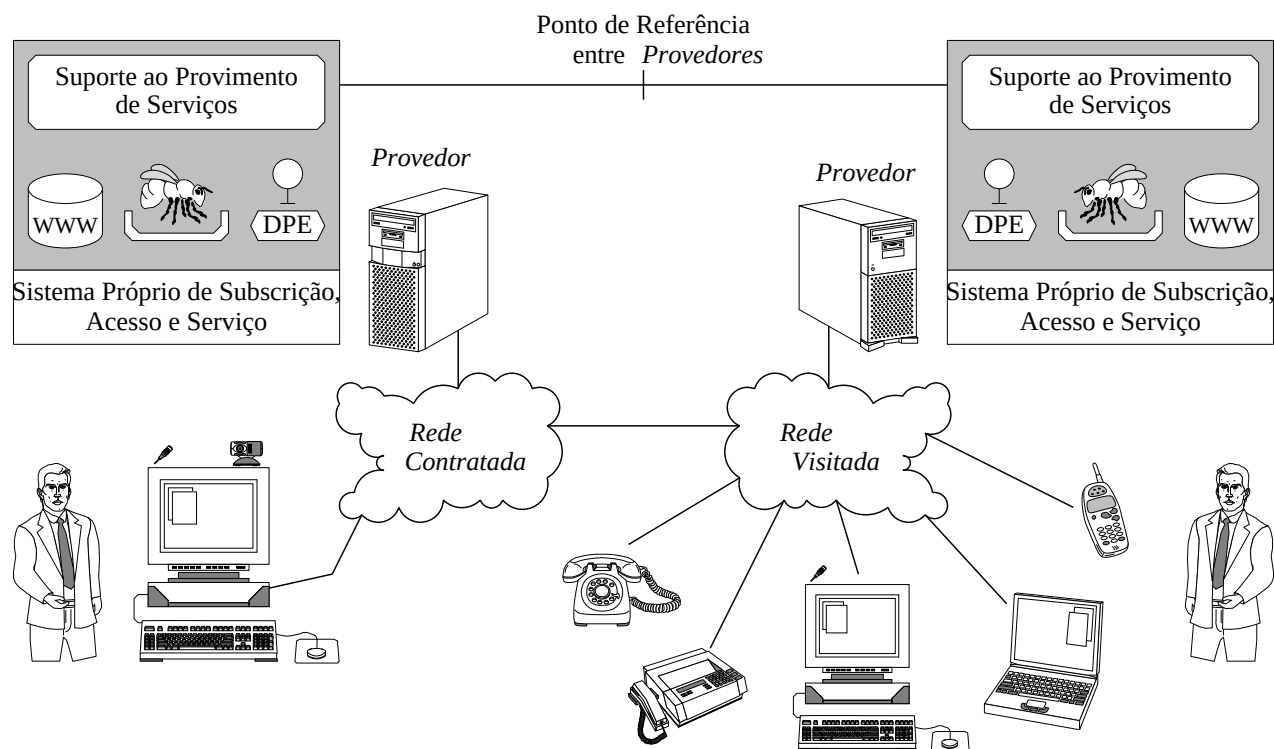


Fig. 4.1: Modelo Proposto para Suporte à Mobilidade.

direcionar o projeto e implementação de uma arquitetura de serviço na qual assinantes possam mudar para domínios nos quais eles não tenham serviços contratados e, ainda assim, utilizarem os seus serviços com suporte do provedor a partir do qual estabeleceram acesso. Apresentamos neste presente capítulo um modelo de suporte à mobilidade pessoal e de serviço para serviços de telecomunicações e de informação.

4.1 Visão Geral do Modelo de Suporte à Mobilidade

Como dito anteriormente, VHE (*Virtual Home Environment*) é definido como um conceito de sistema para portabilidade de serviços personalizados resultante da combinação de capacidades disponíveis no provedor de serviço, nas operadoras de redes e nos equipamentos terminais. Pode ser considerado como a distribuição do perfil do usuário, mantida por um provedor de serviço e distribuída a qualquer momento entre o equipamento terminal, operadora de rede e provedor de serviço. Dentre os componentes que realizam este conceito, estão: *home environment*, usuário, assinante, um ou mais terminais, uma ou mais operadoras de redes de serviço e, possivelmente, um ou mais provedores de serviço.

A Fig. 4.2 ilustra os papéis que os provedores podem desempenhar, ou seja, como provedor

contratado ou como provedor visitado. Provedor contratado é aquele com o qual o usuário contratou serviços e assim dispõe de uma perfil e de um contrato que reflete os direitos concedidos ao usuário pelo subscritor. Já no provedor visitado, o usuário não tem contrato de uso de serviços, mas pode utilizá-lo como ponto de contato inicial para uso de serviços a partir de algum domínio contratado, desde que em conformidade com o ponto de referência entre provedores.

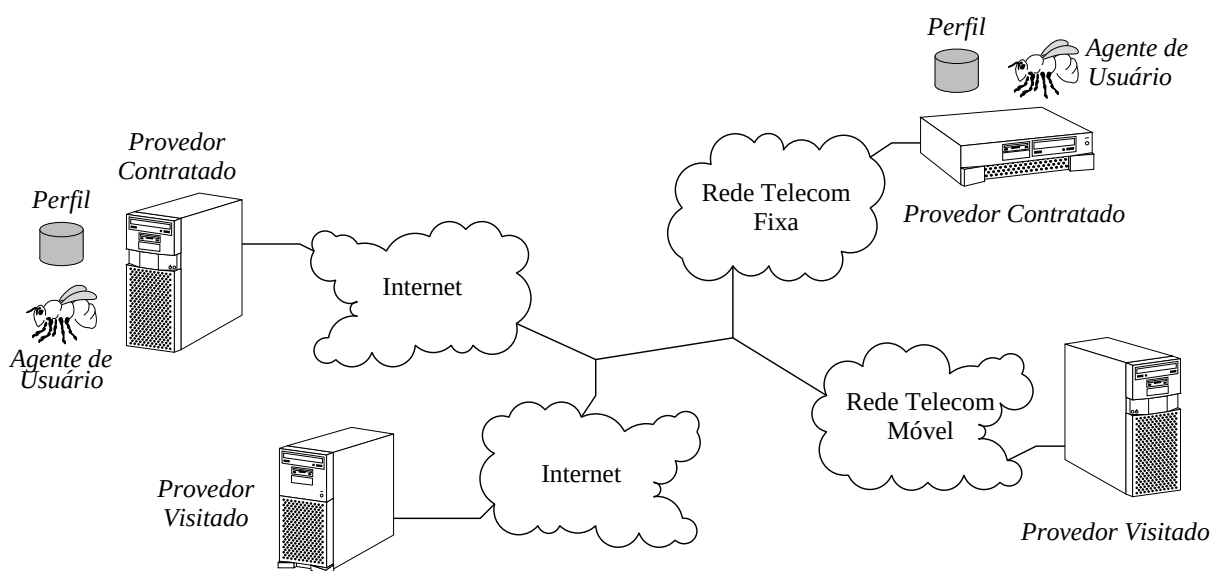


Fig. 4.2: Domínio Contratado de um Usuário.

Um usuário pode ter contrato com vários provedores e assim terá associado uma perfil de usuário e de serviços com cada um deles. No domínio visitado, o provedor visitado proporciona um ponto de contato inicial para um usuário através de seu nome@domínio, que para ser válido na federação, deve ser autenticado pelo provedor contratado. Serviços de algum domínio contratado poderiam então ser disponibilizados no novo domínio, considerando naturalmente a nova infra-estrutura (p. ex., recursos de *hardware* e *software* no provedor e terminal). Neste caso, os provedores além de estarem em conformidade com o PR-Prvds, acordam a contabilidade dos serviços oferecidos através de federação na disponibilização dos mesmos.

Neste trabalho, o suporte à mobilidade na arquitetura de serviço é obtida explorando os seguintes aspectos como parte do ponto de referência entre provedores: **i.** contrato de federação de serviços entre provedores de serviço; **ii.** cada usuário em seu domínio do provedor contratado é representado por um agente de usuário na Fig. 4.2), responsável pelo suporte a mobilidade pessoal; e **iii.** componente capaz de adaptar serviços à nova infra-estrutura do provedor e terminal, suportando assim a mobilidade de serviço.

4.2 Descrição dos Componentes do PR-Prvds

Nesta seção, apresenta-se uma descrição geral dos componentes do Ponto de Referência entre provedores e suas interações com os componentes de subscrição, acesso e serviço do provedor. Estes últimos não fazem parte do PR-Prvds. A caracterização do PR-Prvds consiste na descrição das interfaces IDL dos objetos que o compõem. A Fig. 4.3 apresenta esses componentes e suas interações — 3 componentes fixos e 2 móveis.

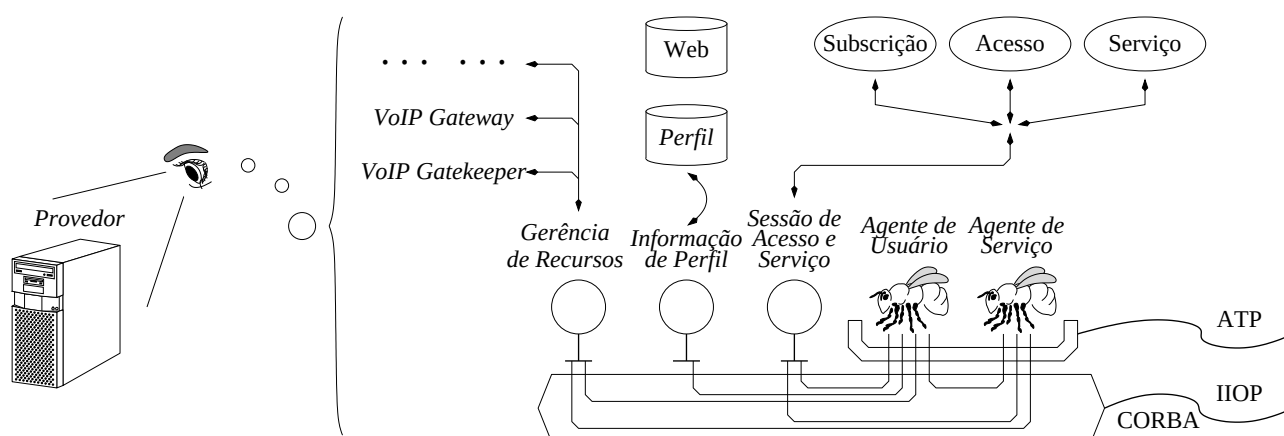


Fig. 4.3: Arquitetura do provedor para Suporte à Mobilidade.

Componente Agente de Usuário

É um componente de serviço que representa o usuário na rede de serviços em cada provedor contratado, e é acessível a partir de um endereço de rede fornecido por um serviço de diretório ou serviço de nomes. O agente de usuário age como um ponto único de contato na adaptação dos serviços de um domínio contratado em um domínio visitado para o usuário.

O agente de usuário negocia o modelo e aspectos suportados pela sessão de acesso e serviço no provedor, além de gerenciar as preferências do usuário no acesso, uso e adaptação dos serviços. Também permite o uso de serviços a partir de diferentes terminais incluindo o suporte do registro do usuário nesses terminais para receber convites.

O agente de usuário pode participar de uma ou mais sessões de acesso e/ou serviço simultaneamente, mas esta limitado pelo número máximo de sessões previamente contratadas no provedor contratado. Num primeiro momento em que o usuário estabelece contrato de serviços em algum provedor, a própria sessão de subscrição do provedor se encarrega de registrar o usuário, retornando seu número de identificação única ou nome@domínio e dados de segurança

associados. Em um segundo momento, o usuário tem o seu agente de usuário instanciado e personalizado no sentido de que pode informar outros provedores contratados nos quais poderá receber convite de outros usuários. Isto sugere que todos os provedores que estão em conformidade com o PR-Prvds, terão a possibilidade de redirecionar um convite que porventura não encontrou o usuário.

A base de conhecimento inserida no agente de usuário é de responsabilidade do provedor contratado, dado que é este provedor que tem conhecimento dos serviços que oferece e como podem ser adaptados a outros domínios. Com base nesse conhecimento, o agente de usuário irá perceber o ambiente visitado, e inferir que serviços podem ser adaptados.

As operações suportadas pelo componente agente de usuário são independentes do serviço, e se encarregam das seguintes funcionalidades:

- estabelecer um relacionamento confiável entre ele, agente de usuário e o provedor pela interação com o componente Sessão de Acesso e Serviço (**SessAssServ**), permitindo assim, negociar o modelo e aspectos suportados pela sessão de acesso e serviço no provedor, assim como obter as referências dos componentes Informação de Perfil (**InfoPrfl**) e Gerência de Recursos (**GercRcrs**);
- através de informações obtidas dos componentes **SessAssServ**, **InfoPrfl** e **GercRcrs**, disponibilizar ao usuário por meio da sessão de acesso e/ou serviço do provedor quais os serviços possíveis de serem adaptados às novas condições;
- considerar as preferências do usuário no acesso, uso e adaptação dos serviços, desde que o modelo e aspectos suportados pelas sessões do provedor o permitam;
- permitir o uso de serviços a partir de diferentes terminais, incluindo o suporte do registro do usuário nesses terminais para que possa receber convites;
- receber convites para participar de sessões iniciadas por outros usuários, tomando então a decisão de alertar o usuário acerca do convite, ou deixá-lo armazenado em uma caixa de entrada para ser exibido quando o usuário estiver com alguma sessão em aberto.

Propriedades (**Properties**) associadas com o domínio, serviços, sessões e usuário são enviadas e/ou recebidas como parâmetros. Assumindo que o agente de usuário coleta informações para com base nelas, informar que tipos de serviços podem ser adaptados no domínio visitado. O agente de usuário percebe o ambiente através dos sensores, e atua fornecendo a lista de serviços que podem ser adaptados a partir do domínio do provedor contratado.

Componente Agente de Serviço

É um componente de serviço que modela uma variedade de aplicações de tal forma a adaptar o serviço ao novo domínio, considerando a perfil do usuário na configuração desses serviços. Isto pode ser conseguido seja pela personalização ou reconfiguração de serviços já existentes.

O agente de serviço traz flexibilidade, permitindo a distribuição de *software* e/ou provisão de serviço sob demanda configurável, portanto, não só o arranjo da rede de acesso mas o dispositivo final do usuário devem ser considerados. Através do suporte apropriado, o agente de serviço pode interagir com outras aplicações ou com o usuário, por exemplo através de interface preestabelecida ou baseada em interface *homem-máquina*.

As operações suportadas pelo componente de agente de serviço são dependentes do serviço, e se encarregam das seguintes funcionalidades:

- através do suporte apropriado (p. ex., interface preestabelecida ou interface homem-máquina), interagir com outras aplicações ou com o usuário;
- através de informações obtidas do componente agente de usuário, requerer a reserva de recursos (*GercRcrs*) e informar ao usuário através da sessão de acesso e/ou serviço (*SessAssServ*) que o início e/ou retomada de um serviço está em andamento.

Componente Informação de Perfil

É um componente de serviço a partir do qual se obtém as preferências do usuário sobre o acesso e uso de serviços. A partir deste componente, também podem ser obtidas informações acerca do contrato de serviços com relação ao uso de infra-estrutura específica de *hardware* e *software* (p. ex., *gateway* para VoIP), tanto para usuários que têm contrato de serviços no domínio quanto para os que não tem. Em ambos os casos, o usuário pode solicitar o uso de serviços de um outro domínio do provedor contratado, embora salvaguardando a avaliação do agente de usuário.

Para a adaptação do serviço ao novo domínio e às capacidades do terminal, mesmo que o usuário não disponha de serviços contratados neste provedor, um contrato de serviços entre provedores terá que ser estabelecido de modo a garantir a adaptação do serviço a esses usuários. Para esses usuários, o uso de serviços se dá após sua autenticação no provedor contratado.

Neste sentido, considerando um usuário que não disponha de contrato de serviços neste provedor, a adaptação de um dado serviço considerando a infra-estrutura presente e as capacidades

do terminal só se dará mediante garantia por parte de um contrato *default* de serviços de que recursos estão disponíveis a estes usuários na adaptação de serviços.

As operações suportadas pelo componente Informação de Perfil são independentes do serviço, e se encarregam das seguintes funcionalidades:

- fornecer ao agente de usuário informações relativas a perfil do usuário, como por exemplo, serviços contratados e infra-estrutura que pode ser usada na adaptação de serviços;
- fornecer informações dos serviços disponíveis e suportados no provedor e que, eventualmente, possam ser exportados para outros domínios (provedor visitado) para fins de adaptação;
- fornecer informações do perfil do serviço relativas a um usuário em particular, bem como das capacidades de um terminal suportado pelo provedor e pela rede de acesso.

Componente Gerência de Recursos

É um componente de serviço a partir do qual se obtém informações da infra-estrutura específica de *hardware* e/ou *software* no suporte da mobilidade de serviços (p. ex., *gateway* para Voz sobre IP). Também é a partir deste componente que o agente de serviço reserva o recurso, assim como gerencia seu ciclo de vida e sua operação.

Deste modo, este componente em conjunto com o agente de serviço, este componente adapta o serviço às novas condições de arranjo de rede, provedor e capacidade do terminal, através da reserva, operação e gerenciamento de recursos específicos de *hardware* e/ou *software* necessárias ao suporte da mobilidade de serviço. Não obstante, o uso desses recursos é dependente de informações contidas no contrato *default* de serviço ou particular para um dado usuário.

As operações suportadas pelo componente Gerência de Recursos são independentes do serviço, e se encarregam das seguintes funcionalidades:

- fornece informações específicas de componentes *hardware* (p. ex., referência do componente) e/ou *software* necessários no suporte à adaptação de serviços;
- permitir a reserva de um recurso (p. ex., um número de IP e *port* de um *gateway* VoIP);
- gerenciar o ciclo de vida e operação do recurso reservado, por exemplo, através de um *daemon* específico para cada recurso;
- dar suporte à mobilidade de serviço oferecendo os meios pelos quais a informação processada pelo agente de usuário será disponibilizada no terminal em questão.

Componente Sessão de Acesso e Serviço

É um componente de serviço que permite a troca de informações entre os componentes agente de usuário e agente de serviço com o usuário. Esta troca de informações ocorre através das sessões de acesso e/ou serviço do provedor com o qual o usuário estabeleceu conexão.

Este componente exporta através de sua interface funcionalidades que permitem convidar um participante ou ser convidado a participar de uma sessão em curso. Neste sentido, pode se fazer necessário componentes adicionais com a função de traduzir a semântica da informação pelo agente de usuário para aquela usada nas Sessões de Acesso e/ou Serviço do provedor.

Por exemplo, considerando o acesso a serviços em uma Rede Telecom Fixa, onde o oferecimento de serviços pode se dar por meio de mensagens previamente gravadas e cuja realimentação ocorre pelo teclado do dispositivo, o agente de usuário ao fornecer o leque de serviços que podem ser disponibilizados no domínio visitado, o fará mediante parâmetros definidos na interface oferecida por este componente. Daí a necessidade de um tradutor para converter, por exemplo, esta lista de serviços em mensagens gravadas.

As operações suportadas pelo componente Sessões de Acessos e Serviços são independentes do serviço, e se encarregam das seguintes funcionalidades:

- transcrever por meio de interface algumas das funcionalidades que caracterizam uma sessão de acesso e/ou serviço de um provedor, e por meio do qual os componentes agente do usuário e agente de serviço interagem com o usuário no terminal;
- receber do agente de usuário a lista de serviços que podem ser adaptados ao terminal por intermédio da infra-estrutura de *hardware* e *software* do provedor;
- fornecer ao agente de usuário o serviço escolhido pelo usuário e, assim, o agente de usuário pode então solicitar a migração do agente de serviço para o domínio em questão;
- encaminhar à sessão de acesso e/ou serviço convites endereçados a usuários que estejam com alguma sessão de acesso e/ou serviço em andamento no provedor;
- encaminhar ao componente responsável pela autenticação no provedor, a solicitação de autenticação de um usuário requisitando acesso em algum provedor da federação.

Opcionalmente, o componente de serviço Sessão de Acesso e Serviço poderia suportar algumas operações que permitissem alterar as condições de ação estabelecidas no agente de usuário, como por exemplo, mudança do componente preferencial para recebimento de convites.

4.3 Interações entre os Componentes do PR-Prvds

Nesta seção, apresentamos o comportamento dinâmico dos componentes do Ponto de Referência entre provedores. Estas interações durante as diversas fases de acesso e uso de um serviço são complementares à descrição dos componentes apresentada na Seção 4.2. As interações serão apresentadas por meio de um conjunto de cenários separados em categorias de acesso e uso, e serão baseados nas ações tomadas pelo usuário no acesso, estabelecimento e uso de serviços. A evolução dos cenários é apresentada por diagramas de eventos e a especificação formal pela Rede de Petri.

A Rede de Petri é uma ferramenta gráfica e matemática que se adapta bem a um grande número de aplicações nas quais as noções de eventos e de evoluções simultâneas são importantes [27]. Com a Rede de Petri e com o auxílio dos diagramas de troca de mensagens foi possível compreender com detalhes a dinâmica dos vários cenários dentro dos quais um usuário possa se encontrar. Através da Rede de Petri foi possível avaliar o modelo proposto no que se refere a ocorrência de *deadlock* e *loops*, freqüentemente encontrados em ambientes distribuídos [28].

4.3.1 Pré e Pós-Condições nos Cenários

Para cada cenário, condições prévias e um estado posterior são esperados. Condições prévias definem as condições que devem estar estabelecidas para que os eventos de um cenário possam evoluir, ou seja, usualmente outros cenários são completados antes que o cenário em questão seja estabelecido. Já as condições posteriores são caracterizadas pelo estado dos vários componentes do cenário após os vários eventos que o compõem terem se completado.

A evolução dos cenários também será apresentada por diagramas de eventos, usados para descrever a sequência com que ocorrem os eventos que compõem um cenário. As setas acompanhadas de rótulos descrevem as ações tomadas pelos objetos, podendo ser internas ou envolverem interações com outros objetos descritos no ponto de referência. Informações ou mensagens de erro retornadas na invocação das operações aparecem entre colchetes (p. ex., [exceção], [lista de parâmetros], etc.).

4.3.2 *Login* e *Logout* em/de um Provedor Contratado

Embora os cenários de nosso interesse sejam aqueles nos quais haja necessidade do suporte à mobilidade pessoal e de serviço, apresentamos este cenário com a intenção de enfatizar a

necessidade de uma sessão de acesso no provedor contratado que considere, na rede à qual pertence, os meios possíveis para atender seus usuários através dos terminais suportados.

Tanto o processo de *login*, quanto o de *logout* necessitam que o provedor seja antes contactado. Para isso, assumimos que a partir de um componente (p. ex., *browser*, serviço de discagem) previamente instalado no terminal, o usuário possa contactá-lo a fim de obter a referência de um ponto inicial de contato. Os passos descrevendo este cenário são apresentados a seguir:

1. usuário contacta o provedor, fornecendo por exemplo nome@domínio do provedor;
2. terminal recebe confirmação de conexão, por exemplo, recebendo a referência da interface de algum componente no domínio do provedor.

No processo de *login* do usuário, assumimos que o terminal do mesmo já detém a referência do ponto de contato inicial no provedor, ou seja, o cenário descrito anteriormente foi estabelecido. Ao iniciar o processo de *login*, os passos descritos em seqüência ou os eventos apresentados no diagrama de eventos de *login* da Fig. 4.4 devem ocorrer.

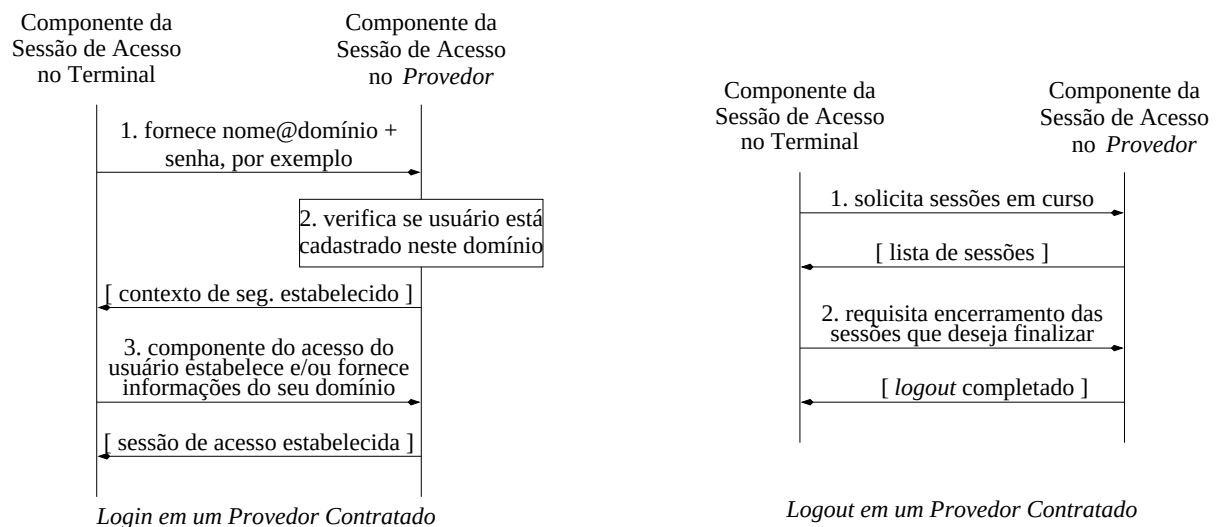
1. o usuário utilizando-se de algum suporte no terminal, fornece seu nome@domínio (ou um número de identificação único) e informações de segurança associadas (p. ex., senha) para que as mesmas possam ser endereçadas ao ponto de contato inicial no provedor;
2. de posse das informações do usuário, o provedor verifica se o mesmo já possui serviços contratados neste domínio e, em caso afirmativo, uma sessão de acesso é estabelecida.

Caso contrário, assume-se que o usuário não tem serviços contratados no domínio, e portanto o provedor assume a condição de provedor visitado para este usuário. O processo de *login* de um usuário em um provedor visitado será descrito na próxima seção;

3. algum componente da sessão de acesso do domínio do usuário fornece ao provedor informações sobre o domínio do usuário, tais como capacidades do terminal, infra-estrutura de suporte a execução de serviços e funcionalidades e aplicações disponíveis.

O provedor reconhece o recebimento dessas informações, e o componente da sessão de acesso no domínio do usuário pode então informá-lo do estabelecimento do acesso.

Ainda na Fig. 4.4, o diagrama de eventos de *logout* mostra como um usuário pode solicitar a finalização das sessões de acesso e por conseguinte de serviço que tenha estabelecido como o provedor. Os passos seguintes descrevem este processo:

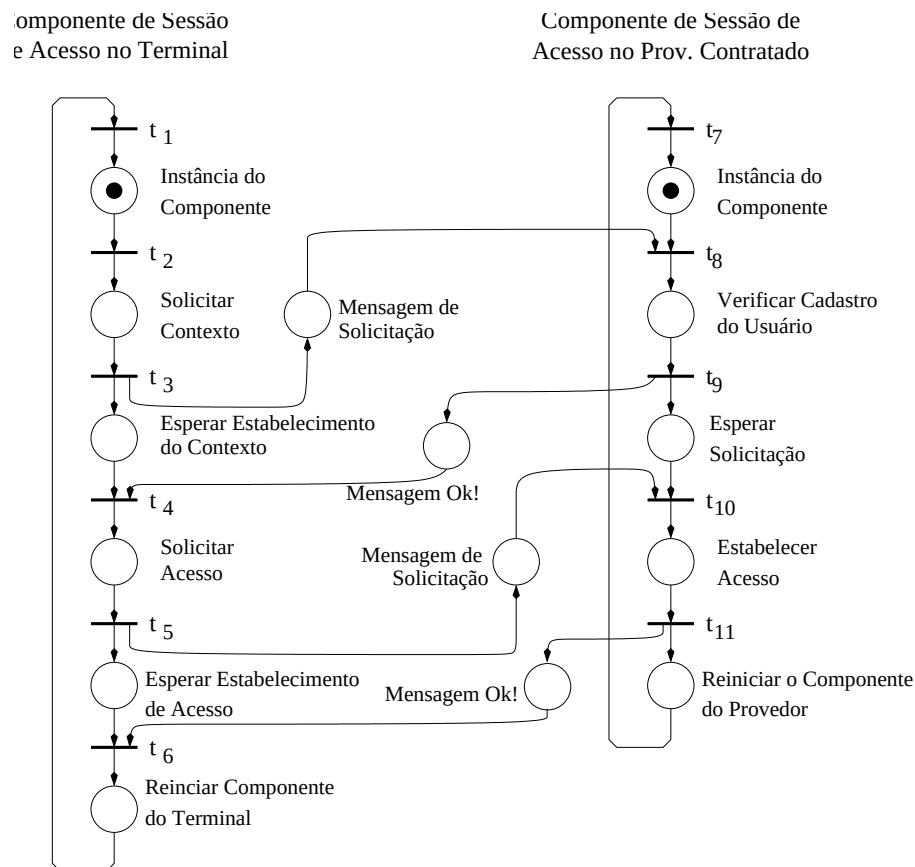
Fig. 4.4: *Login e Logout em/de um Provedor Contratado.*

1. o usuário utilizando-se de uma sessão de acesso, inicia o processo de *logout* verificando que sessões de acesso e por conseguinte de serviço, estão em andamento no provedor;
2. o usuário requisita a finalização da(s) sessão(ões) de acesso e/ou serviço, informando as que deseja finalizar. Instruções para o usuário podem ocorrer no sentido de informá-lo da dependência entre sessões de acesso e serviço.

No provedor, os componentes referentes a(s) sessão(ões) são destruídos ou perdem sua validade, e um sinal de reconhecimento é enviado para o terminal, caracterizando para o usuário que o processo de *logout* foi completado.

A Fig. 4.5 ilustra a Rede de Petri para o cenário de *login* em um provedor contratado anteriormente visto. Como pode ser observado, a dinâmica inicia somente quando o componente de sessão de acesso no terminal é instanciado. Ao fornecer dados na sessão de *login* a transição adjacente na seqüência do fluxo de marcação é sensibilizada (t_2), por conseguinte ao caminhar ocorre transição de estado. Neste estado (Solicitar Contexto), o componente encapsula os dados do usuário para encaminhá-los ao provedor contratado (t_3). Nesta transição, duas marcas partem da mesma nas direções objeto do terminal e do provedor contratado.

Percebe-se que para o objeto do provedor contratado não há dinâmica sem a marcação vinda do terminal do usuário. Na verdade, a dependência se dá nas duas vias, ou seja, mensagens são bloqueantes. O provedor contratado então verifica o cadastro do usuário enviando uma mensagem para o objeto do terminal do usuário através sensibilização da transição t_9 , esta transição também envia uma marca para o próximo estado do provedor visitado, o qual fica

Fig. 4.5: Rede de Petri para *Login* em/de um Provedor Contratado.

aguardando uma próxima solicitação. Depois de estabelecer contexto a transição t_5 solicita o acesso para o usuário. Por fim, a transição t_6 leva ao estado de uma nova sessão de acesso.

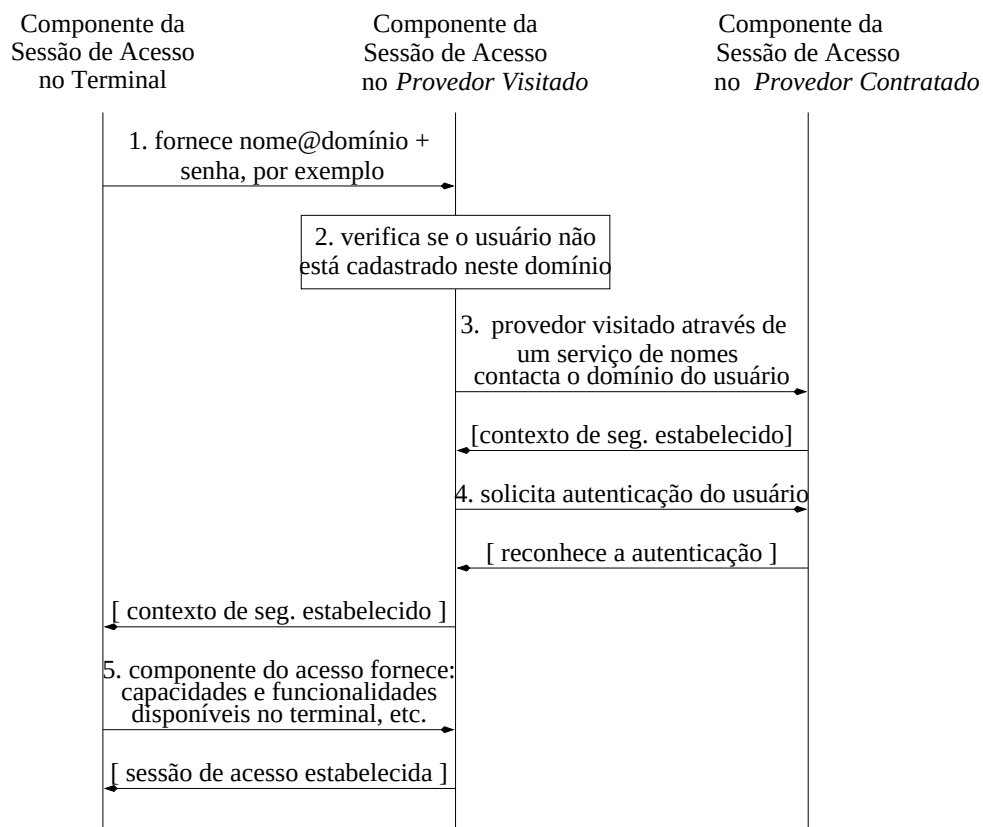
A Rede de Petri para o cenário de *logout* em um provedor contratado anteriormente visto, têm os passos similares ao mostrado na rede de petri, Fig. 4.5, para o cenário de *login* no provedor contratado.

4.3.3 *Login* em um Provedor Visitado

Este cenário permite que o usuário estabeleça uma sessão de acesso em um provedor visitado, ou seja, o usuário não tem contrato de uso de serviços neste domínio. Como pré-condição, assumi-se que o terminal do usuário já possui a referência do ponto de contato inicial no provedor, ou seja, o cenário de contactar o provedor já foi estabelecido.

Ao iniciar o processo de *login*, os passos descritos em sequência ou os eventos apresentados no diagrama de eventos de *login* da Fig. 4.6 descrevem sua evolução.

1. o usuário utilizando-se de algum suporte no terminal, fornece seu nome@domínio (ou um

Fig. 4.6: *Login* em um Provedor Visitado.

número de identificação único) e informações de segurança associadas (p. ex., senha) para que as mesmas possam ser endereçadas ao ponto de contato inicial no provedor;

2. de posse das informações do usuário, o provedor verifica que o mesmo não possui serviços contratados neste domínio, e assim assume a condição de provedor visitado para o usuário;
3. busca através do serviço de nomes o domínio fornecido, ou seja, a referência do componente **SessAssServ**. Informações de contexto de segurança são trocadas para que dados do usuário possam ser submetidos a autenticação;
4. dados identificando o usuário são submetidos a autenticação no provedor contratado de modo a garantir que o usuário tenha contrato de serviços com o domínio fornecido.

Uma vez obtida a autenticação, o endereço de rede do agente de usuário é fornecido ao provedor visitado, de modo que a migração do agente poderá ser solicitada no caso do usuário requerer o uso de serviços do domínio identificado;

5. algum componente da sessão de acesso do domínio do usuário fornece ao provedor informações sobre o domínio do usuário, tais como: capacidades e funcionalidades do terminal,

infra-estrutura de suporte a execução de serviços e aplicações disponíveis.

O provedor reconhece o recebimento dessas informações, e o componente da sessão de acesso no domínio do usuário pode então informá-lo do estabelecimento do acesso.

Uma vez que o usuário tenha sido identificado e a sessão de acesso tenha sido completada, outros cenários poderão ser estabelecidos, por exemplo, que serviços de um provedor contratado podem ser adaptados considerando as funcionalidades disponíveis no provedor visitado e terminal.

A Fig. 4.7 ilustra a Rede de Petri para o cenário anteriormente visto. Como pode ser observado, a dinâmica inicia somente quando o componente de sessão de acesso no terminal é instanciado. Ao fornecer dados na sessão de *login* a transição adjacente na seqüência do fluxo de marcação é sensibilizada (t_2), por conseguinte ao caminhar ocorre transição de estado. Neste estado (Solicitar Contexto), o componente encapsula os dados do usuário para encaminhá-los ao provedor visitado (t_3). Nesta transição, duas marcas partem da mesma nas direções objeto local e remoto.

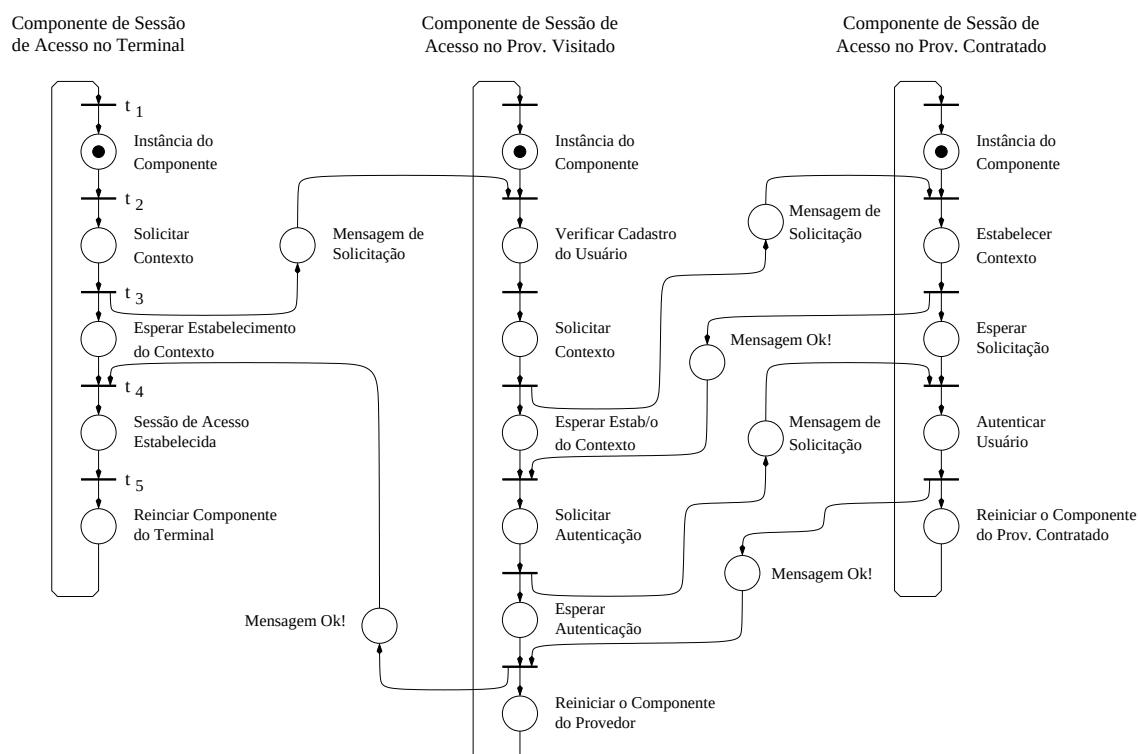


Fig. 4.7: Rede de Petri para *Login* em/de um Provedor Visitado.

Percebe-se que para o objeto remoto (componente no provedor visitado) não há dinâmica sem a marcação vinda do terminal do usuário. O mesmo raciocínio pode ser aplicado entre o

componentes do provedor visitado e do provedor contratado. Na verdade, a dependência se dá nas duas vias, ou seja, mensagens são bloqueantes. Voltando ao componente no domínio do usuário, a marcação prepara a transição t_4 para ser sensibilizada, ou seja, só será sensibilizada quando do retorno da invocação do provedor visitado. Por fim, a transição t_5 leva ao estado de uma nova sessão de acesso.

4.3.4 Logout de um Provedor Visitado

Os passos que descrevem a finalização das sessões de acesso e por conseguinte de serviço que um usuário tenha estabelecido como o provedor visitado, são os mesmos apresentados na Seção 4.3.2, bem como o diagrama de eventos apresentado na Fig. 4.4.

4.3.5 Listar Serviços Contratados e/ou Adaptáveis

Este cenário permite ao usuário requisitar a lista de serviços contratados no provedor contratado ou serviços possíveis de serem adaptados no provedor visitado. Não é objetivo deste cenário descrever como o usuário contrata os serviços ou se torna assinante dos mesmos. Como cada provedor possui seu próprio sistema de subscrição, assim como de acesso e/ou de serviço, a contratação de serviços está circunscrita a cada provedor.

Para listar os serviços, assumimos que o usuário já estabeleceu uma sessão de acesso com o provedor visitado ou com o provedor contratado. A evolução deste processo se dá pelos passos descritos a seguir ou pelos eventos apresentados no diagrama de eventos da Fig. 4.8.

1. um componente no terminal requisita no provedor a lista de serviços contratados através de opção correspondente na aplicação. A lista de serviços retornada inclui os serviços contratados mais serviço opcional, ou seja, a opção de serviço de outro provedor;
2. se o usuário estabeleceu sessão de acesso com algum provedor contratado e está solicitando serviço opcional, faz-se necessário fornecer o endereço de um outro provedor contratado a partir do qual deseja-se usar serviços contratados que possam ser adaptados às capacidades do terminal e do provedor ao qual se estabeleceu conexão. Neste caso, os dados do usuário terão que ser autenticados no provedor a partir do qual deseja-se usar serviços, a fim de que a migração do referido agente de usuário possa ser requerida.

Mas, caso tenha estabelecido uma sessão de acesso com um provedor visitado, a sua identificação já contempla o domínio do qual deseja usar serviços contratados que possam ser adaptados às capacidades do terminal e do provedor ao qual está conectado;

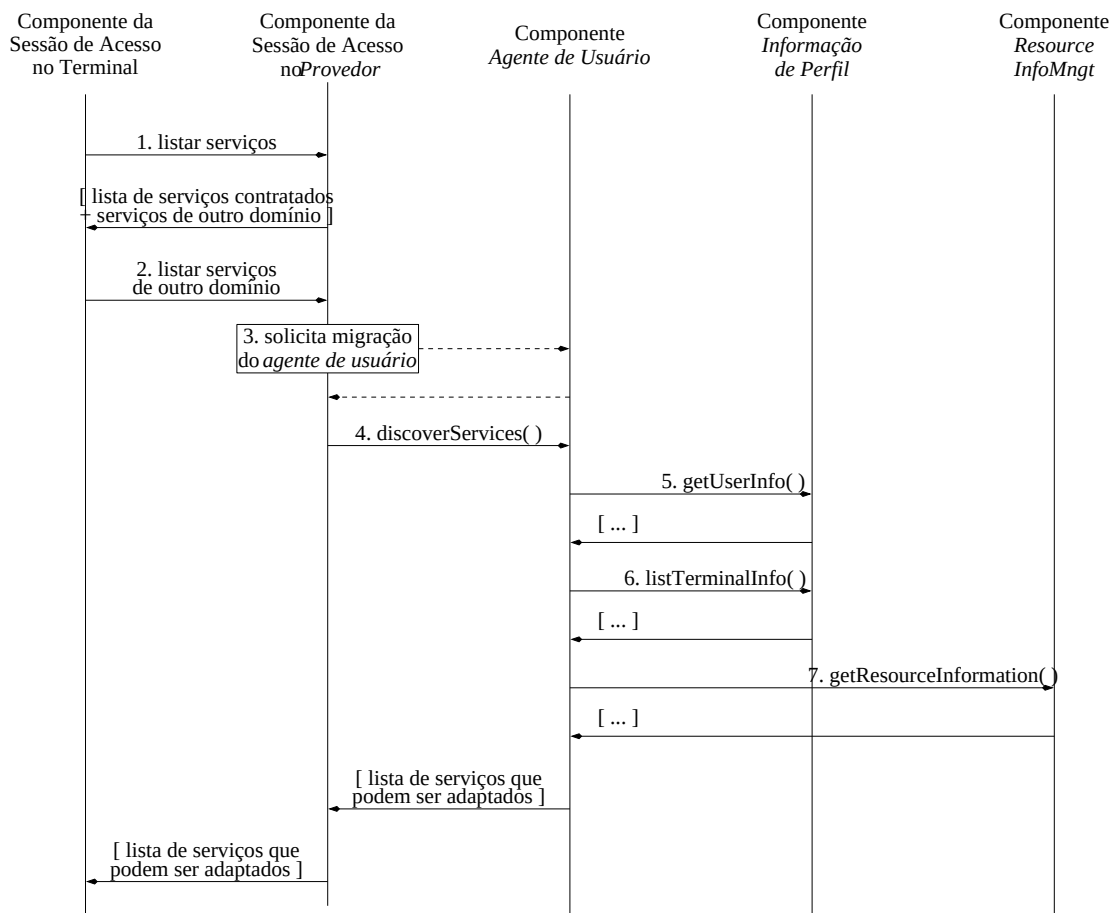


Fig. 4.8: Listar serviços contratados e/ou adaptáveis.

3. para que uma nova lista de serviços possa ser disponibilizada, o provedor precisa solicitar a migração do agente de usuário de forma que o mesmo possa avaliar que serviços são possíveis de serem adaptados, considerando as capacidades do terminal, a infra-estrutura do novo domínio e a perfil do usuário do provedor contratado especificado;
4. após o componente da sessão de acesso no provedor visitado receber a informação de que o agente de usuário migrou e já se instalou na agência, o mesmo solicita ao agente de usuário a lista de serviços que podem ser adaptados ao conjunto provedor visitado e terminal através da operação `discoverServices()` do agente de usuário. Após obter as informações necessárias à análise, a operação `discoverServices()` do objeto agente de usuário retorna a lista de possíveis serviços a serem adaptados às capacidades do terminal considerando o suporte da infra-estrutura do provedor;
5. o agente de usuário, com o intuito de fornecer a lista de serviços solicitada, obtém informações do usuário invocando a operação `getUserInfo()` no objeto `InfoPrfl` ;

6. ainda sobre o objeto `InfoPrfl`, informações das capacidades do terminal corrente são obtidas pelo agente de usuário invocando a operação `getTrmnInfo()`;
7. obtenção de informações acerca de que infra-estrutura de *hardware* e *software* (p. ex., *gateway* p/ voz sobre IP) pode ser utilizada pelo usuário no provedor, invocando a operação `getResourceInformation()` no objeto `GercRcrs`.

A Fig. 4.9 ilustra a Rede de Petri para o cenário anteriormente visto. Observa-se que a dinâmica inicia quando o componente de sessão de acesso no terminal é instanciado. O estado seguinte (t_2) é sensibilizado, e este solicita a lista de serviços, assim que obtém a lista, solicita a lista de serviços contratados em outro domínio (t_3). Então a sessão de acesso no Provedor solicita a migração do agente, para que o mesmo possa descobrir os serviços adaptáveis.

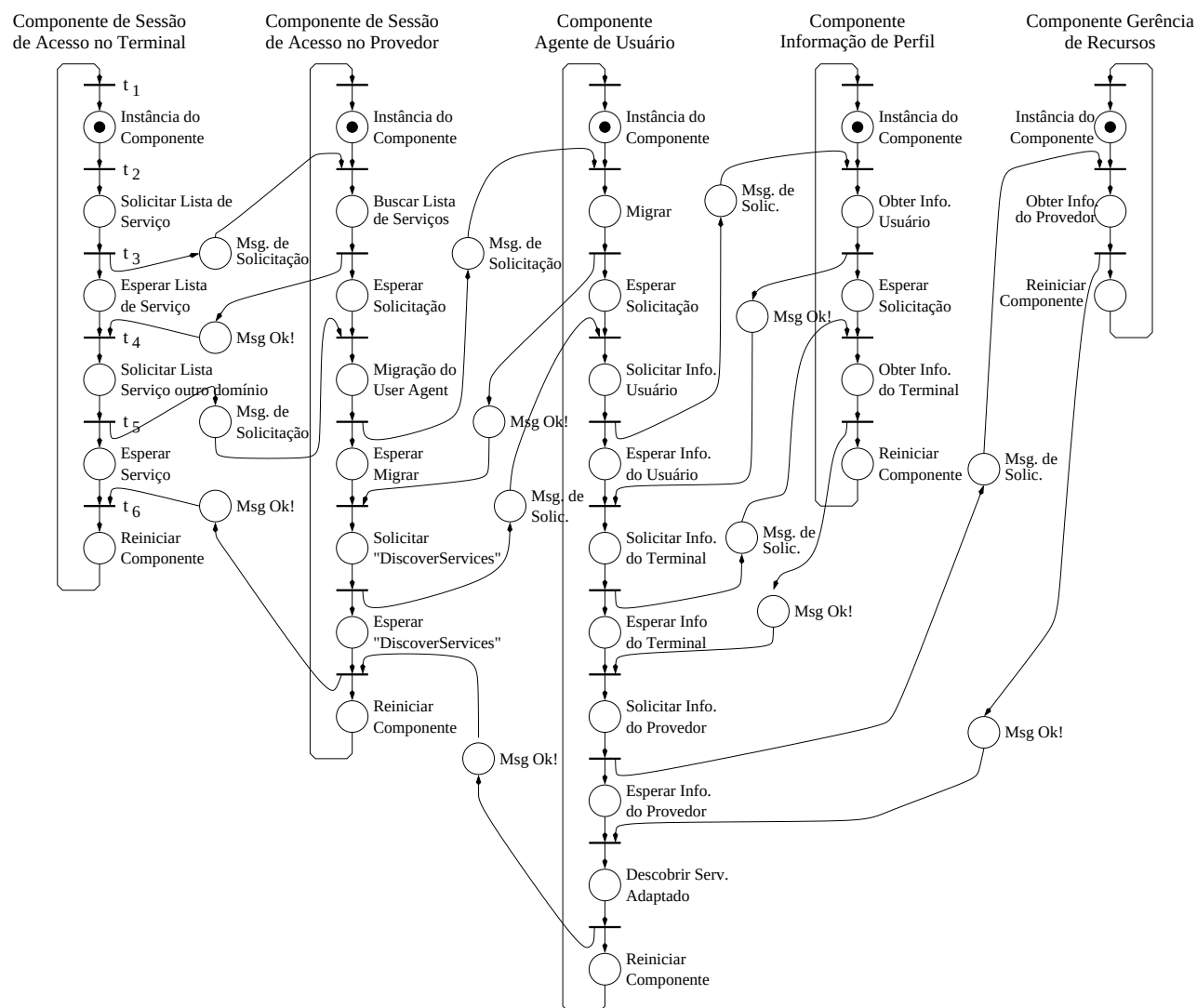


Fig. 4.9: Rede de Petri para serviços contratados e/ou adaptáveis.

Uma variação deste cenário retrata o cenário em que o agente de usuário migra para um provedor em decorrência de uma programação preestabelecida para o agente. Neste caso, após analisar que possíveis serviços podem ser adaptados no novo domínio, o conjunto de possíveis serviços pode ser disponibilizado ao provedor invocando a operação `givePossibleServices()` no objeto `SessAssServ`. Assim, quando do estabelecimento da sessão de acesso pelo usuário num momento posterior, a lista de serviços já estará disponível, restando ao usuário a seleção.

Os passos que descrevem este processo são apresentados a seguir, enquanto que o diagrama de eventos correspondente pode ser visualizado na Fig. 4.10.

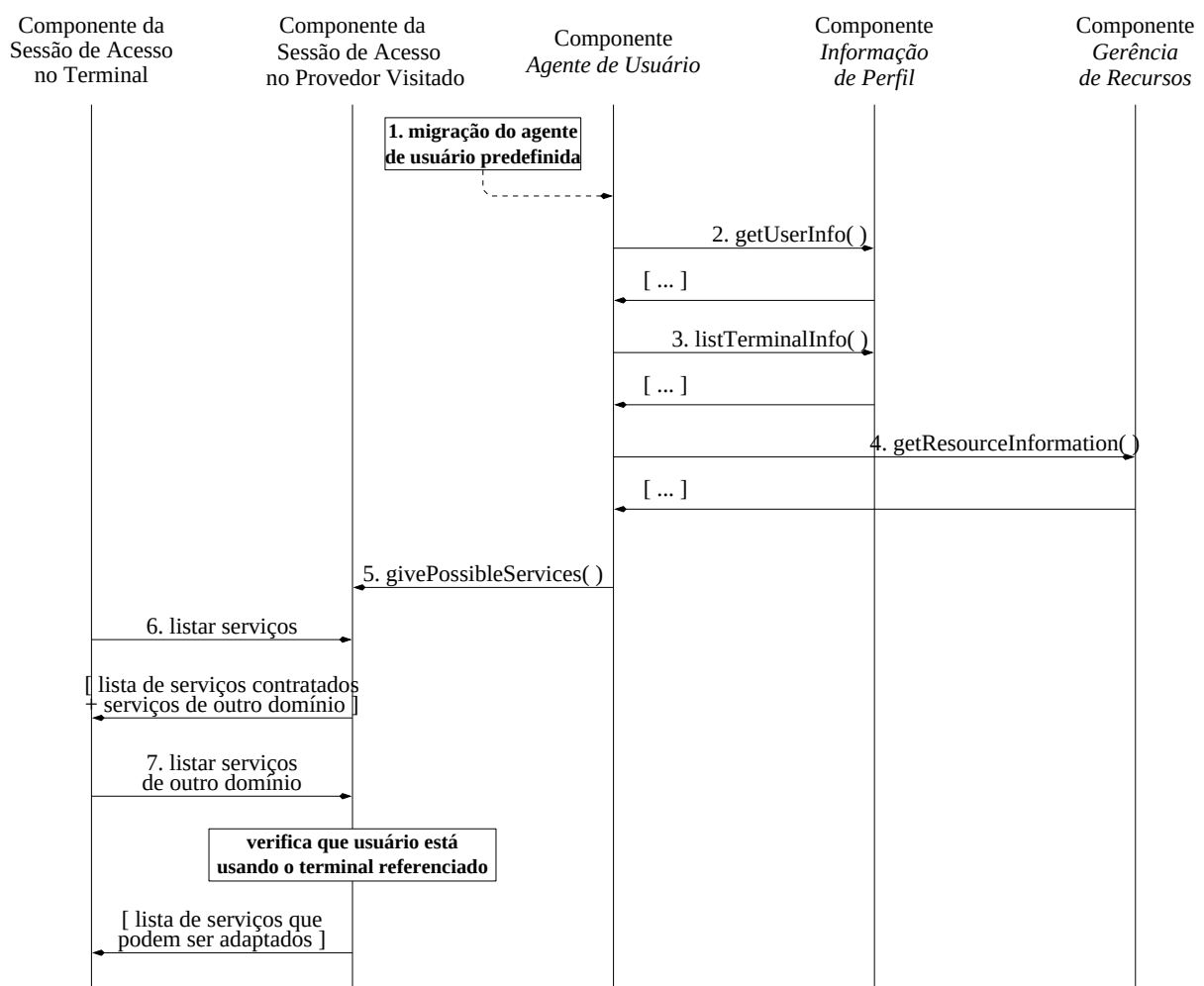


Fig. 4.10: Listar serviços contratados e/ou adaptáveis (migração do agente de usuário predefinida).

1. migração do agente de usuário para um provedor em decorrência de uma programação predefinida para o agente; agente de usuário antecipando as ações do usuário, coleta as informações necessárias à análise da lista de possíveis serviços a serem adaptados às capacidades do terminal registrado com suporte da infra-estrutura oferecida pelo provedor;

2. o agente de usuário com o intuito de fornecer a lista de serviços solicitada obtém informações do usuário invocando a operação `getUserInfo()` no objeto `InfoPrfl` ;
3. ainda sobre o objeto `InfoPrfl` , informações das capacidades do terminal corrente são obtidas pelo agente de usuário invocando a operação `getTrmnInfo()` ;
4. verifica que infra-estrutura de *hardware* e *software* pode ser utilizada pelo usuário no provedor, invocando a operação `getResourceInformation()` no objeto `GercRcrs` ;
5. agente de usuário informa ao objeto `SessAssServ` a lista de serviços que podem ser adaptados no domínio para um usuário, invocando a operação `givePossibleServices()` ;
6. um componente no terminal requisita no provedor a lista de serviços contratados; e se o usuário tem serviços contratados, a lista de serviços retornada inclui os serviços contratados mais serviço opcional, ou seja, a opção de serviço de outro provedor;
7. usuário solicita serviço opcional fornecendo o endereço de algum provedor contratado a partir do qual deseja usar serviços contratados que possam ser adaptados às capacidades do terminal e do provedor ao qual está conectado.

É responsabilidade da sessão de acesso do provedor disponibilizar a lista de possíveis serviços retornada pelo agente de usuário e, assim, pode se fazer necessário componentes adicionais com a função de traduzir a semântica da informação fornecida para aquela utilizada pela sessão de acesso e/ou serviço do provedor corrente.

4.3.6 Selecionar Serviço no Provedor Contratado

Apresenta-se este cenário com a intenção de enfatizar que a seleção de um serviço no provedor contratado está circunscrita ao sistema de acesso e/ou serviço do mesmo, e portanto cabe ao provedor prover suporte a seleção, distribuição, uso e gerenciamento do serviço.

O mesmo é válido para o sistema de armazenamento dos dados de perfil de usuário e de aspectos de configuração dos serviços, exceto que quando extraídos da base de dados por operações definidas nos `InfoPrfl` e `GercRcrs` , serão apresentados conforme o especificado no ponto de referência entre provedores.

De modo geral, para selecionar um serviço previamente contratado no provedor contratado, os passos descritos a seguir devem ocorrer. O diagrama de eventos correspondente aos passos descritos a seguir pode ser visualizado na Fig. 4.11.

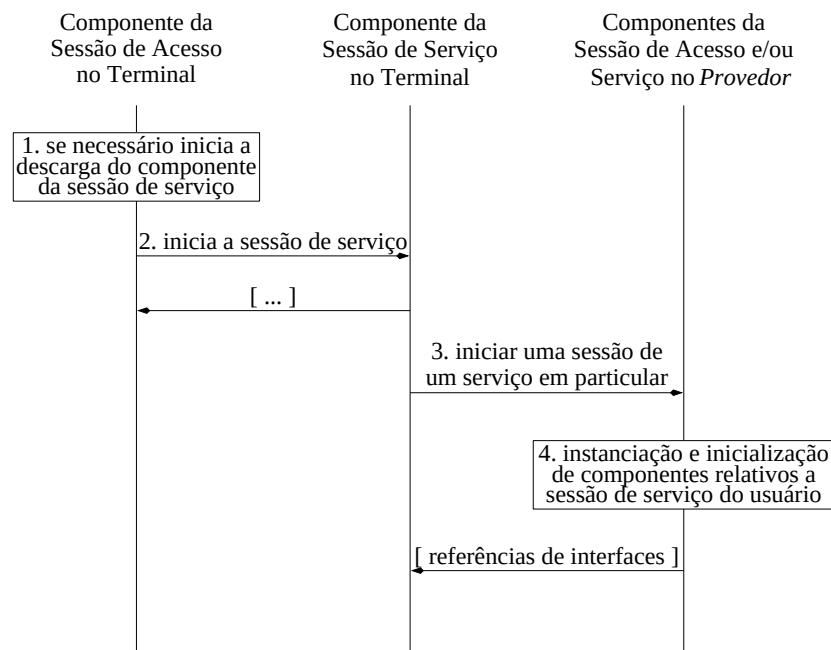


Fig. 4.11: Seleção de Serviço no Provedor Contratado.

1. no terminal, o componente da sessão de acesso verifica a instalação prévia do componente correspondente ao serviço ou de parte dele, e se necessário solicita a descarga do componente no terminal;
2. componente do acesso no terminal inicializa o componente da aplicação fornecendo a interface de algum componente no provedor responsável pela instanciã o e inicializa o de uma sess o do servi o correspondente a do terminal do usu rio;
3. requisi  o para que o componente no provedor respons vel pela sess o do servi o correspondente a do terminal inicie uma sess o de servi o especificada; uma vez sendo instanciadas e iniciadas, as refer ncias dos componentes correspondentes   sess o s o retornadas;
4. instanci o e iniciais o dos componentes do servi o especificado.

O objetivo deste cen rio foi caracterizar a sele  o de um servi o, e para isto assumimos que o usu rio j  havia estabelecido uma sess o de acesso. N o nos preocupamos em nenhum momento, em detalhar os componentes que comp em a arquitetura de servi o de algum provedor em particular, mas somente descrever em linhas gerais as principais opera  es.

A Fig. 4.12 ilustra a Rede de Petri para o cen rio anteriormente visto. Observa-se que a din mica inicia quando o componente de sess o de acesso no terminal   instanciado. O estado seguinte (t_2)   sensibilizado, e este solicita a iniciais o do servi o a sess o de servi o do terminal

e este por sua vez solicita da sessão de acesso/serviço do provedor, retornando a sessão de acesso do terminal o serviço instanciado, para que o mesmo possa ser utilizado.

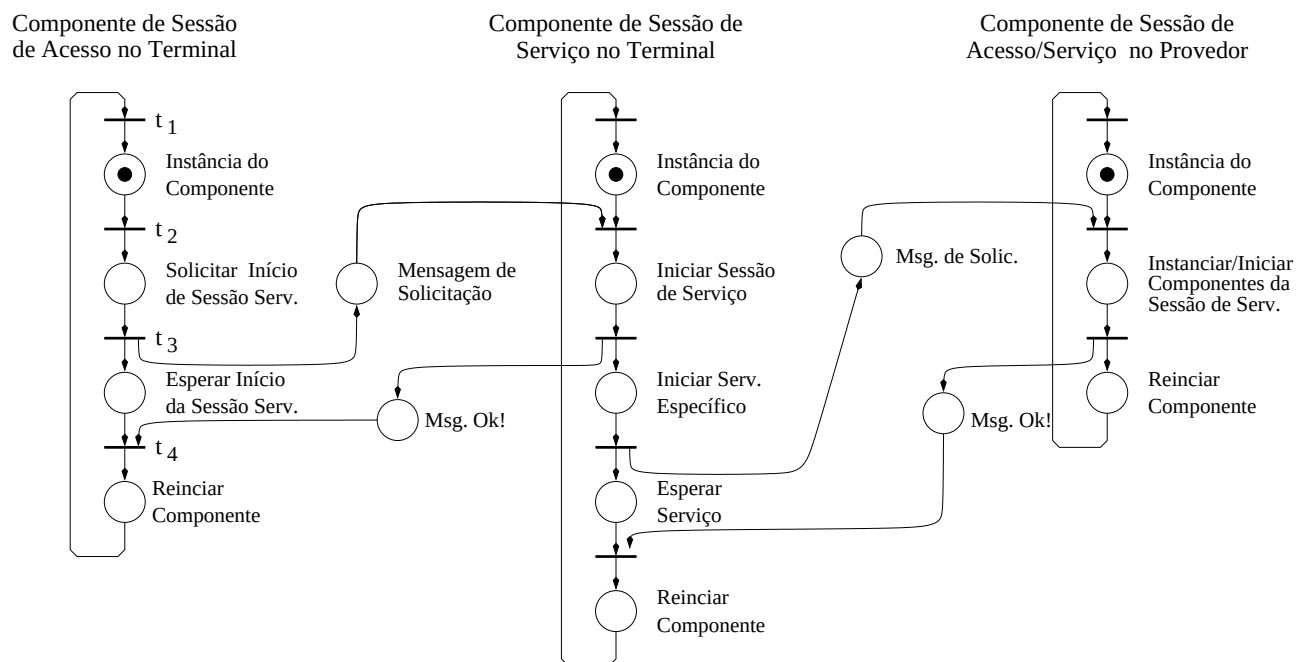


Fig. 4.12: Rede de Petri para Seleção de Serviço no Provedor Contratado.

4.3.7 Selecionar Serviço no Provedor Visitado

Diferentemente do cenário anterior, a seleção de um serviço no provedor visitado não está circunscrita somente ao seu domínio, pois a seleção de um serviço de um outro domínio envolve a verificação das capacidades do terminal, da infra-estrutura do provedor e da perfil do usuário.

Para selecionar um serviço no provedor visitado, assumimos que o usuário já estabeleceu uma sessão de acesso com o provedor como descrito na Seção 4.3.3. Caso o usuário já tenha listado os serviços opcionais disponíveis, o agente de usuário já possui estas informações e assim sendo, selecionar um serviço implica em primeiro solicitar a lista de serviços que podem ser adaptados. Os passos que descrevem este processo são apresentados a seguir, enquanto que o diagrama de eventos correspondente pode ser visualizado na Fig. 4.13.

1. usuário solicita serviço opcional fornecendo o endereço de algum provedor contratado a partir do qual deseja usar serviços contratados que possam ser adaptados às capacidades do terminal e do provedor ao qual está conectado;
2. componente da sessão de acesso do provedor verifica se o usuário já solicitou a lista de serviços opcionais disponíveis; em caso negativo, a lista de serviços é solicitada invocando a operação `discoverServices()` no mesmo objeto;

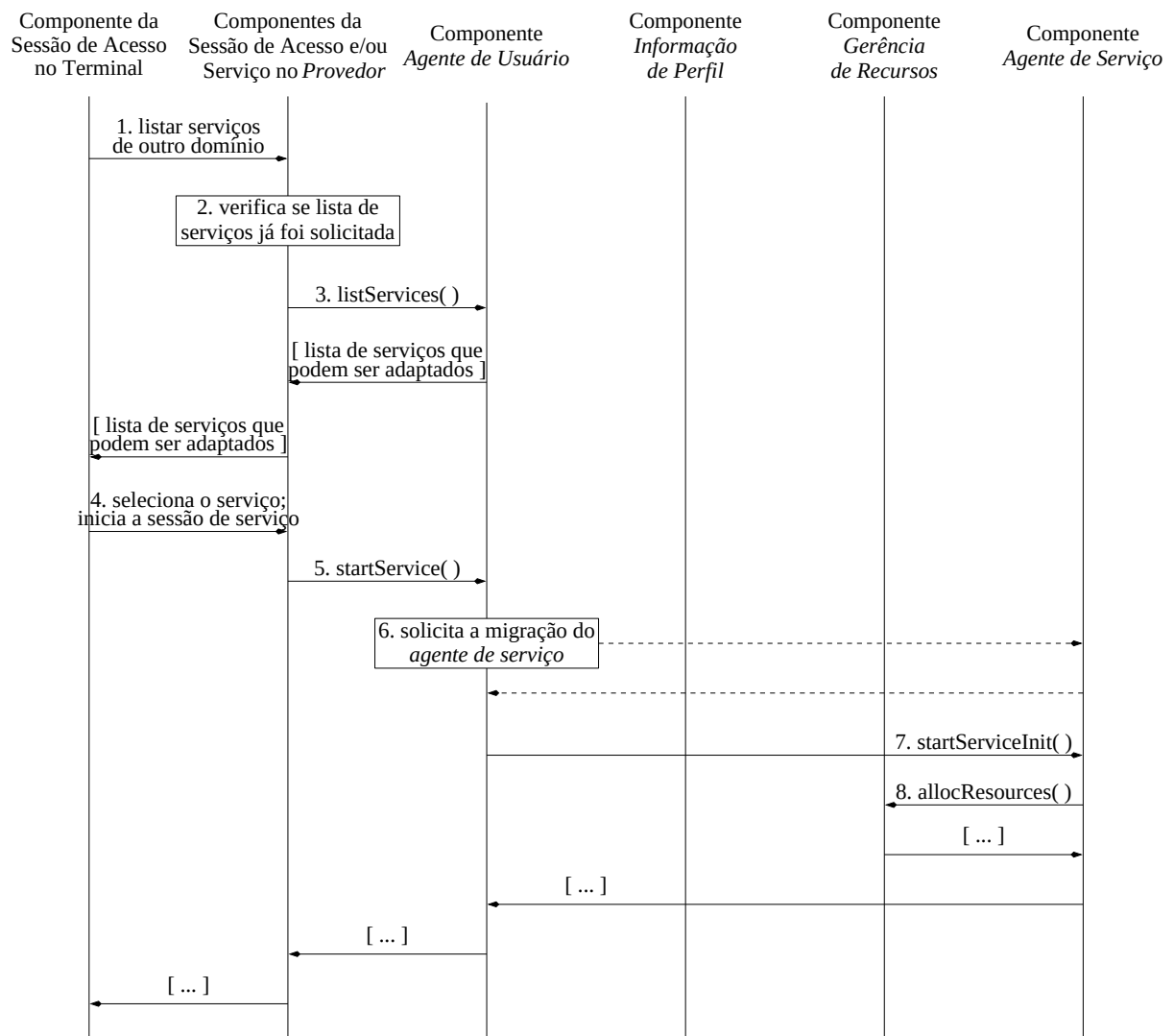


Fig. 4.13: Seleção de Serviço no Provedor Visitado.

3. componente da sessão de acesso e serviço do provedor invoca a operação `listServices()` no objeto `UserAgent` ; a lista de serviços adaptáveis ao novo domínio é retornada para o componente que, por sua vez, por intermédio de outros componentes do provedor pode então ser disponibilizada no terminal do usuário;
4. usuário seleciona serviço, iniciando uma sessão específica para o serviço junto ao provedor;
5. componente da sessão de acesso e serviço do provedor invoca a operação `startService()` no objeto `UserAgent` , uma vez que o serviço terá que ser adaptado às novas condições;
6. agente de usuário solicita a migração do agente de serviço do *retailer home* para em conjunto com o provedor visitado adaptar o serviço considerando as capacidades do terminal;
7. após receber informação de que o agente de serviço migrou e já se encontra instalado

na agência, o agente de usuário solicita o início de um serviço, cujos parâmetros são fornecidos através da operação `startServiceInit()` no objeto `ServAgnt` ;

8. o componente agente de serviço requisita a alocação dos recursos necessários invocando a operação `allocResources()` no objeto `GercRcrs` e, em sendo confirmada a requisição, os objetos `UserAgnt` , `SessAssServ` e o terminal recebem o *status* de que a sessão de serviço está estabelecida. O serviço pode, então, ser gerenciado por alguma interface homem-máquina disponível no terminal para o usuário.

A Fig. 4.14 ilustra a Rede de Petri para o cenário anteriormente visto. Observa-se que a dinâmica inicia quando o componente de sessão de acesso no terminal é instanciado. A transição adjacente do fluxo de marcação é sensibilizada (t_2), por conseguinte ocorre transição de estado, e este solicita a lista de serviço de outro domínio, o componente encapsula os dados do provedor contratado para encaminhá-los ao provedor visitado (t_3).

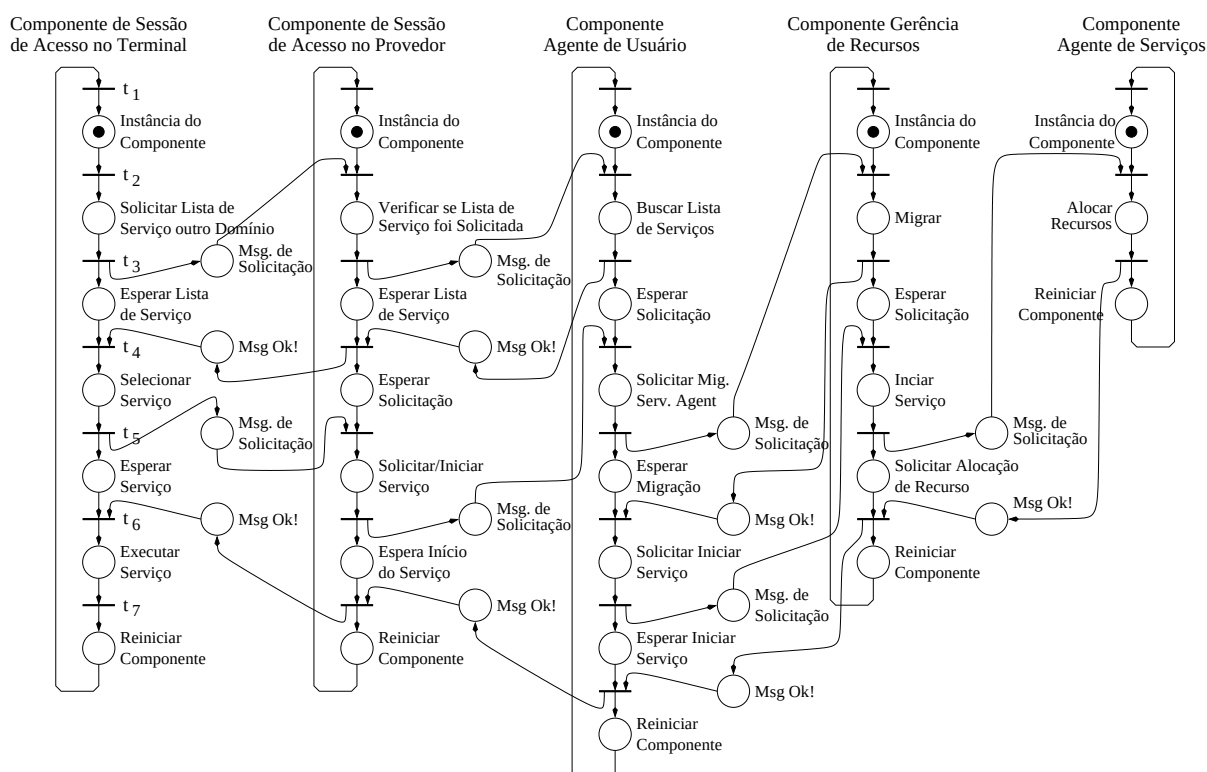


Fig. 4.14: Rede de Petri para Seleção de um Serviço no Provedor Visitado.

Percebe-se que para o objeto remoto (componente no provedor visitado) não há dinâmica sem a marcação vinda do terminal do usuário. O mesmo raciocínio pode ser aplicado entre os componentes do provedor visitado, do agente de usuário, do gerenciador de recursos e do agente

de serviço. A construção da matriz de marcações acessíveis nos mostra a ausência de *deadlocks* e *loops* nesta rede.

4.3.8 Finalizar Serviço no Provedor Contratado

Não estamos preocupados neste cenário em detalhar os componentes que estão envolvidos no processo de finalização de uma sessão de serviço, mas enfatizar que alguns aspectos devem ser considerados independentemente de qual seja o provedor. Além disso, este cenário nos permite familiarizar com o processo de finalização de uma sessão de serviço, e ao mesmo tempo enfatizar a presença de tais componentes que, de algum modo, também serão utilizados na finalização de um serviço no provedor visitado (Seção 4.3.9).

De modo geral, para finalizar uma sessão de serviço estabelecida no provedor contratado e que envolve múltiplos participantes, todos os usuários agrupados na mesma sessão de serviço terão suas sessões terminadas quando em concordância. Os passos descritos a seguir mais o diagrama de eventos da Fig. 4.15, nos mostram como evolui este processo.

1. componente da sessão de acesso e serviço do terminal solicita a finalização do serviço;
2. gerente da sessão de serviço do usuário¹ solicita finalização da sessão para o gerente responsável pela coordenação da mesma sessão de outros usuários envolvidos;
3. gerente geral da sessão de serviço² encaminha a solicitação aos gerentes da sessão de serviço dos demais usuários que estejam envolvidos na sessão;
4. gerentes da sessão de serviços de cada usuário encaminham a solicitação a seus usuários, que poderão decidir em favor ou não da finalização da sessão conforme direitos de propriedade concedidos quando do seu estabelecimento;
5. assumindo a concordância na finalização, o gerente geral informa aos gerentes de sessão de serviço dos usuários o término da sessão;
6. cada gerente da sessão de serviço de cada usuário encaminha a solicitação a seus usuários; nesta fase informações que sintetizem a utilização do serviço (p. ex., serviço utilizado, tempo de uso, participantes envolvidos, etc.) podem ser disponibilizadas para o usuário;

¹Como descrito na Seção, o Gerente da Sessão de Serviço do Usuário guarda o contexto de um participante ou recurso em uma sessão de serviço.

²Como descrito na Seção, o Gerente da Sessão de Serviço provê suporte às capacidades do serviço que são compartilhadas entre os participantes em uma sessão de serviço.

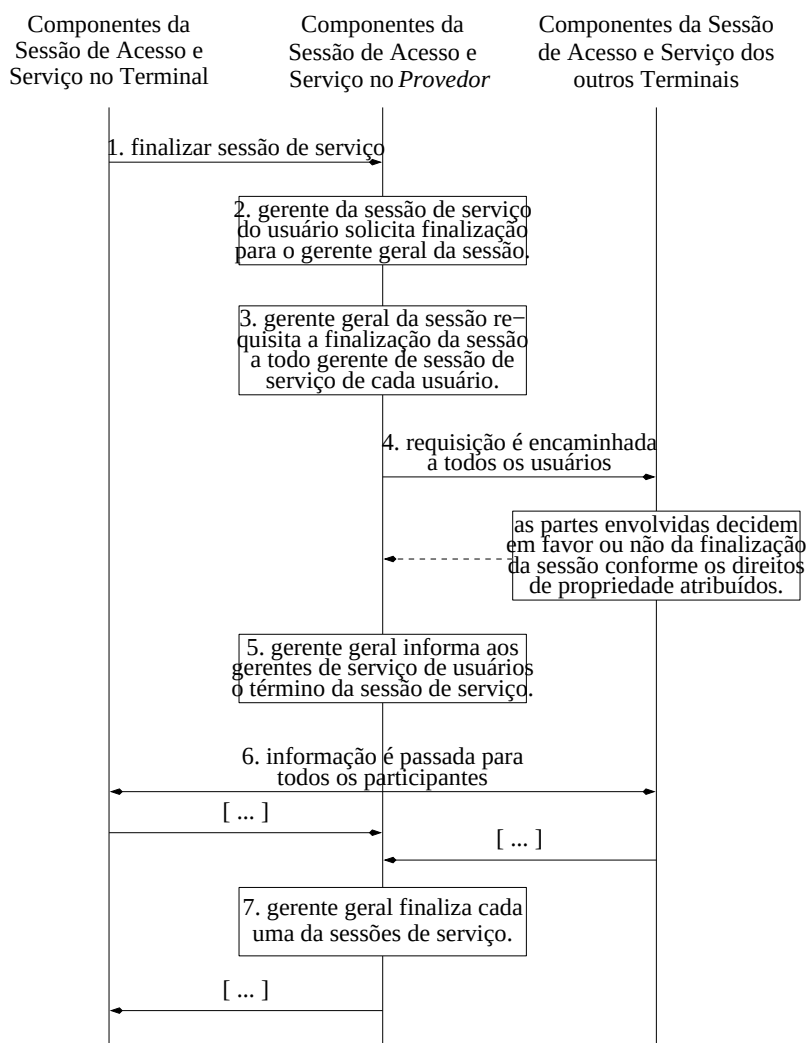


Fig. 4.15: Finalização de Serviço no Provedor Contratado.

7. por fim, o gerente geral finaliza a sessão e eventualmente libera recursos alocados durante o estabelecimento da sessão de serviço.

O objetivo deste cenário foi caracterizar a finalização de um serviço no provedor contratado e, para isto, assumimos que o usuário já havia estabelecido uma sessão de acesso, bem como estivesse em andamento ou suspensa uma sessão de serviço.

A Fig. 4.16 ilustra a Rede de Petri para o cenário anteriormente visto. Observa-se que a dinâmica inicia quando o componente de sessão de acesso no terminal é instanciado. O estado seguinte (t_2) é sensibilizado, e este solicita a finalização do serviço a sessão de acesso e serviço do provedor e este então verifica se o serviço está sendo utilizado por outros através da sessão de acesso de outros terminais, retornando ao solicitante a finalização do serviço.

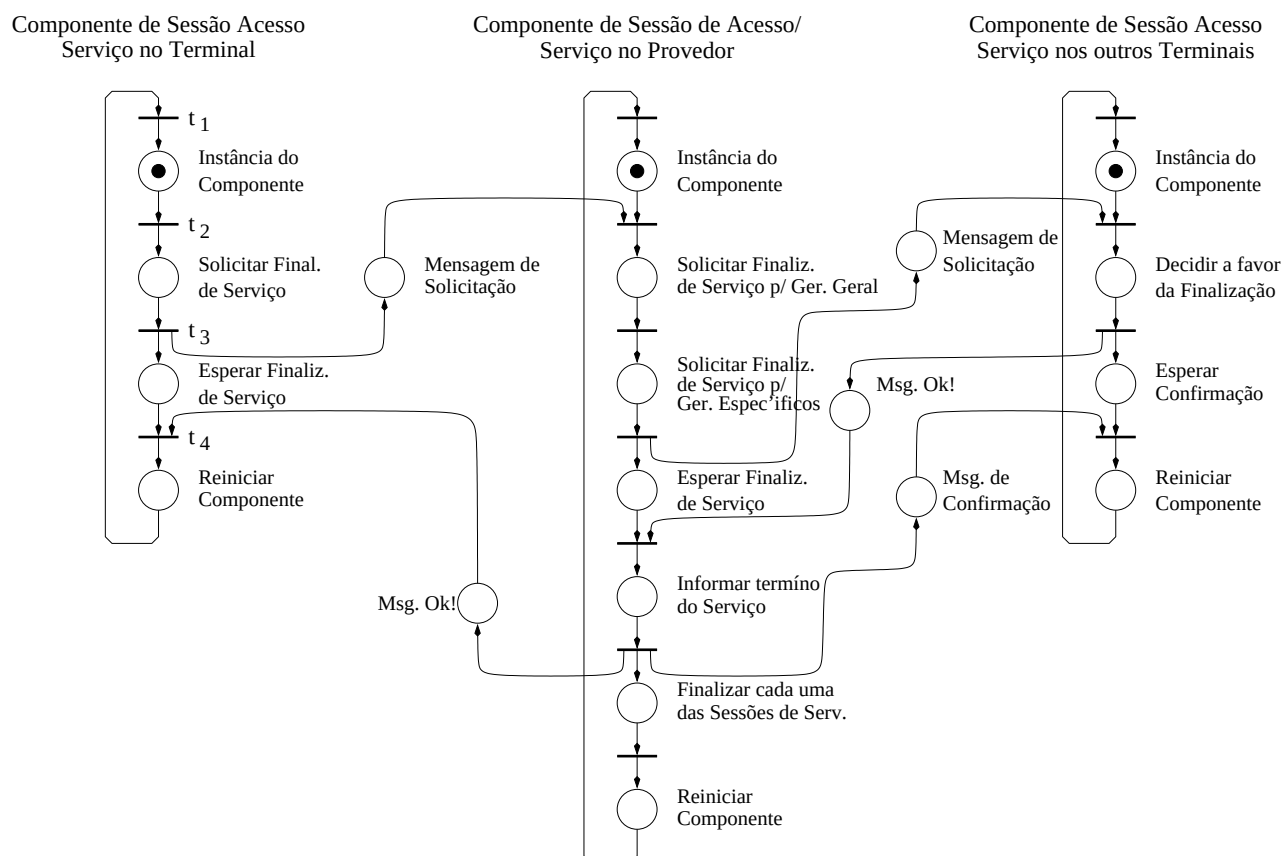


Fig. 4.16: Rede de Petri para Finalização de Serviço no Provedor Contratado.

4.3.9 Finalizar Serviço no Provedor Visitado

Diferentemente do cenário anterior, finalizar uma sessão de serviço no provedor visitado não está circunscrita somente ao seu domínio, envolve também o domínio do provedor contratado a partir do qual o serviço foi disponibilizado. Deste modo, constituem parte do processo não só os aspectos apresentados no cenário anterior (p. ex., encaminhamento da solicitação aos demais usuários que estejam participando), mas também o armazenamento das informações relacionadas ao uso e configuração do serviço na perfil do usuário no provedor contratado.

Para finalizar uma sessão de serviço estabelecida no provedor visitado, assumi-se que o usuário já possui uma sessão de acesso, assim como uma sessão de serviço. Os passos descritos a seguir mais o diagrama de eventos da Fig. 4.17, nos mostram como evolui este processo.

1. passos de 1 a 5 do cenário anterior (Seção 4.3.8);
2. cada gerente da sessão de serviço de cada usuário encaminha a solicitação a seus usuários; nesta fase informações que sintetizem a utilização do serviço (p. ex., serviço utilizado, tempo de uso, participantes envolvidos, etc.) podem ser disponibilizadas para o usuário;

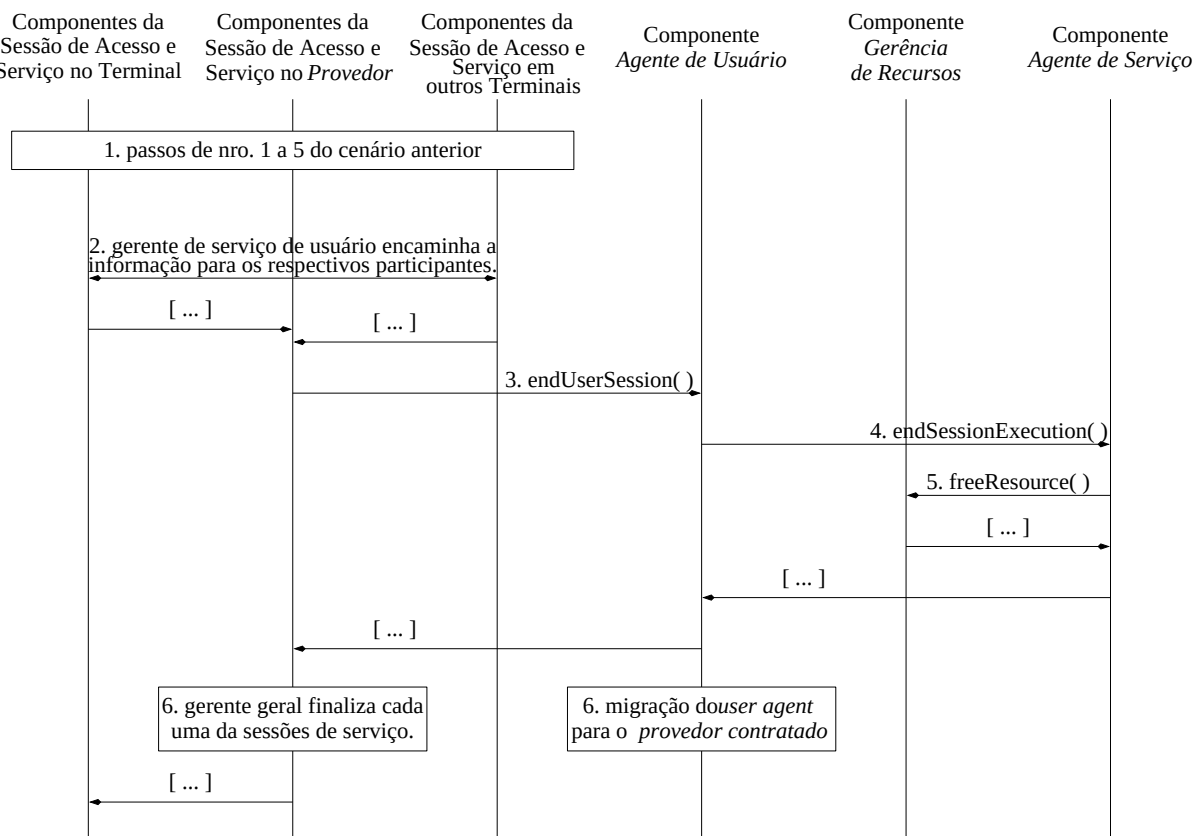


Fig. 4.17: Finalização de Serviço no Provedor Visitado.

3. após os usuários terem sido alertados que a sessão está em processo de finalização, o gerente da sessão de serviço do usuário visitante solicita ao agente de usuário o término do serviço associado a sessão de serviço;
4. agente de usuário finaliza o serviço invocando `endServiceExe()` no objeto `ServAgnt` ;
5. agente de serviço solicita liberação do(s) recurso(s) alocado(s) invocando a operação `freeResource()` no objeto `GercRcrs` ;
6. por fim, o gerente geral finaliza a sessão e eventualmente libera recursos alocados durante o estabelecimento da sessão de serviço;

agente de usuário migra para o provedor contratado a fim de armazenar no perfil do usuário informações de configuração relacionadas ao estabelecimento e uso do serviço. O agente de serviço pode ser finalizado, pois informações relativas ao uso do serviço já foram contabilizadas.

Capítulo 5

Aspectos da Implementação e Resultados

Neste trabalho propõe-se a concepção de agentes móveis inteligentes representando o usuário e atuando na configuração de serviços. Neste contexto, agentes são entidades de *software* autônomas, assíncronas, dotadas de habilidades e objetivos próprios (Fig. 3.2). Adicionalmente, nossos agentes contemplam dois predicados: mobilidade e inteligência.

Este capítulo discute aspectos de implementação da mobilidade do agente, aspectos de implementação da inteligência do agente, e finalmente apresenta cenários de teste que evidenciam a dinâmica das interações entre os componentes da arquitetura proposta. Desta forma, pôde-se validar a arquitetura proposta quanto ao uso de agentes móveis inteligentes no suporte à ubiquidade de serviços multimídia.

5.1 Infra-Estrutura da Arquitetura Proposta

A Fig. 5.1 apresenta os componentes da arquitetura proposta os quais coexistem com sistemas legados de subscrição, acesso e serviço do provedor. O componente Gerência de Recursos é um componente de serviço do qual se obtém informações da infra-estrutura de *hardware* e/ou de *software*, informação de Perfil é um componente de serviço do qual se obtém as preferências do usuário sobre acesso e uso de serviços e o sessão de acesso e serviço é um componente de serviço que permite a troca de informações entre componentes do agente de usuário e agente de serviço. Os componentes agente de usuário e agente de serviço são respectivamente componentes de serviços que representam o usuário na rede e que modela uma variedade de aplicações de tal forma a adaptar o serviço ao novo domínio.

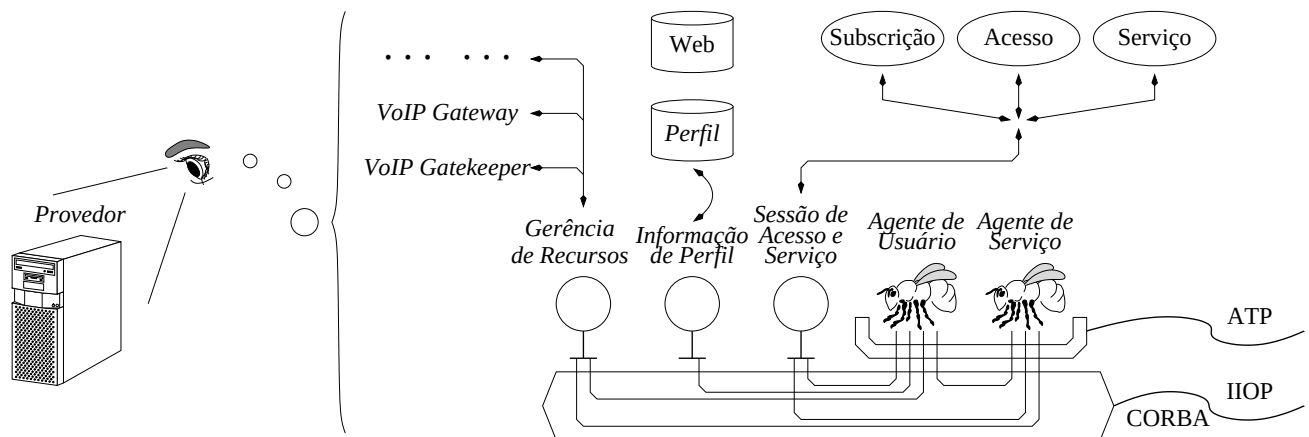


Fig. 5.1: Arquitetura do provedor para Suporte à Mobilidade.

5.1.1 Modelo de Tecnologias

Os componentes desta arquitetura estão em conformidade com a especificação *Common Object Request Broker* (CORBA) [29] do Consórcio *Object Management Group* (OMG). CORBA define uma linguagem neutra, *Interface Definition Language* (IDL), que permite descrever a especificação de um objeto CORBA através de suas interfaces. As interfaces descritas em IDL podem ser implementadas em linguagem para a qual o mapeamento IDL esteja disponível, como Java. Deste modo, aplicações CORBA escritas em diferentes linguagens podem interoperar.

Com a infra-estrutura CORBA em nosso modelo, utilizou-se o ORB disponível no ambiente de desenvolvimento Java™ 2 [30] (Java™ IDL). Suportando a especificação CORBA/IOP [29] e possuindo mapeamento IDL/Java [31], Java™ IDL implementa um ORB para linguagem Java. Para usar Java IDL faz-se necessário a instalação da Plataforma Java™ 2 JDK 1.4 ou superior que já inclui o compilador `idltj`, além das classes `org.omg.CORBA`, `org.omg.CORBA.ORB` e `org.omg.CosNaming`.

Para suportar os agentes, utilizou-se o conceito de Agência da especificação *Mobile Agent Facility* (MAF) [26]. A Agência é um plataforma que pode criar, executar, transferir e finalizar um agente. Na verdade, MAF é o resultado dos esforços de padronização da *ObjectSpace* [32] e da OMG para dentre outros aspectos garantir a interoperabilidade entre as diferentes plataformas, quando em conformidade com a especificação [33].

Para a infra-estrutura de agentes móveis, utilizou-se uma implementação da especificação MAF, a qual contempla objetos que exportam as interfaces `MAFFinder` e `MAFAgentSystem`. A interface `MAFFinder` é um serviço de nomes que define operações para registrar, remover o registro e localizar agentes, lugares e agências (Sistemas Agentes). Enquanto que a interface

MAFAgentSystem define métodos e objetos que suportam tarefas de gerenciamento de agentes tais como: procurar o nome de um sistema agente e receber um agente. Na implementação, agentes são objetos CORBA.

Além da mobilidade, nossos agentes necessitam de conhecimento (inteligência) de modo que possam avaliar a infra-estrutura do provedor e terminal a partir dos quais o usuário solicita um serviço. Para incorporar tal propriedade aos nossos agentes, utilizou-se a plataforma *Java Expert System Shell* (JESS) [19]. O JESS suporta o desenvolvimento de sistemas inteligentes baseados em regras de conhecimento do tipo *se-então*, possibilitando a junção com código Java. A versão utilizada foi JESS 6.0 que é compatível com a versão Java JDK 1.4.

5.1.2 Infra-Estrutura do PR-Prvds

Os componentes Gerência de Recursos, Informação de Perfil e Sessão de Acesso e Serviço (Fig. 5.1) são implementados como objetos servidores CORBA, que dispõem suas funcionalidades através de interfaces definidas em IDL. Juntamente com os componentes que suportam a subscrição, acesso e serviço no provedor. Por fim, os componentes agente de usuário e agente de serviço são agentes móveis dotados de um certo nível de inteligência permitindo-os inferir que serviços podem ser adaptados às novas capacidades do provedor e terminal.

Para os provedores que estejam em conformidade com o PR-Prvds, dois protocolos de aplicação são utilizados para interação entre os domínios: *Internet Inter-ORB Protocol* (IIOP) e *Agent Transfer Protocol* (ATP). ATP é utilizado nas interações entre as agências para transferência dos agentes e IIOP é utilizado para interconexão de ORBs através da Rede TCP/IP, permitindo que haja interoperabilidade entre diferentes implementações de ORB.

As informações trocadas entre os objetos da arquitetura proposta é feita através do serviço de propriedade do CORBA, o *Property Service*. Ele implementa objetos da interface `PropertySet` e da interface `PropertySetDef`, a primeira suporta um conjunto de propriedades, onde uma propriedade (`Property`) é uma associação de nome e valor relacionados a um objeto; e a segunda é uma subclasse de `PropertySet` que expõe as características de cada `Property` provendo operações para recuperar restrições, definições e modificação, a obtenção e configuração de tipos de propriedades.

5.2 Aspectos de Implementação do Agente

Quando o usuário necessita usar serviços oferecidos por seu provedor contratado e o mesmo encontra-se em outro provedor e/ou terminal, o agente de usuário deverá migrar para o provedor visitado para que possa verificar a adaptação dos serviços para o provedor e terminal em questão através de sua inteligência. Nas subseções seguintes apresenta-se os aspectos de implementação da mobilidade e da inteligência do agente.

5.2.1 Aspectos de Implementação da Mobilidade

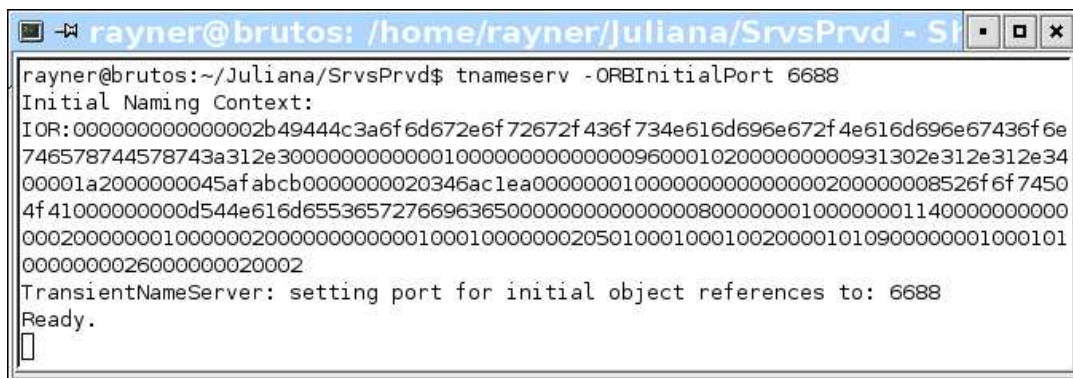
Agentes são considerados programas autônomos que agem em benefício de seus proprietários, usuários ou outros programas e que, de maneira a acompanhar as tarefas designadas por seus proprietários, agentes podem se comunicar com outros agentes, com o seu ambiente e com seus proprietários, segundo Oliveira e Cardozo [24]. Em nosso projeto o agente foi implementado utilizando a uma implementação do *Mobile Agent Facility* (MAF).

O MAF, possui duas interfaces bem definidas, uma o MAFFinder, a qual define métodos e objetos para manter dinamicamente uma base de dados dos nomes e localizações dos agentes, dos lugares e dos sistemas agentes, estes incluem o registro, desregistro e localização de agentes, lugares e sistemas agentes, e a MAFAgentSystem a qual define métodos e objetos que suportam o gerenciamento de agentes estes incluem a recepção, criação, suspensão e término de agentes.

Para instanciar o servidor MAFFinder, agências, agentes, provedores há a necessidade de registrá-los, utiliza-se o *Transient Naming Service*, `tnameserv` (Fig. 5.2) que é o serviço de nomes provido com IDL JAVA sendo uma implementação da especificação *COS Naming Service* que provê uma árvore de diretórios para referências de objetos associando o par nome e referência do objeto. O `tnameserv` tem como parâmetro, `ORBInitialPort`, o qual especifica a porta por onde ele irá escutar para registrar os serviços, neste exemplo usou-se a porta 6688.

A partir do momento em que o serviço de nomes for instanciado, pode-se instanciar o servidor MAFFinder utilizando a mesma porta do `tnameserv` para que o MAFFinder possa ser registrar no `tnameserv` através do parâmetro `ORBInitialPort`, como também o parâmetro `ORBInitialHost` para especificar a máquina em que se encontra o `tnameserv`, como mostra a Fig. 5.3. Com o servidor MAFFinder instanciado pode-se então criar as agências.

De acordo com a arquitetura proposta deve-se instanciar provedores de serviços, onde estes provedores deverão registrar os objetos da arquitetura `SessAssServ`, `InfoPrfl` e o `GercRcrs`. O `SessAssServ` implementa o método `givePossibleServices` que fornece a lista de servi-



```

rayner@brutos: /home/rayner/Juliana/SrvsPrvd - Shell
rayner@brutos:~/Juliana/SrvsPrvd$ tnameserv -ORBInitialPort 6688
Initial Naming Context:
IOR:000000000000002b49444c3a6f6d672e6f72672f436f734e616d696e672f4e616d696e67436f6e
746578744578743a312e30000000000001000000000000096000102000000000931302e312e312e34
00001a2000000045afabcb0000000020346ac1ea0000000100000000000000200000008526f6f7450
4f4100000000d544e616d65536572766963655000000000000080000000100000001140000000000
0002000000010000002000000000000100010000000205010001000100200001010900000001000101
00000000260000000020002
TransientNameServer: setting port for initial object references to: 6688
Ready.

```

Fig. 5.2: Instância do tnameserv.



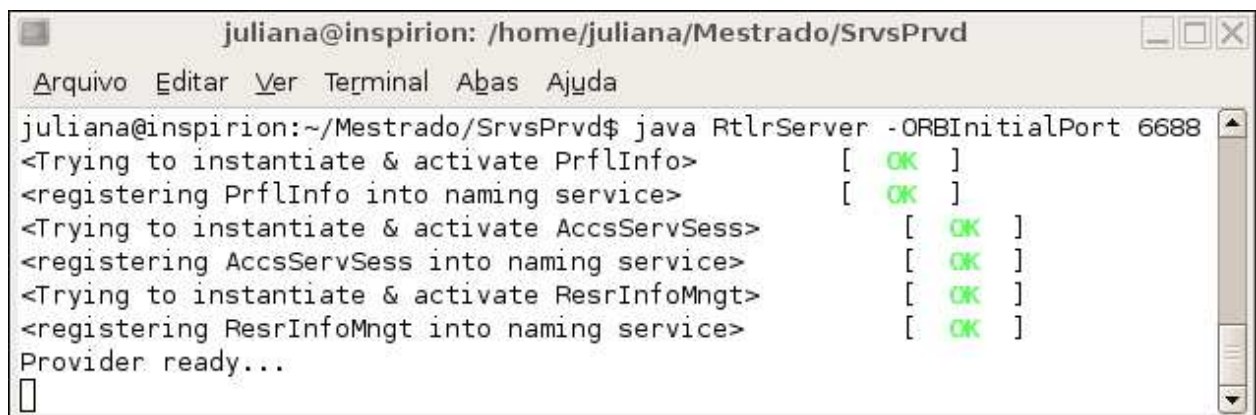
```

rayner@brutos: /home/rayner/Juliana/SrvsPrvd - Shell
rayner@brutos:~/Juliana/SrvsPrvd$ java MAFFinderServer -ORBInitialPort 6688 -ORBIni
tialHost brutos
MAFFinder server ready.
Agent system registered: CosNaming:/beta!Agency

```

Fig. 5.3: Instância do MAFFinder.

ços que podem ser adaptados ao terminal e ao provedor, o `InfoPrfl` implementa os métodos `getUserInfo` que fornece informações associadas ao usuário e o `getTrmnInfo` que fornece informações acerca das capacidades do terminal corrente e o `GercRcrs` implementa o método `getResourcesInformation` fornece informações específicas de componentes de *hardware* e *software*. A Fig. 5.4 mostra o instanciamento de um provedor de serviços, o qual possui os parâmetros `ORBInitialPort` e `ORBInitialHost` para especificar a porta e a máquina referente ao `tnameserv`.



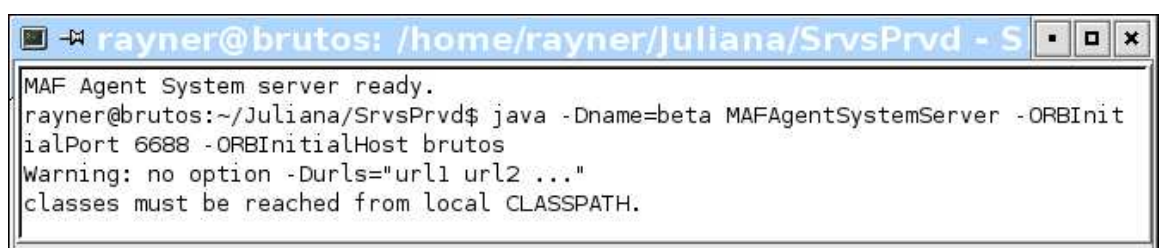
```

juliana@inspirion: /home/juliana/Mestrado/SrvsPrvd
Arquivo  Editar  Ver  Terminal  Abas  Ajuda
juliana@inspirion:~/Mestrado/SrvsPrvd$ java RtlrServer -ORBInitialPort 6688
<Trying to instantiate & activate PrflInfo>          [ OK ]
<registering PrflInfo into naming service>          [ OK ]
<Trying to instantiate & activate AccsServSess>      [ OK ]
<registering AccsServSess into naming service>      [ OK ]
<Trying to instantiate & activate ResrInfoMngt>      [ OK ]
<registering ResrInfoMngt into naming service>      [ OK ]
Provider ready...

```

Fig. 5.4: Instância do Provedor de Serviço PR-Prvds.

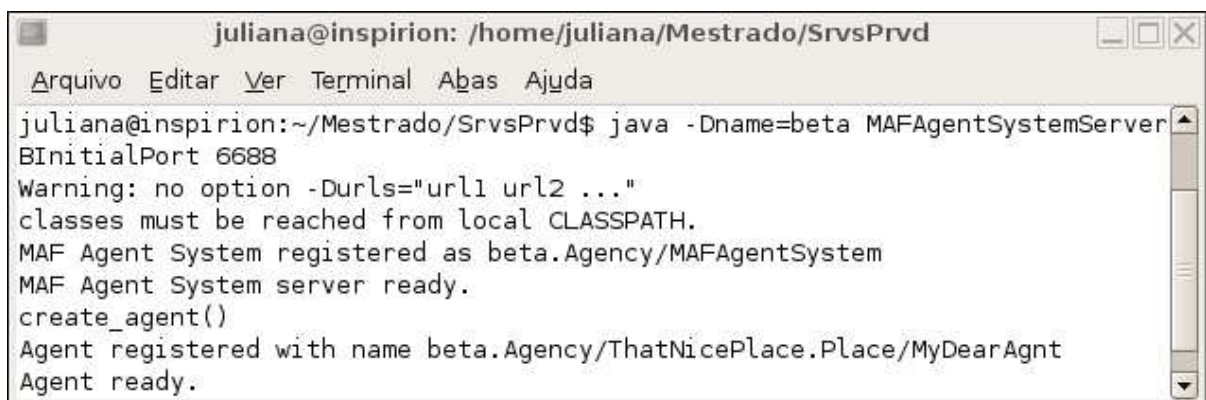
Com o servidor *MAFFinder* instanciado, pode-se então criar agências nos provedores de serviços. Todas as agências deverão ser registradas no servidor *MAFFinder* independente de qual provedor de serviço ela faz parte, se está ou não na mesma máquina, ou seja, só existe um servidor *MAFFinder* instanciado. A Fig. 5.5 mostra a criação e registro de uma agência tendo como parâmetro *ORBInitialPort* e *ORBInitialHost* para especificar porta e máquina referente ao *tnameserv*. Neste contexto utiliza-se um servidor de sistemas agentes para que possa criar agentes dentro de agências, os quais são implementados baseados na interface *MAFAgentSystem* do padrão MAF.



```
rayner@brutos: /home/rayner/Juliana/SrvsPrvd - S
MAF Agent System server ready.
rayner@brutos:~/Juliana/SrvsPrvd$ java -Dname=beta MAFAgentSystemServer -ORBInit
ialPort 6688 -ORBInitialHost brutos
Warning: no option -Durls="url1 url2 ..."
classes must be reached from local CLASSPATH.
```

Fig. 5.5: Criação e registro da agência beta.

Um agente só poderá ser criado dentro de uma agência, ou seja, só é possível criar um agente em uma agência já registrada, não se esquecendo que o mesmo será registrado no servidor *MAFFinder* via o *tnameserv*, pois é ele que sabe localizar o *MAFFinder*. Este agente pode migrar de um provedor de serviços para outro, como também descobrir os serviços que podem ser adaptados. A Fig. 5.6 mostra a criação de um agente, onde *UserAgntServer* cria um agente do tipo do objeto *UserAgnt*, o qual é um objeto da arquitetura proposta, e uma implementação MAF agente.



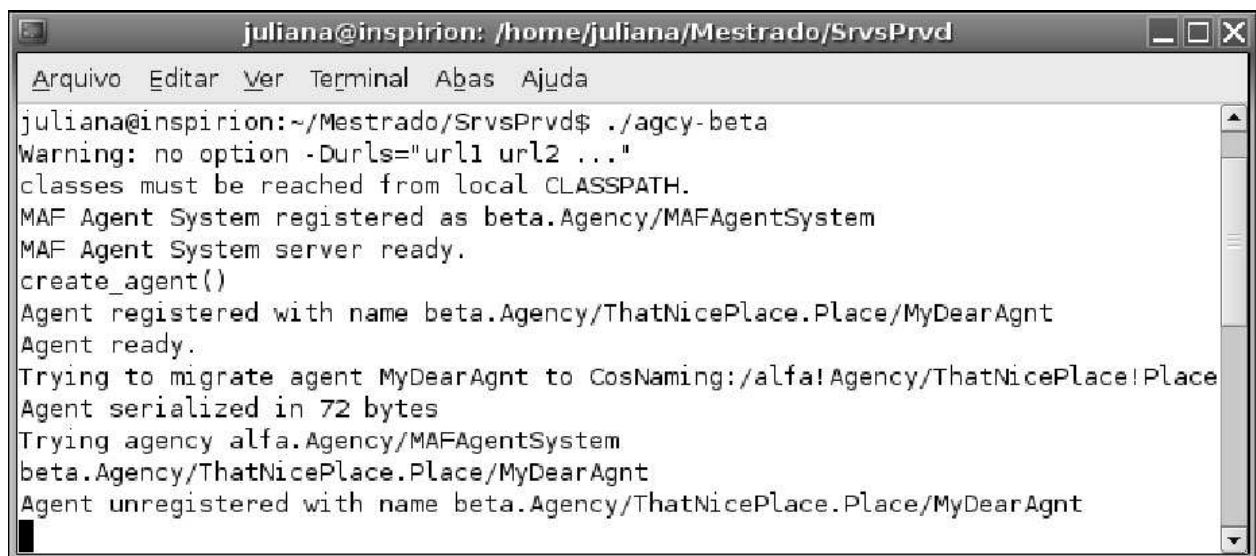
```
juliana@inspirion: /home/juliana/Mestrado/SrvsPrvd
Arquivo Editar Ver Terminal Abas Ajuda
juliana@inspirion:~/Mestrado/SrvsPrvd$ java -Dname=beta MAFAgentSystemServer
BInitialPort 6688
Warning: no option -Durls="url1 url2 ..."
classes must be reached from local CLASSPATH.
MAF Agent System registered as beta.Agency/MAFAgentSystem
MAF Agent System server ready.
create_agent()
Agent registered with name beta.Agency/ThatNicePlace.Place/MyDearAgnt
Agent ready.
```

Fig. 5.6: Criação e registro do agente *MyDearAgnt* na agência Beta.

E por fim para que este agente possa se mover, ou seja, migrar de um provedor de serviços

para outro, o agente foi implementado em JAVA, pois possui o recurso serialização de objetos, a qual coloca o objeto na forma de uma sequência de *bytes* possíveis de serem recuperados quando transmitidos pela rede, que não é tratada no padrão MAF.

Assim quando um agente precisar se mover ele será serializado na sua origem, e quando chegar ao seu destino ele será desserializado. A Fig. 5.7 mostra uma agência desregistrando uma agente, que significa que um agente está se movendo de um provedor de serviços para outro, neste caso ele se moverá de uma agência do provedor de origem para uma agência do provedor de destino. Quando o agente se move, a agência onde ele se encontrava desregistra o agente, e a agência que recebe o registra.



```
juliana@inspirion: /home/juliana/Mestrado/SrvsPrvd
Arquivo  Editar  Ver  Terminal  Abas  Ajuda
juliana@inspirion:~/Mestrado/SrvsPrvd$ ./agcy-beta
Warning: no option -Durls="url1 url2 ..."
classes must be reached from local CLASSPATH.
MAF Agent System registered as beta.Agency/MAFAgentSystem
MAF Agent System server ready.
create_agent()
Agent registered with name beta.Agency/ThatNicePlace.Place/MyDearAgnt
Agent ready.
Trying to migrate agent MyDearAgnt to CosNaming:/alfa!Agency/ThatNicePlace!Place
Agent serialized in 72 bytes
Trying agency alfa.Agency/MAFAgentSystem
beta.Agency/ThatNicePlace.Place/MyDearAgnt
Agent unregistered with name beta.Agency/ThatNicePlace.Place/MyDearAgnt
```

Fig. 5.7: O agente MyDearAgnt migra da agência Beta para Alfa.

5.2.2 Aspectos de Implementação da Inteligência

Para que o agente do usuário tenha condições de analisar se os serviços podem ou não ser adaptados no provedor visitado e/ou terminal, ele necessita de informações a respeito do terminal, do provedor e do perfil do usuário, tais como:

- nome e conjunto de propriedades do serviço;
- conjunto de propriedades do terminal;
- conjunto de propriedades do provedor;
- conjunto de propriedades do perfil do usuário;

- características do conjunto de propriedades, ou seja, cada um dos conjuntos de propriedade anteriores terão suas características.

Estas informações podem ser estruturadas em árvore, conforme Fig 5.8, onde: serviços, terminais, provedor, perfil do usuário, conjunto de propriedades, e características são nós da árvore; serviços, terminais, provedor e usuário são raízes de suas próprias árvores; o conjunto de propriedades são filhos de serviço, terminal, provedor e usuário; e as características são filhas dos conjuntos de propriedades.

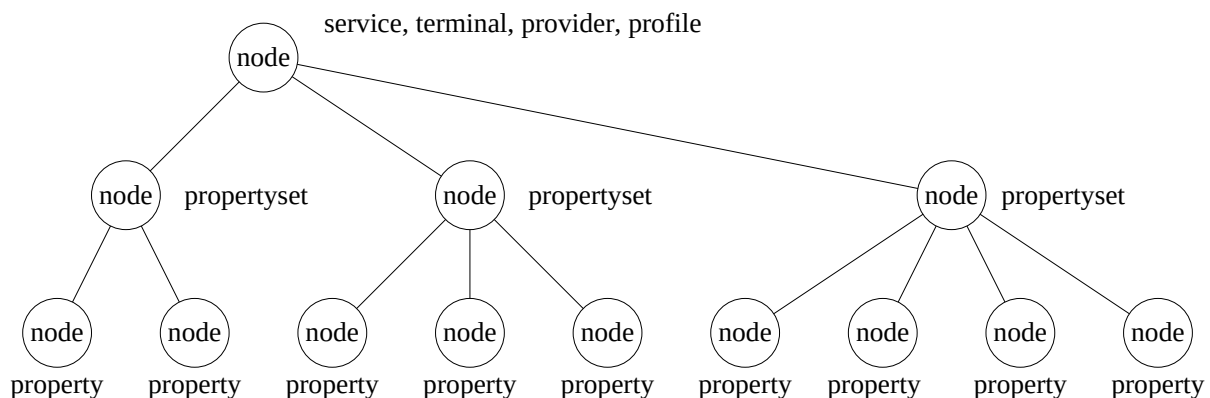


Fig. 5.8: Árvore que Representa a Estrutura do Conhecimento.

Cada nó da árvore possui os campos: **type**, **parent**, **name**, **properties**, o primeiro campo contém o tipo do nó, o qual pode ser: *service*, *terminal*, *provider*, *profile*, *propertyset* ou *property*; o segundo campo, o **parent**, é um campo número que representa o número do nó que corresponde ao nó pai deste nó. Sendo 0 (zero) para o nó raiz da árvore, podendo o nó raiz ser os **types** *service*, *terminal*, *profile* e *provider*; o terceiro campo do nó o **name** contém o nome do correspondente **type**; e por fim o campo **properties** contém a lista de propriedades do correspondente **type**. O nó será um template em JESS, conforme o código a seguir:

```

(deftemplate node (slot type)
                  (slot parent)
                  (slot name)
                  (multislot properties))

```

onde:

- **slot** significa um tipo com apenas uma informação e;
- **multislot** um tipo que contém uma lista de informações.

Como este trabalho se propõe usar serviços de multimídia, se faz necessário definir quais as propriedades de áudio e vídeo que serão analisadas bem como suas características. Sendo o conjunto de propriedades e características para os serviços que acomodam:

- **Áudio:** *encoding*: é uma *string* que descreve o tipo de formato; *sampleRate*: é a taxa de amostragem; *sampleSizeInBits*: é o tamanho da amostra em *bits*; *channel*: é o número de canais; *endian*: é a ordenação do *byte* da amostra; *signed*: indica se a amostra está ou não no formato sinalizado; *dataType*: é o tipo do dado, de acordo com a Fig. 5.9.

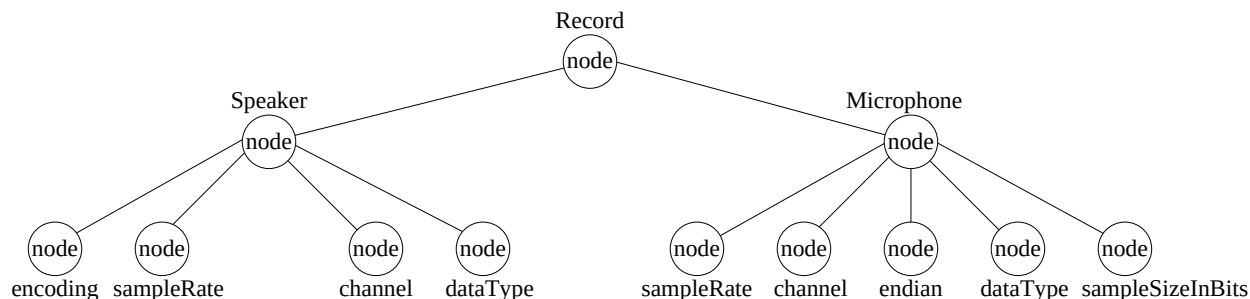


Fig. 5.9: Árvore que Representa um Serviço de Áudio.

- **Vídeo:** *encoding*: é uma *string* que descreve o tipo de formato; *frameRate*: é a taxa de quadros; *frameSizeInBits*: é o tamanho do quadro em *bits*; *endian*: é a ordenação do *byte* da amostra; *dataType*: é o tipo de dado; *mode*: é o modo do vídeo, gráfico ou texto; *size*: é a dimensão do vídeo; *color*: é a quantidade de cores, de acordo com a Fig. 5.10.

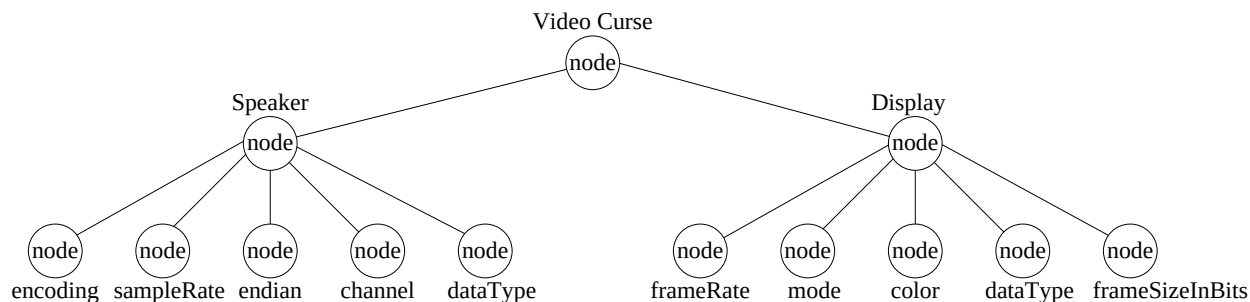


Fig. 5.10: Árvore que Representa um Serviço de Vídeo.

Para exemplificar, suponha que um usuário, Arthur, deseja usar serviços contratados de seu provedor contratado, o qual possui apenas um serviço de vídeo-conferência, em um provedor visitado e em um celular. As informações, ou seja, as propriedades e características relacionadas ao serviço, ao provedor, ao terminal e ao perfil do usuário são apresentadas a seguir na forma de estrutura de árvore apresentada anteriormente.

- Serviço: Vídeo-Conferência

```

1-(node (parent 0) (type service) (name Vídeo-Conferência)
  (properties speaker, microphone, display))

2-(node (parent 1) (type propertyset) (name speaker)
  (properties encoding, sampleRate, endian, dataType))
3-(node (parent 1) (type propertyset) (name microphone)
  (properties encoding channel, sampleSizeInBits, dataType))
4-(node (parent 1) (type propertyset) (name display)
  (properties encoding, mode, color, frameRate))

5-(node (parent 2) (type property) (name encoding) (properties araw))
6-(node (parent 2) (type property) (name sampleRate) (properties 20))
7-(node (parent 2) (type property) (name endian) (properties little))
8-(node (parent 2) (type property) (name dataType) (properties bytearray))

9-(node (parent 3) (type property) (name encoding) (properties araw))
10-(node (parent 3) (type property) (name channel) (properties 5))
11-(node (parent 3) (type property) (name sampleSizeInBits) (properties 14))
12-(node (parent 3) (type property) (name dataType) (properties bytearray))

13-(node (parent 4) (type property) (name encoding) (properties araw))
14-(node (parent 4) (type property) (name mode) (properties grafico))
15-(node (parent 4) (type property) (name color) (properties 65000))
16-(node (parent 4) (type property) (name frameRate) (properties 16))

```

- Provedor: Autumn

```

17-(node (parent 0) (type provider) (name Autumn)
  (properties speaker, microphone, display))

18-(node (parent 17) (type propertyset) (name speaker)
  (properties encoding, sampleRate, endian, dataType))
19-(node (parent 17) (type propertyset) (name microphone)
  (properties encoding channel, sampleSizeInBits, dataType))
20-(node (parent 17) (type propertyset) (name display)
  (properties encoding, mode, color, frameRate))

21-(node (parent 18) (type property) (name encoding) (properties))
22-(node (parent 18) (type property) (name sampleRate) (properties))
23-(node (parent 18) (type property) (name endian) (properties))
24-(node (parent 18) (type property) (name dataType) (properties))

25-(node (parent 19) (type property) (name encoding) (properties))
26-(node (parent 19) (type property) (name channel) (properties))
27-(node (parent 19) (type property) (name sampleSizeInBits) (properties))
28-(node (parent 19) (type property) (name dataType) (properties))

29-(node (parent 20) (type property) (name encoding) (properties))
30-(node (parent 20) (type property) (name mode) (properties))
31-(node (parent 20) (type property) (name color) (properties))
32-(node (parent 20) (type property) (name frameRate) (properties))

```

- Terminal: NET55

```

33-(node (parent 0) (type trminal) (name NET55)
    (properties speaker, microphone, display))

34-(node (parent 33) (type propertyset) (name speaker)
    (properties encoding, sampleRate, endian, dataType))
35-(node (parent 33) (type propertyset) (name microphone)
    (properties encoding channel, sampleSizeInBits, dataType))
36-(node (parent 33) (type propertyset) (name display)
    (properties encoding, mode, color, frameRate))

37-(node (parent 34) (type property) (name encoding) (properties araw))
38-(node (parent 34) (type property) (name sampleRate) (properties 10))
39-(node (parent 34) (type property) (name endian) (properties big))
40-(node (parent 34) (type property) (name dataType) (properties bytearray))

41-(node (parent 35) (type property) (name encoding) (properties araw))
42-(node (parent 35) (type property) (name channel) (properties 4))
43-(node (parent 35) (type property) (name sampleSizeInBits) (properties 18))
44-(node (parent 35) (type property) (name dataType) (properties bytearray))

45-(node (parent 36) (type property) (name encoding) (properties araw))
46-(node (parent 36) (type property) (name mode) (properties texto))
47-(node (parent 36) (type property) (name color) (properties 256))
48-(node (parent 36) (type property) (name frameRate) (properties 20))

```

- Perfil do Usuário: Arthur

```

49-(node (parent 0) (type profile) (name Arthur)
    (properties speaker, microphone, display))

50-(node (parent 49) (type propertyset) (name speaker)
    (properties encoding, sampleRate, endian, dataType))
51-(node (parent 49) (type propertyset) (name microphone)
    (properties encoding channel, sampleSizeInBits, dataType))
52-(node (parent 49) (type propertyset) (name display)
    (properties encoding, mode, color, frameRate))

53-(node (parent 50) (type property) (name encoding) (properties araw))
54-(node (parent 50) (type property) (name sampleRate) (properties 16))
55-(node (parent 50) (type property) (name endian) (properties little))
56-(node (parent 50) (type property) (name dataType) (properties bytearray))

57-(node (parent 51) (type property) (name encoding) (properties araw))
58-(node (parent 51) (type property) (name channel) (properties 6))
59-(node (parent 51) (type property) (name sampleSizeInBits) (properties 24))
60-(node (parent 51) (type property) (name dataType) (properties bytearray))

61-(node (parent 52) (type property) (name encoding) (properties araw))
62-(node (parent 52) (type property) (name mode) (properties grafico))
63-(node (parent 53) (type property) (name color) (properties 65000))
64-(node (parent 54) (type property) (name frameRate) (properties 24))

```

A árvore do terminal, do provedor, do perfil do usuário e de cada serviço são percorridas buscando as propriedades em comuns. Se uma propriedade de um serviço for comum a uma

propriedade do terminal ou do provedor, as características da propriedade são então verificadas. Se as características puderem ser adaptadas, então a propriedade será colocada na lista de propriedades comuns, caso contrário na lista de propriedades diferentes. Se a propriedade do serviço não for comum nem ao terminal nem ao provedor, então esta é colocada na lista de propriedades diferentes. Para cada serviço verificado teremos como resposta:

- adaptado completamente: a lista de propriedades diferentes deverá estar vazia, ou seja, o serviço pode ser oferecido com todos os recursos que ele necessita;
- adaptado de acordo com as propriedades: a lista de propriedades comuns e diferentes não estão vazias, ou seja, o serviço pode ser oferecido mas não com todos os recursos, mas sim adaptado aos recursos existentes;
- não pode ser adaptado: a lista de propriedades comuns está vazia, ou seja, o serviço não pode ser oferecido pois não existe um mínimo de recursos possíveis para a sua execução.

Para que se possa ter uma das respostas anteriores, será necessário que o agente possua uma inteligência capaz de verificar se um determinado serviço pode ou não ser adaptado as novas condições de terminal e/ou provedor de acordo com o perfil do usuário. A inteligência do agente foi desenvolvido em JESS, pois o mesmo suporta o desenvolvimento de sistemas inteligentes baseados em regras. As regras estão organizadas por grupos onde cada grupo tem por objetivo analisar uma determinada situação.

O **Grupo 1** é composto pelas regras que verificam quais propriedades de um serviço pertencem ao terminal e ao perfil do usuário. O código a seguir mostra uma das regras deste grupo que tem como objetivo verificar se a primeira propriedade do serviço pertence ao terminal e ao perfil do usuário, obtendo as informações a respeito do serviço em questão, do terminal e do perfil do usuário, lendo os nós de cada um, caso pertença, será criado um nó para esta propriedade para que se possa verificar as características da mesma. A partir deste ponto as outras propriedades deste serviço serão verificadas por outras regras.

```
(defrule properties-equals-term

(not (root-s ?root))
?cns <- (current-node-service ?ps)

?serv <- (node (type service)
              (parent ?ps)
              (name ?ns)
              (properties ?r $?resto_rs))
```

```

?lsv <- (list-service-finished $?lista&:(not (member$ ?ns $?lista)))

?term <- (node (type terminal)
              (parent 0)
              (name ?nt)
              (properties $?inicio ?r $?fim))

?prfl <- (node (type profile)
              (parent 0)
              (name ?nf)
              (properties $?inicio-1 ?r $?fim-1))

=>
(assert (current-node-service ?serv))
(assert (root-s ?serv))
(assert (root-tp ?term))
(assert (root-prfl ?prfl))
(assert (recurso-current ?r))
(bind ?fato-recr (assert (node (type propertyset)
                              (parent nil)
                              (name ?r)
                              (properties ))))

(assert (recurso-atual ?fato-recr)))

```

O **Grupo 2** é composto pelas regras que verificam quais propriedades de um serviço pertencem ao provedor e ao perfil do usuário. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se a primeira propriedade do serviço pertence ao provedor e ao perfil do usuário, obtendo as informações a respeito do serviço em questão, do provedor e do perfil do usuário, lendo os nós de cada um, caso pertença, será criado um nó para esta propriedade para que se possa verificar as características da mesma. A partir deste ponto as outras propriedades deste serviço serão verificadas por outras regras.

```

(defrule properties-equals-prvd
(not (root-s ?root))
?cns <- (current-node-service ?ps)

?serv <- (node (type service)
              (parent ?ps)
              (name ?ns)
              (properties ?r $?resto_rs))

?lsv <- (list-service-finished $?lista&:(not (member$ ?ns $?lista)))
?prvd <- (node (type provider)
              (parent 0)
              (name ?np)
              (properties $?inicio ?r $?fim))

?prfl <- (node (type profile)
              (parent 0)
              (name ?nf)
              (properties $?inicio-1 ?r $?fim-1))

```

```
=>
(assert (current-node-service ?serv))
(assert (root-s ?serv))
(assert (root-tp ?prvd))
(assert (root-prfl ?prfl))
(assert (recurso-current ?r))
(bind ?fato-recr (assert (node (type propertyset)
                               (parent nil)
                               (name ?r)
                               (properties ))))
(assert (recurso-atual ?fato-recr)))
```

O **Grupo 3** é formado pelas regras que verificam quais propriedades de um serviço não pertencem nem ao provedor e nem ao terminal. O código a seguir mostra uma das regras deste grupo que tem como objetivo verificar se a primeira propriedade do serviço não pertence nem ao provedor e nem ao terminal, obtendo as informações a respeito do serviço em questão, do provedor e do terminal, lendo os nós de cada um, caso seja verdade acrescenta esta propriedade na lista de propriedades diferentes.

```
(defrule properties-differences-term-prvd
(not (root-s ?root))
?cns <- (current-node-service ?ps)
?serv <- (node (type service)
              (parent ?ps)
              (name ?ns)
              (properties ?r $?resto_rs))

?lsv <- (list-service-finished $?lista&:(not (member$ ?ns $?lista)))
?term <- (node (type terminal)
              (parent 0)
              (name ?nt)
              (properties $?list&:(not (member$ ?r $?list))))

?prvd <- (node (type provider)
              (parent 0)
              (name ?np)
              (properties $?list-p&:(not(member$ ?r $?list-p))))

?rd <- (node (type difference)
           (parent ?serv)
           (name ?ns)
           (properties $?list_rd))

=>
(retract ?cns)
(retract ?rd)
(duplicate ?serv (parent ?serv)(properties $?resto_rs))
(assert (node (type difference)
              (parent ?serv)
              (name ?ns)
              (properties ?r $?list_rd)))

(assert (current-node-service ?serv))
(assert (root-s ?serv))
(assert (root-tp ?term))
```

```
(assert (property-new)))
```

O **Grupo 4** é formado pelas regras que verificam quais propriedades de um serviço não pertencem nem ao perfil do usuário. O código a seguir apresenta uma das regras deste grupo que tem por objetivo verificar se a primeira propriedade do serviço não pertence ao perfil do usuário, obtendo as informações a respeito do serviço em questão e do perfil do usuário, lendo os nós de cada um, caso seja verdade acrescenta esta propriedade na lista de propriedades diferentes.

```
(defrule properties-differences-prfl
(not (root-s ?root))
?cns <- (current-node-service ?ps)

?serv <- (node (type service)
              (parent ?ps)
              (name ?ns)
              (properties ?r $?resto_rs))

?lsv <- (list-service-finished $?lista&:(not (member$ ?ns $?lista)))
?prfl <- (node (type profile)
              (parent 0)
              (name ?nf)
              (properties $?list&:(not (member$ ?r $?list))))

?rd <- (node (type difference)
           (parent ?serv)
           (name ?ns)
           (properties $?list_rd))

=>
(retract ?cns)
(retract ?rd)
(duplicate ?serv (parent ?serv)(properties $?resto_rs))
(assert (node (type difference)
              (parent ?serv)
              (name ?ns)
              (properties ?r $?list_rd)))

(assert (current-node-service ?serv))
(assert (root-s ?serv))
(assert (root-prfl ?prfl))
(assert (property-new)))
```

O **Grupo 5** é composto por regras que verificam quais características das propriedades de um serviço pertencem ou não as características das propriedades do terminal/provedor e perfil do usuário. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as características de uma propriedade comum do serviço são satisfeitas pelas características da propriedade do terminal e do perfil do usuário, obtendo as informações a respeito das propriedades do serviço em questão das propriedades do terminal e das propriedades do perfil do usuário.

```

(defrule caract-equals-properties
  (root-s ?root)
  (root-tp ?term-prvd)
  (root-prfl ?prfl)
  (recurso-current ?r)

  ?recr-s <- (node (type propertyset)
                   (parent ?root)
                   (name ?r)
                   (properties ?cr $?resto_crs))

  ?recr-tp <- (node (type propertyset)
                   (parent ?term-prvd)
                   (name ?r)
                   (properties $?inicio ?cr $?fim))

  ?recr-f <- (node (type propertyset)
                  (parent ?prfl)
                  (name ?r)
                  (properties $?inicio-1 ?cr $?fim-1))

  =>
  (duplicate ?recr-s (properties $?resto_crs))
  (assert (root-recr-s ?recr-s))
  (assert (root-recr-tp ?recr-tp))
  (assert (root-recr-pf ?recr-f))
  (assert (property-current ?cr)))

```

O **Grupo 6** é composto pelas regras que verificam se os valores da característica **encoding** podem ser adaptados. O código a seguir apresenta uma das regras deste grupo que tem por objetivo verificar se as características do **encoding** podem ser satisfeitas pelo terminal e pelo perfil do usuário.

```

(defrule properties-caract-encoding-igual
  (root-s ?root)
  (root-recr-tp ?root-recr-t)
  (root-recr-s ?root-recr-s)
  (root-recr-prfl ?root-recr-prfl)
  ?ccr <- (property-current ?c)
  (recurso-atual ?recr)

  ?recr-s <- (node (type property)
                   (parent ?root-recr-s)
                   (name ?c&:(eq ?c encoding))
                   (properties $?inicio ?r $?fim))

  ?recr-t <- (node (type property)
                   (parent ?root-recr-t)
                   (name ?c)
                   (properties $?list&:(member$ ?r $?list)))

  ?recr-f <- (node (type property)
                   (parent ?root-recr-prfl)
                   (name ?c)
                   (properties $?list1&:(member$ ?r $?list1)))

```

```
?recr-r <- (node (type propertyset)
                  (parent nil)
                  (name ?nr)
                  (properties $?lista))
=>
(modify ?recr (properties ?c $?lista))
(assert (node (type property)
              (parent ?recr)
              (name ?c)
              (properties ?r)))
(retract ?ccr))
```

O **Grupo 7** é composto pelas regras que verificam se os valores da característica `sampleRate` podem ser adaptados. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as características do `sampleRate` podem ser satisfeitas pelo terminal, ou seja, se o `sampleRate` da propriedade do serviço e do perfil do usuário for maior do que a do terminal, então a adaptação ocorre com o valor do `sampleRate` do terminal.

```
(defrule properties-caract-sampleRate-menor-term
  (root-s ?root)
  (root-recr-tp ?root-recr-t)
  (root-recr-s ?root-recr-s)
  ?ccr <- (property-current ?c)
  ?rcr <- (recurso-current ?nr)
  (recurso-atual ?recr)

  ?recr-s <- (node (type property)
                    (parent ?root-recr-s)
                    (name ?c&:(eq ?c sampleRate))
                    (properties ?r))

  ?recr-t <- (node (type property)
                    (parent ?root-recr-t)
                    (name ?c)
                    (properties ?r1&:(> ?r ?r1)))

  ?recr-f <- (node (type property)
                    (parent ?root-recr-prfl)
                    (name ?c)
                    (properties ?r2&:(<= ?r1 ?r2)))

  ?recr-r <- (node (type recurso)
                    (parent nil)
                    (name ?nr)
                    (properties $?lista))
=>
(modify ?recr (properties ?c $?lista))
(assert (node (type property)
              (parent ?recr)
              (name ?c)
              (properties ?r1)))
(retract ?ccr))
```

O **Grupo 8** é composto por regras que verificam se os valores da característica `frameRate` podem ser adaptados. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as características do `frameRate`, ou seja, se a taxa de quadros da propriedade do serviço é menor ou igual a do terminal e a do perfil do usuário, caso seja, a adaptação ocorre de acordo com o valor do `frameRate` do serviço.

```
(defrule properties-caract-frameRate-menor-serv
  (root-s ?root)
  (root-recr-tp ?root-recr-t)
  (root-recr-prfl ?root-recr-prfl)
  (root-recr-s ?root-recr-s)
  ?ccr <- (property-current ?c)
  (recurso-atual ?recr)

  ?recr-s <- (node (type property)
                  (parent ?root-recr-s)
                  (name ?c&:(eq ?c frameRate))
                  (properties ?r))

  ?recr-t <- (node (type property)
                  (parent ?root-recr-t)
                  (name ?c)
                  (properties ?r1&:(<= ?r ?r1)))

  ?recr-f <- (node (type property)
                  (parent ?root-recr-prfl)
                  (name ?c)
                  (properties ?r2&:(<= ?r ?r2)))

  ?recr-r <- (node (type recurso)
                  (parent nil)
                  (name ?nr)
                  (properties $?lista))

=>
  (modify ?recr (properties ?c $?lista))
  (assert (node (type property)
                (parent ?recr)
                (name ?c)
                (properties ?r)))
  (retract ?ccr))
```

O **Grupo 9** é composto pelas regras que verificam se os valores de `sampleSizeInBits` podem ser adaptados. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as características do `sampleSizeInBits`, ou seja, se o tamanho da amostragem em bits da propriedade do serviço e do terminal forem maior do que a do perfil do usuário, a adaptação ocorre de acordo com o valor do `sampleSizeInBits` do perfil do usuário.

```
(defrule properties-caract-sampleSizeInBits-menor-prfl
  (root-s ?root)
  (root-recr-tp ?root-recr-t)
  (root-recr-prfl ?root-recr-prfl)
```

```

(root-recr-s ?root-recr-s)
?ccr <- (property-current ?c)
(recurso-atual ?recr)

?recr-s <- (node (type property)
                (parent ?root-recr-s)
                (name ?c&:(eq ?c sampleSizeInBits))
                (properties ?r))

?recr-t <- (node (type property)
                (parent ?root-recr-t)
                (name ?c)
                (properties ?r1&:(> ?r ?r1)))

?recr-f <- (node (type property)
                (parent ?root-recr-prfl)
                (name ?c)
                (properties ?r2&:(> ?r1 ?r2)))

?recr-r <- (node (type propertyset)
                (parent nil)
                (name ?nr)
                (properties $?lista))
=>
(modify ?recr (properties ?c $?lista))
(assert (node (type property)
              (parent ?recr)
              (name ?c)
              (properties ?r2)))
(retract ?ccr))

```

O **Grupo 10** é composto por regras que verificam se os valores de `frameSizeInBits` podem ser adaptados. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as características do `frameSizeInBits`, ou seja, se o tamanho dos quadros em bits da propriedade do serviço e do perfil do usuário forem maior do que a do terminal, a adaptação ocorre de acordo com o valor do `frameSizeInBits` do terminal.

```

(defrule properties-caract-frameSizeInBits-menor-term
(root-s ?root)
(root-recr-tp ?root-recr-t)
(root-recr-prfl ?root-recr-prfl)
(root-recr-s ?root-recr-s)
?ccr <- (property-current ?c)
(recurso-atual ?recr)

?recr-s <- (node (type property)
                (parent ?root-recr-s)
                (name ?c&:(eq ?c frameSizeInBits))
                (properties ?r))

?recr-t <- (node (type property)
                (parent ?root-recr-t)
                (name ?c)
                (properties ?r1&:(> ?r ?r1)))

```

```

?recr-f <- (node (type property)
                (parent ?root-recr-prfl)
                (name ?c)
                (properties ?r2&:(<= ?r1 ?r2)))

?recr-r <- (node (type propertyset)
                (parent nil)
                (name ?nr)
                (properties $?lista))

=>
(modify ?recr (properties ?c $?lista))
(assert (node (type property)
              (parent ?recr)
              (name ?c)
              (properties ?r1)))
(retract ?ccr)

```

O **Grupo 11** é composto pelas regras que verificam se os valores da característica **channels** podem ser adaptados. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as características do **channels**, ou seja, se o número de canais da propriedade do serviço é menor ou igual a do perfil do usuário e a do terminal, a adaptação ocorre de acordo com o valor do **channels** do serviço.

```

(defrule properties-caract-channels-menor-serv
  (root-s ?root)
  (root-recr-tp ?root-recr-t)
  (root-recr-prfl ?root-recr-prfl)
  (root-recr-s ?root-recr-s)
  ?ccr <- (property-current ?c)
  (recurso-atual ?recr)

  ?recr-s <- (node (type property)
                  (parent ?root-recr-s)
                  (name ?c&:(eq ?c channel))
                  (properties ?r))

  ?recr-t <- (node (type property)
                  (parent ?root-recr-t)
                  (name ?c)
                  (properties ?r1&:(<= ?r ?r1)))

  ?recr-f <- (node (type property)
                  (parent ?root-recr-prfl)
                  (name ?c)
                  (properties ?r2&:(<= ?r ?r2)))

  ?recr-r <- (node (type propertyset)
                  (parent nil)
                  (name ?nr)
                  (properties $?lista))

  =>
  (modify ?recr (properties ?c $?lista))
  (assert (node (type property)
                (parent ?recr)
                (name ?c)

```

```

                (properties ?r)))
(retract ?ccr))

```

O **Grupo 12** é composto por regras que verificam se os valores da característica **endian** podem ser adaptados. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as características do **endian** são diferentes entre a propriedade do serviço e a do terminal, então não ocorre adaptação para esta propriedade, e ela é então colocada na lista de propriedades diferentes para este serviço.

```

(defrule properties-caract-endian-difference-term
(root-s ?root)
(root-recr-tp ?root-recr-t)
(root-recr-s ?root-recr-s)
?ccr <- (property-current ?c)
?rcr <- (recurso-current ?nr)
?ra <- (recurso-atual ?recr)

?recr-s <- (node (type property)
                (parent ?root-recr-s)
                (name ?c&:(eq ?c endian))
                (properties ?r))

?recr-t <- (node (type property)
                (parent ?root-recr-t)
                (name ?c)
                (properties ?r1&:(not (eq ?r ?r1))))

?rd <- (node (type difference)
            (parent ?root)
            (name ?ns)
            (properties $?list_rd))

?recr-r <- (node (type propertyset)
              (parent nil)
              (name ?nr)
              (properties $?lista))
?cns <- (current-node-service ?ps)
?serv <- (node (type service)
            (parent ?ps)
            (name ?ns)
            (properties ?c1 $?resto_rs))

=>
(modify ?recr (parent ?rd)(properties ?c $?lista))
(assert (node (type property)
              (parent ?recr)
              (name ?c)
              (properties ?r)))

(modify ?rd (properties ?nr $?list_rd))
(assert (property-new))
(retract ?recr-s)
(retract ?ra)
(retract ?rcr)
(retract ?ccr)
(retract ?cns)

```

```
(duplicate ?serv (parent ?serv) (properties $?resto_rs)))
```

O **Grupo 13** é composto por regras que verificam se os valores da característica **signed** podem ser adaptados. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as características do **signed** são diferentes entre a propriedade do serviço e a do perfil do usuário, então não ocorre adaptação para esta propriedade, e ela é então colocada na lista de propriedades diferentes para este serviço.

```
(defrule properties-caract-signed-difference-prfl
  (root-s ?root)
  (root-recr-prfl ?root-recr-prfl)
  (root-recr-s ?root-recr-s)
  ?ccr <- (property-current ?c)
  ?rcr <- (recurso-current ?nr)
  ?ra <- (recurso-atual ?recr)

  ?recr-s <- (node (type property)
                  (parent ?root-recr-s)
                  (name ?c&:(eq ?c signed))
                  (properties ?r))

  ?recr-f <- (node (type property)
                  (parent ?root-recr-prfl)
                  (name ?c)
                  (properties ?r1&:(not (eq ?r ?r1))))

  ?rd <- (node (type difference)
              (parent ?root)
              (name ?ns)
              (properties $?list_rd))

  ?recr-r <- (node (type propertyset)
                (parent nil)
                (name ?nr)
                (properties $?lista))

  ?cns <- (current-node-service ?ps)
  ?serv <- (node (type service)
              (parent ?ps)
              (name ?ns)
              (properties ?c1 $?resto_rs))

  =>
  (modify ?recr (parent ?rd)(properties ?c $?lista))
  (assert (node (type property)
                (parent ?recr)
                (name ?c)
                (properties ?r)))

  (modify ?rd (properties ?nr $?list_rd))
  (assert (property-new))
  (retract ?recr-s)
  (retract ?ra)
  (retract ?rcr)
  (retract ?ccr)
  (retract ?cns))
```

```
(duplicate ?serv (parent ?serv) (properties $?resto_rs)))
```

O **Grupo 14** é composto por regras que verificam se os valores da característica `dataType` podem ser adaptados. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as características do `dataType` da propriedade do serviço for igual a do perfil do usuário, e a do terminal a adaptação ocorre para esta propriedade.

```
(defrule properties-caract-dataType-igual
  (root-s ?root)
  (root-recr-tp ?root-recr-t)
  (root-recr-prfl ?root-recr-prfl)
  (root-recr-s ?root-recr-s)
  ?ccr <- (property-current ?c)
  (recurso-atual ?recr)

  ?recr-s <- (node (type property)
                  (parent ?root-recr-s)
                  (name ?c&:(eq ?c dataType))
                  (properties ?r))

  ?recr-t <- (node (type property)
                  (parent ?root-recr-t)
                  (name ?c)
                  (properties ?r))

  ?recr-f <- (node (type property)
                  (parent ?root-recr-prfl)
                  (name ?c)
                  (properties ?r))

  ?recr-r <- (node (type propertyset)
                  (parent nil)
                  (name ?nr)
                  (properties $?lista))

  =>
  (modify ?recr (properties ?c $?lista))
  (assert (node (type property)
                (parent ?recr)
                (name ?c)
                (properties ?r)))
  (retract ?ccr))
```

O **Grupo 15** é composto por regras que verificam se os valores da característica `mode` podem ser adaptados. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as características do `mode`, ou seja, se o modo de transmissão for diferente entre as propriedades do serviço do terminal então não ocorre adaptação para esta propriedade, e ela é então colocada na lista de propriedades diferentes para este serviço.

```
(defrule properties-caract-mode-difference-term
  (root-s ?root)
  (root-recr-tp ?root-recr-t)
```

```

(root-recr-s ?root-recr-s)
?ccr <- (property-current ?c)
?rcr <- (recurso-current ?nr)
?ra <- (recurso-atual ?recr)

?recr-s <- (node (type property)
                (parent ?root-recr-s)
                (name ?c&:(eq ?c mode))
                (properties ?r))

?recr-t <- (node (type property)
                (parent ?root-recr-t)
                (name ?c)
                (properties ?r1&:(not (eq ?r ?r1))))

?rd <- (node (type difference)
            (parent ?root)
            (name ?ns)
            (properties $?list_rd))

?recr-r <- (node (type propertyset)
                (parent nil)
                (name ?nr)
                (properties $?lista))

?cns <- (current-node-service ?ps)
?serv <- (node (type service)
            (parent ?ps)
            (name ?ns)
            (properties ?c1 $?resto_rs))

=>
(modify ?recr (parent ?rd)(properties ?c $?lista))
(assert (node (type property)
              (parent ?recr)
              (name ?c)
              (properties ?r)))

(modify ?rd (properties ?nr $?list_rd))
(assert (property-new))
(retract ?recr-s)
(retract ?ra)
(retract ?rcr)
(retract ?ccr)
(retract ?cns)
(duplicate ?serv (parent ?serv) (properties $?resto_rs)))

```

O **Grupo 16** é composto pelas regras que verificam se os valores da característica **color** podem ser adaptados. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as características do **color**, ou seja, se o número de cores da propriedade do serviço for menor ou igual a do terminal e do perfil do usuário, a adaptação ocorre de acordo com o valor do **color** do serviço.

```

(defrule properties-caract-color-menor-serv
  (root-s ?root)
  (root-recr-tp ?root-recr-t)

```

```

(root-recr-prfl ?root-recr-prfl)
(root-recr-s ?root-recr-s)
?ccr <- (property-current ?c)
(recurso-atual ?recr)

?recr-s <- (node (type property)
                (parent ?root-recr-s)
                (name ?c&:(eq ?c color))
                (properties ?r))

?recr-t <- (node (type property)
                (parent ?root-recr-t)
                (name ?c)
                (properties ?r1&:(<= ?r ?r1)))

?recr-f <- (node (type property)
                (parent ?root-recr-prfl)
                (name ?c)
                (properties ?r2&:(<= ?r ?r2)))

?recr-r <- (node (type propertyset)
                (parent nil)
                (name ?nr)
                (properties $?lista))
=>
(modify ?recr (properties ?c $?lista))
(assert (node (type property)
              (parent ?recr)
              (name ?c)
              (properties ?r)))
(retract ?ccr))

```

O **Grupo 17** é composto pelas regras que verificam se os valores da característica **size** podem ser adaptados. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as características do **size** da propriedade do serviço e do perfil do usuário forem maior do que a do terminal, a adaptação ocorre de acordo com o valor do **size** do terminal.

```

(defrule properties-caract-size-menor-term
(root-s ?root)
(root-recr-tp ?root-recr-t)
(root-recr-prfl ?root-recr-prfl)
(root-recr-s ?root-recr-s)
?ccr <- (property-current ?c)
(recurso-atual ?recr)

?recr-s <- (node (type property)
                (parent ?root-recr-s)
                (name ?c&:(eq ?c size))
                (properties ?r1 ?r2))

?recr-t <- (node (type property)
                (parent ?root-recr-t)
                (name ?c)
                (properties ?r3&:(> ?r1 ?r3) ?r4&:(> ?r2 ?r4)))

```

```

?recr-f <- (node (type property)
                (parent ?root-recr-prfl)
                (name ?c)
                (properties ?r5&:(<= ?r3 ?r5) ?r6&:(<= ?r4 ?r6)))

?recr-r <- (node (type propertyset)
                (parent nil)
                (name ?nr)
                (properties $?lista))

=>
(modify ?recr (properties ?c $?lista))
(assert (node (type property)
              (parent ?recr)
              (name ?c)
              (properties ?r3 ?r4)))
(retract ?ccr))

```

O **Grupo 18** é composto por regras que verificam se os valores de `maxDataLength` podem ser adaptados. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as características do `maxDataLength`, ou seja, se o número máximo do tamanho do dado da propriedade do serviço e do terminal forem maior do que a do perfil do usuário, a adaptação ocorre de acordo com o valor do `maxDataLength` do perfil do usuário.

```

(defrule properties-caract-maxDataLength-menor-prfl
  (root-s ?root)
  (root-recr-tp ?root-recr-t)
  (root-recr-prfl ?root-recr-prfl)
  (root-recr-s ?root-recr-s)
  ?ccr <- (property-current ?c)
  ?rcr <- (recurso-current ?nr)
  (recurso-atual ?recr)

  ?recr-s <- (node (type property)
                  (parent ?root-recr-s)
                  (name ?c&:(eq ?c maxDataLength))
                  (properties ?r))

  ?recr-t <- (node (type property)
                  (parent ?root-recr-t)
                  (name ?c)
                  (properties ?r1&:(> ?r ?r1)))

  ?recr-f <- (node (type property)
                  (parent ?root-recr-prfl)
                  (name ?c)
                  (properties ?r2&:(> ?r1 ?r2)))

  ?rd <- (node (type difference)
             (parent ?root)
             (name ?ns)
             (properties $?list_rd))

  ?recr-r <- (node (type propertyset)
                 (parent nil)

```

```

                (name ?nr)
                (properties $?lista))
=>
(modify ?recr (parent ?rd)(properties ?c $?lista))
(assert (node (type property)
              (parent ?recr)
              (name ?c)
              (properties ?r2)))

(modify ?rd (properties ?nr $?list_rd))
(assert (property-new))
(retract ?rcr)
(retract ?ccr)

```

O **Grupo 19** é composto pelas regras que verificam se todas as propriedades e características do serviço foram percorridas para dizer se ele pode ou não ser adaptado e de que forma. O código a seguir mostra uma das regras deste grupo que tem por objetivo verificar se as listas de propriedades comuns e de propriedades diferentes são maiores do que zero. Se for verdade o serviço pode ser adaptado parcialmente, ou seja, com todas as propriedades que estão na lista de propriedades comuns, obedecendo as características que contiverem nesta lista.

```

(defrule can-adapt-properties-equals
?serv <- (node (type service)
               (parent ?ps)
               (name ?ns)
               (properties ))

?rc <- (node (type equal)
            (parent ?prc)
            (name ?ns)
            (properties $?list_rc&:(> (length$ $?list_rc) 0)))

?rd <- (node (type difference)
            (parent ?prd)
            (name ?ns)
            (properties $?list_rd&:(> (length$ $?list_rd) 0)))

=>
(printout t "O serviço " ?ns " pode ser adaptado
para os recursos em comum. A lista
de recursos adaptaveis é: " ?list_rc crlf)

(assert (response ?ns)))

```

O **Grupo 20** é composto pelas regras que atualizam as informações para que outro serviço possa ser analisado. O código a seguir mostra uma das regras deste grupo que tem por objetivo verifica se o serviço atual, ou seja, se o serviço que esta sendo analisado, foi totalmente percorrido, se sim coloca-o na lista de serviços finalizados, para que possa analisar outro serviço, e também atualiza informações para análise do próximo serviço.

```

(defrule change-service

```

```

?resp <- (response ?ns)
?lsv <- (list-service-finished $?list)
?cns <- (current-node-service ?pcns)

?serv <- (node (type service)
              (parent ?pcns)
              (name ?ns)
              (properties))

?rz <- (root-s ?root)
?nr <- (property-new)
=>
(retract ?lsv)
(assert (list-service-finished ?ns $?list))
(retract ?serv)
(retract ?cns)
(retract ?resp)
(assert (current-node-service 0))
(retract ?rz)
(retract ?nr)

```

5.3 Cenário de Validação da Arquitetura Proposta

Para validar a arquitetura proposta considera-se que a Internet constitui a fonte dos serviços que podem ser adaptados a outros domínios, além de funcionar como elemento integrador das várias arquiteturas de rede envolvidas e duas classes de serviços:

- a de vídeo conferência, onde inicialmente é estabelecida entre dois usuários que fazem uso de terminais capazes de abrigar *web browser* com suporte a *applets* JAVA e objetos distribuídos (Fig. 5.11). Depois um dos usuários move-se para um outro terminal que não mais apresenta o mesmo suporte e infra-estrutura computacional, surge então a necessidade de adaptação do serviço a nova infra-estrutura do terminal e do provedor;

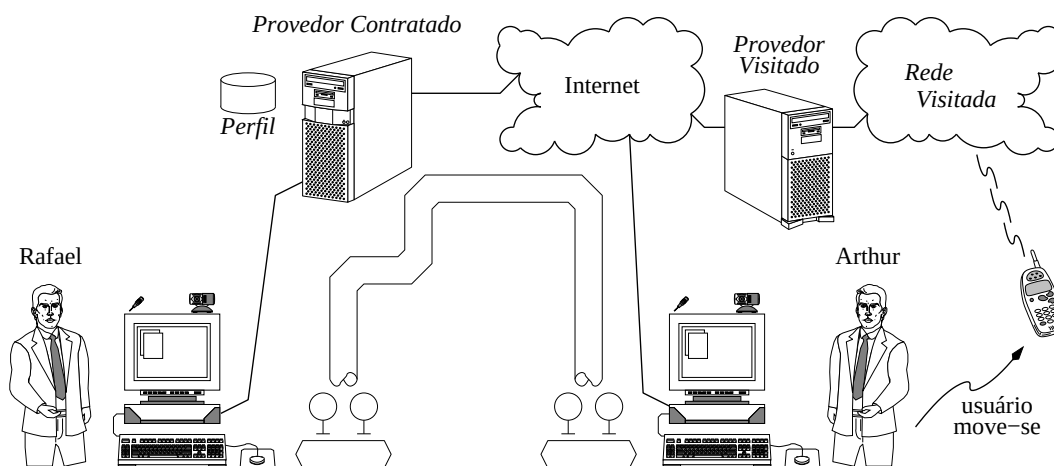


Fig. 5.11: Suporte à Mobilidade no Serviço de Vídeo-Conferência.

- a de conteúdo sob demanda onde apenas um usuário está envolvido, assim uma única sessão de comunicação é estabelecida, e por consequência, um fluxo será solicitado e ativado entre os componentes no domínio do usuário e do provedor (Fig. 5.12).

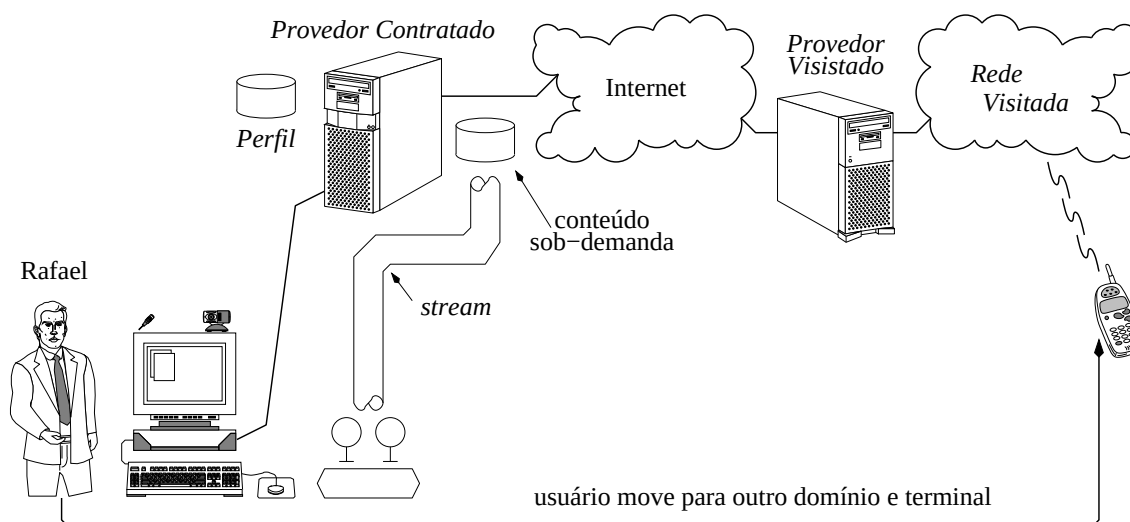


Fig. 5.12: Suporte à Mobilidade no Serviço de Conteúdo Sob Demanda.

Dentre as duas classes de serviços consideradas, a diferença principal reside no fato de que a primeira envolve múltiplos participantes, enquanto que a segunda não. Para avaliar o protótipo nas situações em que se faz necessário à mobilidade de pessoal e de serviço, aborda-se alguns cenários onde o serviço pode adaptar completamente, parcialmente ou não adaptar o usuário estando no seu provedor contratado ou em um provedor visitado.

As próximas seções apresentam sessão de serviço no provedor contratado e no provedor visitado para validar cenários de acordo com a mobilidade pessoal e/ou serviço.

5.4 Sessão de Serviço no Provedor Contratado

Considere uma infra-estrutura composta por terminais com diferentes recursos computacionais. Estes terminais podem ser estações de trabalho dotadas de câmera, microfone e alto-falante, por celulares e palms com suporte para vídeo e áudio. Os provedores estão representados por máquinas que apresentam uma configuração de *hardware* mais do que suficiente para suportar os sistemas legados de subscrição, acesso e serviço, além dos componentes que compõem o ponto de referência entre provedores.

Para que se possa fazer uso de serviços faz necessário, que a agência, agente do usuário sejam criados no provedor contratado de acordo como os passos a seguir:

- primeiro cria-se a agência para receber o agente de usuário, como mostra a Fig. 5.13, ela é registrada no servidor de agências MAF;

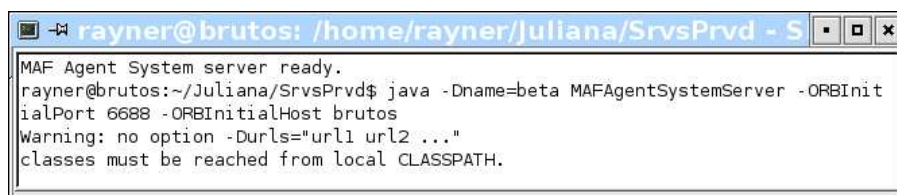


Fig. 5.13: Agência Beta é criada no Provedor Contratado.

- quando o usuário acessa o provedor contratado o agente do usuário é criado na agência beta (Fig. 5.14) e quando o usuário solicita o uso de um serviço o componente de sessão de acesso solicita a lista de serviços do agente de usuário

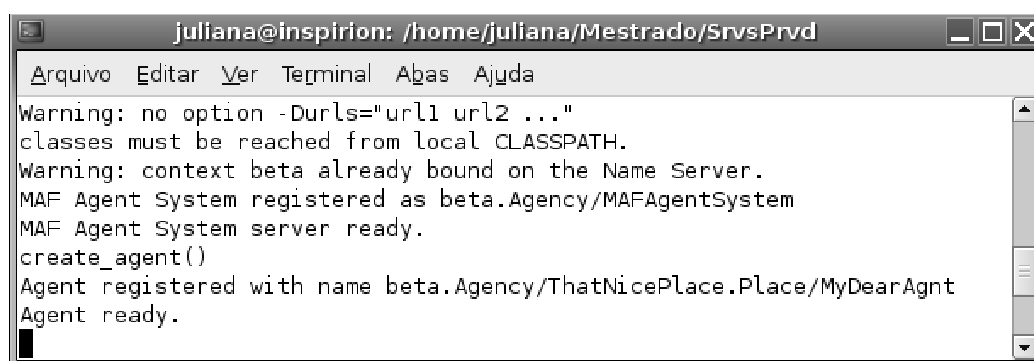


Fig. 5.14: O agente é criado na Agência Beta.

Agora suponha que dois usuários desejam estabelecer uma sessão de acesso sobre seu provedor contratado e iniciam um serviço de áudio-vídeo-conferência, as características do serviço são mostradas na Fig. 5.15.

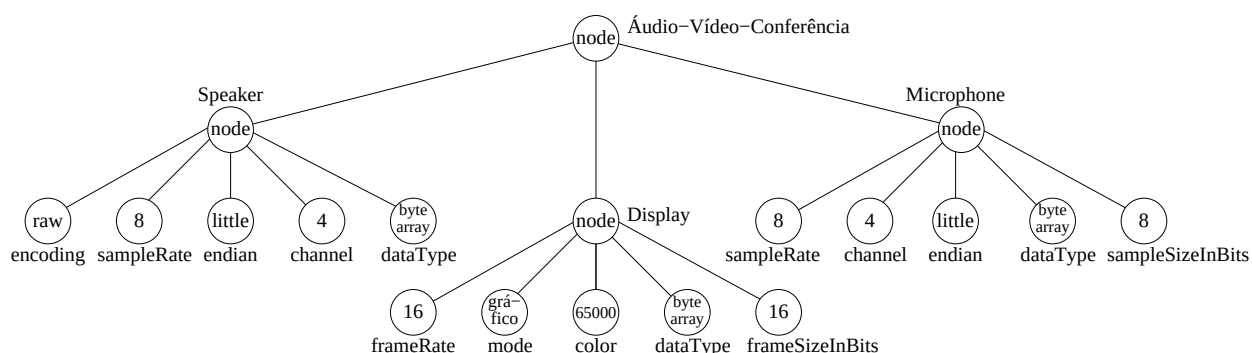


Fig. 5.15: Árvore que Representa as Propriedades do Serviço de Áudio-Vídeo-Conferência.

Nas subseções seguintes serão analisados cenários onde um destes usuários interrompe o serviço e o retoma a partir de um outro terminal para o qual será analisado a adaptação do serviço, sem que o agente de usuário tenha que migrar, pois não há mudança de provedor.

5.4.1 Serviço Completamente Adaptável no Provedor Contratado

Considerando o cenário exposto na Seção 5.4, onde dois usuários estão fazendo uso de um serviço de Áudio-Vídeo-Conferência, após um certo tempo um dos usuários interrompe o serviço e o retoma a partir de uma estação de trabalho, a qual faz parte da infra-estrutura computacional proposta e cuja as propriedades são mostradas na Fig. 5.16.

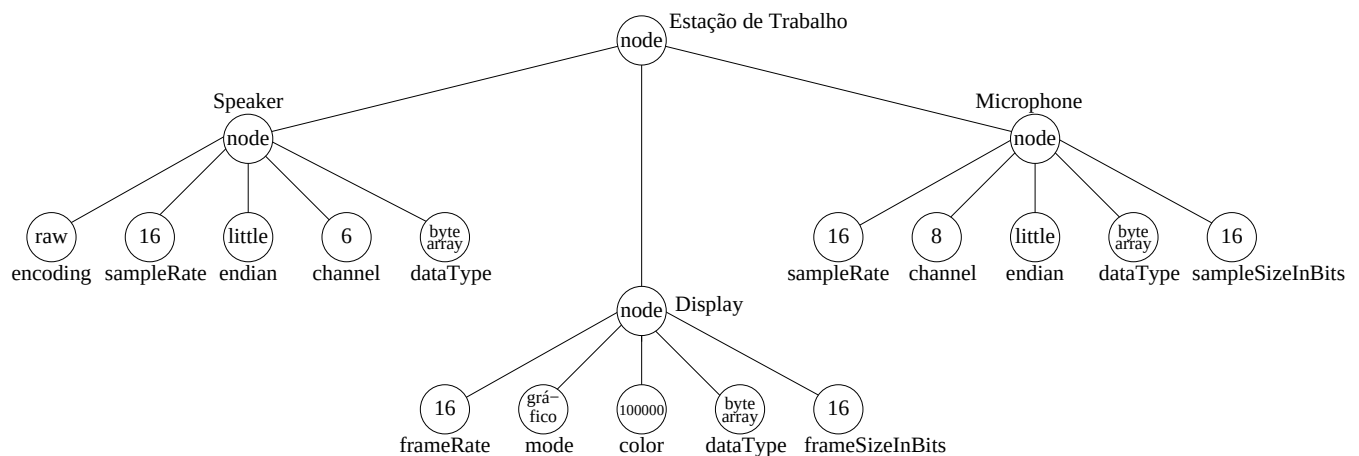


Fig. 5.16: Árvore que Representa as Propriedades da Estação de Trabalho.

A estação de trabalho possui todos os recursos necessários para a continuação da áudio-vídeo-conferência, pois todas as suas propriedades são iguais ou superiores as do serviço, lembrando que quando a característica é numérica ele se adapta de acordo com o menor, neste caso o serviço. A Fig. 5.17 mostra a resposta a adaptação de serviço.

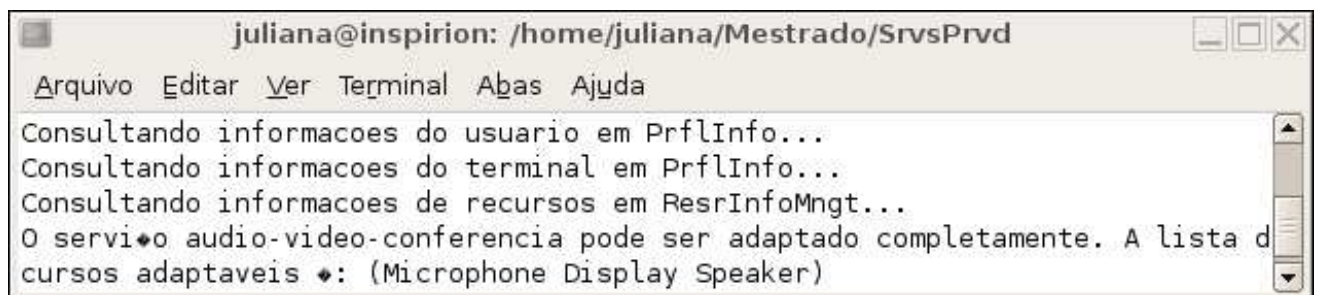


Fig. 5.17: Serviço Completamente Adaptável no Provedor Contratado.

5.4.2 Serviço Parcialmente Adaptável no Provedor Contratado

Em um segundo momento um dos usuários que estava participando do uso do serviço de Áudio-Vídeo-Conferência, exposto anteriormente, interrompe o serviço e o retoma a partir de um palm, onde suas características são mostradas na Fig. 5.18.

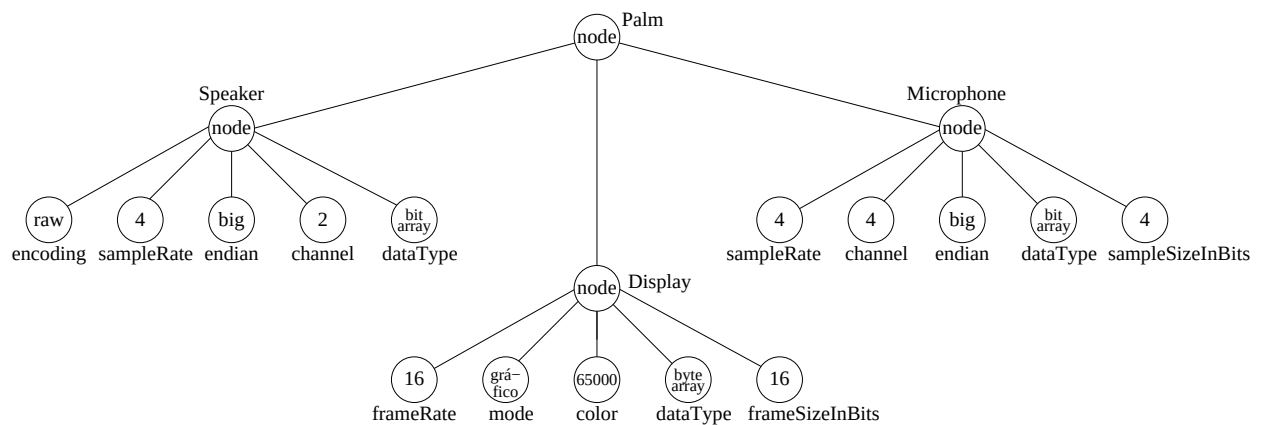


Fig. 5.18: Árvore que Representa as Propriedades do Palm.

O palm não possui suporte de áudio necessário para uma adaptação completa, pois as propriedades `sampleRate`, `sampleSizeInBits` possuem características inferior a do serviço, mas sendo possível dar continuidade ao serviço de áudio-vídeo-conferência a partir das capacidades do terminal. A Fig. 5.19 mostra a resposta a adaptação de serviço.

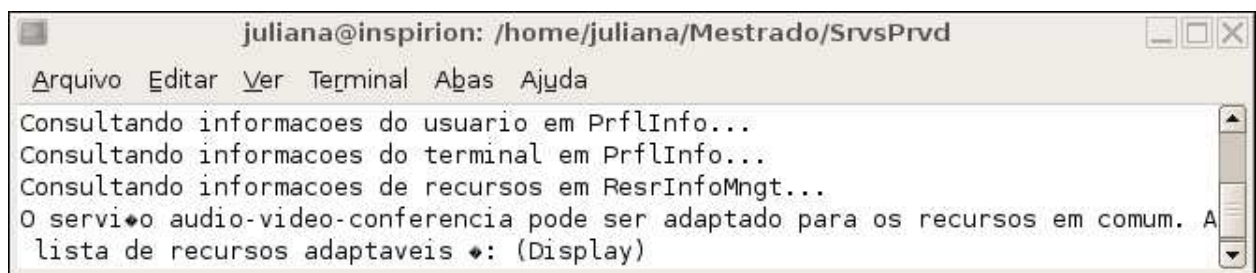


Fig. 5.19: Serviço Parcialmente Adaptável no Provedor Contratado.

5.4.3 Serviço Não Adaptável no Provedor Contratado

Em um terceiro momento um dos usuários interrompe o serviço e o retoma a partir de um celular, com pouco suporte computacional e que faz parte da infra-estrutura computacional proposta, suas características são mostradas na Fig. 5.20.

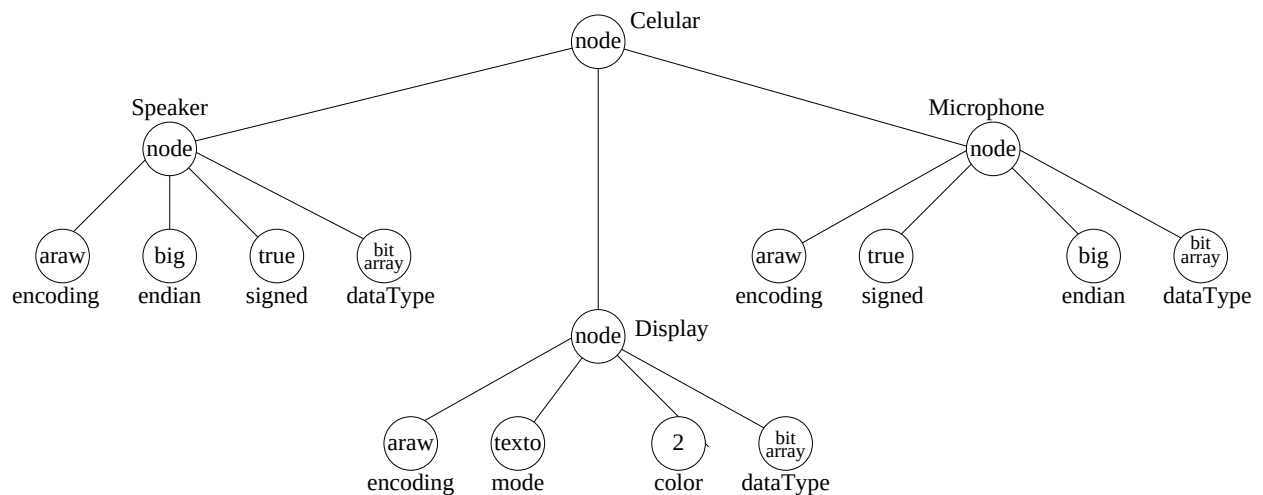


Fig. 5.20: Árvore que Representa as Propriedades do Celular.

O celular não possui as capacidades mínimas para a adaptação do serviço de áudio-vídeo-conferência, pois as propriedades que possui em comum não são iguais e outras ele não possui. A Fig. 5.21 mostra que o serviço não pode ser adaptado para este terminal, portanto o usuário não consegue dar continuidade ao serviço.

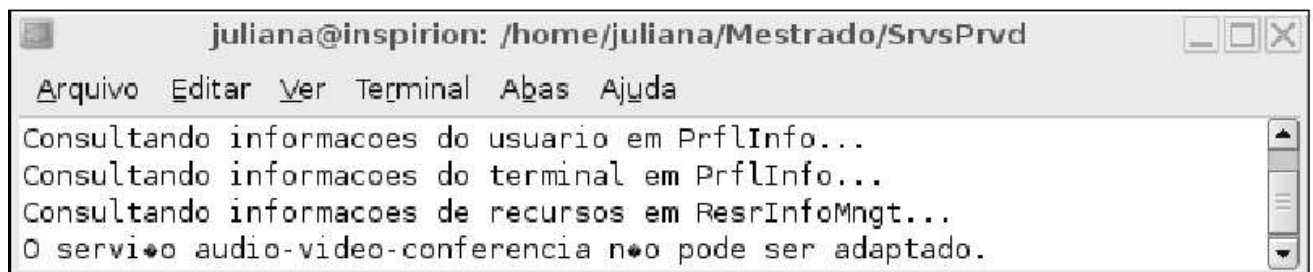


Fig. 5.21: Serviço Não Adaptável no Provedor Contratado.

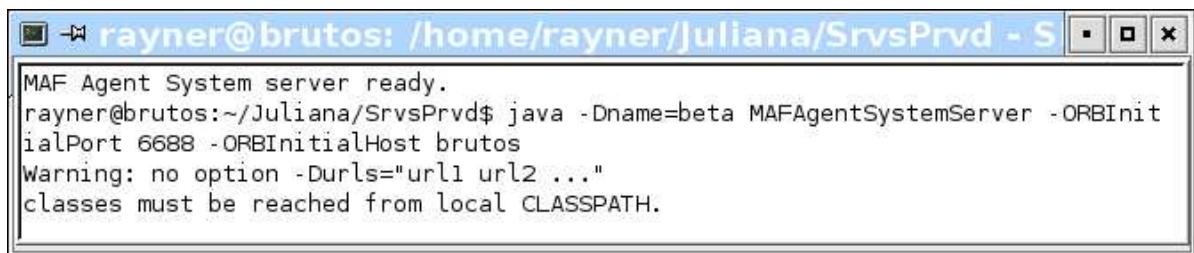
5.5 Sessão de Serviço no Provedor Visitado

Esta seção apresenta cenários que estejam de acordo com a infra-estrutura apresentada na Seção 5.4, nestes cenários dois usuários fazem uso de um serviço de áudio-vídeo-conferência (Fig. 5.15) contratado em seu provedor contratado, em um determinado momento um dos usuários se move para um outro provedor, sendo este um provedor visitado. Neste caso há a necessidade do agente do usuário migrar, e avaliar o suporte do provedor, bem como o perfil do usuário e do terminal.

Para que serviços possam ser analisados se faz necessário, que as agências, agente do usuário

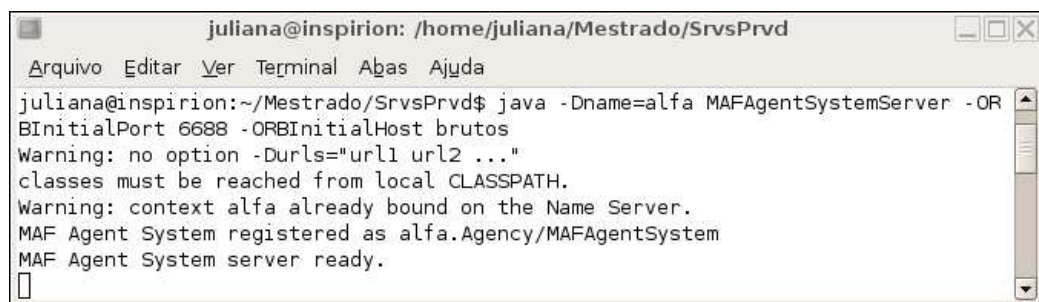
sejam criados e solicitada a migração do agente do usuário da agência referente ao seu provedor contratado para a agência do provedor visitado, de acordo como os passos a seguir:

- primeiro cria-se duas agências para receber agentes de usuários, como mostra as Fig. 5.22 e 5.23, cada agência estará localizada em um provedor e registrado no servidor MAF;



```
rayner@brutos: /home/rayner/Juliana/SrvsPrvd - S
MAF Agent System server ready.
rayner@brutos:~/Juliana/SrvsPrvd$ java -Dname=beta MAFAgentSystemServer -ORBInit
ialPort 6688 -ORBInitialHost brutos
Warning: no option -Durls="url1 url2 ..."
classes must be reached from local CLASSPATH.
```

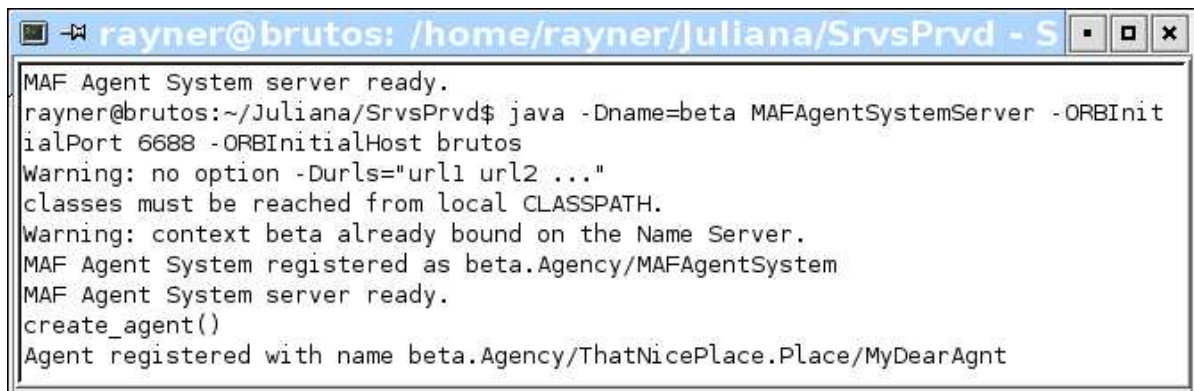
Fig. 5.22: Agência Beta é criada no Provedor Contratado.



```
juliana@inspirion: /home/juliana/Mestrado/SrvsPrvd
Arquivo Editar Ver Terminal Abas Ajuda
juliana@inspirion:~/Mestrado/SrvsPrvd$ java -Dname=alfa MAFAgentSystemServer -OR
BInitialPort 6688 -ORBInitialHost brutos
Warning: no option -Durls="url1 url2 ..."
classes must be reached from local CLASSPATH.
Warning: context alfa already bound on the Name Server.
MAF Agent System registered as alfa.Agency/MAFAgentSystem
MAF Agent System server ready.
```

Fig. 5.23: Agência Alfa é criada no Provedor Visitado.

- quando o usuário acessa o provedor contratado o agente do usuário é criado na agência beta (Fig. 5.24) e quando o usuário solicita o uso de um serviço o componente de sessão de acesso solicita a lista de serviços do agente de usuário;



```
rayner@brutos: /home/rayner/Juliana/SrvsPrvd - S
MAF Agent System server ready.
rayner@brutos:~/Juliana/SrvsPrvd$ java -Dname=beta MAFAgentSystemServer -ORBInit
ialPort 6688 -ORBInitialHost brutos
Warning: no option -Durls="url1 url2 ..."
classes must be reached from local CLASSPATH.
Warning: context beta already bound on the Name Server.
MAF Agent System registered as beta.Agency/MAFAgentSystem
MAF Agent System server ready.
create_agent()
Agent registered with name beta.Agency/ThatNicePlace.Place/MyDearAgnt
```

Fig. 5.24: O agente é criado na Agência Beta.

- quando o usuário se move para o provedor visitado, pede-se ao agente do usuário para migrar (Fig. 5.25), levando consigo a lista de serviços contratados no provedor contratado, a Fig. 5.26 mostra a agência alfa recebendo o agente;

```

juliana@inspirion: /home/juliana/Mestrado/SrvsPrvd/Cliente
Arquivo Editar Ver Terminal Abas Ajuda
juliana@inspirion:~/Mestrado/SrvsPrvd/Cliente$ java -Dagencia_de=beta -Dagencia_
_para=alfa -Dagente=MyDearAgnt -Dlugar=ThatNicePlace Migrar_Agente -ORBInitialPo
rt 6688 -ORBInitialHost brutos
MyDearAgnt migrou para alfa
juliana@inspirion:~/Mestrado/SrvsPrvd/Cliente$

```

Fig. 5.25: O agente migra de Beta para a Alfa.

```

juliana@inspirion: /home/juliana/Mestrado/SrvsPrvd
Arquivo Editar Ver Terminal Abas Ajuda
BInitialPort 6688 -ORBInitialHost brutos
Warning: no option -Durls="url1 url2 ..."
classes must be reached from local CLASSPATH.
MAF Agent System registered as alfa.Agency/MAFAgentSystem
MAF Agent System server ready.
Receiving agent MyDearAgnt
create_agent()
Agent registered with name alfa.Agency/ThatNicePlace.Place/MyDearAgnt
Agent ready.

```

Fig. 5.26: A Agência Alfa recebe o Agente.

5.5.1 Serviço Completamente Adaptável no Provedor Visitado

Considerando o cenário exposto na Seção 5.5, onde dois usuários estavam fazendo uso de um serviço de Áudio-Vídeo-Conferência e após um certo tempo um dos usuários interrompe o serviço e o retoma em um provedor visitado, e a partir de uma estação de trabalho a qual faz parte da infra-estrutura computacional proposta. As propriedades do terminal, uma Estação de Trabalho, para qual um dos usuários migrou são mostradas na Fig. 5.27.

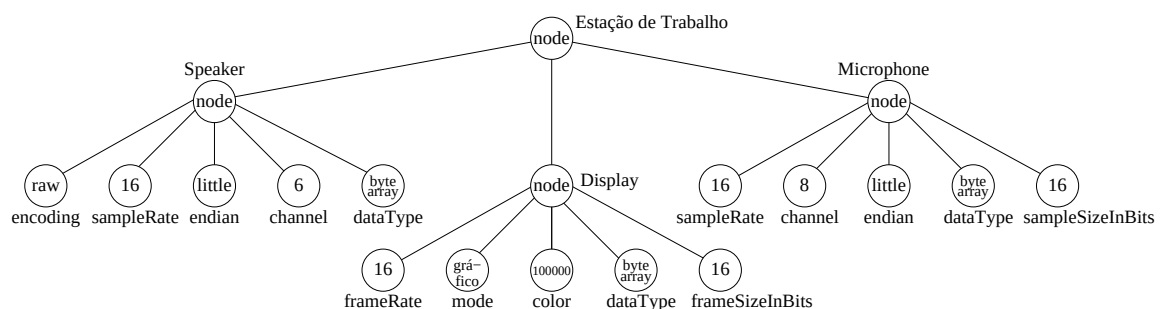


Fig. 5.27: Árvore que Representa as Propriedades da Estação de Trabalho.

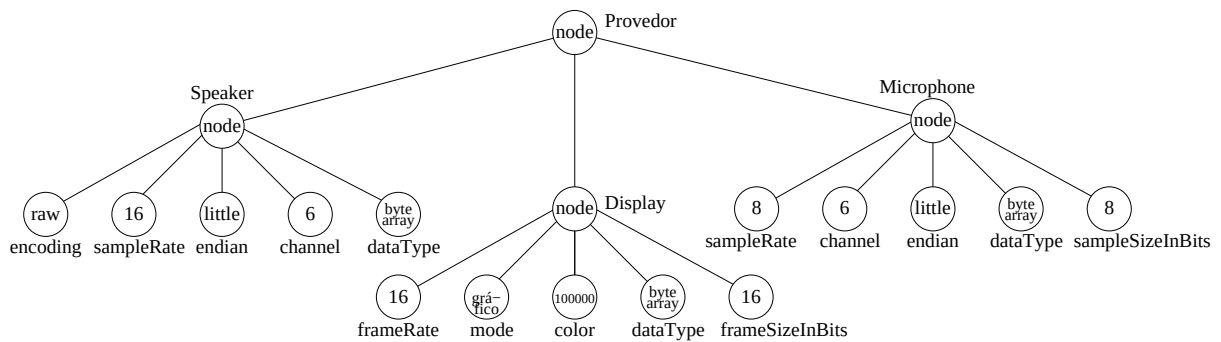


Fig. 5.28: Árvore que Representa as Propriedades do Provedor Visitado.

A Fig. 5.28 mostra as propriedades do provedor visitado, para o qual o usuário migrou.

As propriedades do perfil do usuário no provedor visitado são mostradas na Fig. 5.29.

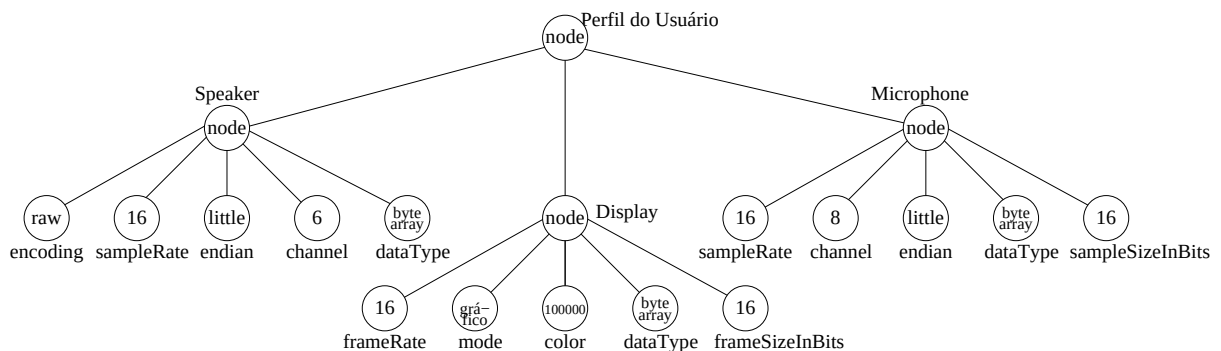


Fig. 5.29: Árvore que Representa as Propriedades do Perfil do Usuário.

As propriedades do terminal, do provedor e do perfil do usuário satisfazem as necessidades do serviço, assim possuindo todos os recursos necessários para a continuação da áudio-vídeo-conferência. A Fig. 5.30 que o serviço pode ser adaptado completamente pelo provedor, pelo perfil do usuário, como também pelo terminal.

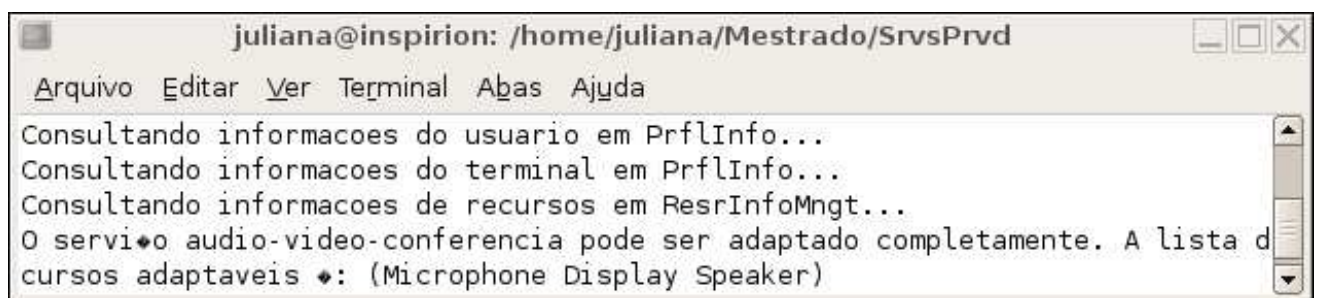


Fig. 5.30: Serviço Completamente Adaptável no Provedor Visitado.

5.5.2 Serviço Parcialmente Adaptável no Provedor Visitado

Em um segundo momento um dos usuários se move para um provedor visitado, e deseja continuar a fazer uso do serviço de áudio-vídeo-conferência em um palm. O agente de usuário necessita verificar as características do provedor, do perfil do usuário e do terminal para dizer se o serviço pode ser adaptado. As características do perfil do usuário para este provedor, do provedor e do terminal são mostradas nas Figs. 5.31, 5.32 e 5.33 respectivamente.

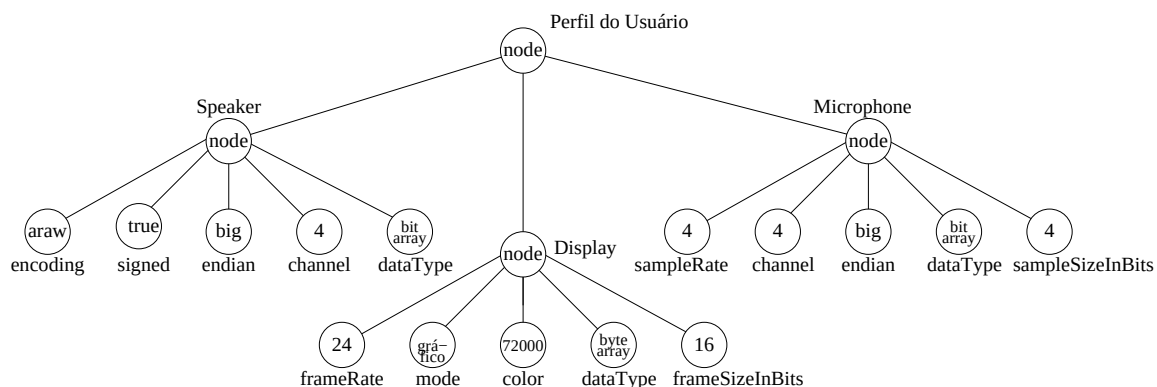


Fig. 5.31: Árvore que Representa as Propriedades do Perfil do Usuário.

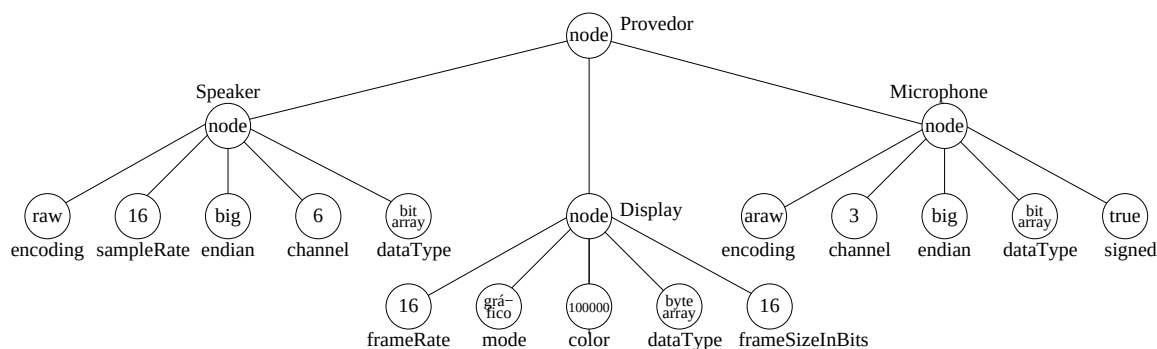


Fig. 5.32: Árvore que Representa as Propriedades do Provedor Visitado.

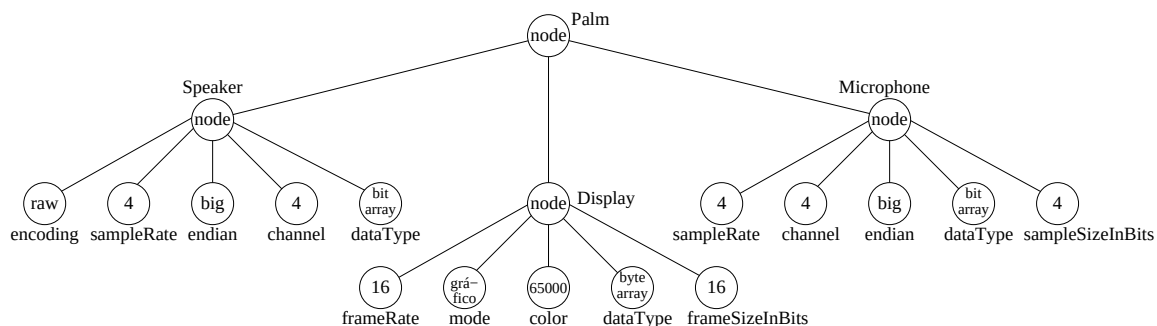


Fig. 5.33: Árvore que Representa as Propriedades do Palm.

O agente de usuário consulta as informações a respeito do provedor, do perfil do usuário e do terminal, constata que o serviço não pode ser adaptado completamente, pois nenhum dos três possui características suficiente para executar o serviço com todos os recursos necessários. O palm não satisfaz a propriedade `frameRate`, o perfil do usuário a do `sampleRate` e o provedor a do `framesizeInBits`. A Fig. 5.34 mostra a adaptação do serviço neste novo cenário.

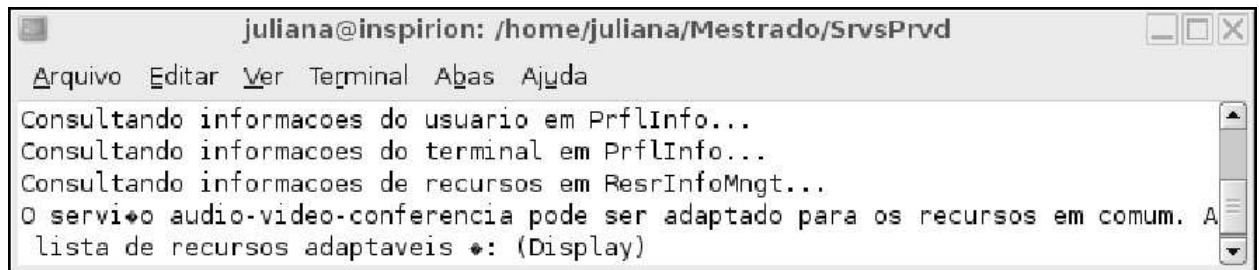


Fig. 5.34: Serviço Parcialmente Adaptável no Provedor Visitado.

5.5.3 Serviço Não Adaptável no Provedor Visitado

Em um terceiro momento um dos usuários se move para um outro provedor, ou seja, para um outro provedor visitado, e deseja continuar fazer uso do serviço de áudio-vídeo-conferência em um celular com pouco recurso computacional.

As características do terminal ao qual o usuário fará uso para acessar o serviço, um celular, são apresentadas na Fig. 5.35.

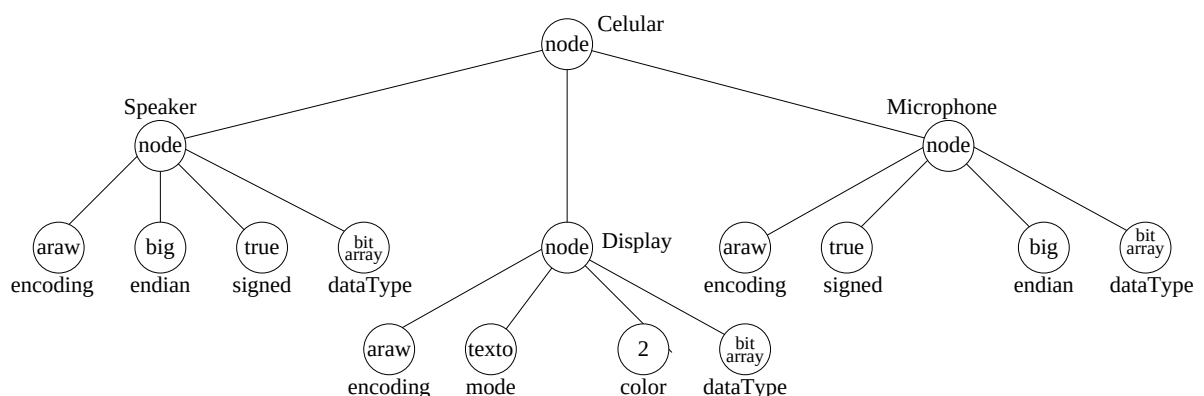


Fig. 5.35: Árvore que Representa as Propriedades do Celular.

As características do provedor visitado para o qual o usuário se moveu e de onde deseja continuar a usar o serviço são apresentadas na Fig. 5.36.

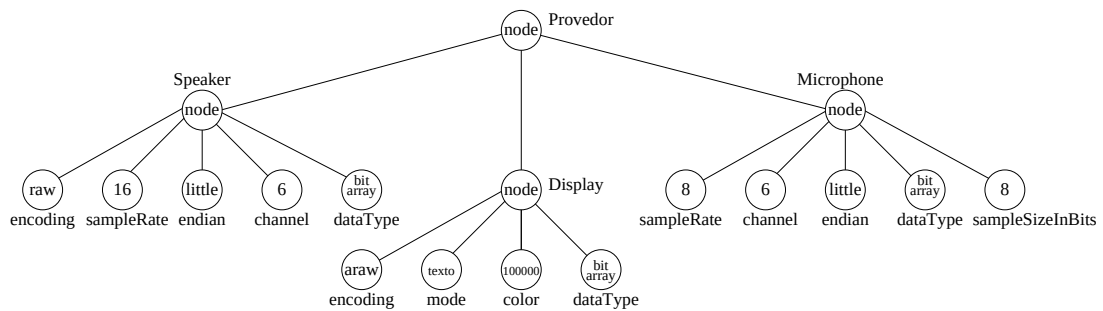


Fig. 5.36: Árvore que Representa as Propriedades do Provedor Visitado.

As características do perfil do usuário para o provedor visitado em questão são apresentadas na Fig 5.37.

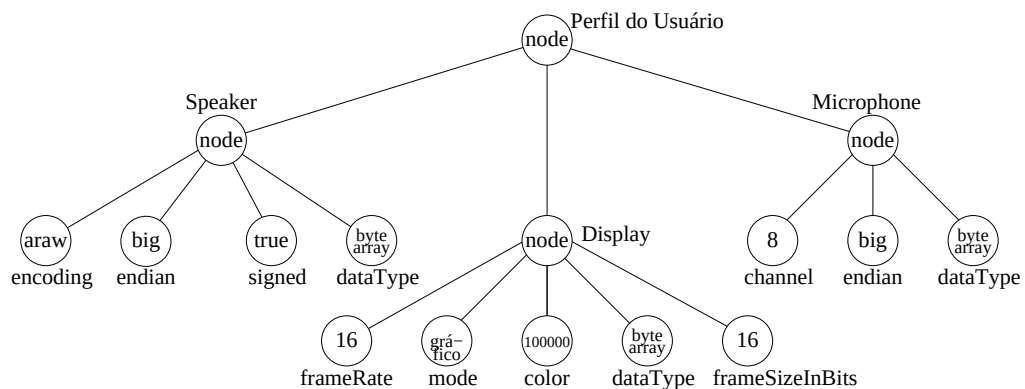


Fig. 5.37: Árvore que Representa as Propriedades do Perfil do Usuário.

Quando o agente de usuário migra e consulta as informações a respeito do provedor, do perfil do usuário e do terminal, constata que o serviço não pode ser adaptado, pois as características do serviço não podem ser satisfeitas em conjunto pelo terminal, pelo provedor e pelo perfil do usuário. A Fig. 5.38 mostra a adaptação do serviço neste novo cenário.

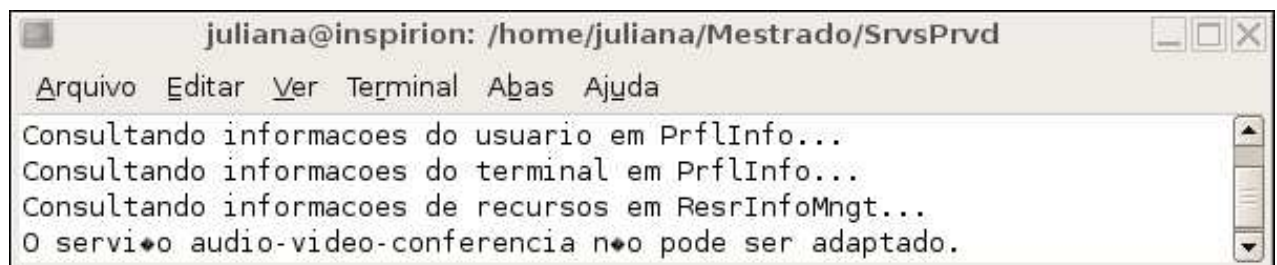


Fig. 5.38: Serviço Não Adaptável no Provedor Visitado.

5.6 Avaliação e Comentários

Com base na arquitetura proposta e na busca da ubiquidade de serviços a partir de outros domínios devidamente acordados na federação de serviços, é desejado trazer de um outro domínio os componentes necessários a adaptação de um dado serviço. Assim nossa proposta se baseia na descentralização dos serviços de controle e gerenciamento de *software*, trazendo o controle, gerenciamento e capacidade de adaptação tão próximo quanto possível dos seus recursos através do agente de usuário.

Para que a proposta possa ser alcançada, é necessário que os provedores estejam em conformidade com o ponto de referência entre provedores, o qual necessita da instalação e configuração de um ambiente para processamento distribuído e de uma agência em cada provedores. O suporte ao processamento distribuído é necessário não só para garantir a interoperabilidade entre sistemas heterogêneos, mas com o intuito de permitir que a arquitetura proposta coexista e interopere com sistemas legados presentes nas redes, provedores e terminais. E a agência suporta aspectos do comportamento do agente incluindo execução, comunicação e mobilidade.

Com relação aos agentes estes são responsáveis pela realização de parte de um processo pragmático e com interfaces de comunicação bem definidas, estas características inerentes ao paradigma de agentes conduzem a propriedades tais como modularidade, flexibilidade e encapsulamento. A modularidade e o encapsulamento foram obtidos através da implementação do agente de usuário, já a flexibilidade foi obtida pela migração do agente, onde o mesmo se move de acordo com a percepção do ambiente, e adaptando o serviço por possuir uma inteligência.

Assim, a implementação permitiu constatar a adequação da arquitetura proposta baseada em agentes móveis inteligentes, em relação a flexibilidade da arquitetura no suporte a migração, adaptação e configuração do serviço decorrentes da mobilidade pessoal e de serviço.

Capítulo 6

Conclusão

Cada vez mais as pessoas prezam pela mobilidade, exigindo dos serviços de telecomunicações e de informação um caráter ubíquo. Este trabalho teve como proposta a integração de ambientes de comunicação que permita o acesso a um conjunto de serviços a partir de diferentes localizações e dispositivos. Para avaliar a proposta, implementa-se um protótipo onde aspectos de mobilidade e de adaptação do serviço são investigados, baseado na junção dos paradigmas de objetos distribuídos e de agentes móveis inteligentes. Cada paradigma oferece vantagens em diferentes aspectos, e assim em conjunto podem prover o máximo de flexibilidade no suporte à mobilidade de aplicações distribuídas configuráveis como os serviços emergentes de telecomunicações. Isto nos permite dizer que é possível oferecer uma infra-estrutura de suporte à mobilidade aos serviços de informação e de telecomunicações em um ambiente de comunicação integrado.

Dentre os aspectos relevantes desta proposta, destacamos:

- proposição do uso da arquitetura que coexiste com sistemas legados de subscrição, acesso e serviço no provedor, preservando a infra-estrutura já existente;
- o agente de serviço traz flexibilidade ao permitir a distribuição do *software* e/ou provisão do serviço sob demanda configurável;
- descentralização dos serviços de controle e gerenciamento de *software*, trazendo o controle, gerenciamento e capacidade de adaptação tão próximo quanto possível dos seus recursos através do agente de usuário;
- provedores podem ou não permitir o uso de recursos de *hardware/software* dependendo das capacidades do terminal.

Um outro aspecto diz respeito aos agentes móveis, ou seja, código que viaja através da rede realizando tarefas do usuário, mantendo sempre seu estado que é uma das contribuições deste trabalho. Trata-se de uma tecnologia importante para a computação móvel e acesso remoto. Neste caso, usuários remotos podem realizar o trabalho *off-line* e despachar o agente móvel para realizar o trabalho em um provedor distante através de uma simples conexão de rede. Esta tecnologia tem um impacto importante na rede, pois pode colaborar para uma redução das necessidades de recursos de rede.

A abordagem baseada em agentes de *software* constitui-se uma alternativa para o desenvolvimento de sistemas que representam características tais como: rápido desenvolvimento e rápida manutenção; independência de plataforma; facilidade de integração e uso; flexibilidade e eficiência e suporte à computação móvel.

Os agentes inteligentes são módulos de *software* que facilitam a automação de tarefas complexas do usuário. O nível de inteligência dos agentes varia desde módulos pré-programados, cujo o comportamento pode ser definido em detalhe pelo usuário, até os agentes inteligentes adaptáveis que podem detectar padrões de uso, adaptando-se automaticamente às necessidades e modo de trabalhar do usuário. O maior benefício dos agentes inteligentes é a habilidade que possuem de realizar trabalho útil a qualquer tempo o computador esteja sem atividades ou, quando o tráfego na rede encontra-se com baixa demanda.

A inteligência do agente desenvolvida em JESS, além de ter um código menor do que se fosse desenvolvido em outro paradigma e de ser de fácil entendimento, a estrutura em árvore fez com que o código possa ser acrescido de novas regras, sem a necessidade de modificar a estrutura já existente, bem como as regras.

No que se refere a implementação do *user agent* usando a plataforma JESS, cabe como ressalva a necessidade de coordenação dos nomes de serviços, de recursos contemplados nos terminais bem como de recursos de infra-estrutura oferecida pelos provedores, pois em muito facilita a geração da base de conhecimento. Por fim, nosso agente mantém o estado quando da sua reinstanciação remota, ou seja, o agente de fato sofre uma migração. Tal funcionalidade se deve ao uso do JESS.

Este trabalho teve também como contribuição a descrição formal através de redes de Petri das interações entre os componentes da arquitetura utilizada, podendo desta forma obter um maior detalhamento destas interações bem como a visibilidade de que estas interações não entram em *deadlocks*.

Referências Bibliográficas

- [1] David Clark, Joseph Pasquale, and et al. “Strategic Direction in Networks and Telecommunications”. *ACM Computing Surveys*, 28(4), Dezembro 1996.
- [2] Luís F. Faina. “*Uma Arquitetura para Suporte à Ubiquidade dos Serviços de Telecomunicações Baseada na Arquitetura TINA e em Agentes Móveis*”. Tese de Doutorado, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, Campinas, Brasil, Dezembro 2000.
- [3] M. Khedr and A. Karmouch. “ACAI: Agent-Based Context-Aware Infrastructure for Spontaneous Applications.”. *Journal of Network and Computer Applications*, 2005.
- [4] Harry Chen, Tim Finin, Anupam Joshi, Lalana Kagal, Filip Perich, and Dipanjan Chakraborty. “Intelligent Agents Meet the Semantic Web in Smart Spaces.”. *IEEE Internet Computin*, 8(6):69–79, 2004.
- [5] Harry Chen, Tim Finin, and Anupam Joshi. An intelligent broker for context-aware systems, September 06 2003.
- [6] H. Nwana. "software agents: An overview". In *knowledge and Engineering Review*. Knowledge and Engineering Review, Novembro 1996.
- [7] C. Hewitt. "viewing control structures as patterns of passing messages". In *Artificial Intelligence*, pages 323–364, 1977.
- [8] Jorge Muniz Barreto. “*Inteligência Artificial. No limiar do XXI*”. Duplic, Florianópolis, 3 edição edition, 2001.
- [9] William B. Gevarter. “*Artificial Intelligence, Expert Systems, Computer Vision and Natural Language Processing*”. Noyes Publications, 1984.
- [10] Rosina Weber. “*Pesquisa Jurisprudencial Inteligente*”. Tese de doutorado, Universidade Federal de Santa Catarina, UFSC, Florianópolis, Brasil, 1998.

- [11] Elaine Rich. *"Inteligência Artificial"*. McGraw-Hill, São Paulo, 1988.
- [12] E. Turban. *"Decision Support and Expert Systems. Management Support System"*. Prentice-Hall, 4 edition, 1995.
- [13] George F. Luger. *"Inteligência Artificial: Estruturas e Estratégias para a solução de Problemas Complexos"*. Bookman, 2004.
- [14] Stuart Russell and Peter Norving. *"Inteligência Artificial"*. Campus, Rio de Janeiro, 2 Edição edition, 2004.
- [15] Donald A. Waterman. *"A guide to expert systems"*. Addison-Wesley, Boston, 1985.
- [16] J. Durkin. *"Expert System Design and Development"*. Prentice-Hall, USA, 1994.
- [17] Limin Fu. *"Netural Networks in Computer Intelligence"*. MC Graw Hill, 1994.
- [18] Gary Riley. *"A Tool for Building Expert Systems"*, 2005. URL at <http://www.ghg.net/clips/CLIPS.html>.
- [19] Ernest J. Friedman-Hill. *"Jess, The Expert System Shell to the Java Platform"*. Sandia National Laboratories, 2001. URL at <http://herzberg.ca.sandia.gov/jess>.
- [20] Tim Finin, Yannis Labrou, and James Mayfield. *"Software Agents"*, chapter "KQML as an Agent Communication Language", pages 291–316. AAAI Press, Cambridge, 1997.
- [21] Solange Oliveira Rezende. *"Sistemas Inteligentes: Fundamentos e Aplicações"*. Manole, 1 Edição edition, 2002.
- [22] Renato Antônio Rabuske. *"Inteligência Artificial"*. Editora da UFSC, Florianópolis, 1995.
- [23] Stan Franklin and Art Graesser. "is it an agent, or just a program?p: A taxonomy for autonomous agents". In Springer-Verlag, editor, *Third International Workshop on Agent Theories, Architectures and Languages*, 1996.
- [24] Paulo C. Oliveira and Eleri Cardozo. "Mobile Agent-Based Systems: An Alternative Paradigm for Distributed Systems Development.". In *15 Simpósio Brasileiro de Redes de Computadores - SBRC'97*, pages 508–524, May 1997.

- [25] Stênio Firmino Pereira Filho. “Avaliação de Ambientes Servidores para Agentes Móveis”. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação - USP, São Carlos - SP, 2001.
- [26] OMG. “ Mobile Agent Facility Specification”. Document, Object Management Group, Jan. 2000. URL at <http://www.omg.org>.
- [27] Janette Cardoso and Robert Valette. “*Redes de Petri*”. Editora da UFSC, Florianópolis, 1997.
- [28] Juliana Oliveria de Carvalho and Luís Fernando Faina. “Agentes Móveis Inteligentes no Suporte a Ubiquidade dos Serviços de Telecomunicações.”. In *XIII SEMINCO - Seminário de Computação*, pages 117–128, Blumenau, SC, Setembro 2004.
- [29] OMG. “Common Object Request Broker: Architecture and Specification – Version 2.3.1”. Document formal/99-10-07, Object Management Group, Oct. 1999. <http://www.omg.org>.
- [30] Sun Microsystems. “*Java™ 2 Software Development Kit*”. Sun Microsystems, 2000. <http://www.java.sun.com/products/>.
- [31] OMG. “Java Language Mapping to OMG IDL”. Document formal/99-07-59, Object Management Group, July 1999. URL at <http://www.omg.org>.
- [32] Object Space. “Voyager Core Technology”. Technical Overview Version 1.0, Object Space, Inc., Dezembro 1997. <http://www.objectspace.com>.
- [33] Johnny Wong, Guy Helmer, Venkatraman Naganathan, and Srinivas Polavarapu. “SA-MART Mobile Agent Facility”. *The Journal of Systems and Software*, 56, Setembro 2001.