Universidade de Lisboa

Faculdade de Ciências

Departamento de Informática

Personal Unified Multimedia Applications

projecto realizado na

Portugal Telecom Inovação, S. A.

por

Bruno Duarte

Mestrado em Engenharia Informática

2007

Universidade de Lisboa

Faculdade de Ciências

Departamento de Informática

Personal Unified Multimedia Applications

projecto realizado na

Portugal Telecom Inovação, S. A.

por

Bruno Duarte

Projecto orientado pelo Engenheiro Paulo Chainho
e co-orientado pelo Professor Doutor Luís Correia

Júri:
Professor Doutor Luís Correia
Engenheiro Paulo Chainho
Professora Doutora Teresa Chambel

Mestrado em Engenharia Informática

2007

Declaração

Bruno Duarte, aluno n.º 30370 da Faculdade de Ciências da Universidade de Lisboa, declara ceder os seus direitos de cópia sobre o seu Relatório de Projecto em Engenharia Informática, intitulado "Personal Unified Multimedia Applications", realizado no ano lectivo de 2006/2007 à Faculdade de Ciências da Universidade de Lisboa para o efeito de arquivo e consulta nas suas bibliotecas e publicação do mesmo em formato electrónico na Internet.

FCUL, 30 de Setembro de 2007

Paulo Chainho, supervisor do projecto de *Bruno Duarte*, aluno da Faculdade de Ciências da Universidade de Lisboa, declara concordar com a divulgação do Relatório do Projecto em Engenharia Informática, intitulado "Personal Unified Multimedia Applications".

Lisboa, 30 de Setembro de 2007

There is nothing so powerful as an idea whose time has come.
Yes, even if it's a bad idea.

.

John McCarthy, Turing Award (1971) pela sua
contribuição no campo da Inteligência Artificial

Agradecimentos

Ao Paulo Chainho por ter sido um orientador presente e participativo, por ter dado sempre o apoio necessário para aprender e desenvolver o projecto e que teve uma grande responsabilidade no sucesso deste estágio.

Ao Professor Luís Correia que sempre se mostrou disponível e prestou muito apoio, em especial aquando da realização deste documento.

A toda a equipa da PT Inovação com quem tive o enorme prazer de trabalhar e que são uma eficaz fonte de apoio para os pequenos desafios de um dia-a-dia de trabalho.

A todos os meus amigos e em especial aos amigos especiais! Seja pelo apoio ao longo do percurso académico ou pelo simples facto de marcarem presença na nossa vida, os amigos tornam a nossa vida mais colorida.

À minha fantástica família por me terem proporcionado todo o apoio necessário. São eles a principal razão da não existência de maus momentos na minha vida.

E, a ti...

Resumo

Este documento descreve o projecto realizado no âmbito da disciplina de Projecto em Engenharia Informática do segundo ano do curso Mestrado em Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa.

O projecto é desenvolvido no Pólo de Lisboa da Portugal Telecom Inovação. Esta instituição tem como objectivo ser líder nas áreas de conhecimento estratégicas para o desenvolvimento dos negócios em que o Grupo Portugal Telecom está presente.

O trabalho de investigação e desenvolvimento é enquadrado no projecto IP-JIB, uma plataforma para fornecimento de serviços inteligentes na rede de telecomunicações de próxima geração do Grupo Portugal Telecom.

Em concreto, o trabalho consistiu no estudo e concepção de um serviço que permita ao utilizador controlar o comportamento de todas as suas telecomunicações, dependente do contexto em que este se encontra.

Relata-se o estudo que existiu no campo das redes de próxima geração e na concepção ou adaptação de um motor de regras que representasse a lógica de decisão de serviço. Neste âmbito, pretende-se esclarecer o leitor sobre a necessidade e a forma de concepção deste sistema normalmente associado à Inteligência Artificial.

PALAVRAS-CHAVE:
IMS, JAIN SLEE, communications manager, rules engine

Abstract

This document describes the project developed in the course *Projecto em Engenharia Informática* (Informatic Engineering Project) that took place in the second year of the course *Mestrado em Engenharia Informática* (Informatic Engineering Masters) of the University of Lisbon Faculty of Science.

The project took place at the Lisbon Pole of Portugal Telecom Inovação. The main goal of this company is to be leader in strategic research areas to develop the core business of the Portugal Telecom Group.

This research and development work is related to the project IP-JIB, a platform to deploy intelligent services in the Next Generation Networks Portugal Telecom's solution.

The main goal of the work was to study and develop a service that allowed the end user to manage the behavior of all his communications, in according with his context.

This document describes the study in the Next Generation Networks and the adaption or development of a Rules Engine to control a service business logic.

KEYWORDS:
IMS, JAIN SLEE, communications manager, rules engine

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Especificações

Capítulo 1

Introdução

Neste primeiro capítulo descreve-se muito sucintamente o projecto (motivação e objectivos) e a organização do documento.

1.1 Motivação

A revolução tecnológica nos últimos anos obrigou as organizações a alterar o seu modelo de negócio. Na área das telecomunicações o utilizador está a tornar-se cada vez mais exigente, obrigando os grandes *players* das telecomunicações a adaptarem os seus serviços às necessidades pontuais deste. Consequentemente, este segmento de mercado está cada vez menos rentável.

Com o intuito de responder às crescentes necessidades do utilizador, combater pequenos operadores e minimizar o custo de operação, as telecoms estão a apostar em tecnologias de comunicação de próxima geração assentes em redes de comutação de pacotes.

Com o empenho de organizações como a 3GPP, o TISPLAN e o IETF, tem-se vindo a afirmar a arquitectura de suporte ao paradigma de redes de próxima geração IP Multimedia Subsystem (IMS).

Estando neste momento várias organizações a estudar e planear a migração para o IMS, existem muitos desafios interessantes e que merecem ser alvo de profundo estudo.

Este documento relata em detalhe o projecto levado a curso na instituição PT Inovação, nomeadamente no estudo e desenvolvimento de componentes para a plataforma de serviços de comunicação inteligente.

1.2 Objectivos

O conceito de *Personal Unified Multimedia Applications* (PUMA) consiste em permitir a personalização de um conjunto unificado de procedimentos de aplicações multimédia que são independentes do tipo de comunicação, do tipo de informação multimédia e do tipo de

dispositivos e respectivas interfaces de utilizador usadas. Estas aplicações são adaptativas ao contexto em que se encontra o utilizador e terão numa próxima geração capacidade de prever e facilitar o comportamento deste.

Este conceito pode ser concretizado e visto pelo utilizador através da metáfora do *Assistente Pessoal*, sempre disponível para satisfazer as necessidades do utilizador, de acordo com o seu perfil e comportamento.

O *Assistente Pessoal* pode agir em nome do utilizador e operar com um comportamento definido pelo próprio ou automaticamente sugerido a este, tais como:

- gestão das comunicações e contactos do utilizador;
- gestão de tarefas e de compromissos registadas na agenda do utilizador;
- aprendizagem e definição automática do perfil de utilizador;
- gestão de comunicações e tarefas definidas no perfil de utilizador.

O projecto foca-se na concepção e implementação de um motor de regras enquadrado na plataforma actual de desenvolvimento de serviços e que seja apropriado ao desenvolvimento do tipo de aplicações supracitadas.

Ao longo do projecto este componente será validado e enquadrado no desenvolvimento de uma aplicação de gestão de comunicações orientada para o utilizador final. O principal objectivo desta aplicação será a organização e gestão de todo o tipo de comunicações do utilizador.

Em suma, os objectivos deste projecto são os seguintes:

- aquisição de conhecimentos nas normas e tecnologias em torno do IMS;
- concepção e desenvolvimento ou integração de motor de regras;
- concepção e desenvolvimento de aplicação que usufrua das funcionalidades do motor de regras desenvolvido.

O projecto é enquadrado nas mais recentes normas para as Redes IP de Próxima Geração, com destaque para a arquitectura IP Multimedia Subsystem e servidores de aplicações JAIN SLEE. Dentro da PT Inovação o projecto está enquadrado na plataforma IP-JIB, anteriormente identificado por eXtensible Multimedia Application Server (XMAS).

1.3 Organização do documento

O documento está organizado da seguinte forma:

- Capítulo 2 - *Contexto Institucional*: Neste capítulo identificam-se aspectos relevantes do contexto institucional, nomeadamente informação geral sobre o estágio, a instituição de acolhimento e a equipa em que o aluno está integrado.
- Capítulo 3 - *Contexto Tecnológico e Científico*: Neste capítulo contextualiza-se o leitor para as várias tecnologias que são objecto de discussão ao longo deste documento.

- Capítulo 4 - *Contexto Específico do Projecto*: Neste capítulo descreve-se o projecto IP-JIB da PT Inovação no qual o aluno desenvolve o seu projecto individual, nomeadamente o PUMA.

- Capítulo 5 - *Planeamento*: Neste capítulo apresenta-se em detalhe o plano de trabalho, as metas a atingir e o processo de desenvolvimento previsto.

- Capítulo 6 - *Trabalho Realizado*: Neste capitulo descreve-se o trabalho realizado no estágio na PT Inovação no âmbito da disciplina PEI. Relata-se a integração de um motor de regras e o desenvolvimento de uma aplicação que usufrui das funcionalidades do motor de regras.

- Capítulo 7 - *Trabalho Futuro*: Neste capítulo relata-se o desenvolvimento futuro planeado, algumas ideias que poderão melhorar os produtos desenvolvidos e desenvolver novos produtos.

- Capítulo 8 - *Considerações Finais*: Por fim, neste capítulo, resume-se as tecnologias estudas e o projecto desenvolvido.

- Anexo A - *Apreciação do Estágio*: Este apêndice tem como objectivo enquadrar o projecto desenvolvido no âmbito da disciplina de Projecto de Engenharia Informática. Consequentemente, é feita uma apreciação pormenorizada do estágio.

Capítulo 2

Contexto Institucional

Neste capítulo identificam-se aspectos relevantes do contexto institucional, nomeadamente informação geral sobre o estágio, a instituição de acolhimento e a equipa em que o aluno está integrado.

2.1 Descrição do Estágio

Este relatório foi escrito no âmbito da disciplina de Projecto em Engenharia Informática (PEI) do Mestrado em Engenharia Informática, leccionado na Faculdade de Ciências da Universidade de Lisboa. Esta disciplina tem uma duração anual e pressupõe a realização de 48 unidades de crédito ECTS[1], estando prevista uma carga horária de 28 horas semanais. No decorrer deste estágio a carga semanal média foi de 36 horas.

O estágio decorreu entre 1 de Setembro de 2006 e 31 de Maio de 2007 no pólo de Lisboa da instituição de acolhimento Portugal Telecom Inovação, situada no parque tecnológico Tagus Park (Oeiras).

A orientação pedagógica esteve a cargo do Professor Doutor Luís Correia do Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa e a orientação técnica a cargo do Engenheiro Paulo Chainho, colaborador da instituição de acolhimento.

2.2 Instituição

A Portugal Telecom Inovação, S.A. (PT Inovação) é a Empresa PT vocacionada para a criação de novos serviços e soluções, contribuindo para o aumento da competitividade e liderança das empresas do Grupo Portugal Telecom.

Fundada em Maio de 1999, com sede em Aveiro, a PT Inovação tem por missão promover o processo de Inovação ao nível de serviços, tecnologias e operações, através do desenvolvi-

[1] European Credit Transfer and Accumulation System, sistema europeu de caracterização da carga horária prevista para realização de uma cadeira.

mento de competências nas disciplinas e sectores do mercado das Telecomunicações e das Tecnologias de Informação. Contribuir para o desenvolvimento da Sociedade de Informação e do Conhecimento e para a criação de novas áreas de negócio a ela associadas são também objectivos da PT Inovação, cuja actividade visa garantir a diferenciação das Empresas PT face à concorrência, reforçando a sua actividade nos mercados nacional e internacional.

A PT Inovação é uma empresa moderna, cuja credibilidade assenta em mais de 50 anos de experiência tecnológica acumulada em telecomunicações. Durante décadas, os resultados da actividade influenciaram decisivamente a modernização tecnológica do Sistema Nacional de Telecomunicações a todos os níveis.

A PT Inovação detém competências em investigação aplicada, integração tecnológica e desenvolvimento de serviços e soluções, prestação de serviços de engenharia e de formação. Da lista dos principais produtos fazem parte sistemas e soluções integradas para as Redes Inteligentes (Próxima Geração/Convergência), Rede de Acesso, Soluções Multimédia e IP, Redes e Serviços Móveis, Business Intelligence, Gestão de Redes, Sistemas e Tecnologias de Informação, Engenharia de Software e processos do negócio das telecomunicações.

A PT Inovação promove a cooperação com Universidades e outros Institutos de I&D nacionais e internacionais, assumindo-se como verdadeiro agente da transferência do conhecimento para o mercado e a indústria.

Integram a PT Inovação cerca de 400 colaboradores, na sua maioria Quadros Superiores especialistas em Telecomunicações e Sistemas de Informação, com uma média etária inferior a 37 anos. Além da sede em Aveiro, a Empresa tem pólos em Lisboa e no Porto e opera ainda a subsidiária PT Inovação Brasil, São Paulo.

A PT Inovação detém os Certificados de Conformidade da APCER (Associação Portuguesa de Certificação) que lhe garantem a Certificação Global no Sistema de Gestão de Qualidade ISO 9001:2000 e a Certificação Ambiental segundo a norma ISO 4001.

2.3 Equipa de Trabalho

O projecto em que o aluno está integrado é totalmente desenvolvido na instituição de acolhimento, no departamento de Plataformas e Protocolos de Aplicação, e existe o envolvimento de dois outros departamentos, totalizando um total de 15 colaboradores.

A equipa de trabalho do projecto IP-JIB, sob orientação do Engenheiro Paulo Chainho, é constituída por cinco arquitectos de *software*[2] e um responsável pela *Quality Assurance* e *Integration Tests*[3].

Pontualmente existem reuniões de planeamento e integração com o departamento responsável pelo projecto IP-JIB, nomeadamente o departamento de Serviços e Plataformas para Redes Móveis situado em Aveiro.

[2] Alexandre Mendonça, Eduardo Martins, João Moura, Luís Barreiro e Neutel Rodrigues
[3] Toni Barata

Capítulo 3

Contexto Tecnológico e Científico

Neste capítulo o contextualiza-se o leitor para as várias tecnologias que foram estudadas no decorrer desta disciplina e que são objecto de discussão ao longo deste documento.

3.1 IP Multimedia Subsystem

O Internet Protocol (IP) já atingiu um estado de ubiquidade há muito tempo. De acordo com a Internet Society o IP é usado para interligar mais de um bilião de pessoas. O melhor da Internet é interoperabilidade em larga escala, permitindo que pessoas com diferentes terminais possam comunicar entre elas.

A expectativa de uma arquitectura IP para disponibilização de serviços multimédia foi aumentado cada vez mais. Neste contexto, surgiu a exigência por parte dos utilizadores de serviços personalizados, interactivos e multimédia em qualquer terminal e em qualquer local, criando assim novos requisitos e desafios para os operadores. Para responder aos novos requisitos, surgiu a arquitectura IP Multimedia Subsystem (IMS).

O IMS é um novo subsistema definido pelo grupo 3GPP e consiste numa arquitectura para uma infra-estrutura de rede que permite a convergência de dados, som, redes e tecnologias sobre uma rede IP.

A arquitectura do IMS foi desenhada de modo a permitir comunicações em tempo-real e serviços multimédia inteligentes (e.g., vídeo-chamadas, mensagens instantâneas, conferência, presença, push-to-talk). Esta arquitectura disponibiliza estes serviços para o utilizador final através de mecanismos como negociação e gestão de sessões, qualidade de serviço e gestão de mobilidade.

3.1.1 Arquitectura

Resumidamente, o objectivo essencial do IMS é permitir um meio seguro e fiável de terminais e aplicações comunicarem entre si. Existindo assim a facilidade de os operadores disponibilizarem múltiplos serviços aos utilizadores maximizando a utilização dos equipamentos através de uma arquitectura horizontal.

A arquitectura IMS é suportada por dois sistemas chave: o Call Session Control Function (CSCF) e o Home Subscriber Server (HSS). Na arquitectura IMS, representada na 3.1, a General Switched Telephony Network (GSTN) interliga-se com o Media Gateway (MGW) e o Media Gateway Function (MGCF).

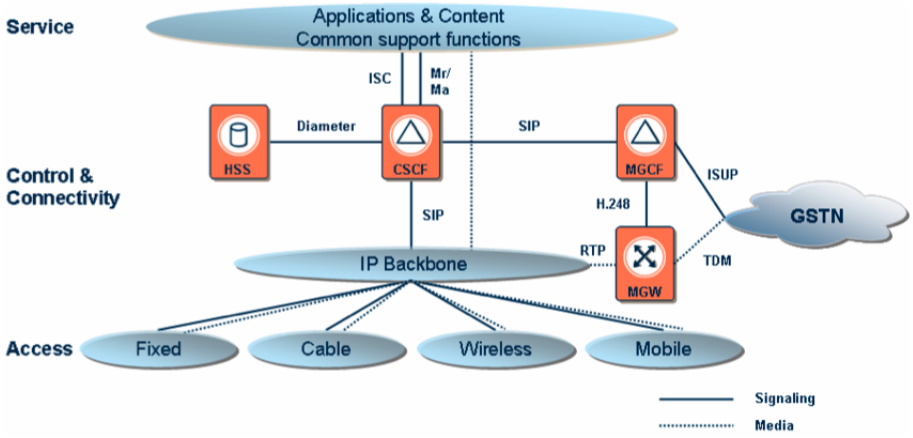

Figura 3.1: Arquitectura IMS

Call Session Control Function

O Call Session Control Function é o coração da arquitectura IMS e é usado para processar as sessões SIP. O SIP é um protocolo do controlo da camada aplicação (sinalização) para criar, modificar e terminar sessões com um ou mais participantes. A principal função do CSCF é controlar as sessões dos terminais e das aplicações na rede. O controlo de sessão inclui o encaminhamento das mensagens SIP, subsequente monitorização das sessões, comunicação com os componentes de suporte à autorização e interligação com o HSS.

Tal como ilustrada a 3.2 o CSCF poderá ter três funções distintas: Serving, Interrogation e Proxy Call Session Control Function.

O Serving-CSCF (S-CSCF) é o componente principal para o aprovisionamento das mensagens SIP e é o coração de toda a infra-estrutura IMS. Para além de ser o intermediário nas sessões SIP, em conjunto com o o P-CSCF e o S-CSCF, tem também a responsabilidade de encaminhar, traduzir e manter as sessões, interagir com outros serviços e efectuar a tarifação.

O Interrogating-CSCF é o primeiro ponto de contacto para terminais registado nessa

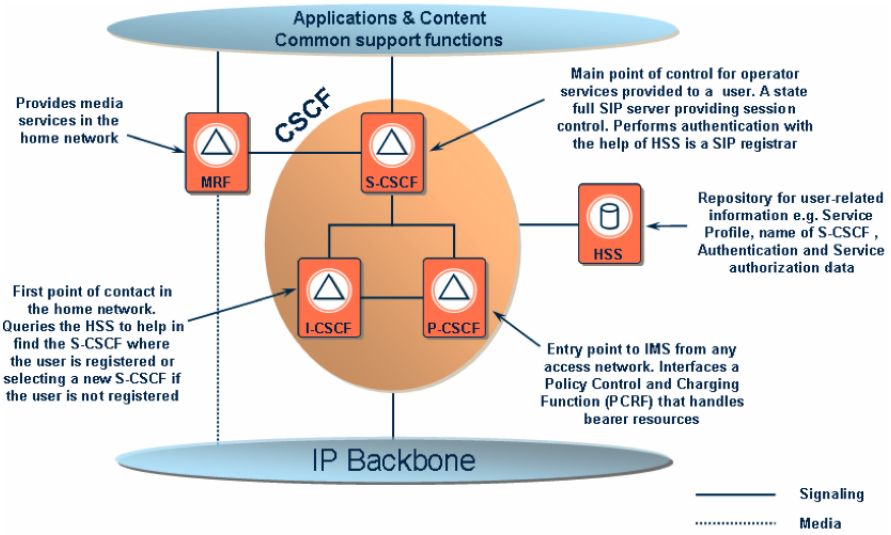

Figura 3.2: CSCF na arquitectura IMS

rede, verificando no HSS qual o S-CSCF do utilizador e encaminhando o pedido para o S-CSCF em questão.

O Proxy-CSCF (P-CSCF) é o proxy SIP e serve como primeiro ponto de contacto para os terminais na infra-estrutura IMS. A principal responsabilidade do P-CSCF é disponibilizar ao terminal uma transmissão segura de mensagens SIP, comunicação com os componentes de suporte à autorização e suporte à compressão de mensagens SIP. P-CSCF encaminha todas as mensagens recebidas para o S-CSCF ou para o I-CSCF. O P-CSCF também valida e encaminha sessões com carácter de emergência.

Home Subscriber Server

O Home Subscriber Server (HSS) consiste numa base de dados que contem a informação dos utilizadores e subscritores necessária para as entidades da rede gerirem as sessões. Tem como principais responsabilidades a identificação dos utilizadores ou entidades, autorização, autenticação, gestão de mobilidade, suporte ao estabelecimento de sessões, suporte ao aprovisionamento de serviços e suporte à autorização de serviços.

SIP Application Server

Todas as aplicações e serviços no IMS são executadas no SIP applications server. Um servidor pode estar dedicado a um único serviço ou disponibilizar vários serviços. No IMS; também é possível combinar serviços de diferentes servidores para criar uma experiência de utilizador unificada.

Os principais benefícios do SIP application server são a relativa facilidade de criação de serviços e a centralização.

Breakout Gateway Control Function

Tal como se constata na figura 3.3, o Breakout Gateway Control Function (BGCF) é responsável pelas sessões com as redes GSTN. Este componente do IMS decide também qual a rota das sessões iniciadas na infra-estrutura IMS com destino a uma rede de comutação de circuitos (redes de telefones normais ou redes moveis).

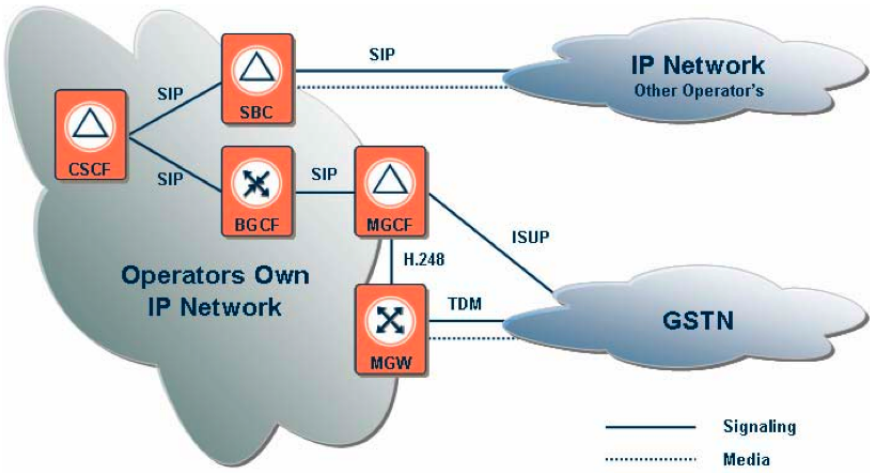

Figura 3.3: Outros componentes da arquitectura IMS

Media Resource Function

O Media Resource Function (MRF) disponibiliza conteúdos de media à rede e tem a capacidade de gerir e processar *streams* de som, vídeo e *text-to-speech* e permite ainda criar em tempo real novos conteudos multimedia. O MRF pode ser interpretado como a junção de dois componentes, o Media Resource Function Controller (MRFC) - integração com o S-CSCF - e o Media Resource Function Processor (MRFP) - funcionalidades de adaptação e criação de conteudos multimédia.

Session Border Controller

Com funcionalidades e controlo de fronteiras, na infra-estrutura IMS o Session Border Controllers (SBC) é um *gateway* que está colocado na fronteira da rede de um operador com a função de interligação à rede de outros operadores. O SBC tem também funcionalidades de tradução de endereços (endereços públicos e privados e endereços IPv4 e IPv6).

Media Gateway Control Function

O Media Gateway Control Function (MGCF) é o nó central do *gateway* para a rede PSTN (Public Switched Telephone Network) e tem como função controlar os recursos associados às sessões entre diferentes redes e diferente media (e.g., rede Time Division Multiplexing e

a rede IP). Tal como se ilustra na 3.3, o MGCF interage com o CSCF, o MGW e as redes GSTN.

Media Gateway

O MGCF controla o Media Gateway (MGW) que disponibiliza interligação entre as diferentes redes e diferente media e disponibiliza também adaptação entre os diferentes protocolos de transporte de media (e.g., RTP/UDP/IP e TDM).

3.1.2 Estandardização

Diferentes elementos e componentes na arquitectura IMS foram definidos por diferentes entidades.

O grupo 3GPP e o grupo 3GPP2 definiram os requisitos e a arquitectura de topo. A Open Mobile Aliance (OMA) está focada na definição de aplicações e serviços sobre a arquitectura. O Internet Engineering Task Force (IETF) é o responsável pelos protocolos na cama de rede.

3.1.3 Serviços

Em termos técnicos o IMS não apresenta nada de novo, os conceitos técnicos já estão muito discutidos na estandardização da arquitectura e a tecnologia já existe há muito tempo. Até há data o IMS tem existido de uma maneira pouco visível, não existindo ainda muitas redes IMS em produção. No entanto, começa a surgir a noção que o IMS proporcionará vários benefícios para o utilizador final através de novos serviços.

Vendedores, operadores e produtores estão a mostrar um maior interesse no IMS pois este irá revolucionar o mundo das telecomunicações com a criação de novos e inovadores serviços que acrescentam valor para utilizador final.

A transição da industria das telecomunicações de uma fase mais tradicional baseada em serviços de voz e de mensagens para uma nova fase com novos e excitantes serviços multimédia já começou. O serviços de voz e mensagens estão a ser complementados pela próxima geração de serviços de vídeo-chamadas, mensagens de voz e vídeo, mensagens instantâneas, conferência, presença, push-to-talk, partilha de recursos (aplicações, ficheiros, vídeos), jogos para multi-jogadores, aplicações de localização. Não se deve esquecer também o sem número de combinações possíveis com estes serviços, por exemplo um jogo para multi-jogadores com conferência.

O IMS permite também a criação de novos serviços entre as redes de telecomunicações móveis e fixas. Um exemplo deste serviço pode ser uma vídeo-chamada entre um telefone fixo e um telefone móvel.

3.1.4 Benefícios

Operadores

- Time-to-Market - A criação de novos serviços, nas infra-estruturas existentes, é um processo moroso e com elevado custo. A estandardização dos protocolos de comunicação entre os diferentes componentes da arquitectura, dos interfaces e funções disponibilizadas por serviços comuns diminui significativamente o esforço de criar e disponibilizar novos serviços ao utilizador final.

- Menor custo - A falta de estandardização no ambiente de criação de serviços resulta na criação de ambientes verticais, que para além de tornar a gestão mais difícil também duplica funcionalidades entre produtos.

  A arquitectura IMS permite um comportamento reutilizável, novos serviços usam os componentes e funcionalidades já existentes na plataforma. Os operadores apenas precisarão de uma plataforma para disponibilizar todos os seus serviços. A redução de complexidade da plataforma também reduz significativamente o custo de posse e suporte da mesma.

- Liberdade de escolha - Outra vantagem desta arquitectura é a flexibilidade de escolha entre vendedores, para componentes ou para a plataforma como todo. Deixam de existir os problemas de interoperabilidade dentro da infra-estrutura do operador!

- Integração e Interoperabilidade entre serviços - O problema da integração ou interoperabilidade entre serviços (mesmo de operadores diferentes) já não existirá. Os serviços serão integrados de forma transparente e com poucos custos associados, mais uma vez devido à existência da estandardização.

Utilizadores

- Serviços - O utilizador terá à sua disponibilidade uma panóplia de novos e inovadores serviços.

- Multimédia - O utilizador deixa de estar limitados aos tradicionais serviços de voz ou de mensagens escritas. Existe a possibilidade de novos meios de comunicação e várias sessões em simultâneo.

- Entidade única - O utilizador apenas terá de utilizar uma única identidade para se identificar no operador (qualquer que seja a rede, serviço ou localização).

- Comunicações personalizadas - O utilizador poderá definir através de que recurso pretende ser contactado, de acordo com o seu contexto.

- Roaming - O roaming sempre foi um tópico com diversas complicações a nível de integração entre operadores. A arquitectura define o comportamento para o roaming,

permitindo ao utilizador em roaming usar todos os serviços disponibilizados pela sua operadora.

- Interligação móvel fixo - As duas redes tenderão a convergir e a experiência do utilizador tenderá a ser unificada qualquer que seja a rede onde se encontra ou se pretende ligar.

3.2 Servidor de Aplicações

Um servidor de aplicações é por definição uma plataforma para execução de aplicações e serviços. Adicionalmente, a plataforma pode disponibilizar à aplicação lógica de negócio ou de acesso à informação. O principal benefício de um servidor de aplicações é a facilidade que proporciona para criar aplicações, disponibilizando funcionalidades e APIs normalmente estandardizadas.

Para as aplicações que executa, esta plataforma providencia um contentor que implementa serviços para garantir certas propriedades fundamentais numa aplicação, como a persistência, segurança ou fiabilidade, entre outras. Desta forma quem desenvolve uma aplicação para um servidor deste tipo concentra-se apenas nas funcionalidades que esta vai fornecer.

Apesar de o termo "servidor de aplicações"se referir a todas as plataformas, este termo tem sido muito associado à plataforma Sun JEE e a plataformas de serviços de comércio electrónico ou de sistemas de gestão de conteúdos.

3.2.1 JAIN SLEE

Java APIs for Integrated Networks (JAIN) é uma actividade da Java Community Process (JCP)[1] que tem como objectivo desenvolver APIs para criar serviços de telecomunicações.

JAIN é uma iniciativa do movimento nas telecomunicações para criar serviços, analogamente à Internet. A existência de padrões abertos, deverá resultar numa crescente participação de criadores de serviços resultando em mais, melhores e mais personalizados serviços nas telecomunicações.

O principal objectivo das JAIN APIs é abstrair a rede onde opera o serviço, para que os serviços possam ser criados independentemente da rede onde vão operar (seja PSTN ou as redes de próxima geração).

A iniciativa JAIN resultou na criação de cerca de 40 APIs, que se encontram em vários estados de estandardização, que vão desde a especificação de protocolos de rede (e.g., SIP e TCAP) até APIs mais abstractas tal como *call control and charging* ou mais especificas como o Service Logic Execution Environment (SLEE).

[1] Processo com participação da comunidade para desenvolver e rever especificações da tecnologia Java

O SLEE é uma especificação de API que define uma *framework* para o desenvolvimento de serviços para telecomunicações. A primeira versão foi desenvolvida na comunidade JCR sobre o Java Specification Request (JSR) 22 e foi liderado por David Ferry da empresa Open Cloud (Nova Zelândia) e por Swee Lim da empresa Sun Microsystems. A versão 1.1 encontra-se no estado final sobre o JSR 240 e foi liderada por David Ferry da empresa Open Cloud. Nas especificações estão envolvidas muitas outras empresas das quais se destacam a IBM, Motorola, Nokia, NTT Corporation, Red Hat, Siemens e Vodafone.

A empresa Open Cloud desenvolveu e disponibilizou à comunidade uma referência de implementação SLEE baseada na arquitectura Enterprise Java Beans (EJB)[2].

Especificação

Os quatro elementos básicos do SLEE são os *Resource Adaptors* (Adaptadores de Recursos), *Events* (Eventos), *Activity Contexts* (Contexto de Actividades) e o *Runtime Environment* (Ambiente de Execução) que inclui os *Service Building Blocks* (Blocos de Construção de Serviços). A Figura 3.4 ilustra os elementos na arquitectura e a relação entre estes.

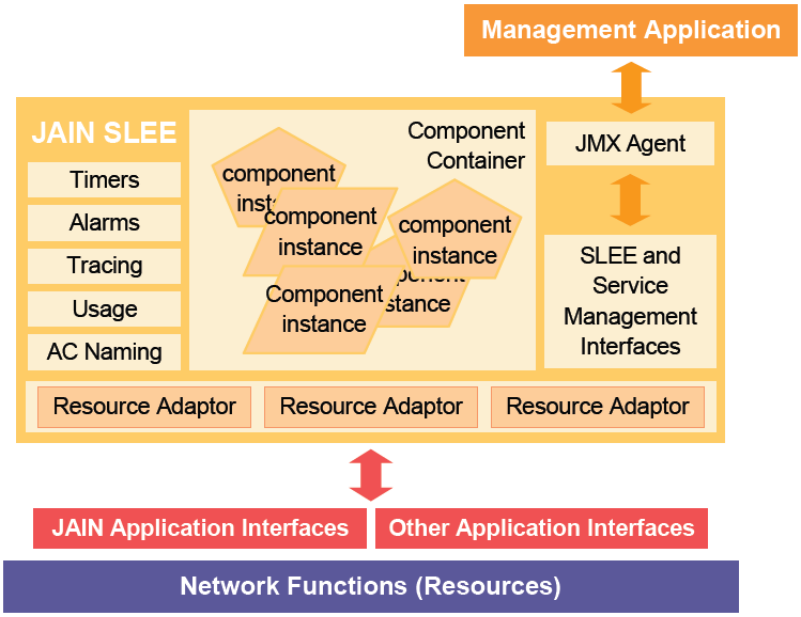

Figura 3.4: Aquitectura JAIN SLEE

Os *Resource Adaptors* (RAs) são responsáveis por comunicar com entidades externas ao sistema SLEE. Os RAs podem enviar e receber eventos com o exterior (comunicação bidireccional). Quando recebem um evento do exterior, os RAs submetem o evento recebido para os *Activity Contexts* (ACs) como um objecto representativo desse evento. Os *Service Building Blocks* (SBBs) que se encontram no ambiente de execução têm interfaces com os ACs, que são usados para entregar o objecto representativo do evento ao SBB.

[2]Tema estudado neste documento

O AC é uma entidade lógica do sistema que recebe e reencaminha os eventos recebidos para os componentes SBB. O reencaminhamento é efectuado pelo componente do SLEE *event router*. Os eventos podem ser duplicados e reencaminhados para vários SBBs.

Em complemento aos elementos mencionados, a especificação SLEE define também *Facilities* SLEE. A responsabilidade destes elementos é disponibilizar aos criadores de RAs ou SBBs ferramentas que interagem com o próprio SLEE e que permitem executar tarefas comuns a todos (e.g., alarmes de estado do SLEE ou ferramentas de temporização).

Resource Adaptors

Os RAs comunicam com as entidades externas ao sistema SLEE (e.g., componentes de rede, protocolos de comunicação, directórios ou base de dados). De acordo com a especificação SLE, um RA é uma implementação especifica de um vendedor de um *Resource Adaptor Type*. Uma instancia de um RA numa plataforma SLEE é identificado como uma entidade SLEE.

O *Resource Adaptor Type* (RA Type) declara os tipos de eventos que podem ser recebidos e disparados e as actividades que as instancias do RA vão criar. Quando o RA introduz uma actividade no SLEE, este tem também de disponibilizar o objecto representativo do evento, o tipo de evento e a especificação da actividade. Este comportamento e especificação apenas está definido na versão 1.1 da especificação SLEE.

Events

O objectos de eventos contêm a informação que é passada de uma entidade SLEE para outra. Apenas as entidades SBB podem consumir e produzir eventos dentro do SLEE. Dentro do SLEE, as entidades de RAs, o próprio contentor SLEE e as *Facilities* apenas podem produzir eventos.

Cada evento é representado por um objecto de evento e um tipo de evento. O tipo de evento determina como o contentor SLEE irá reencaminhar o evento (que elementos SBB irão receber o objecto de evento).

Os eventos produzidos estão associados a ACs. Apenas SBBs associados aos ACs onde o evento é produzido recebem o evento. Caso esteja definido como um evento inicial, o SLEE primeiro cria um objecto SBB que associa a um AC onde vai produzir o evento.

Activity Context

Os conceitos associados a actividades consistem em duas entidades lógicas, actividade e contexto de actividade, as suas representações lógicas são os objectos de actividade e os objectos de interface de contexto de actividades, respectivamente.

Uma actividade representa um conjunto de eventos associados entre si. A representação lógica desta entidade é o objecto de actividade, criado por RAs ou SLEE *Facilities*.

Um contexto de actividade representa a actividade dentro do SLEE e contém informação que as entidades SBB queiram partilhar entre elas. As entidades SLEE podem aceder à informação guardada em contextos de actividades através do objecto de interface de contexto de actividades.

Uma entidade SBB pode usar o objecto de interface de contexto de actividades genérico ou estender o interface para definir a informação que vai ser partilhada com as outras entidades.

Os objectos de actividades são tipicamente criados por eventos de rede. Os RAs escutam eventos na rede e criam o objecto de actividade. Estes objectos são associados ao contexto de actividade. O SLEE é responsável por entregar os eventos aos objectos SBB. Vice versa, um objecto SBB pode aceder a um objecto de interface de contexto de actividade e obter o objecto de actividade actual. Para produzir eventos para os RAs, e consequentemente para entidades externas ao SLEE, os SBBs produzem eventos associados a um objecto de interface de contexto de actividades.

Runtime Environment e *Service Building Blocks*

De acordo com a especificação, o ambiente de execução SLEE tem de disponibilizar as APIs especificadas aos componentes SBB. Apenas as APIs são especificadas, ficando responsabilidade do criador da implementação SLEE disponibilizar outras APIs.

Actualmente, uma implementação SLEE tem de disponibilizar as seguintes APIs: Java 2 Platform, Standard Edition, v1.3 APIs (e versão 1.4 na especificação 1.1); Java Naming and Directory Interface v1.2 Standard Extension; Java Management Extensions v1.2; Java API for XML Processing v1.0; Java Database Connectivity v1.0 (e versão 2.0 na especificação 1.1).

A classe abstracta SBB faz parte do componente SBB (que também inclui interfaces locais em descritores de *deployment*) e contém a lógica de processamento de eventos definida pelo criador. O criador define uma classe abstracta SBB para cada componente SBB. O ambiente de execução é responsável por criar as entidades SBB, para isso implementando estas classes abstractas. Este processo pode ser comparado à criação de componentes EJB. Tal como na especificação EJB, o ambiente de execução é responsável por implementar os métodos abstractos e por criar instancias que processem os eventos recebidos.

Cada classe abstracta SBB implementa a classe concreta javax.slee.Sbb. O métodos concretos contêm a lógica de aplicação do componente. Os métodos abstractos servem para produzir eventos, gerir persistência no contentor, criar relações pai-filho entre SBBs, gerir perfis e aceder ao contexto das actividades. O contentor utiliza introinspeção e dados no descritor de *deployment* para criar o componente.

Implementações

Considera-se uma implementação JAIN SLEE uma plataforma que cumpra todos os requisitos da especificação JAIN SLEE.

- Red Hat Mobicents - A empresa Red Hat adquiriu recentemente a única implementação de código aberto da especificação JAIN SLEE, a plataforma Mobicents.

  O Mobicents disponibiliza à indústria um robusto modelo de componentes e um ambiente para execução de serviços de telecomunicações. Complementa as plataformas JEE e permite a convergência de som, imagem e dados nos serviços de próxima geração.

  Apesar da plataforma JAIN SLEE poder ser implementado de raiz, a equipa decidiu optar por desenhar a sua implementação no topo do servidor de aplicações JEE da JBoss (adquirida também pela Red Hat), outro projecto de código aberto. Desta forma foi possível ter uma implementação completamente funcional encurtando o esforço e tempo necessários. O Mobicents herda do servidor de aplicações JEE os mecanismos para gestão de *threads*, persistência, *clustering* e ciclo de vida dos componentes.

  Esta plataforma posiciona-se como uma plataforma com alto desempenho para disponibilização de aplicações no contexto das redes de próxima geração e do IMS.

  Existe a possibilidade de monitorização e gestão da plataforma através da especificação JAIN SLEE JMX e interfaces Simple Network Management Protocol (SNMP). Estas funcionalidades permitem aos operadores de telecomunicações optarem pela platform para os seus sistemas Operations Support Systems (OSS) e Network Management Systems (NMS).

  Para além da indústria das telecomunicações, o Mobicents é a plataforma indicada para uma variedade de problemas do domínio Event Driven Architecture (EDA).

- OpenCloud Rhino - O núcleo da plataforma Rhino é um contentor JAIN SLEE desenhado e optimizado para aplicações baseadas em eventos. O Rhino é o primeiro produto a cumprir integralmente a especificação JAIN SLEE 1.0 e implementa todas as funcionalidades incluídas na especificação (incluindo as opcionais). O Rhino foi concebido de forma a cumprir os requisitos de serviços das redes de próxima geração.

  O contentor JAIN SLEE Rhino segue os padrões associados e é um ambiente de execução de serviços que suporta, gere e executa os novos serviços associados ao mercado das telecomunicações.

  A plataforma Rhino é também um servidor de aplicações desenhado para permitir uma fácil integração numa infra-estrutura IMS. Este *design*, apresenta aos vendedores e integradores varias vantagens em termos de avanços tecnológicos e oportunidades comerciais.

  Em complemento, este contentor é disponibilizado pela empresa que lidera as especificações JAIN SLEE e, por isso, se encontra em vantagem em comparação com qualquer

outra implementação.

Aspectos Chave da Especificação

De seguida destacam-se as vantagens mais interessantes do JAIN SLEE:

- Fiabilidade Elevada disponibilidade é um dos principais objectivos dos serviços de telecomunicações e só pode ser conseguida se o tempo para detectar e recuperar de uma falha seja o mais curto possível. Historicamente a indústria previne-se para falhas de *hardware* mas as falhas com origem de *software* afectam significativamente a disponibilidade dos serviços (40%).

  A complexidade dos sistemas de *software* está a aumentar e uma consequência directa é o aumento da complexidade dos serviços. O custo para criar serviços que garantam uma disponibilidade elevada não deve ser subestimado.

  Assim sendo, em conjunto com os modelos existentes que permitem lidar com as falhas de *hardware*, a arquitectura do SLEE foi desenhada tendo em consideração este novo requisito.

- Serviços de Próxima Geração A necessidade de introduzir novos serviços nas redes de telecomunicações nos últimos anos e a explosão da Internet originou o desejo de criar serviços que usufruíssem destas duas infra-estruturas em simultâneo.

  Existem porém requisitos que foram objecto de estudo aquando do desenvolvimento do SLEE:

  - Portabilidade de serviços - O desenvolvimento de serviços não deverá continuar a estar restringido a interfaces proprietárias;
  - Convergência de redes - O desenvolvimento de serviços deverá ser independente da rede onde vai operar (e.g., PSTN ou baseada em comutação de pacotes);
  - Acesso Seguro à Rede - Serviços dentro e fora da rede principal de telecomunicações devem poder aceder de forma segura aos recursos da infra-estrutura;

- Desenvolvimento de Aplicações De modo a satisfazer os requisitos de desempenho e disponibilidade, o programador tem acesso a APIs que permitem que a lógica da aplicação:

  - seja simples;
  - continue a funcionar aquando de falhas de *harware* e de processo;
  - execute independentemente do nó do *cluster* em que se encontra;
  - execute de forma concorrente;

  Estes requisitos do modelo de programação para desenvolver aplicações e os requisitos do servidor de aplicações foram desenvolvidos tendo em conta os objectivos supracitados.

Componentes da Aplicação:

- Confiar no modelo de faltas de modo a simplificar o comportamento da aplicação em caso de falhas;
- Não existe a necessidade de gerir a concorrência;
- O estado da aplicação pode ser replicado;
- Existem formas claras de sincronização de estado;

Servidor de Aplicações:

- Através do modelo de faltas, migra os componentes e dados das aplicações entre os nós do *cluster*;
- Gere a concorrência de execução dos componentes das aplicações;
- Garante que faltas numa aplicação não afectam as restantes aplicações no servidor;
- Permite actualização de aplicações sem que seja necessário terminar ou suspender a execução destas;

Especificação SLEE:

- Modelo de eventos assíncronos ;
- Gestão do estado dos componentes pelo contentor SLEE;
- *Garbage collection* dos componentes da aplicação gerido pelo contentor SLEE;
- Interacção por eventos com componentes exteriores;
- Gestão do ciclo de vida do servidor, serviços e estado do contentor;
- Actualização de serviços gerida pelo contentor de forma a não se perder nenhuma actividade;
- Independência da rede, tecnologia e componentes exteriores através da arquitectura de adaptação de recursos (*resource adaptors*);

- Modelo de Programação
  - Simples e orientado a objectos;
  - Suporte ao desenvolvimento de componentes de aplicação por terceiros;
  - Servidor de aplicações gere o estado da aplicação;
  - Transaccional;
  - Suporte a unidades de trabalho independentes;
  - Independente da rede de telecomunicações;

- Estandardização APIs abertas e estandardizadas escondem a complexidade das redes no nível aplicação. O uso de sinalização e protocolos de gestão estandardizados são a chave para permitir a flexibilidade e criatividade necessária para os serviços de próxima geração.

- Independência da Rede Para minimizar o custo de substituição de *hardware* e *software*, a especificação foi definida de modo a abstrair-se da rede. A integração deste contentor e de novas aplicações pressupõe uma integração inicial com as redes existentes. A integração é possível através da *framework* de adaptadores de recursos.

- Portabilidade Os operadores e os vendedores de equipamento de telecomunicações têm as suas preferencias em termos de *hardware* e sistemas operativos. As aplicações criadas para um ambiente JAIN SLEE têm de ter as seguintes características:

  – Executar em qualquer plataforma que cumpra a especificação sem ser necessário alterações à aplicação;

  – Não existir a necessidade de alterar código para funcionar em diferentes plataformas JAIN SLEE;

  – Têm de ser isoladas de arquitecturas *hardware* ou APIs de nível sistema;

3.2.2 JEE

As organizações precisam estender os seus produtos, reduzir os custos e reduzir o time-to-market para disponibilizar serviços que correspondam às necessidades dos seus clientes, parceiros, empregos ou fornecedores.

Tipicamente, as aplicações que disponibilizam estes serviços têm de combinar sistemas de informação existentes com novas funcionalidades. Estes serviços precisam:

- Apresentar uma alta disponibilidade;

- Garantir os níveis de segurança necessários para manter a privacidade dos utilizadores e integridade da informação;

- Fiáveis e escaláveis, para não correr o risco de se perder transacções;

Normalmente estes serviços são concebidos como aplicações distribuídas separadas em camadas, apresentação ao utilizador, acesso a recursos e dados e uma camada intermédia onde se define a lógica de negócio da aplicação. Nesta camada intermédia implementa-se novos serviços que utilizam funcionalidades comuns e acedem da mesma forma à informação.

Uma plataforma que respeite a especificação JEE, reduz o custo e complexidade de desenvolvimento de serviços, resultando numa maior facilidade de criar novos serviços ou evoluir os actuais. A especificação JEE atinge este objectivo através da disponibilização dos seguintes elementos:

- Modelo de Programação de Aplicações JEE - Modelo de programação estandardizado para desenvolver serviços numa plataforma JEE;
- Plataforma JEE - Plataforma estandardizada para manter e gerir aplicações;
- Testes de Compatibilidade para Plataformas e Aplicações JEE - Um conjunto muito alargado e preciso de testes garante que tanto a plataforma como as aplicações desenvolvidas cumprem a especificação;
- Implementação de Referência JEE - Disponibilização de uma referência de implementação de uma plataforma JEE para demonstrar as capacidades e as definições operacionais de uma potencial plataforma.

Red Hat/JBoss Application Server

A plataforma JEE está madura o suficiente para facilitar a criação de novas oportunidades de baixo custo nesta área. O Red Hat/JBoss Application Server, para além de ser de código aberto e contar com a contribuição da comunidade, é um dos servidores de aplicações mais populares.

A Red Hat/JBoss combinou o melhor que o *open source* tem com o melhor do *software* comercial, numa metodologia a que a empresa chama de Professional Open Source. Esta metodologia consiste em adaptar as vantagens do desenvolvimento em código aberto com um suporte (comercial) de excelência, complementado com acções de formação e serviços de suporte e consultoria.

Este servidor de aplicações é um elemento chave numa infra-estrutura informática de baixo custo. Esta infra-estrutura engloba também o sistema operativo Linux, o servidor Web Apache e a base de dados MySQL. O Red Hat/JBoss AS permite uma plataforma JEE com um imbatível custo zero, permitindo uma flexível e escalavel infra-estrutura sem os proibitivos preços das licenças. Sendo assim, é possível manter os custos em *software* (e.g., formação) ao mesmo tempo que se aumenta a capacidade da infra-estrutura.

Uma tecnologia chave desta plataforma é o JBoss AS Clustering que, sem necessitar de qualquer modificação às aplicações Java, permite configurar uma grelha de máquinas para trabalharem em *cluster*.

Os programadores podem aumentar a sua produtividade e simplicidade de código ao usar este servidor de aplicações (com todas as outras tecnologias). Ao contrário de outros servidores de aplicações que limitam o conhecimento do programador à especificação pública do servidor, o Red Hat/JBoss AS é baseado em padrões e com uma filosofia de *design* e implementação transparente. Significa assim que os programadores podem focar-se na lógica da sua aplicação enquanto usufruem de todas as vantagem deste servidor.

Em suma, o Red Hat/JBoss AS é um dos mais populares e produtivos servidores de aplicações, apresentando o menor custo em comparação com os seus concorrentes. é um dos

componentes essenciais para uma infra-estrutura *open source*, em conjunto com o sistema operativo Linux, servidor Web Apache e a base de dados MySQL.

EJB, Servlet e JSP

Para desenvolvimento de páginas Web em plataformas Java existem três tecnologias chave que se descreve de seguida.

Enterprise JavaBeans (EJB) é uma especificação Java API para a plataforma JEE. A especificação detalha como um servidor de aplicações pode disponibilizar a objectos no lado do servidor, conhecidos como JavaBeans, funcionalidades de conexão remota (CORBA), persistência, transacções, controlo de concorrência, eventos usando Java Messaging Service (JMS), serviços de nome e directório, segurança e instalação no servidor de aplicações.

A API Java Servlet permite ao programador adicionar conteúdo dinâmico a um servidor Web usando a plataforma Java. O conteúdo gerado é normalmente HTML, mas pode ser outro tipo de dados como XML ou mesmo dados binários. Servlet é a alternativa Java a tecnologias como o CGI ou ASP para conteúdo Web dinâmico. Esta API já disponibiliza todas as funcionalidades das tecnologias alternativas como por exemplo controlo de sessões (e.g., HTTP Cookies).

Java Server Page (JSP) é a tecnologia para controlar o conteúdo e aparência de uma página Web através do auxilio de Servlets. JSP é directamente comparável com a tecnologia da Microsoft ASP. Na plataforma Java um programa é executado no servidor Web que define o conteúdo da página.

3.3 Regras

Linguagens de programação baseadas em regras e sistemas baseados em regras sempre tiverem um papel muito importante na história da ciência da computação e na evolução da informática. Desde os Sistemas Especialista às base de dados dedutivas, a teoria e a prática da inferência automática baseada em representações simbólicas teve uma história muito rica e continuará a ser uma tecnologia chave no futuro.

Devido às inovações possibilitadas pela Internet e, mais recentemente, a Web Semântica, existem cada vez mais oportunidades de estudo nesta área. As regras são uma peça fundamental!

Uma regra é uma representação de uma acção que ocorrerá caso se verifiquem certas condições. A regra é tipicamente dividida em duas partes: condições e acções. Quando a condição ou conjunto de condições são cumpridas, a acção é executada.

Existem várias formas de representação de regras. Neste estudo aborda-se a representação através de linguagens declarativas.

3.3.1 Programação e Linguagens Declarativas

À programação declarativa estão normalmente associados dois significados distintos, ambos de uso corrente.

De acordo com uma definição, um "programa"é declarativo se descreve "o que algo é"ao invés de "como algo é". Por exemplo, uma página HTML é considerada declarativa porque descreve o que deve conter (texto, imagens, etc.) e não como a apresentação da página deve ser composta. Esta é uma abordagem diferente das linguagens imperativas que explicitam um algoritmo para atingir um objectivo. Nas linguagens declarativas explicita-se o objectivo e a implementação do algoritmo fica a cargo de alguma aplicação. Na linguagem SQL, descreve-se as propriedades e os dados que se deseja obter e não o processo de extrair os dados da base de dados.

De acordo com a segunda definição, um programa é "declarativo"se foi escrito puramente numa linguagem de programação funcional, numa linguagem de programação lógica ou numa linguagem de programação com restrições. A expressão "linguagem declarativa"é por vezes usada para descrever estes tipos de programação como um grupo.

No entanto, estas duas definições sobrepõem-se. Em particular, a linguagem de programação restrita e, não tão visível, a linguagem de programação lógica focam-se na descrição das propriedades da solução desejada, não especificando a implementação do algoritmo que calcula a solução. Todavia, a grande maioria das linguagens de programação lógicas e as linguagens de programação restritas têm funcionalidades de descrição de algoritmos e detalhes de implementação, deixando de ser tão estritos como a primeira definição.

A linguagem de programação declarativa é muito associada a este modelo de programação uma vez que é o meio utilizado na programação declarativa para descrever o problema.

No entanto, uma linguagem de programação declarativa pode ser considerada declarativa de vários pontos de vista. Prolog é declarativa pois o programador simplesmente define relações e efectua "questões"sobre essas relações, sem ser necessário definir como computar a resposta. Linguagens de programação funcional também podem ser consideradas declarativas pois associam relações entre *input* e *output*, sem ser necessário definir uma ordem de avaliação de operações. Linguagens de programação orientadas a dados (e.g., SQL) são consideradas declarativas pois não se define como obter os dados mas sim quais os dados a obter.

A definição para uma linguagem de programação declarativa é quase sempre dúbia. Por exemplo, o Prolog pode ser usado para definir algoritmos de computação através de regras recursivas ou do operador de corte! A linguagem SQL também pode não ser considerada declarativa devido aos comandos INSERT e DELETE (que dependem da ordem de execução).

Apesar de vários esforços, ainda não existe uma linguagem declarativa estandardizada para expressar regras que possam ser interpretadas por um "compilador"ou motor de regras. Existe assim a necessidade de se especificar uma tecnologia de representação de regras que seja independente de metodologias e processos. No entanto, a regra normalmente é

estruturada em duas partes usando Lógica de Primeira Ordem para a representação do conhecimento, como se pode observar na Especificação 1.

Especificação 1 Exemplo de uma regra em Linguagem Declarativa

IF

- condicoes

THEN

- accoes

De seguida analisam-se algumas tentativas de resolução do problema de estandardização de representação de regras através de uma linguagem declarativa. Estas linguagens ainda não estão a ser adoptadas pelos criadores de motores de regras mas prevê-se que num futuro próximo isso venha a acontecer.

W3C Rule Interchange Format

O objectivo do grupo de trabalho do W3C para o formato Rule Interchange Format (RIF) é definir uma especificação de uma linguagem de representação de regras que seja útil no contexto actual e que seja extensível o suficiente para ser adequada à evolução da tecnologia actual.

De acordo com os requisitos e os casos de uso do formato RIF, o *design* do formato assenta numa arquitectura baseada em camadas e tendo em conta o conceito de dialecto.

O dialecto é uma linguagem de representação de regras com um sintaxe e uma semântica bem definidas. Estas semânticas têm de ser modeladas teoricamente, comprovadas teoricamente ou operacionais, nesta ordem de preferência. Alguns dialectos podem ser extensões de outros (sintaxicamente e semanticamente). Obrigatoriamente, todos os dialectos têm de estender directa ou indirectamente o dialecto principal (RIF Core).

Eventualmente, espera-se que o dialecto RIF cubra um número importante de paradigmas de especificações baseadas em regras e linguagens de programação baseadas em regras.

RMI Rule Markup Language

A Rule Markup Language (RuleML), responsabilidade do grupo Rule Markup Initiative (RMI), é um linguagem desenhada para expressar regras em XML para dedução, rescrita e futuras tarefas de transformações inferidas. O Rule Markup Initiative é um grupo de indivíduos e grupos da indústria e de universidades formado com o objectivo de desenvolver uma Linguagem de regras para Web, utilizando XML, e que permita transformações de e para outras especificações de regras e linguagens proprietárias.

Uma vez que tanto o RIF como o RuleML têm muitos objectivos comuns, existe neste momento uma pressão muito forte para se aproximar o mais possível uma linguagem da outra. O ideal seria a especificação de uma única linguagem que cumprisse todos os objectivos que as duas iniciativas delinearam.

3.3.2 Motor de Regras

Representação do Conhecimento é a área da IA que estuda como o conhecimento é representado e manipulado. Sistemas Especialista utilizam a representação do conhecimento para facilitar a codificação do conhecimento para uma base de conhecimento com o objectivo de inferir conclusões do mesmo (*reasoning*), ou seja pode-se processar os dados na base de conhecimento para inferir conclusões. Os motores de regras que utilizam uma abordagem baseada em regras para implementação do Sistema Especialista são também conhecidos como Sistemas de Produção de Regras.

O termo Motor de Regras é algo ambíguo uma vez que pode ser aplicado a qualquer sistema que utiliza regras (qualquer que seja a utilização) para aplicar sobre data e produzir um resultado. O livro "How to Build a Business Rules Engine (2004)"de Malcolm Chisholm exemplifica esta ambiguidade. O livro é na verdade sobre como construir e alterar um esquema de base de dados para permitir regras de validação de dados. Enquanto um Sistema de Produção de Regras é um tipo de Motor de Regras e também um Sistema Especialista, a validação e avaliação de expressões por um "Motor de Regras"mencionada anteriormente não é um Sistema Especialista.

Um Sistema de Produção de Regras é turing completo e foca-se na representação de conhecimento para expressar proposições e lógica de primeira ordem de uma forma concisa, declarativa e não ambígua. O cérebro de um Sistema de Produção de Regras é um Motor de Inferência capaz de escalar a um grande número de regras e factos. O Motor de Inferência combina factos e dados com regras para inferir conclusões que resultarão numa acção.

O processo de *matching* dos actuais ou novos factos com as regras é chamado de Pattern Matching, e é executado pelo Motor de Inferência. Alguns dos algoritmos mais comuns para Pattern Matching são o linear, o RETE e o LEAPS.

Por fim, os Sistemas de Gestão de Regras de Negócio acrescentam um valor adicional aos motores de regras ao disponibilizar um sistema de criação, gestão, instalação, colaboração e análise de regras ao utilizador final.

Porquê usar?

Tenta-se agora elucidar o leitor para algumas das dúvidas mais frequentes em relação ao uso ou não de um motor de regras.

- Quais são as vantagens de um motor de regras?
  - Programação Declarativa - Os motores de regras permitem descrever "O que fazer ?"e não "Como o fazer?".

    A principal vantagem do uso de um motor de regras prende-se com a facilidade de expressar soluções. Na maioria do caso estando associado a problemas complexos e consequentemente com uma maior complexidade na representação da solução (as regras são mais fáceis de escrever e ler do que código funcional).

Os sistemas de regras são capazes de revolver problemas muito complexos, apresentando a explicação de como a solução foi derivada e cada decisão/passo até se encontrar a solução (algo que não se verifica noutros sistemas de inteligência artificial como as redes neuronais ou mesmo o cérebro humano).

- Separação da Lógica e dos Dados - Os dados estão representados nos objectos de domínio e a lógica está representada nas regras.

  Este é um princípio que quebra o conceito de programação orientado a objectos, que poderá ser considerado uma enorme vantagem ou uma desvantagem dependendo do ponto de vista. A lógica de regras poderá ser mantida e alterada no futuro sem que para isso seja preciso adaptar a representação dos dados.

- Velocidade e Escalabilidade - Os algoritmos RETE e LEAPS (assim como todos os descendentes) disponibilizam formas muito eficientes de fazer *matching* de regras de acordo com os objectos de domínio. Estes algoritmos são especialmente eficientes quando existem computações repetitivas (os algoritmos recordam os *matches*) anteriores.

- Centralização de Conhecimento - Ao se usar regras, cria-se um repositório de conhecimento que, visto não estar estritamente conectado a um domínio, pode ser reaproveitado. As regras podem ser escritas de uma forma simples (não comprometendo a eficiência) que até podem ser aproveitadas para documentação.

- Integração de Ferramentas - Hoje em dia os motores de regras integram-se com ferramentas de gestão, validação e assistência à construção de regras. Existem também vários motores de regras com ferramentas de auditorias e de depuração.

- Regras "Naturais- Ao se criar modelos de objectos e linguagens especificas de domínio, é possível representar regras muito próximas da linguagem natural. Existe a possibilidade de definir uma linguagem especifica do domínio para facilitar pessoas não técnicas e especialistas de um domínio especifico a criar e gerir as regras.

- Quando se deve usar um motor de regras?

  A resposta mais comum para esta questão é "quando não existe através da programação tradicional uma solução satisfatória para resolver o problema". A razão para não existir uma solução satisfatória poderá ser uma das seguintes:

  - A representação do problema é complicada. O problema em questão poderá não ser muito complexo mas não existe uma forma fácil de representar o problema.

  - O problema está para além de qualquer solução baseada num algoritmo obvio. Se é um problema complexo para resolver ou se o problema não é completamente percebido.

  - A lógica é alterada com muita frequência. Mesmo que a lógica seja muito simples (não terá obrigatoriamente de ser), se muda com muita frequência. Em muitas organizações existem muitas versões de certas aplicações e num curto espaço de tempo. Um motor de regras pode permitir a agilidade necessária e de uma forma segura.
  - Existem muitos especialistas de domínios, mas são todos não técnicos. Muitos domínios de negócio são tão complexos que só especialistas nessa área conseguem representar a solução para um problema, infelizmente quase todos estes especialistas não são técnicos. Com as linguagens especificas do domínio é possível uma fácil adaptação destes especialistas à representação do problema de uma forma lógica e associada a uma motor de regras.

  Nas aplicações modernas de programação orientada a objectos, um motor de regras será a chave para a parte de lógica de negócio (muitas vezes a mais complexa de representar). Este é o inverso do conceito de programação orientada a objectos onde a lógica de negócio é encapsulada nos próprios objectos que representam a informação. No entanto, não se defende o esquecimento de todas as praticas de programação orientada a objectos. Nas aplicações de mundo real a lógica de negócio é apenas uma parte da aplicação.

  Se simplesmente o problema é demasiado complicado e não existe na programação orientada a objectos algoritmos ou padrões para o representar, um motor de regras poderá também ser uma boa opção.

- Quando não se deve usar um motor de regras?

  Os programadores não se podem esquecer que o motor de regras é apenas uma parte da complexidade de uma aplicação ou solução. Os motores de regras não são desenhados para tratar de *workflows* ou processos de trabalho.

  Uma vez que os motores de regras são dinâmicos, por vezes são adaptados a vários problemas da informática. Há no entanto que salientar que, apesar do fantástico novo mundo de oportunidades que os motores de regras proporcionam, não se pode esquecer que a principal razão de existência do motor de regras é a possibilidade de representar lógica através de uma linguagem declarativa.

Representação do conhecimento

As regras são escritas usando lógica de primeira ordem. Emil Leon Post[3] foi o primeiro a desenvolver um sistema de inferência expresso em símbolos e, como consequência disso, foi capaz de provar que qualquer sistema lógico (incluindo matemático) pode ser expresso num sistema deste tipo.

[3](1897-1954) Matemático e Lógico

Uma proposição é uma afirmação que pode ser classificada como verdadeira ou falsa. Se a verdade pode ser determinada apenas da afirmação então esta é classificada de "afirmação fechada". Em termos de programação isto é uma expressão que não refere variáveis: "10 == 2 * 5".

Expressões da qual a avaliação depende de uma ou mais variáveis (factos) são "afirmações abertas", uma vez que não se consegue determinar a veracidade da afirmação até que as variáveis possam ser avaliadas tal como se pode constatar na Especificação 2.

Especificação 2 Afirmação Aberta

```
País.continente == "Europa"
```

Na linguagem SQL, podemos considerar o retorno das expressões como as acções da regra, onde as condições são representadas pela *query* SQL.

Especificação 3 *Query* SQL

```
SELECT * FROM País WHERE País.continente == "Europa"
```

Para cada linha retornada, o motor inferiu que os países estão no continente Europeu. O diagrama na Figura 3.5 ilustra como a expressão SQL e a tabela Países são representadas em termos de Motores de Inferência.

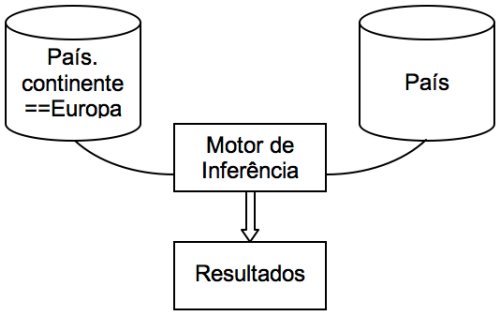

Figura 3.5: SQL no contexto dos Motores de Inferência

Em Java pode-se então afirmar que a proposição mais simples é do tipo "variável operador valor". Este tipo de proposições podem ser combinadas conjuntivamente ou disjuntivamente, logicamente representadas como "&&"e "||".

A Especificação 4 ilustra uma afirmação composta, com duas afirmações disjuntivamente conectadas.

Lógica Preposicional não é Turing completa pois não se pode expressar um critério para a estrutura dos dados, limitando os problemas que se podem definir. Lógica de Primeira Ordem estende a Lógica Preposicional com dois novos conceitos de quantificação para permitir que expressões definam estruturas, quantificador universal e quantificador existencial. O quantificador universal permite verificar a veracidade de todos os elementos, i.é, semelhante

Especificação 4 Afirmação Composta com duas afirmações disjuntivamente conectadas
País(Continente=="Europa") || País(Dimensão<="82000km")

à instrução "forall"do Java. O quantificador existencial verifica a existência de um elemento (que ocorre pelo menos uma vez).

Red Hat/JBoss Rules

O Red Fat/JBoss Rules é um motor de regras de código aberto e baseado em padrões.

Disponibiliza um sistema de gestão de regras de negócio (BRMS) para acesso, criação e gestão de regras, diferentes versões de regras e sistema de *backup*. O BRMS permite a análise de negócio e auditoria de regras na infra-estrutura instalada. Muito importante em plataformas complexas e domínios específicos com regras muito elaboradas. Este motor de regras suporta também várias linguagens de representação de regras e tabelas de decisão (ficheiros tipo Excel com representação das regras), tornando-o num produto flexível para se adoptar às novas oportunidades de negócio.

Das principais características do Red Hat/JBoss Rules destacam-se:

- Motor de Regras - Implementação de uma variante do algoritmo RETE com um elevado desempenho de indexação e optimização. Possibilidade de alteração dinâmica de regras (adicionar e remover) em tempo de execução. Suporte para regras temporais que são avaliadas de acordo com certas restrições ou num certo período de tempo.

- Drools[4] Rule Language - Suporta todas as funcionalidades básicas das linguagens de representação de regras já mencionadas e utiliza objectos Java para representação de factos, condições e funções (possibilitando assim uma fácil integração com as plataformas existentes). Suporte a linguagens naturais para representação de regras (linguagens especificas do domínio) definidas pelo Geração de representação de regras em DRL através de representação em ficheiros Excel e Open Office Decision Tables.

- Business Rules Management System (BRMS) - Sistema de Gestão de Regras de Negócio para auxiliar os utilizadores na gestão das regras de negócio dentro da sua infra-estrutura. Funcionalidades de persistência e gestão de regras, categorização, múltiplas versões de regras e diferentes estados para as regras ou grupos de regras.

Java Rule Engine API

A especificação Java Rule Engine API (JSR94) define uma API para motores de regras na plataforma JavaSE. A API estabelece as operações fundamentais e comuns aos motores de regras. Este grupo de operações é baseado nos casos de uso em que os clientes têm de ser capazes de executar o ciclo básico de utilização de regras num motor de regras, que consiste

[4]Identificação inicial do motor de regras

na avaliação de regras, adicionar objectos ao motor, iniciar a avaliação de regras e obter os resultados.

O primeiro *input* do motor de regras é uma colecção de regras a que se dá o nome de conjunto de regras executáveis (rule execution set). As regras no conjunto de regras executáveis têm de ser expressas utilizando uma linguagem para representação de regras. Esta especificação pretende abordar e facilitar a interoperabilidade entre motores de regras mas não aborda o tipo de linguagem a usar para representação de regras, uma vez que ainda não existe uma estandardização plenamente aceite pela comunidade.

A especificação tem como requisito ser flexível o suficiente para se adequar a qualquer tipo de motores de regras e não se restringe a uma semântica do ciclo de execução de regras ou a linguagem de representação de regras. Consequentemente, a especificação é o mais abstracta possível e não se restringe a implementações ou metodologias de gestão especificas. A especificação permite também motores de regras que estejam em execução em Java Virtual Machines (JVM) remotas.

Os principais objectivos da especificação são:

- Facilitar a integração de motores de regras em aplicações que utilizem tecnologia Java;

- Facilitar a comunicação e estandardização entre vendedores de motores de regras;

- Facilitar a integração da tecnologia de motores de regras em outros JSRs para suportar o modelo de programação declarativa;

- Disponibilizar modelos de implementações;

- Suportar actuais e novos motores de regras através da API harmonizada e que cumpre as necessidades actuais sem comprometer a extensibilidade no futuro;

Capítulo 4

Contexto Específico do Projecto

Neste capítulo descreve-se em detalhe a plataforma IP-JIB da PT Inovação, os componentes desta e o seu enquadramento da solução IMS da instituição, na qual o aluno desenvolve o seu projecto individual.

4.1 IP-JIB

O advento da convergência das redes fixas e das redes móveis (TISPAN)[1] conjugado com a rápida evolução e maturação das tecnologias de informação para telecomunicações (OSA/Parlay[2] e JAIN), tornaram finalmente possível a concretização de visões e conceitos perspectivados há mais de uma década para a realização de um ecossistema aberto de serviços de telecomunicações através das emergentes Plataformas de Fornecimento de Serviços.

O IP-JIB é um projecto que tem como principal objectivo a disponibilização de uma plataforma para fornecimento de serviços inteligentes de comunicação (texto, som e/ou imagem).

O principal objectivo deste projecto é criar uma solução, baseada na arquitectura IMS, para substituição da actual infra-estrutura de comunicações do Grupo PT. Todavia, antes de ser possível substituir a infra-estrutura actual, existe o requisito estratégico de ter todos os serviços visíveis para o utilizador final concretizados nesta nova solução.

Os resultados deste projecto ambicionam viabilizar o fornecimento dos serviços existentes de forma mais eficiente e, acima de tudo, a sua capacidade de fomentar novas oportunidades de negócio com novas aplicações (em particular aplicações multimédia), fornecidas pelos próprios operadores ou por terceiros.

[1] *Telecommunication and Internet converged Services and Protocols for Advanced Networking* - Este comité trata de todos os aspectos técnicos relacionados com as redes de próxima geração, como a arquitectura, a sinalização, a numeração, os serviços, a qualidade de serviço e a gestão.

[2] O grupo Parlay é um consórcio técnico que especifica APIs para a rede de telecomunicações. O objectivo deste grupo é especificar APIs independentes da tecnologia de comunicação e da linguagem de programação.

Do ponto de vista técnico o IP-JIB disponibiliza a infra-estrutura necessária para a criação de um ecossistema de serviços distribuídos e colaborativos. As novas aplicações são construídas através da composição de serviços com diferentes níveis de abstracção das tecnologias de rede. Para o efeito, a plataforma agrega as diferentes funcionalidades das redes e as diferentes fontes de conteúdos interligando-se com a infra-estrutura de sistemas de informação do operador. No contexto da arquitectura IMS o IP-JIB tem o papel de Application Server.

Sobre este vasto número de funcionalidades são disponibilizados aos criadores de aplicações um conjunto uniforme e normalizado de interfaces de programação que facilitem a concepção, desenvolvimento e aprovisionamento de novas aplicações.

A mesma plataforma pode fornecer múltiplas aplicações de voz e dados com conteúdos de diferentes origens e para diferentes terminais ligados a diferentes redes de acesso incluindo ADSL, PSTN, GSM e UMTS.

Um dos aspectos chave do IP-JIB consiste na sua facilidade de criação de novas aplicações que usem o maior número possível das funcionalidades disponíveis na rede e nos terminais de modo a promover a criatividade e tirar partido de conhecimentos específicos de segmentos de mercado. São usados modelos de programação apropriados para a natureza assíncrona (programação orientada para eventos) das aplicações típicas de telecomunicações. Adicionalmente são suportados modelos de programação síncronos tradicionais das aplicações de sistemas de informação Web, sendo possível a combinação de diferentes modelos numa mesma aplicação.

A plataforma IP-JIB é desenvolvida no âmbito da especificação JAIN SLEE e requer um servidor de aplicações JAIN SLEE. De momento, o IP-JIB pode ser suportado pela plataforma Red Hat Mobicents e pela plataforma Open Cloud Rhino.

Sempre que possível, são suportadas interfaces de programação normalizadas ou familiares para os programadores de modo a endereçar uma comunidade de programadores o mais vasta possível. Simultaneamente, é possível disponibilizar diferentes níveis de granularidade sobre um mesmo serviço de acordo com os níveis de exigência de cada aplicação.

As aplicações e serviços IP-JIB são, sempre que possível, independentes das tecnologias de rede usadas, tornando possível a adição, alteração e remoção de protocolos sem impacto nos serviços e aplicações existentes. O serviços são acessíveis pelo maior número possível de terminais e a lógica da aplicação é também independente desses terminais, tornando possível a utilização de novos terminais sem impacto nas aplicações existentes.

Como se pode constatar na Figura 4.1, a arquitectura do servidor de aplicações IP-JIB é composta pelos seguintes níveis de abstracção:

- Core - componentes para gestão do ciclo de vida do servidor, subscritores, aplicações e outro tipo de funcionalidades nucleares do servidor (e.g., gestão da configuração do servidor e rede, monitorização, segurança, desempenho, tolerância a faltas e balanceamento de carga). Este componente é suportado em grande parte pelo servidor de

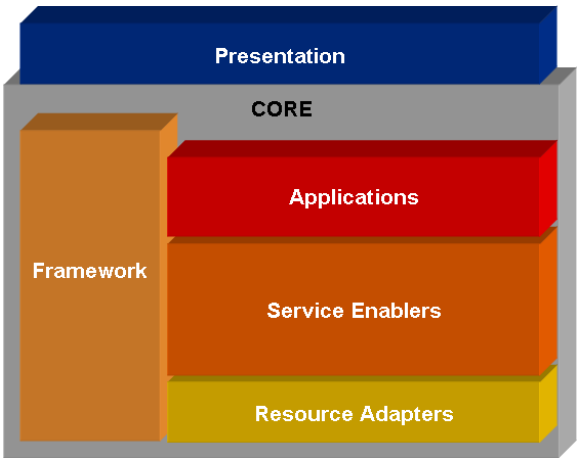

Figura 4.1: Arquitectura do Servidor de Aplicações IP-JIB

aplicações JAIN SLEE em que a plataforma se encontra instalada;

- Framework - disponibilização de funcionalidades transversais, incluindo aprovisionamento (de serviços, aplicações e utilizadores), descoberta e orquestração de serviços e gestão de conflitos entre serviços e aplicações;
- Resource Adaptores - componentes com interfaces de acesso a recursos externos. Disponibilizam as funcionalidades dos recursos externos aos componentes presentes na plataforma JAIN SLEE;
- Service Enablers - componentes para disponibilização de funcionalidades de mais alto nível (com abstracção sobre as componentes Resource Adaptores). O papel destes componentes é disponibilizar às aplicações um interface comum relativamente a recursos externos com os mesmos objectivos;
- Aplications - componentes com a lógica da aplicação que reutilizam componentes de outros níveis da arquitectura em particular as componentes Service Enablers;
- Presentation - componentes que disponibilizam funcionalidades de apresentação e gestão de dados e conteúdos. Suportadas por tecnologias Java para Web (EJB, Servlet e JSP) com interligação ao servidor de aplicações e aos repositórios de dados.

Apresenta-se de seguida a lista de componentes da plataforma actual.

4.1.1 Core

De momento, as funcionalidades core descritas anteriormente são suportadas pelos seguintes servidores de aplicações:

- Red Hat Mobicents - Servidor JAIN SLEE de código aberto suportado pelo servidor de aplicações Red Hat JBoss AS.

- Open Cloud Rhino - Solução comercial da empresa líder da especificação JAIN SLEE. Servidor de aplicações JAIN SLEE construído de raiz (com aproveitamento de algumas bibliotecas de domínio público).

4.1.2 Framework

- FrameworkProvisioning - Componente com funcionalidades de *provisioning* transversal a toda a plataforma
- Cache - Componente de gestão de cache interna do sistema transversal a toda a plataforma

4.1.3 Resource Adaptores

Os Resource Adaptores da plataforma IP-JIB são:

- Asterisk RA - Um Private Branch Exchange (PBX) é um centro de distribuição telefónica pertencente a uma empresa que não inclua como sua actividade o fornecimento de serviços telefónicos ao público em geral. O Asterisk é um PBX baseado em IP que disponibiliza todas as funcionalidades dos normais PBX. A arquitectura do Asterisk foi desenhada para permitir uma flexibilidade máxima e suportar serviços de voz sobre IP em vários protocolos.
- Diameter RA - Remote Authentication Dial In User Service (RADIUS) é um protocolo de Autenticação, Autorização e gestão de contas de utilizadores para aplicações com acesso ou mobilidade IP. O protocolo Diameter é derivado do protocolo RADIUS com várias melhorias em muitos aspectos e é considerado o protocolo de Authentication, Authorization e Accounting (AAA) de próxima geração. Para além de AAA, na arquitectura IMS o Diameter é usado para interacção entre vários componentes.
- InoVox RA - O Inovox é a solução PT Inovação para assegurar as funcionalidades de Media Resource Function (MRF) que, numa arquitectura IMS, é constituído pelos componentes Media Resource Function Processor (MRFP) e Media Resource Function Controller (MRFC). O Inovox é um Media Server IP genérico, que disponibiliza uma vasta gama de funcionalidades de processamento de diferentes tipos de media (tocar/gravar anúncios de áudio e vídeo, reconhecimento de fala, síntese de texto, conferência, fax e transcoding) que facilitam a implementação de serviços multimédia avançados.
- JDBC RA - Java Database Connectivity (JDBC) é a especificação de uma API para conexão de aplicações Java a base de dados. A aplicação utiliza a interface especificada para construir os comandos SQL que são depois enviados ao Sistema de Gestão da Base de Dados. O retorno dos comandos SQL utiliza a mesma interface.

- SIP RA - O protocolo Session Initiation Protocol (SIP) é de nível aplicação e tem capacidades de criação, modificação e encerramento de sessões com um ou mais participantes. Pode ser utilizado para criar sessões com dois participantes, com vários participantes ou sessões de difusão (e.g., incluem sessões de voz sobe IP, distribuição ou conferência multimédia). O protocolo SIP foi desenhado de forma a ser independente do protocolo nível transporte, pode ser usado com TCP, UDP ou mesmo SCTP.

  Em Novembro de 2000, o SIP foi aceite pelo 3GPP como protocolo de sinalização e é agora o elemento permanente na arquitectura IMS.

- XCAP Client RA - XML Configuration Access Protocol (XCAP) é um conjunto de convenções para mapear documentos XML e os seus componentes a endereços HTTP. Devido à sua estrutura, primitivas HTTP normais podem ser usadas para manipulação de dados.

- XMPP RA - Extensible Messaging and Presence Protocol (XMPP) é um protocolo baseado em XML para troca de mensagens instantâneas e eventos de presença. É o núcleo do protocolo Jabber que está neste momento presente em vários serviços de mensagens instantâneas e presença.

4.1.4 Service Enablers

Os Service Enablers da plataforma IP-JIB são:

- AAA - Service Enabbler de abstracção do protocolo de Autenticação, Autorização e gestão de contas de utilizadores (através do RA Diameter e interface Rf e Ro).

- GUP - O 3GPP Generic User Profile é a estandardização de dados relacionados com utilizador para gerir a experiência individual deste com os serviços associados. Usa o protocolo JDBC (através do RA JDBC e interfaces MySQL e PostgreSQL) e o protocolo XCAP (através do RA XCAP).

- Message - Service Enabler de abstracção de protocolos de mensagens instantâneas à plataforma. Usa o protocolo XMPP (através do RA XMPP).

- Presence - Service Enabler de abstracção de protocolos de presença à plataforma. Usa o protocolo XMPP (através do RA XMPP) e o protocolo SIMPLE (através do RA SIP).

- Session - Service Enabler de abstracção de protocolos de controlo e gestão de sessões. Usa o protocolo SIP (através do RA SIP).

4.1.5 Applications

As aplicações da plataforma IP-JIB são:

- Conference - Serviço de conferência áudio e vídeo com funcionalidades de mensagens instantâneas, presença e troca de recursos partilhados (e.g., *whiteboard*, ficheiros e ambiente de trabalho).
- PC2phone - Serviço de tarifação (e.g., sessões chamadas com voz e vídeo e sessões conferência).

4.1.6 Presentation

As componentes de apresentação da plataforma IP-JIB são:

- webadmin - Página de administração geral do sistema (e.g., gestão de utilizadores, de perfis de serviço e de tarifação).
- webconf - Página de administração e utilização do serviço conferência.

4.2 Personal Communication Manager

Pretende-se que o Personal Communication Manager (PCM) seja uma aplicação de Comunicação Personalizada que permita ao utilizador definir o comportamento mais desejável das suas comunicações, através de regras configuráveis pelo próprio.

As regras são definidas por condições e acções, sendo a acção de uma regra executada quando alguma ou todas as condições dessa mesma regra são satisfeitas. Deste modo, o utilizador pode definir inúmeras regras de acordo com as suas necessidades particulares.

As regras podem ser aplicadas tanto para chamadas de entrada, como para chamadas de saída. As condições podem ser em função do endereço do chamado ou do chamador, em função do dia/hora, em função do estado da ligação, em função do contexto ou de forma incondicional. Para cada condição pode-se aplicar também a "não-condição"(e.g., Ausente e Não Ausente).

A Tabela 4.1 transcreve exemplos de condições do serviço PCM.

A Tabela 4.2 transcreve exemplos de acções do serviço.

Com todas as suas funcionalidades activas o PCM utiliza os service enablers: GUP, Session, Presence e AAA. Indirectamente o serviço utiliza os RAs: JDBC, SIP, XMPP, Diameter e InoVox.

Pretende-se que o PCM tenha também capacidades de gestão de comunicações SMS, MMS, e-mail e mensagens instantâneas, estando para já apenas dependente deste tipo de capacidades dentro da plataforma.

Para gestão do serviço utiliza-se uma interface Web associada ao resto da plataforma de apresentação do sistema. Poderá também ser possível o uso de outras interfaces, como por exemplo administração por voz ou directamente nos terminais.

Tabela 4.1: Exemplo de condições no serviço PCM

Tipo	Condição
Terminal	Desligado
	Ocupado
Chamado	Não Atende
	Endereço
Chamador	Endereço
	Na Lista Negra
Contexto	Ausente
	Não Telefonar
	Não Incomodar
	Online
Incondicional	N/A

Tabela 4.2: Exemplo de acções no serviço PCM

Tipo	Acção
Encaminhamento	Encaminhar para Terminal Alternativo
	Encaminhar para Outro Destinatário
	Encaminhar para Caixa de Correio
Multimédia	Tocar Mensagem
	Tocar Ring Tone
Genérica	Rejeitar Chamada

4.3 Rule Engine

O projecto PUMA foca-se na concepção e implementação de um motor de regras apropriado a aplicações para personalização de um conjunto unificado de procedimentos e a aplicações multimédia que são independentes do tipo de comunicação, do tipo de informação multimédia usado e do tipo de dispositivos e respectivas interfaces de utilizador usadas.

Estas aplicações são adaptativas ao contexto em que se encontra o utilizador. Ou seja, a aplicação pode ter um comportamento diferente de acordo com cada contexto de utilizador e este comportamento pode ser independente da vontade do utilizador.

O objectivos deste subprojecto são a concepção e desenvolvimento ou integração de motor de regras para JAIN SLEE e IMS. Este componente deverá disponibilizar as funcionalidades dos motores de regras às aplicações e serviços desenvolvidos na plataforma IP-JIB.

De salientar que o motor de regras deverá apresentar a melhor eficiência possível em tempo de execução. Para se conseguir esta eficiência será necessário desenvolver um mecanismo de pré-compilação de regras e armazenamento em memória temporária.

Capítulo 5

Planeamento

Neste capítulo apresenta-se em detalhe o plano de trabalho, as metas a atingir e o processo de desenvolvimento previsto.

5.1 Plano pormenorizado

Este projecto esteve dividido em dois *Work Packages* (WP). O primeiro *Work Packages* (WP1) diz respeito ao desenvolvimento ou adaptação de um Motor de Regras para integrar na plataforma de comunicação inteligente IP-JIB. O segundo *Work Package* (WP2) é referente a uma aplicação que deverá usufruir das funcionalidades do Motor de Regras para permitir a personalização de um conjunto unificado de procedimentos de aplicações multimédia.

Para uma calendarização pormenorizada do projecto, consultar a Figura 5.1. Como se poderá observar na consulta, o WP1 teve uma estimativa de 7 homem/mês e a conclusão do WP2 teve uma estimativa de 5 homem/mês, perfazendo o total de 12 homem/mês.

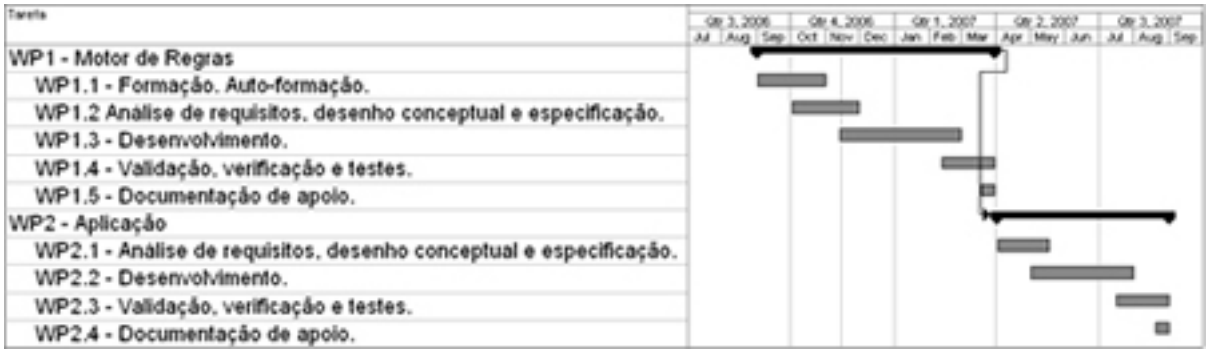

Figura 5.1: Diagrama de Gantt do projecto PUMA

Apesar de não estar previsto na calendarização, o WP2 poderia ser iniciado antes do

WP1 estar concluído. Esta opção facilitaria a identificação de mais requisitos para o Motor de Regras, evitando desta forma que estes requisitos tivessem sido identificados numa fase muito avançada do processo de desenvolvimento do WP1.

É importante referir que no planeamento não estava previsto eventuais alterações ao processo de desenvolvimento em ambas as WPs ou quaisquer outras tarefas.

Por forma a existir um acompanhamento formal no decorrer do projecto, foram compiladas um conjunto de metas que se encontram especificadas na Tabela 5.1.

Meta	Objectivo
M1 (aos 2 meses)	Relatório com estudo sobre tecnologias para Motor de Regras
M2 (aos 3 meses)	Relatório de requisitos e especificação do Motor de Regras
M3 (aos 6 meses)	Desenvolvimento do Motor de Regras terminado
M4 (aos 7 meses)	Relatório com o resultado dos testes efectuados ao Motor de Regras
M5 (aos 8,5 meses)	Relatório de requisitos e especificação da Aplicação
M6 (aos 11 meses)	Desenvolvimento da Aplicação terminado
M7 (aos 12 meses)	Relatório com o resultado dos testes efectuados à Aplicação

Tabela 5.1: Tabela de metas do projecto PUMA

Uma vez que não estava prevista a elaboração deste documento no âmbito do projecto PUMA e este não foi realizado durante o estágio, optou-se por não contabilizar o esforço desta tarefa no planeamento.

5.2 Processo de desenvolvimento

Apesar de estar integrado na equipa *core* de desenvolvimento, os dois *Work Packages* são da inteira responsabilidade do aluno, assim como todas as tarefas subjacentes.

No entanto, existiu pontualmente a necessidade de envolver no desenvolvimento outros componentes da plataforma, que estavam sobre a responsabilidade de outro programador, assim como a possibilidade de envolver outro colaboradores em qualquer uma das fases de desenvolvimento das WPs.

Este projecto está planeado de forma a que o processo de desenvolvimento adoptado seja uma adaptação do modelo em cascata, dividindo-se nas seguintes fases:

- Análise de requisitos;
- Desenho conceptual;
- Implementação e Integração;
- Testes;

No entanto, devido à complexidade e limitações das tecnologias a adoptar, existe a possibilidade de se evoluir para um modelo de processo de desenvolvimento em espiral. Regressa-

se desta forma à fase de desenho conceptual e evoluir-se-à os protótipos desenvolvidos à data.

De salientar também que visto o projecto estar inserido numa plataforma muito dinâmica, onde as necessidades do utilizador vão evoluindo e novos objectivos são delineados, finalizada a fase de desenvolvimento, poderá ser imperativo adoptar um modelo de processo de desenvolvimento incremental.

Não existe na PT Inovação uma metodologia de trabalho formal. O processo de desenvolvimento e ferramentas de trabalho são adaptados a cada departamento e tipo de projecto a desenvolver.

5.3 Ferramentas de trabalho

As ferramentas de trabalho utilizadas são as normalmente utilizadas no desenvolvimento de *software*.

Na fase inicial de cada *Work Package* existiu a necessidade de utilização de ferramentas de modelação UML. O aluno optou por usar a solução *open source* Star UML e a ferramenta Enterprise Architect.

Como ferramenta de desenvolvimento de *software* em Java foi utilizado o Eclipse Integrated Development Environment com a extensão para desenvolvimento JAIN SLEE.

Capítulo 6

Trabalho Realizado

Neste capitulo descreve-se o trabalho realizado no estágio na PT Inovação no âmbito da disciplina PEI. Relata-se a integração de um motor de regras e o desenvolvimento de uma aplicação que usufrui das funcionalidades do motor de regras.

6.1 Rule Engine

O primeiro WP do planeamento consiste na integração ou desenvolvimento de um motor de regras para a plataforma da PT Inovação suportada por um servidor de aplicações JAIN SLEE.

O principal objectivo deste desenvolvimento é disponibilizar as capacidades dos motores de regras à plataforma actual, permitindo criar novos e inovadores serviços. Desde serviços simples que tenham na sua lógica de negócio um comportamento baseado em regras, a serviços mais complicados de inferência do comportamento do utilizador, um motor de regras poderá ser a ferramenta indicada.

Na Tabela 6.1[1] transcreve-se a lista simplificada inicial de requisitos funcionais associados a este componente.

A primeira decisão de concepção foi a de integrar um motor de regras na nossa plataforma ou desenvolver um motor de regras que cumpri-se os requisitos actuais. Devido principalmente a todas as vantagens que o motor de regras JBoss Rules apresentava, entre as quais a inexistência de custo e o seu desenvolvimento baseado em padrões, optou-se por integrar este motor de regras.

Como método de interacção com o motor de regras JBoss Rules optou-se por utilizar o JSR94. Apresar de este apresentar algumas limitações e apenas disponibilizar as funcionalidades básicas dos motores de regras, não se prevê no futuro a necessidade de aproveitamento de outro tipo de funcionalidades ou funcionalidades especificas de um motor de regras

[1] Classificação de um a três sendo que três é considerada a unidade com mais valor

Tabela 6.1: Lista de requisitos do Rule Engine

#	Requisito	Importância	Prioridade
1	Abstracção na plataforma do motor de regras utilizado	3	1
2	Formato normalizado de interacção com o motor de regras	3	3
3	Abstracção na plataforma da linguagem declarativa utilizada	2	1
4	Formato normalizado de representação de regras	3	3
5	Abstracção na plataforma do repositório de regras utilizado	3	3
6	Representação das regras em linguagem natural	2	1
7	Representação das regras em tabelas de decisão	1	1
8	A aplicação envie um facto para avaliação e obtenha o resultado	3	3
9	A aplicação envia uma lista de factos para avaliação e obtenha o resultado	3	3
10	Associação de uma identificação de sessão de regras a um utilizador	3	3
11	Associação de uma identificação de sessão de regras a uma aplicação	3	2
12	Associação de uma identificação de sessão de regras a um conjunto de aplicações	2	2
13	A aplicação carrega para memória temporária uma sessão de regras	3	2
14	A aplicação descarrega da memória temporária uma sessão de regras	3	2
15	A aplicação modifica uma sessão de regras em memória temporária	1	2
16	A aplicação verifica se uma sessão de regras se encontra em memória temporária	3	3
17	A Aplicação carrega para memória temporária um conjunto de sessões de regras	2	1
18	A aplicação carrega da memória temporária um conjunto de sessões de regras	2	1
19	Interface de gestão de regras por utilizador	3	3
20	Interface de gestão de regras por aplicação	2	2
21	*Backup* automático de todas as regras do sistema	3	1
22	Registo de alterações efectuadas no conteúdo do repositório de regras	2	2
23	*Backup* de regras pelo utilizador e para o utilizador	1	1

Ainda não existe neste momento uma linguagem de programação estandardizada para utilização em motores de regras. Existe um esforço da comunidade académica e empresarial para finalizar o desenvolvimento e adoptar as Linguagem RIF e a RuleML. No entanto, uma vez que as linguagens ainda não estão finalizadas e a adaptação destas, devido à sua complexidade, deverá ser da responsabilidade da equipa de desenvolvimento do motor, optou-se por utilizar a linguagem Drools Rule Language que é especifica do motor de regras adoptado.

Existem dois modos de representação da linguagem, texto normal ou através de uma linguagem XML. A representação XML tem a desvantagem de não suportar linguagens naturais ou tabelas de decisão. Optou-se no entanto por usar a linguagem XML pois prevê-se em breve portar o repositório actual para uma base de dados XML com acesso através do RA XCAP.

Este novo componente tem uma arquitectura muito semelhante aos componentes já existentes mas com algumas adaptações. A arquitectura actual baseia-se na utilização de recursos externos abstraindo o mesmo através de um componente intermédio a que se chama service enabler. Como se pode constatar na Figura 6.1, o componente divide-se em três subcomponentes.

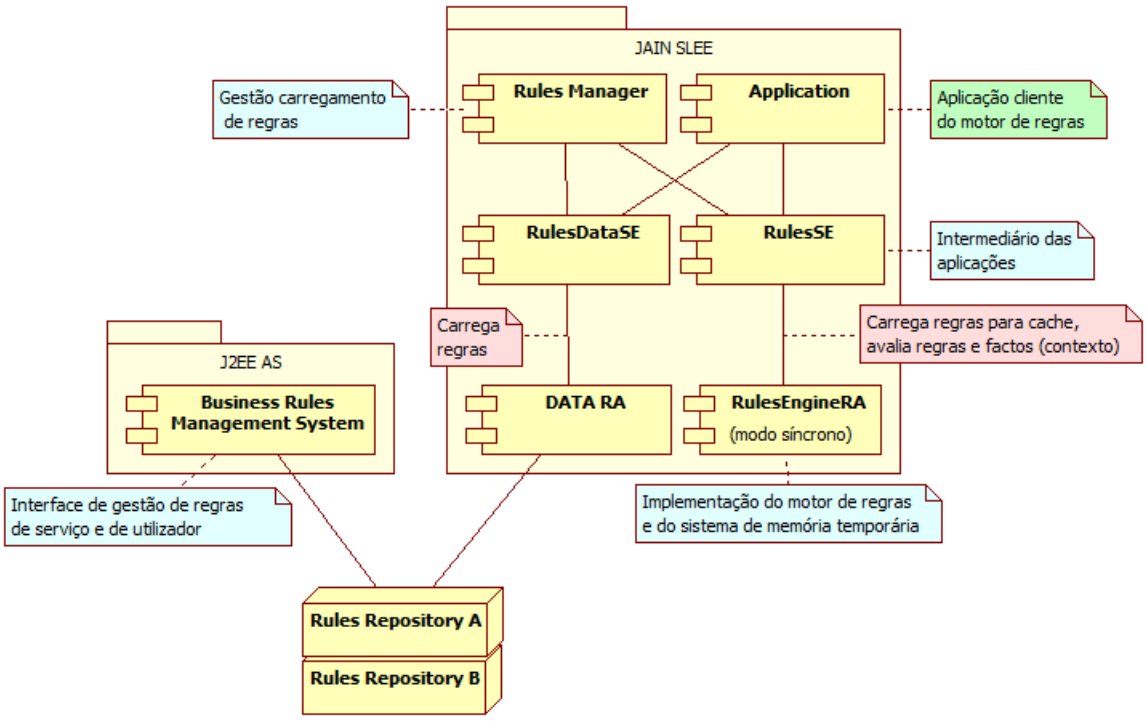

Figura 6.1: Rule Engine: Arquitectura do componente

O Rule Engine RA é responsável pela interacção com o motor de regras e pelo sistema de gestão de memória temporária. Por limitações da especificação JAIN SLEE, tanto a

integração do componente externo como o sistema de memória temporária só podiam ser integrados em RAs. Optou-se por fazer um RA com as duas funcionalidades para evitar interacções desnecessárias entre o RA de cache e o RA de motor de regras (os RAs só podem comunicar através de uma entidade terceira). Na Figura 6.2 pode-se observar o diagrama de classes do RA.

Como se pode constatar pela figura, a comunicação com o RA é feita sincronamente através da interface que o JAIN SLEE disponibiliza para o efeito. Sendo que o RA apenas disponibiliza os métodos:

- createSession(String sessionId, Reader drlReader) - método para criar uma sessão de regras associada à identificação recebida;
- createSession(String sessionId, Reader drlReader, Reader dslReader) - método para criar uma sessão de regras associada à identificação recebida e utilizando a definição da linguagem natural especifica;
- destroySession(String sessionId) - método para destruir uma sessão de regras associada à identificação recebida;
- verifySession(String sessionId) - método para verificar a existência de uma sessão de regras associada à identificação recebida;
- executeRules(String sessionId, List<RAFact> conditions): List<RAFact> actions - método para avaliar uma sessão de regras associada à identificação e factos recebidos, retornando as acções caso existam.

O Rule SE tem como responsabilidade mediar todas as relações que possam ocorrer entre uma aplicação e as funcionalidades de regras. A esta responsabilidade acrescentaram-se as funcionalidades de interacção com o JDBC RA para carregar e guardar as regras no repositório de regras e gestão das identificações de sessões de regras. Na Figura 6.3 pode-se observar o diagrama de classes do SE.

Como se pode constatar na figura a interface disponibilizada às aplicações consiste apenas em dois métodos:

- executeRules(RulesClientId clientId, List<Fact> condition): List<Fact> actions - método que disponibiliza a capacidade de avaliação de factos do motor de regras às aplicações da plataforma;
- executeRules(String clientId, List<Fact> condition): List<Fact> actions - método semelhante ao anterior mas neste caso a aplicação cliente sabe a identificação da sessão de regras.

O terceiro componente é o Rules Manager que tem como responsabilidade gerir a criação ou destruição de sessões de regras. Esta aplicação através de escuta eventos exteriores ou de

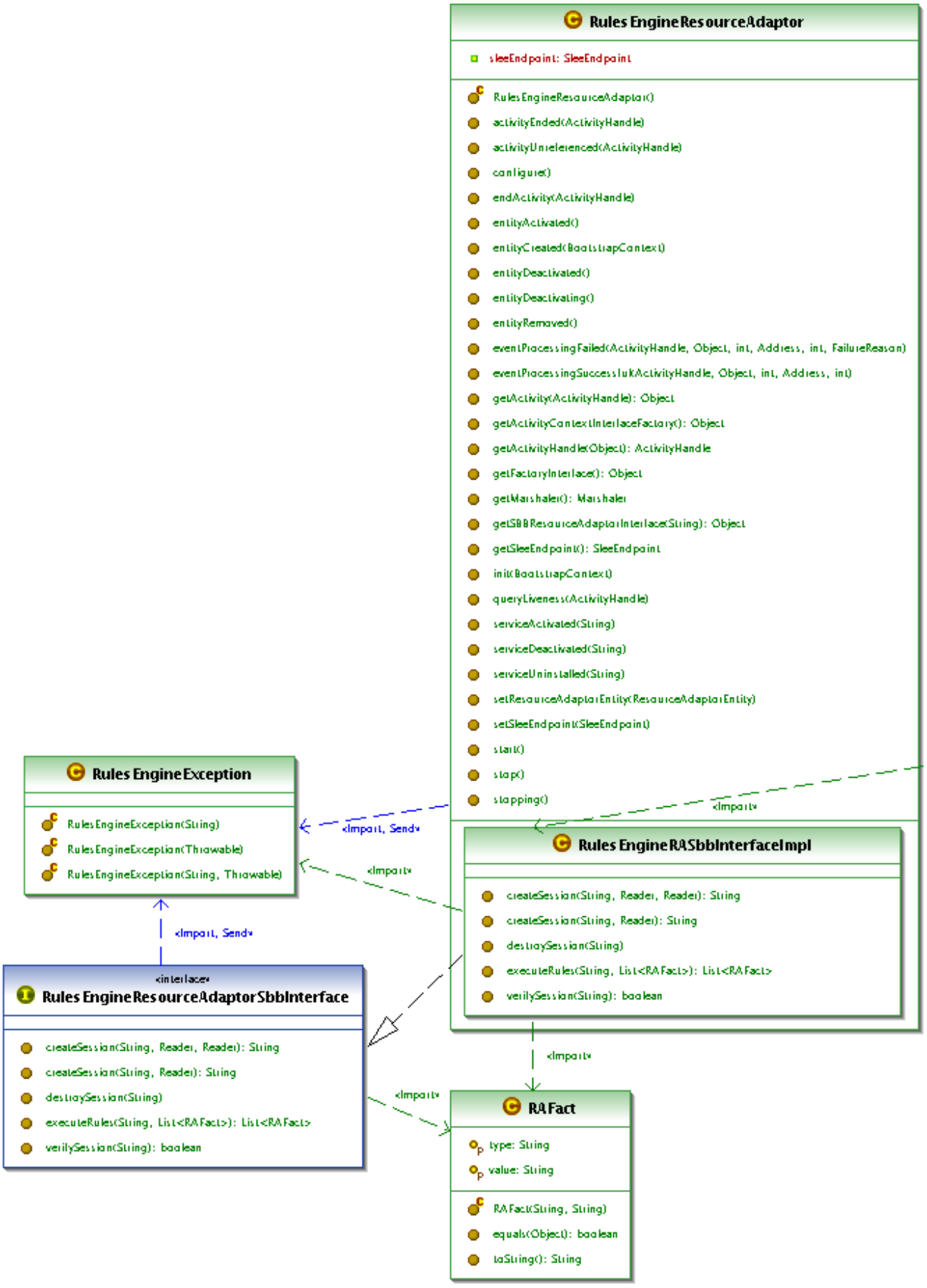

Figura 6.2: Rule Engine Resource Adaptor: Diagrama de Classes

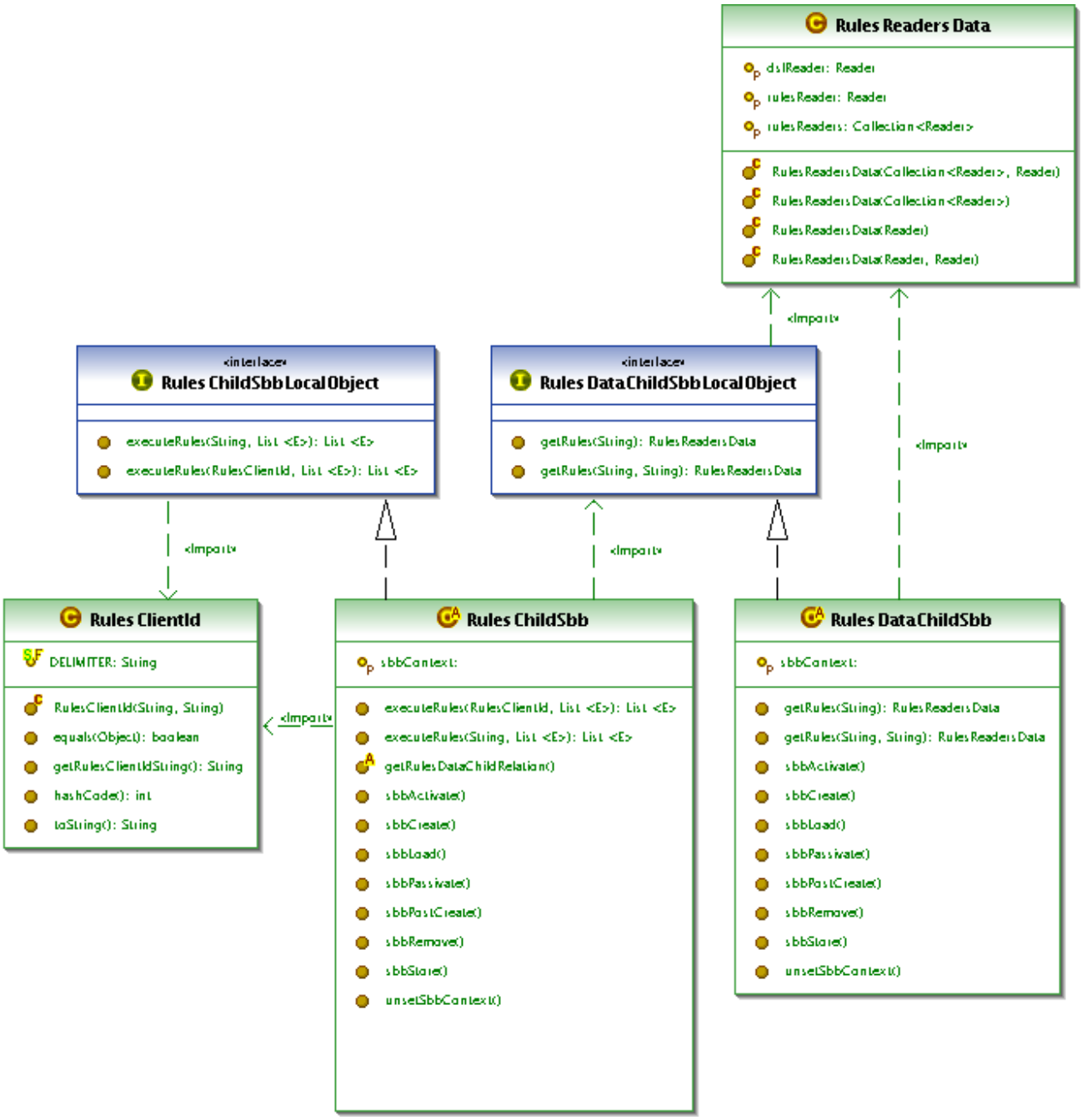

Figura 6.3: Rule Engine Service Enabler: Diagrama de Classes

acordo com o contexto poderá decidir carregar para memória temporária sessões de regras de aplicações ou utilizadores. O Rules Manager comunica apenas com o Rule Engine RA.

Existem dois modos de funcionamento do motor de regras no sistema, com ou sem memória temporária. No modo sem memória temporária o conjunto de regras é carregada e as sessões de regras são criadas quando necessárias e destruídas de seguida. Na Figura 6.4 pode-se observar o diagrama de sequência deste modo de funcionamento.

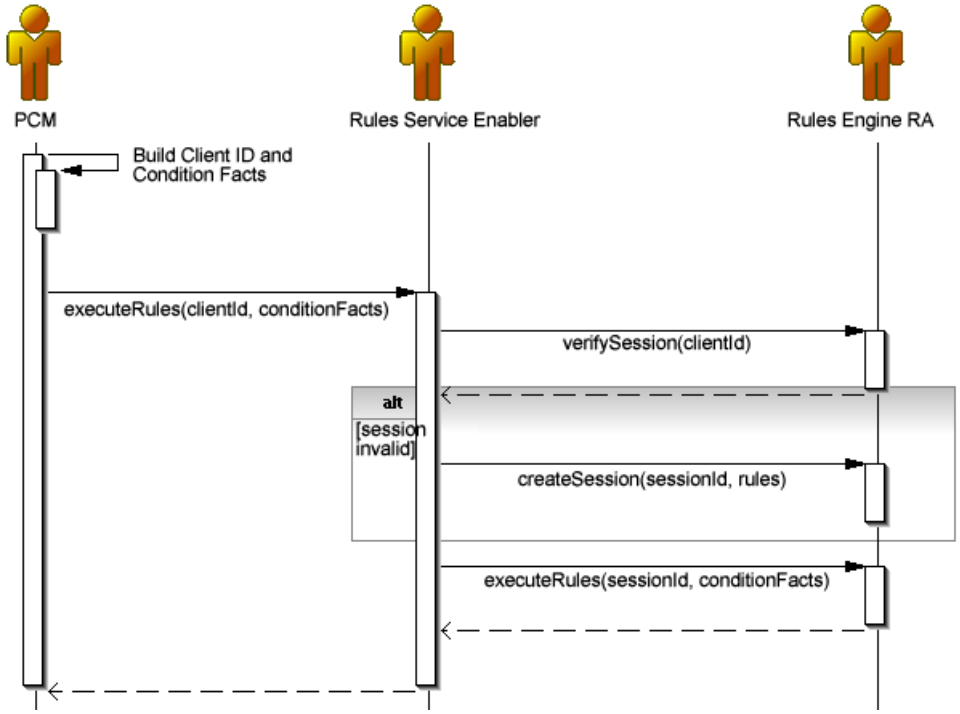

Figura 6.4: Rule Engine: Diagrama de sequência sem memória temporária

No segundo modo de funcionado, com o Rules Manager activo, as regras são carregadas previamente e as sessões criadas antes de serem utilizadas. O Rules Manager é também responsável por destruir as sessões em memória temporária. Foi necessário criar este modo de funcionamento por duas razões: a localização do repositório de regras poderá ser distante e demorar-se demasiado tempo a carregar as regras e a criação das sessões de regras (compilação da regra em linguagem interna do próprio motor de regras) é um processo lento. Na Figura 6.5 pode-se observar o diagrama de sequência deste modo de funcionamento.

A interface de gestão de regras é enquadrada no âmbito do desenvolvimento do serviço PCM.

Efectuou-se uma avaliação a repositórios baseados no padrão JSR283 (Java Content Repository) mas optou-se por efectuar o desenvolvimento numa base relacional normal, com possibilidade de evoluir no futuro para uma tecnologia XCAP. Sendo assim, todos os requisitos associados com o repositório de regras serão cumpridos pelo Sistema de Gestão de Base de Dados e fogem ao âmbito do desenvolvimento deste componente.

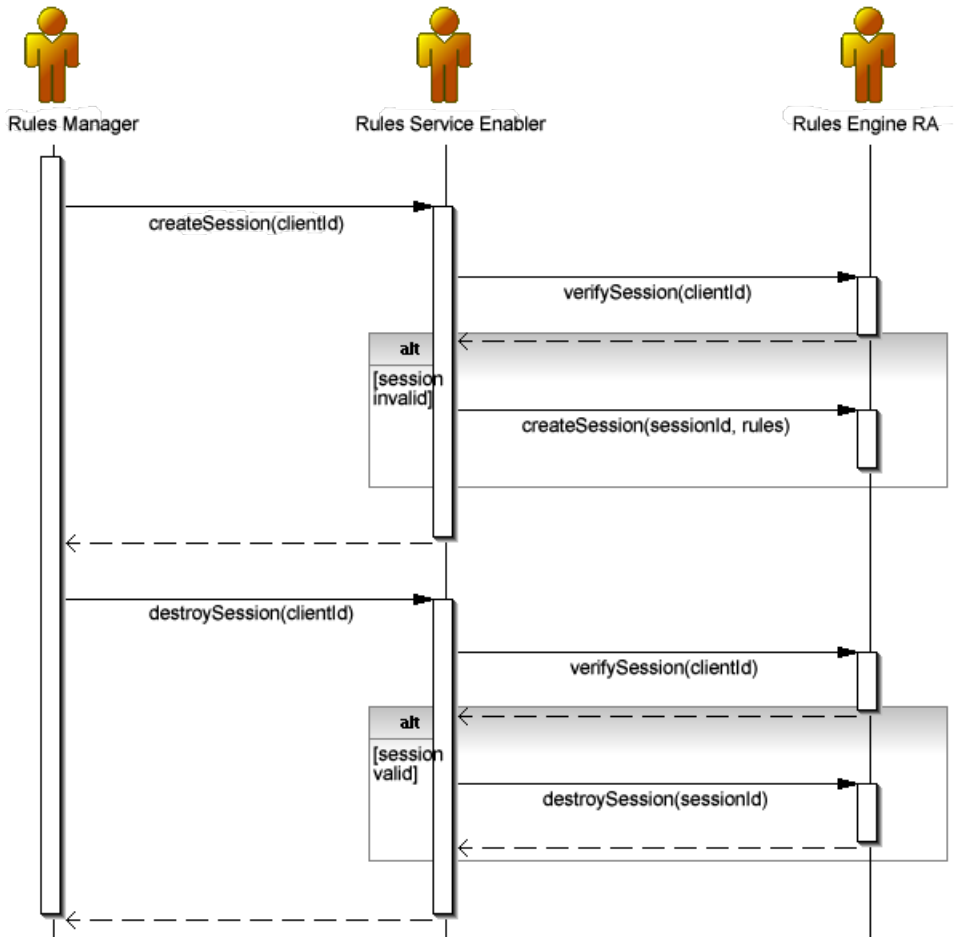

Figura 6.5: Rule Engine: Diagrama de sequência com memória temporária

6.2 Personal Communication Manager

O PCM é um dos serviços chave da plataforma IP-JIB e com muitas e diversificadas funcionalidades.

O serviço PCM já existia numa versão mais simplificada e com menos funcionalidades. O aluno ficou responsável pela continuidade do desenvolvimento deste serviço.

Devido à sua complexidade, o PCM tem várias dependências de outros componentes da plataforma. Nos RAs da plataforma o PCM depende do INOVOX RA para interligação com o Media Resource Function, JDBC RA para comunicação com a base de dados principal, SIP RA para gestão das sessões do utilizadores e do XMPP RA funcionalidades de presença. Nos SEs o PCM depende do GUP para gestão de perfis de utilizador, do Presence para funcionalidades de presença e do Session para gestão das sessões dos utilizadores. Por fim, para funcionalidades de *provisioning* o PCM depende do FrameworkProvisioning e para apresentação depende do webadmin.

A versão do PCM aqui estudada depende também de todo o componente de regras desenvolvido e relatado na secção anterior deste documento.

Sendo o serviço PCM constituído por uma aplicação, a arquitectura é muito semelhante ao diagrama de dependências de classes na Figura 6.6.

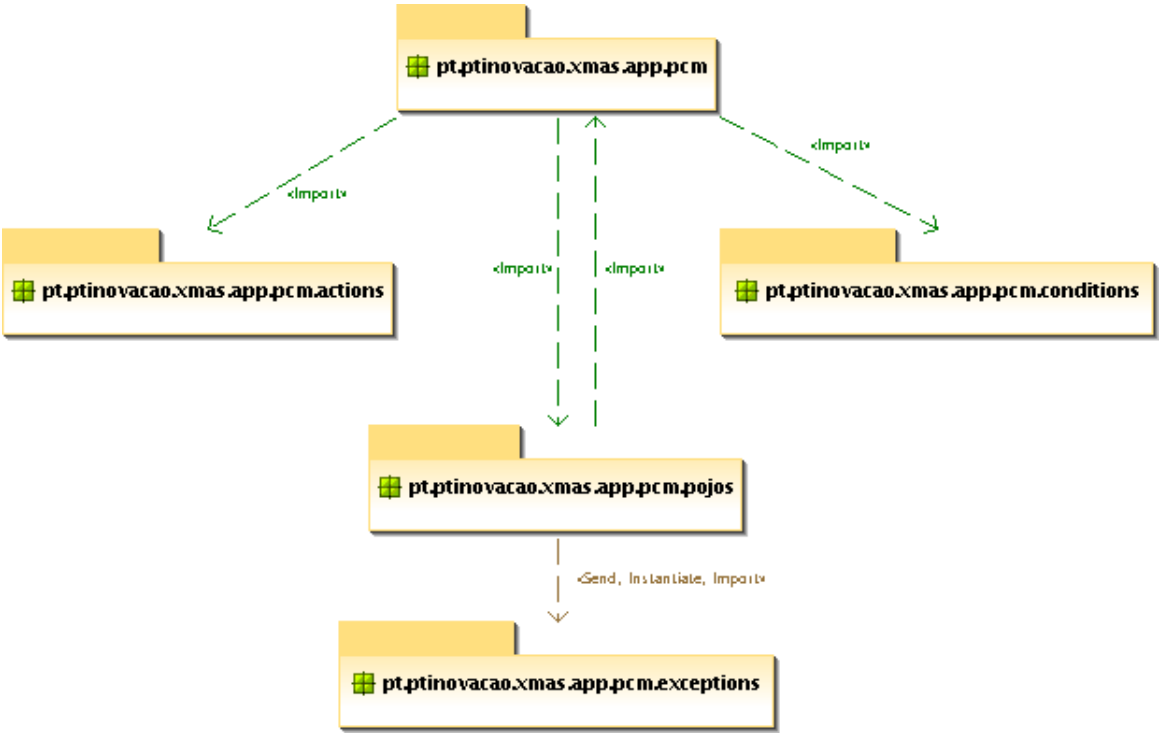

Figura 6.6: Serviço PCM: Diagrama de dependências de classes

Existe um Root Sbb para o serviço que tem como responsabilidade orquestrar a aplicação,

através de uma máquina de estado global, e interagir com os Child SBB e os componentes exteriores ao serviço.

O pacote 'conditions' é constituído pelos Child SBBs responsáveis por obter o contexto do utilizador, que entre outras coisas pode incluir o estado do serviço, por exemplo. O pacote 'actions' é constituído por um Child SBB por cada acção executável. Optou-se por esta arquitectura modelar pois oferece a flexibilidade necessária num serviço deste tipo. Por exemplo, é possível adicionar uma nova acção ao serviço sem ser necessário alterar a lógica aplicacional.

Numa execução normal do serviço PCM, a aplicação é iniciada com um evento exterior à plataforma (e.g., chamada) e depois de avaliar o contexto do utilizador decide se toma ou não alguma acção.

A decisão da acção a tomar pode ser baseada em decisões anteriores. Vejamos um exemplo: caso o utilizador tenha uma regra 'se ausente encaminhar para terceiro' e o PCM já tomou a decisão de encaminhar para terceiro mas a chamada é rejeitada e o PCM tem de avaliar novamente o contexto do utilizador, esta regra já não será aplicada.

Tal como descrito anteriormente o PCM é a aplicação cliente do motor de regras e para além da máquina de estado não tem muita mais lógica de decisão.

A linguagem de representação de regras utilizada tem uma versão XML com a qual o motor de regras trabalha (em conjunto com o PCM) e como se pode observar na Especificação 5 é uma linguagem declarativa. Esta linguagem XML é do próprio motor de regras e não tem qualquer semelhança implícita com o RIF ou o RuleML.

Para além da versão XML, o motor de regras Red Hat/JBoss Rules tem uma linguagem própria e pode utilizar também linguagem especificas do domínio. Apesar do PCM não ter essa característica, a regra representada em XML na Especificação 5 poderia ser representada na linguagem própria do motor na Especificação 6. Na Especificação 7 pode-se constatar um exemplo de uma representação com linguagem específica do domínio.

A lógica de decisão está toda do lado do motor de regras, tendo sido esta a principal razão de escolha de um motor de regras nesta aplicação. O PCM é informado de qual a acção e através de um Child SBB utiliza os SEs e RAs para executar as suas acções (e.g., SIP RA e Session para tocar um *ringback* tone localizado no MRF).

As regras são criadas fora do componente e são criadas manualmente através das indicações do utilizador. A edição de regras do utilizador é efectuada nas páginas de administração do serviço PCM, que estão englobadas no portal de administração de utilizador da plataforma IP-JIB.

A apresentação da página é efectuada através de tecnologia JSP. A máquina de estado do portal e lógica de decisão do negócio são suportados através de tecnologia Servlet. Por fim, a interacção com o servidor de aplicações JAIN SLEE e o servidor de aplicações JEE é efectuada através de EJB.

A página principal do serviço PCM divide-se em três secções. A primeira secção é

Especificação 5 Serviço PCM: Exemplo de uma regra em XML

```
<rule name="Example">
<lhs>
<column object-type="Fact">
<literal field-name="type"evaluator="=="value="away"/>
<literal field-name="value"evaluator="=="value="true"/>
</column>
<not>
<column object-type="Fact">
<literal field-name="type"evaluator="=="value="working"/>
<literal field-name="value"evaluator="=="value="true"/>
</column>
</not>
</lhs>
<rhs>
assert( new Fact("action", "forward") );
assert( new Fact("address", "mobile@brunoduarte.com") );
</rhs>
</rule>
```

Especificação 6 Serviço PCM: Exemplo de uma regra em DRL

```
rule "Example"
when
Fact(type="away", value=true)
Fact(type="working", value=false)
then
assert( new Fact("action", "forward") );
assert( new Fact("address", "mobile@brunoduarte.com") );
end
```

Especificação 7 Serviço PCM: Exemplo de uma regra em DSL

```
rule "Example"
when
I'm away and I'm not working
then
Forward to mobile@brunoduarte.com
end
```

destinada a configurações com carácter dinâmico ou pontual. Na segunda secção pode-se definir grupos de regras (e.g., período de trabalho ou período de férias). A última secção da página principal apresenta a lista de administração de regras (adicionar, alterar ou remover). No caso da Figura 6.7 pode-se observar a página sem grupos e regras.

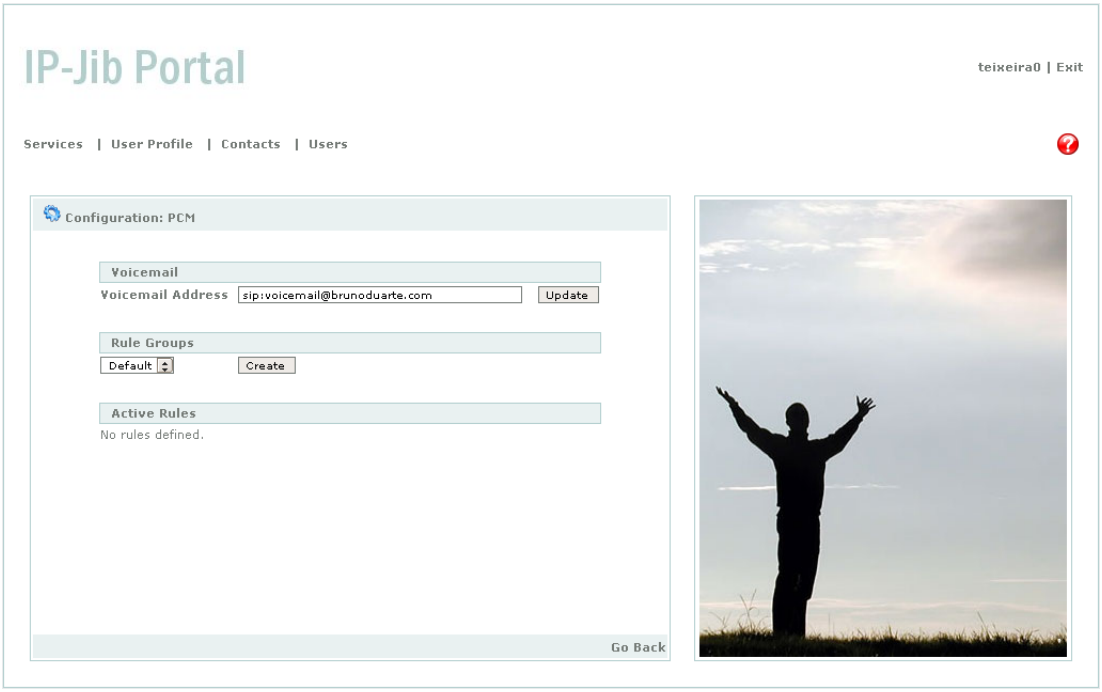

Figura 6.7: Serviço PCM: Página principal

Depois de se criar um grupo, ao se clicar em "Adicionar Regra"navega-se para a página capturada na Figura 6.10. Como se pode observar no exemplo, a primeira secção desta página contém a identificação da regra e uma expressão (actualizada dinamicamente) que representa o comportamento da regra.

Na segunda secção da página pode-se administrar as condições da regra (adicionar, remover ou escolher tipo de condição). Cada condição é também representada negativamente e expressa em regra automaticamente. Complementado a lista de condições é possível definir o tipo de avaliação, se todas as condições têm de se verificar ou basta apenas uma.

Na terceira secção da página define-se qual a acção a executar e as configurações da mesma.

Toda esta página é alterada em tempo real dinamicamente (expressão da regra, condições e acções).

Pode-se observar na Figura 6.8 a lista de possíveis condições.

Pode-se observar na Figura 6.9 a lista de possíveis acções.

É possível criar um conjunto muito vasto de regras diferentes, com várias condições (positivas e negativas) e acções mais complexas. No exemplo da Figura 6.10 pode-se também

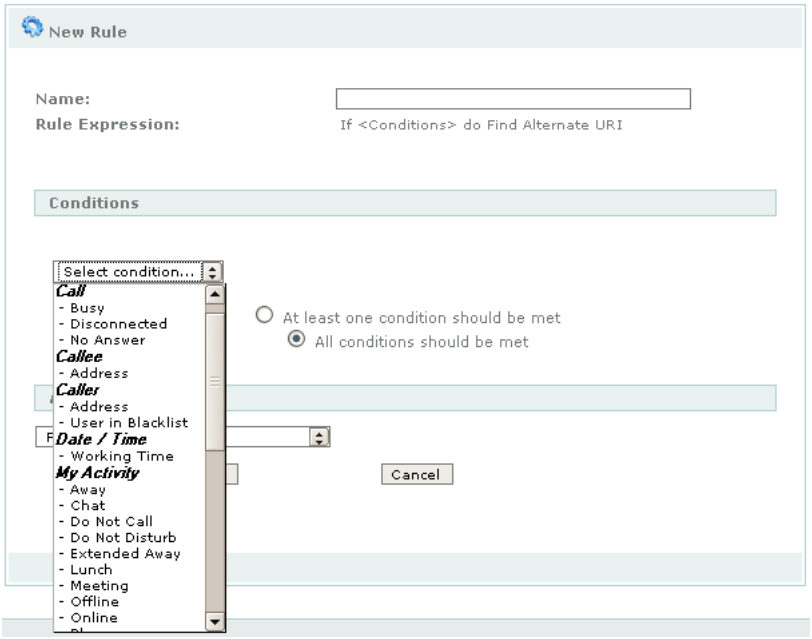

Figura 6.8: Serviço PCM: Lista de condições

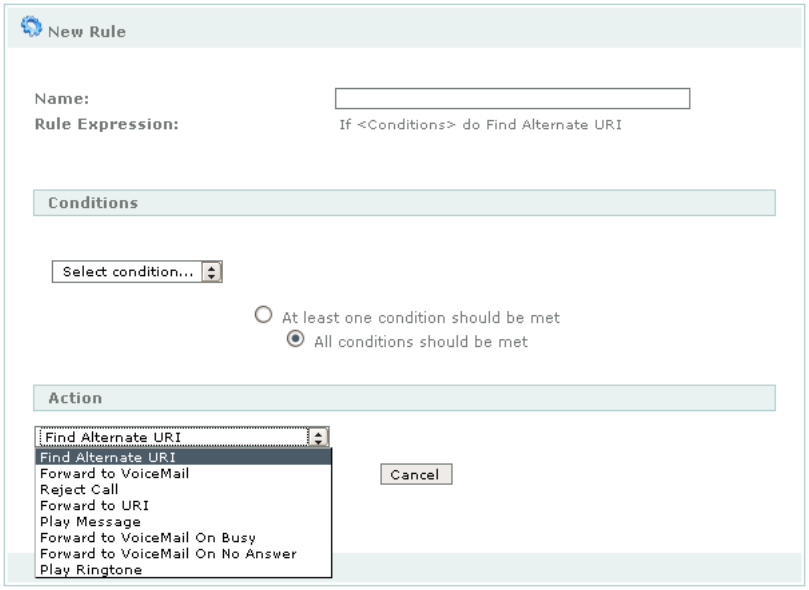

Figura 6.9: Serviço PCM: Lista de acções

verificar uma condição baseada em expressão regular da identificação do chamador.

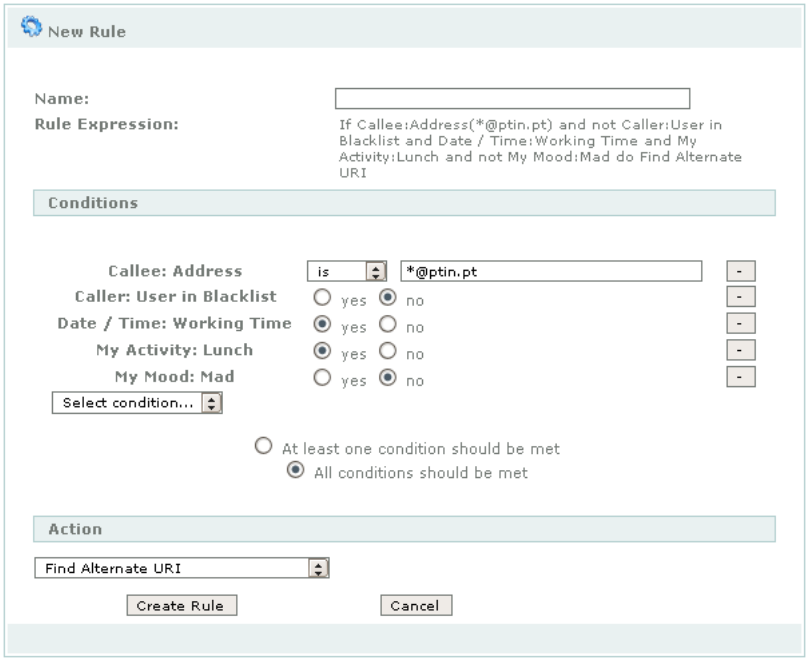

Figura 6.10: Serviço PCM: Exemplo de um regra complexa

São efectuadas validações à coerência da regra e do conteúdo dos campos onde é permitida escrita e não é possível criar regras incoerentes. O exemplo de um utilizador que tenta criar uma regra sem condições pode ser observado na Figura 6.11. Esta verificação é também feita no lado do servidor.

Como se pode verificar na Figura 6.12, regressando à página inicial de administração do serviço PCM, já é possível observar a lista de regras criadas. Outra funcionalidade interessante na página principal é a facilidade de se verificar o conteúdo da regra através de um *tooltip*.

Por fim, é importante frisar que a Internacionalização e Localização da página de administração não foi descurada. Na Figura 6.13 pode-se observar a página do serviço PCM em língua Portuguesa.

Ao longo do desenvolvimento deste componente foi também criado um componente de criação automática de documentação que integra com a plataforma (apesar de não prestar nenhuma funcionalidade em tempo de execução). Este componente utiliza documentação elaborada em formato XML DocBook para gerar modelos de livros em vários formatos (e.g., TXT, HTML e PDF).

A criação de manual de instalação, integração e administração (para uso interno) foi elaborado utilizando o componente de documentação supracitado.

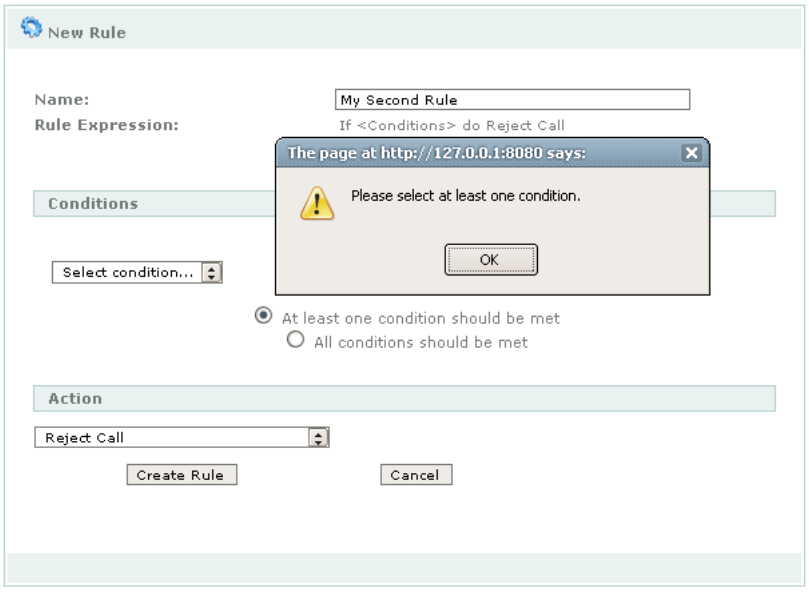

Figura 6.11: Serviço PCM: Verificação na criação ou edição de regras

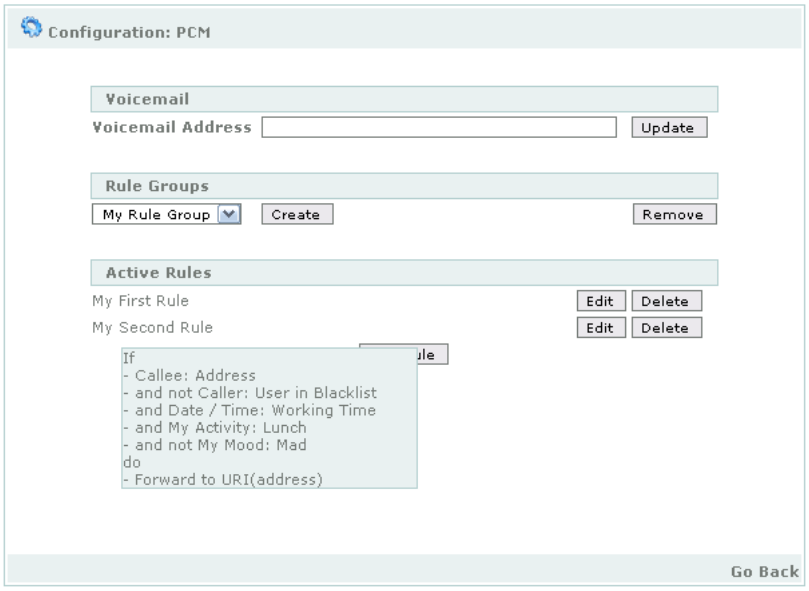

Figura 6.12: Serviço PCM: Página Principal com lista de regras e *tooltip*

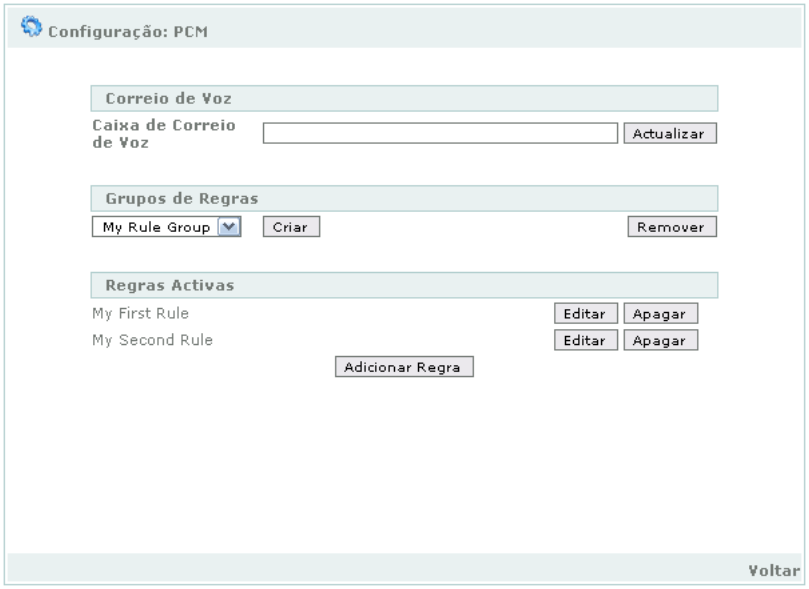

Figura 6.13: Serviço PCM: Página Principal em língua Portuguesa

Capítulo 7

Trabalho Futuro

Neste capítulo relata-se o desenvolvimento futuro planeado, algumas ideias que poderão melhorar os produtos desenvolvidos e desenvolver novos produtos.

7.1 Rule Engine

Neste momento existem duas preocupações em relação ao motor de regras: o interface JSR e a linguagem de regras.

Apesar de neste momento o interface JSR para motores de regras ser suficiente para o as necessidades actuais da plataforma IP-JIB, o interface é muito limitado e apenas permite usufruir das funcionalidades básicas dos motores de regras.

Para precaver no futuro a necessidade de usufruir de funcionalidades específicas de um motor de regras e o interface o impossibilitar, deveria ser criado um interface para uso interno. Este interface desenvolvido em casa deveria usar a API própria do motor de regras e deveria ser estendido conforme as necessidades da plataforma IP-JIB. A principal vantagem desta decisão prende-se com a não dependência do limitado API estandardizado. A principal desvantagem é o esforço necessário para fazer este desenvolvimento.

A linguagem de regras é também alvo de preocupação. As estandardizações das linguagens RuleML e RIF ainda não estão finalizadas e isso limita os programadores às linguagens proprietárias e especificas de cada motor de regras. Um estudo sobre a viabilidade do desenvolvimento de uma linguagem própria e ajustada às necessidades da plataforma IP-JIB deveria ser efectuado.

A principal vantagem do desenvolvimento de uma linguagem própria seria a possibilidade de criar uma linguagem muito simples e adaptada às necessidades das aplicações que usariam o motor de regras. Como desvantagens verifica-se o esforço acrescido de criar um tradutor da linguagem interna para a linguagem especifica do motor de regras.

Alternativamente, poderia ter grande interesse o estudo do desenvolvimento de uma forma de adopção das versões actuais das linguagens RuleML e RIF. A vantagem deste

desenvolvimento seria o facto de se poder criar um repositório de regras (quase) à prova de futuras alterações a este componente. A principal desvantagem deste desenvolvimento é o esforço necessário para o mesmo e o risco de não existir retorno, devido ao estado prematuro de ambas as linguagens.

Poderão ser efectuados diversos estudos e "proof-of-concepts"em relação a motores de regras, IMS e JAIN SLEE. No caso do IP-JIB poderia ser interessante integrar outros motores de regras com outras funcionalidades e efectuar exaustivos testes de desempenho a cada um destes.

Deveria também ser efectuado um estudo a motores de regras usados dentro da plataforma IP-JIB mas que executam a avaliação de regras fora desta. Esta funcionalidade será particularmente interessante quando a avaliação de regras demorar muito tempo e/ou exigir muitos recursos, correndo-se o risco de atrasar o normalmente funcionamento dos serviços na plataforma IP-JIB.

Existe um Rules Engine RA desenvolvido pela equipa da plataforma Mobicents. Este RA é restrito ao motor de regras usado (Red Hat/JBoss Rules) e não cumpre todos os requisitos necessários para uma correcta execução na plataforma IP-JIB (não tem repositório de regras em XML, não tem sistema de memória temporária e a interacção com o RA é assíncrona). Poderia ser de grande interesse comparar ao pormenor ambas as soluções e tentar acordar um interface comum (RA Type) ou mesmo a própria implementação.

Por fim, à semelhança do que vai acontecer com todos os repositórios no projecto, o repositório do motor de regras terá de ser portado para uma base de dados XDM, passando a utilizar o RA XCAP para aceder às regras.

7.2 Personal Communication Manager

Neste momento não existe integração entre o PCM e o serviço AAA. A tarifação do serviço PCM apenas pode ser efectuada através de uma tarifa plana de utilização. Uma integração com o serviço AAA iria permitir criar outros planos de tarifação baseados na utilização do utilizador e criação de subserviços PCM (e.g., apenas utilização da acção *ringback tone*).

A principal limitação do serviço PCM prende-se com o conjunto de condições e acções possíveis e o facto de as condições apenas serem avaliadas quando existe uma comunicação.

Seria muito interessante poder controlar as comunicações de um utilizador mesmo quando este não está a receber uma chamada. Alguns casos possíveis seriam a execução de uma das acções supracitadas de acordo com condições de localização do utilizador, data (calendário), trânsito, cotações na bolsa ou finanças pessoais.

Com as funcionalidades actuais do PCM facilmente se poderá adicionar condições de estado da caixa de correio electrónico de voz (e.g., mais de X mensagens), capacidade de um terminal ou mensagens instantâneas. Às acções, a adicionar, poderá acrescentar-se mensagens de correio electrónico (e-mail), SMS, MMS ou mensagens instantâneas,

Deveria efectuar-se um estudo em relação ao melhor mecanismo de orquestração de acções (para evitar incompatibilidades). A verificação ao nível apresentação poderá ser demasiado penosa.

Um projecto muito ambicioso e que elevaria o motor de regras para um papel ainda mais importante seria a aprendizagem do comportamento do utilizador e inferência de regras. Este componente deveria avaliar toda a história comportamental de utilização do serviço PCM e sugerir novas regras ao utilizador (e.g., inferir a regra equivalente ao utilizador rejeitar todas as chamadas do emprego depois das 22h).

Uma vez que os utilizadores nem sempre têm acesso a um terminal com internet, deveria ser disponibilizado um interface de voz que permitisse controlar todo o serviço.

Capítulo 8

Considerações Finais

Este último capítulo resume as tecnologias estudas e os projecto desenvolvido.

8.1 Tecnologia

O IMS consiste numa arquitectura para as redes de próxima geração e está neste momento a ser implementado em telecoms por todo o mundo. A arquitectura, baseada em camadas horizontais, pretende ser o mais flexível e reutilizável possível. O IMS é composto por vários componentes separados em termos funcionais. O departamento em que o aluno desenvolveu o seu trabalho tem como responsabilidade o componente *Application Server* da solução IMS da PT Inovação.

A solução *Application Server* da PT Inovação é uma plataforma suportada por um *Application Server* JAIN SLEE e tem a designação de IP-JIB.

O JAIN SLEE é uma especificação para servidores aplicacionais com arquitecturas baseadas em eventos. Este é considerado o ambiente perfeito para serviços de telecomunicações.

O IP-JIB pretende ser o núcleo da rede IMS e tem como responsabilidade a disponibilização de serviços aos utilizadores finais. Dos actuais serviços da plataforma IP-JIB destacam-se o serviço de conferência, o serviço de tarifação e o serviço PCM do qual o aluno é o responsável.

8.2 Projecto

Numa fase inicial, o projecto consistiu na adaptação e integração do motor de regras Red Hat/JBoss Rules na plataforma IP-JIB. O principal objectivo do projecto foi a disponibilização das capacidades dos motores de regras actuais para o desenvolvimento de novos serviços na actual plataforma IP-JIB.

A adaptação do motor de regras consistiu no desenvolvimento de três componentes. O primeiro componente foi a criação de um *Resource Adaptor* para interacção com o motor

de regras. De seguida desenvolveu-se um componente (*Service Enabler*) de abstracção das funcionalidades de regras. Por fim, o terceiro componente tem como responsabilidade o carregamento e gestão de regras em memória temporária.

A segunda *Work Package* do projecto consistiu na criação de uma aplicação que utilizasse as funcionalidades do motor de regras. O serviço PCM foi o primeiro a utilizar as funcionalidades do motor de regras para gerir a lógica de negócio do serviço. A principal razão de uso do motor de regras, para além da gestão da lógica de negócio do serviço, foi a separação da lógica de negócio da lógica de dados. Esta separação dá a flexibilidade necessária para se adicionar novas funcionalidades ao serviço sem que seja necessário alterar toda a aplicação.

Tanto o componente Rule Engine como o serviço PCM estão em fase de produção e cumprem todos as necessidades e requisitos iniciais.

O trabalho futuro de maior interesse nesta área seria o estudo da possibilidade de utilização do motor de regras para aprendizagem do comportamento do utilizador e, por exemplo, a sugestão de novas regras no serviço PCM.

Apêndices

Apêndice A

Apreciação do Estágio

Este apêndice tem como objectivo enquadrar o projecto desenvolvido na disciplina de Projecto de Engenharia Informática. Consequentemente, é feita uma apreciação pormenorizada do estágio.

A.1 Enquadramento académico

Das várias propostas de estágio no âmbito da cadeira de PEI, este era o estágio que oferecia um plano de trabalho mais enquadrado no plano curricular do MEI. Este estágio destacava-se pela existência de um plano de trabalho concreto (durante toda a duração da cadeira PEI) e pela possibilidade de ser enquadrado num projecto de tecnologias emergentes.

Uma vez que o estágio não oferecia um ambiente académico de investigação mas também não se tratava de um ambiente de trabalho, foi possível ao aluno desenvolver as competências técnicas e simultaneamente complementar a formação com as *soft skills* inerentes a um ambiente de trabalho formal.

É importante referir que este não se tratava de um estágio com um projecto focado na produção, existiu a constante preocupação de formação e auto-formação do aluno.

Pelas características supracitadas, este foi o tipo de estágio que mais se adequava ao objectivo de complemento da formação académica da disciplina de PEI.

A.2 Estágio

Nesta secção analisam-se em detalhe todas as variáveis que envolveram o estágio durante o projecto, nomeadamente a instituição de acolhimento, a equipa de trabalho, as metodologias de trabalho e o projecto desenvolvido.

A.2.1 Instituição

Das muitas instituições que acolhem estagiários, a PT Inovação, pelo seu objectivo e método de trabalho, é sem dúvida uma das mais indicadas.

Esta instituição tem com frequência projectos de estágio curriculares e profissionais desenvolvidos em conjunto com a Associação InovaRia[1] e no âmbito do Programa Talento.

Durante o estágio foram dadas todas as condições necessárias para desenvolver as tarefas atribuídas. Em termos de equipamento e local de trabalho não há nada de relevante a assinalar.

A.2.2 Equipa

O facto de equipa ser muito informal e sem funções rígidas permitiu adquirir conhecimentos e formação *on-job* com grande facilidade e sobre vários assuntos (mesmo os que não estavam directamente relacionados com o estágio).

A existência de um responsável pelo *Quality Ansurance* e*Integration Tests*, para além de facilitar o *deployment* no cliente, garante que nenhum código entra em produção sem antes passar todos os testes funcionais e de integração

O Director de Departamento e Gestor do Projecto define as *guidelines* e a arquitectura do(s) projecto(s). Positivamente, há a destacar o seu empenho, acompanhamento e a activa participação nas decisões de concepção do projecto.

Existe, no entanto, uma pequena lacuna no acompanhamento mais próximo ao estagiário. As pequenas decisões de concepção (e.g., código fonte) ou as dificuldades encontradas a um nível mais baixo muitas vezes não podem ou conseguem ser auxiliadas pela restante equipa. Sente-se assim a falta de uma pessoa que desempenhe uma função intermediária (entre o estagiário e o Gestor do Projecto). Não se pode no entanto esquecer que esta variável influência positivamente as aptidões autodidactas e de independência do estagiário.

A.2.3 Metodologia de Trabalho

O projecto desenvolvido foi sendo acompanhado por uma plataforma de trabalho colaborativo. Esta plataforma é apoiada por um sistema tipo wiki (onde é mantida a documentação) e por um sistema de gestão de requisitos, tarefas, testes e defeitos do projecto. Para além de servir de repositório de informação e de gestão do projecto, esta plataforma permite também uma melhor gestão das tarefas individuais dos membros da equipa (com o conhecimento dos restantes).

Normalmente, o processo desenvolvimento de *software* para cada componente é muito semelhante ao processo cascata. Sendo este um projecto muito dinâmico e surgindo com facilidade novos requisitos, poderá ser necessário mais do que uma interacção para o mesmo componente do projecto.

[1] http://www.inova-ria.pt

Na fase inicial o Gestor de Projecto, define os requisitos de primeiro nível e atribui a responsabilidade de análise mais profunda a um dos membros da equipa. Se, da investigação e da definição e análise de requisitos, surgir a decisão de concepção de um novo componente ou a reformulação de um componente existente, passa-se para a fase de desenho do componente. Na maioria dos casos, a fase de desenho involve toda a equipa de desenvolvimento, o que proporciona uma valiosa troca de ideias e uma mais correcta definição do desenho do sistema. A concepção (terceira fase) e os testes unitários (inicio da fase de testes) são da responsabilidade de um dos membros da equipa. A restante fase de testes, com testes funcionais e de integração, fica a cargo do membro responsável pela área.

As principais desvantagens da concepção individual prendem-se com a maior dificuldade em transmitir o conhecimento do resultado do trabalho e a não interacção ao longo desta fase com os restantes membros da equipa. Embora os resultados aparentem ser mais rápidos, numa perspectiva a longo prazo, esta poderá não ser a melhor opção. Utilizando por exemplo o método de trabalho *pair programming*, os prazos serão alargados mas o resultado terá melhores métricas de qualidade.

A.2.4 Projecto Desenvolvido

O planeamento de trabalho inicial foi cumprido em pleno. Tendo existido as condições necessárias ao desenvolvimento de outras tarefas, tempo e disponibilidade por parte do aluno e dos orientadores, foram desenvolvidas muitas outras tarefas relacionadas com o projecto.[2]

Para além de todas as vantagens citadas neste documento, pela sua natureza e complexidade, este projecto permitiu a interacção e trabalho em conjunto com equipas de empresas internacionais.

A.3 Apreciação Global

Para a instituição de acolhimento, o resultado do estágio foi positivo e está neste momento incorporado na principal solução IMS da empresa (com perspectivas de entrar em produção).

Para o estagiário, foi uma óptima oportunidade de ser enquadrado num projecto de tecnologias emergentes onde pôde adquirir conhecimentos sobre vastas áreas do mundo das telecomunicações. Para além do planeamento inicial que foi totalmente cumprido, através de tarefas não planeadas, o estagiário adquiriu competências técnicas em áreas relacionadas.

Em suma, analisando estes meses de estudo e trabalho facilmente se avalia o estágio como extremamente positivo.

[2]Deliberadamente, a maioria das tarefas não planeadas não é relatada neste documento ou existem apenas as referências necessárias à boa compreensão dos raciocínios expostos

Acrónimos

3GPP	The 3rd Generation Partnership Project
AC	Activity Context
ACI	Activity Context Interface
API	Application Programming Interface
AS	Application Server
BGCF	Breakout Gateway Control Function
CSCF	Call Session Control Function
DRL	Drools Rule Language
DSL	Domain Specific Language
EDA	Event Driven Architecture
EJB	Enterprise Java Beans
GSTN	General Switched Telephony Network
HSS	Home Subscriber Server
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
JAIN	Java Advanced Intelligent Networks (anteriormente: Java APIs for Integrated Networks)
JCR	Java Community Process

JEE Java Enterprise Edition

JSP Java Server Pages

JSR Java Specification Request

JVM Java Virtual Machine

MGCF Media Gateway Control Function

MGW Media Gateway

MRF Media Resource Function

MRFC Media Resource Function Controller

MRFP Media Resource Function Processor

NGN Next Generation Network

NMS Network Management Systems

OMA Open Mobile Aliance

OSS Operations Support Systems

PCM Personal Communication Manager

PSTN Public Switched TelephoneNetwork

PUMA Personal Unified Multimedia Applications

RA Resource Adapter

SBB Service Building Block

SBC Session Border Controller

SE Service Enabler

SIP Session Initiation Protocol

SLEE Service Logic Execution Environment

SNMP Simple Network Management Protoco

WME Working Memory Element

XML Extensible Markup Language

Bibliografia

[1] Simplified guide to the java 2 platform, enterprise edition. Technical report, Sun Microsystems, 1999.

[2] Delivering business value to the telecommunications industry with sun javatm enterprise system. Technical report, Sun Microsystems, 2004.

[3] Ims overview and applications. Technical report, 3G Americas, 2004.

[4] Ip multimedia subsystem white paper. Technical report, Motorola, 2005.

[5] Jboss application server. Technical report, JBoss, 2005.

[6] Introduction to ims. Technical report, Ericsson, 2007.

[7] Jboss rules fact sheet. Technical report, JBoss, 2007.

[8] Gonzalo Camarillo and Miguel-Angel Garcia-Martin. *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds.* John Wiley and Sons Ltd, second edition, 2005.

[9] Paulo Chainho. Xmas - definição de requisitos e arquitectura. Technical report, PT Inovação, Abril 2005.

[10] David Ferry, David Page, Swee Boon Lim, and Phelim O'Doherty. Jain slee tutorial. Technical report, Open Cloud and Sun Microsystems, 2003.

[11] Steven Grover. Java in communications: Jain slee. Technical report, Sun Microsystems, 2005.

[12] JAINSLEE.org. Url - www.jainslee.org, 2006.

[13] JBoss. Url - www.jboss.com, 2006.

[14] Dan Leih and Dave Halliday. Standards, protocols, architecture and functions of the ip multimedia subsystem. Technical report, Motorola, 2006.

[15] Swee Boon Lim and David Ferry. Jain slee 1.0 specification. Technical report, Sun Microsystems, Inc; Open Cloud Limited, 2002.

[16] Michael Maretzke. Jain slee techonology overview. Technical report, 2005.

[17] Alexandre Mendonça, Bruno Duarte, Eduardo Martins, Luis Barreiro, Neutel Rodrigues, Paulo Freire, Paulo Chainho, and Toni Barata. Comunicações multimédia personalizadas. *Saber e Fazer*, Novembro 2006.

[18] Mobicents. Url - www.mobicents.org, 2006.

[19] Tony Morgan. *Business Rules and Information Systems: Aligning IT with Business Goals*. Addison Wesley Professional, 2002.

[20] Miikka Poikselka, Aki Niemi, Hisham Khartabil, and Georg Mayer. *The IMS: IP Multimedia Concepts and Services*. John Wiley and Sons Ltd, 2006.

[21] Alex Toussaint. Java rule engine api tm jsr-94. Technical report, BEA Systems, 2003.